UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Università degli Studi di Padova

Department of Information Engineering

Master Thesis in Control Systems Engineering

TITOLO

Supervisor:

Master Candidate: Matteo Grandin

1 Abstract

Contents

2 Listing of figures

3 Acknowledgments

Thank you thank you thank you thank you very much

4 Introduction

Spiegazione della struttura della tesi e dei metodi utilizzati.

5 Context Analysis

5.1 Problem Statement

This thesis project is placed in the context of autonomous vehicles and autonomous driving: more specifically we want to target the problem of lane following in a vehicle equipped with a frontal camera. In particular, we want to estimate the heading error of the vehicle with respect to the center of the lane and the structure of the road ahead of the vehicle, using only local information about the car and the road. The idea behind the need of an accurate estimation of these road parameters is that they can be used in a standard control strategy to perform lane following or other driving behaviors.

The objective is to design a system which satisfy a set of reasonable constraint in a real drive situation:

- The system should be able to accurately estimate the car position and orientation in the lane. The accuracy should be measurable and quantifiable using some appropriate performance metrics.
- The system should be able to estimate the structure of the lane ahead using some appropriate parameters, with an accuracy measurable and quantifiable using some appropriate performance metrics.
- The system should operate in a real-time environment, with low latency and high frequency, in order to be able to handle possible obstacles and changes in the road.
- The estimation process should only use local information about the vehicle and the road, without the use of any other localization system or map; in order to increase the robustness in a loss-of-signal situation, and to mimic the way human drivers approach the same task.

5.2 Scaled vehicle assembly

Descrizione del veicolo, da valutare se mettere nella sezione dei test/risulati

5.3 Vehicle model and characterization of the heading error

The entire car assembly is composed of the vehicle and a frontal camera. The camera is mounted on the vehicle in a fixed position, located in an elevated point for better visibility and pointing ahead towards the road. The vehicle is a standard car with 4-wheel drive, with standard suspensions and frontal steering.

In the literature there exists several vehicle models, that span across a wide range in complexity and degrees of freedom [1]. The more accurate models are used when the vehicle is driving close to the limit conditions, for example at very high speeds and accelerations in a racing scenario; the less accurate, but simpler models can be used if the vehicle is driving at low speeds in average conditions. In this thesis we will use a simple model, and more specifically the *bicycle model* [2] to describe the vehicle, the reason behind this choice is that we are interested in developing a general framework that is focused on the vision system and basic physical quantities of the vehicle like speed and steering angle; in addition to this, in the model vehicle we were working on we were able to directly control only the steering angle and the vehicle velocity.

5.3.1 The Bicycle Model

The bicycle model represents the foundation for vehicle modelling, the more complete models are build on top of this model by removing or lightening some of its assumptions. The bicycle model takes a 4-wheel model and combines the front and rear wheels respectively to form a 2-wheeled model (hence the name). We can therefore deal with only 2 wheels and 1 steering angle instead of 4 wheels and 2

steering angles. The bicycle model is based on the following assumptions:

- The vehicle is operating on a 2-dimensional plane. This is a reasonable assumption, it means that the vehicle does not move in the vertical direction.
- The vehicle is a rigid body with its mass concentrated in the center of mass.
- No-slip condition: there is no lateral or longitudinal slip in the tires; therefore we can assume the velocity of the wheels acts in the same direction that the wheel is facing in.

The idea is to consider only the vehicle speed and steering angle as inputs, and use the model to estimate the x and y position as wheel as the vehicle heading direction using the model.

Using the aforementioned assumptions, it is possible to derive the following equations of motion for the bicycle model:

$$\dot{x} = v \cos\theta \tag{1}$$

$$\dot{y} = v \sin\theta \tag{2}$$

$$\dot{\theta} = \frac{v}{L} \tan\delta \tag{3}$$

with the following symbol definitions:

- θ = vehicle yaw angle, i.e. heading direction
- L = wheelbase, the distance between the wheels
- δ = steering angle
- v = vehicle speed

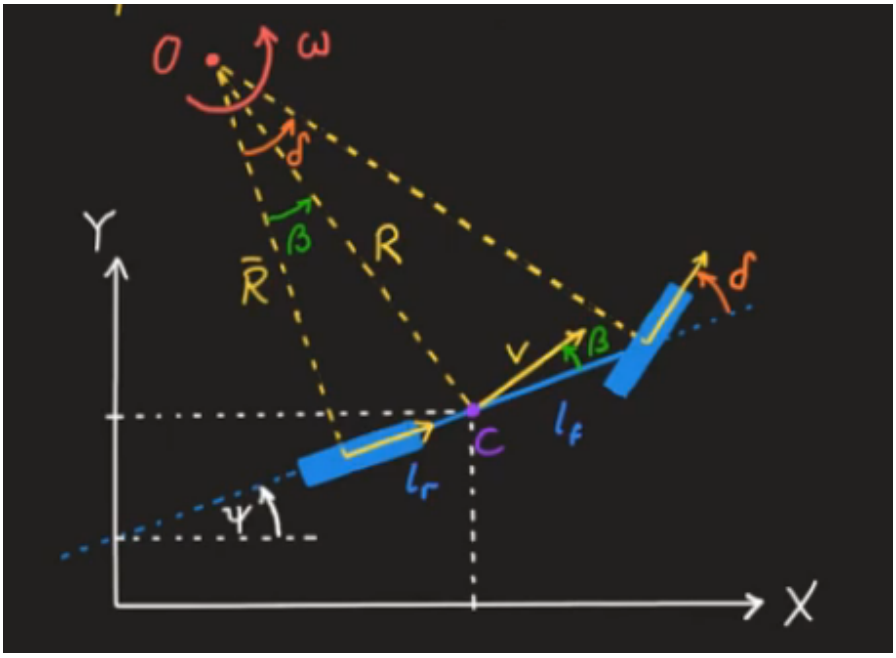

Figure 1: Bicycle model

5.3.2 Pure Pursuit and Heading Error

Pure pursuit is a tracking algorithm that works by calculating the curvature that will move a vehicle from its current position to some goal position. The whole point of the algorithm is to choose a goal position that is some distance ahead of the vehicle on the path. The name pure pursuit comes from the analogy that we use to describe the method. We tend to think of the vehicle as chasing a point on the path some distance ahead of it - it is pursuing that moving point. That analogy is often used to compare this method to the way humans drive. We tend to look some distance in front of the car and head toward that spot. This lookahead distance changes as we drive to reflect the twist of the road and vision occlusions[3].

The heading error (HE) is the key parameter to be estimated, as it is the starting point of the pure pursuit control strategy[4, 5]. The HE is defined as

the angle between the vehicle heading direction and the line that connects the lookahead point and the center of rear axle of the vehicle. The lookahead point is defined as the point on the path on a fixed (or variable) distance ahead of the vehicle.

$$\delta = \tan^{-1}\left(\frac{L\dot{\theta}}{v}\right) = \tan^{-1}\left(\frac{L}{R}\right) = \tan^{-1}\left(\frac{2L\sin(\alpha)}{l_c}\right) \tag{4}$$

The distance of the lookahead point is a delicate parameter, since a big value can lead to cutting curves at low speed and a too small value can lead to unstable behaviors at higher speeds. Moreover, the accuracy of the estimation is also affected by this distance, since it becomes more difficult to identify the lines which are far ahead. For these reasons an analysis on the lookahead distance is necessary.

In the practical implementation used to generate the training datasets in the simulator, we assume to have an *optimal* path[1] that follows the center of the lane. The implementation follows the same approach explained in [3]; in particular, the basic steps of the algorithm are:

1. Calculate the closest point p_0 on the path to the vehicle position, using the standard euclidean distance.

2. Calculate the lookahead point on the path, using the lookahead distance l_c from p_0. In this case we consider the arc length[2]

3. Convert the lookahead point to the vehicle coordinate system

4. Calculate the heading error

[1]The optimality of the path will be discussed in more detail in later sections.

[2]The reasons behind the use of the arc length rather than the standard euclidean distance is that the existence and uniqueness of the lookahead point is guaranteed only by using the arc length. Moreover, for small curvature paths, the metrics lead to very similar values.

Figure 2: Heading Error

It is important to note that this process makes use of the position and orientation of the vehicle in the global frame of reference; therefore it can be used only in a simulated environment. This is indeed the algorithm used to generate the training datasets.

5.3.3 Camera

The camera is the main sensor used to perform the heading error estimation. In the model car it is positioned about 20 cm above the ground. It is oriented 20 degrees downward in order to see also the closer section of the road. The camera is a standard *Raspberry Pi Camera* module. The technical specifications of the camera are the following:

- Still resolution: 8 Megapixels
- Video capture resolution: up to 1080p30, used at 640x480p at 30fps (max 90fps)
- Focal length: 3.04 mm
- Horizontal field of view: 62.2 degrees
- Vertical field of view: 48.8 degrees
- F-stop: 2.0

The native resolution used is 640x480 since it allows for a better frame rate, however the images will be further reduced in size in order to increase performance. The simulated camera is mounted in the exact same position of the real one, and has been programmed to have the same field of view and frame rate.

5.4 Simulation environment

The simulation environment is a key component of the estimation process, it is not directly related to the online estimation itself, but it is used to generate the training datasets; the accuracy of the estimation is therefore extremely correlated with the quality of the dataset and, consequently, with the quality of the simulation.

For the simulation environment, we use the *Gazebo simulator*[6], which is a powerful simulator normally used in robotics. The reasons behind the use of this simulator are the fact that it's open source and does not require extreme hardware to run, moreover we were able to use assets that were given to us by the organizers of the challenge. In the following are listed the main characteristics of the simulation environment:

- It is a physics-based simulator

Figure 3: Example image from the camera

- The simulation is modular, allowing for the use of different models
- *Gazebo* is fully integrated with *ROS*, this allows for easy access to all the sensors and actuators of the simulated vehicle and control of the simulation.
- The basic environment is composed of the model of the car and the streets, depending on the need it is possible to add other vehicles, pedestrians, and ramps.
- The car model is composed as a group of links/joints, with their respective physical properties. All the virtual sensors and actuators are mounted on the car model and are accessible through the *ROS* network.
- The virtual camera is mounted in the exact same position of the real one, and it's configured to have the same field of view and frame rate. The flow of images is easily accessible through *ROS*.

The simulator has been run on a laptop with an *Intel Core i7-7500U* processor, an *NVIDIA GeForce GTX 950M* graphics card and 16 GB of RAM.

6 Optimal path

6.1 Why do we need a path to follow?

The basic idea of having an *optimal* path is being able to localize the vehicle on the path and to sample the heading error looking at the path ahead.

Since we are only interested in estimating the heading error with respect to the center of the road it does not seem necessary to have an optimal path; however there are several reasons why one would prefer a different path to the centerline. The first one is that the centerline could be an infeasible trajectory to follow for the vehicle, with an aggressive curvature or discontinuous first derivative. The

Figure 4: Simulator image

second one is that the path method is very easy to scale in situations where the centerline is not clear or defined, for example when dealing with intersections or unmarked roads. The third is that it's possible to have some degrees of control over the path, for example on the smoothness or the maximum curvature. Moreover, the path concept is useful when planning far ahead in the future or when considering obstacle avoidance. Finally, using a path is very convenient from the implementation point of view, because it decouples the camera image from the localization and path tracking.

6.2 Why clothoids

Common path-planning methods usually generate obstacle-free path, but with no or very little concern about path feasibility or optimality so that it is usually necessary to apply some kind of transform algorithm to locally smooth such a path. Various path-smoothing algorithms are proposed in the literature: cubic splines [7], intrinsic splines [8], Bezier's curves [9], quintic Bezier splines [10], and clothoids. The main advantage of clothoids over other smoothing methods in path-planning applications is linear change of their curvature, which is of great importance for transportation of people or heavy and sensitive loads since it prevents abrupt changes in the centripetal acceleration and forces experienced by a vehicle increasing driving comfort. Clothoids are very attractive in path-smoothing applications as they are easy to follow because of their linearly changing curvature [11].

Clothoids also have advantages over other smoothing techniques in sense of vehicles optimal motion planning—by applying the Maximum Principle from the optimal control theory, for forward motion and differential drive vehicle, one can find that the necessary condition for trajectory to be time optimal yields clothoids [12]. Further, Boissonnat et al. [13] studied the shortest plane paths joining two given positions with given tangent angles and curvatures along which the tangent

angle and the curvature are continuous and the derivative of the curvature is bounded. They showed that at a point where such a path is of class $\mathcal{C}^3$, it must be locally a piece of a clothoid or a line segment. Similarly, Fraichard and Scheuer [14] showed that with requirements of continuous curvature and bounded both curvature and its derivative, the shortest path consist of line segments, circular arcs, and clothoids [15].

6.3 Clothoid definition

A Clothoid, also known as Euler or Cornu spiral, is a plane, invariant spiral curve defined in parameter form. As explained in [16] Its curvature κ can be expressed as a linear function of its arc length s:

$$\kappa(s) = \kappa_0 + \sigma s, \quad \kappa_0,\, \sigma \in \mathbb{R} \tag{5}$$

where σ is the sharpness of the clothoid and κ_0 is the initial curvature at $s = 0$. The tangent or winding angle θ with respect to its arc length is given as:

$$\theta(s) = \theta_0 + \kappa_0 s + \frac{\sigma s^2}{2}, \quad \theta_0,\, \kappa_0,\, \sigma \in \mathbb{R} \tag{6}$$

where θ_0 is the initial tangent angle at $s = 0$. Thus, a general clothoid can be expressed in paremetric form as in [17]:

$$\mathbf{F}(s) = \begin{pmatrix} x_0 + \int_0^s \cos\left(\theta_0 + \kappa_0 u + \frac{\sigma u^2}{2}\right) du \\ y_0 + \int_0^s \sin\left(\theta_0 + \kappa_0 u + \frac{\sigma u^2}{2}\right) du \end{pmatrix} \tag{7}$$

Based on the above definition, it is possible to define the elementary clothoid segment $\mathbf{F}_{\mathcal{E}}(s)$ with $\sigma = \sigma_{\mathcal{E}} > 0$, $s \geq 0$, $\kappa_0 \geq 0$, $\theta_0 = 0$, $(x_0, y_0) = (0, 0)$ and $\theta \in (0, \pi/2]$

$$\mathbf{F}_{\mathcal{E}}(s) = \sqrt{\frac{\pi}{\sigma_{\mathcal{E}}}} \mathbf{R}\left(-\frac{\kappa_0^2}{2\sigma_{\mathcal{E}}}\right) \begin{pmatrix} \delta_c(s) \\ \delta_s(s) \end{pmatrix} \tag{8}$$

where $\delta_c(s) = C_f(\frac{\sigma_{\mathcal{E}} s + \kappa_0}{\sqrt{\pi\sigma_{\mathcal{E}}}}) - C_f(\frac{\kappa_0}{\sqrt{\pi\sigma_{\mathcal{E}}}})$ and $\delta_s(s) = S_f(\frac{\sigma_{\mathcal{E}} s + \kappa_0}{\sqrt{\pi\sigma_{\mathcal{E}}}}) - S_f(\frac{\kappa_0}{\sqrt{\pi\sigma_{\mathcal{E}}}})$. Here, $\mathbf{R}(\theta)$ is the standard planar rotation matrix, and $C_f(x)$ and $S_f(x)$ are a pair of nonnegative functions named Fresnel integrals [18]:

$$C_f(s) = \int_0^s \cos\left(\frac{\pi\xi^2}{2}\right) d\xi,\ S_f(s) = \int_0^s \sin\left(\frac{\pi\xi^2}{2}\right) d\xi. \tag{9}$$

6.4 Implementation

The context of our particular application is the following: we are given a map of a scaled city, with roads, intersections, roundabouts, crosswalks, etc... Alongside with the map we are also given a set of waypoints in the form of a directed graph. Every waypoint corresponds to an (x, y) position on the map, more specifically to point in the center of a lane. All the waypoints are placed at about $30cm$ from each other. Every waypoint also incorporate a list of edges that connects it to other waypoints, in such a way that only consecutive waypoints are connected by an edge and only if they follow they same lane, the only exceptions being the points associated to roundabouts and intersections as well as the start point and the end point.

The goal of the algorithm is to take 2 waypoints as input and return a path that connects them. To achieve this goal, we start by finding the shortest path in the graph of waypoints that connects them[3]; to do this we use the Dijkstra algorithm [19].

The second and last step to create the *optimal* path is to fit clothoid segments between the waypoints of the shortest path. Since the Fresnel integrals in equation (9) are two transcendental functions, they can't be solved analytically, however there are several approximation methods in the literature that achieves good results [15, 16, 20]. In our implementation we used the python library *pyclothoids* [21],

[3]We restrict ourselves to the case where at least one path exists between any two waypoints, which was always true in our scenario, except if we considered the end point as starting point

which is a python wrapper of the C++ library by Bertolazzi [22–25]. In figure 5 it is possible to see the result of the algorithm, the path is designed to cross most of the waypoints.

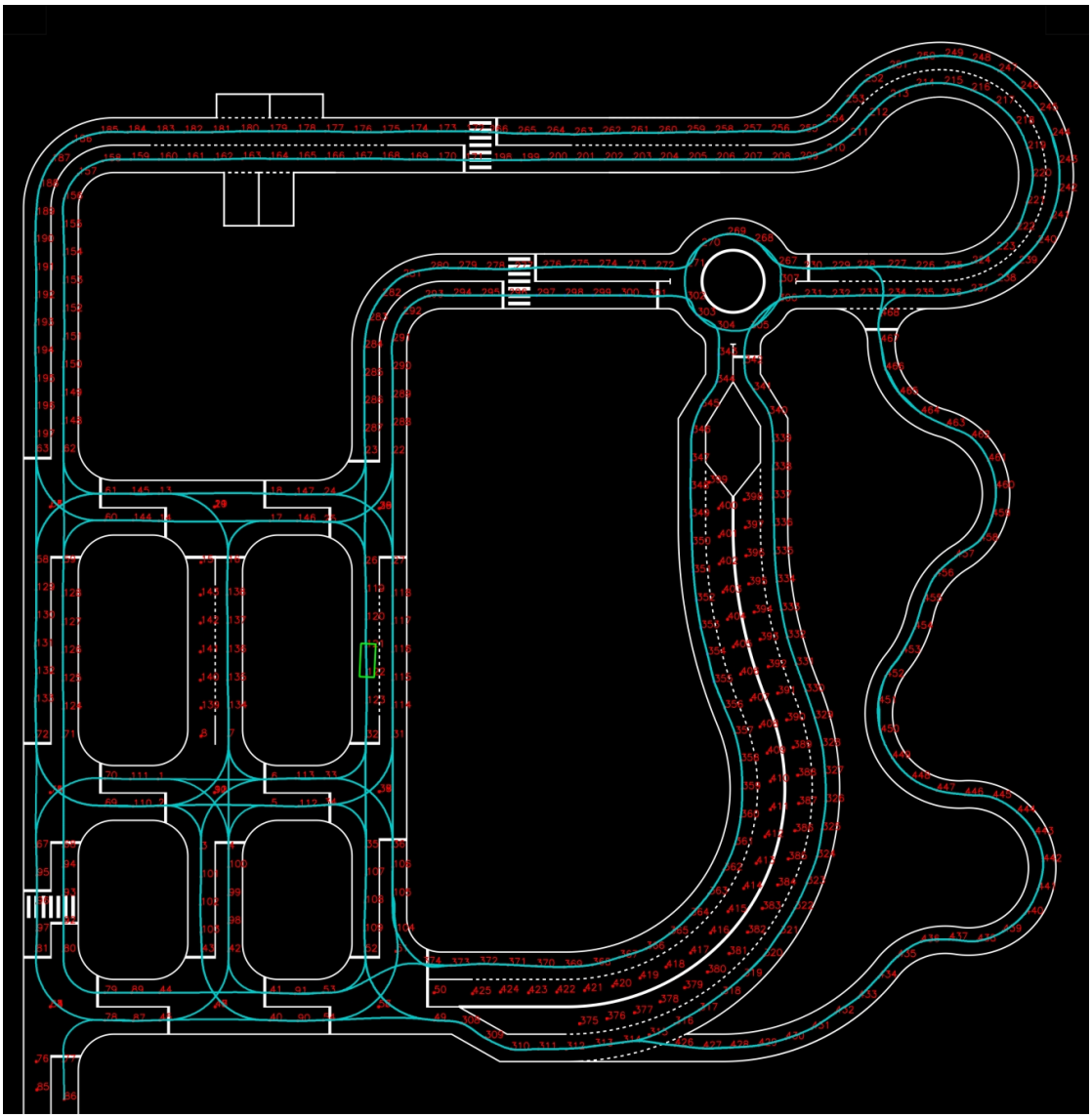

Figure 5: Map and waypoints, with the optimal path passing through most of the waypoints

7 Performance metrics

Definizione delle metriche di performance

Read code not pdf here... -L2 norm that we have both in simulation and hopefully in the real world thanks to a precise localization system (MG idea) -The ones explained in 'On Performance Evaluation Metrics for Lane Estimation' -The ones explained in section 5 of 'Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation' -they used mean absolute error in position, standard deviation of error in position, standard deviation of error in rate of change of lateral position the idea of the rate of change is very interesting -We should consider the 2 cases: the case of the heading error estimation and the case of the actual lane detection, there should be 2 different performance metrics for each one of them -We should consider the fact that we have the simulation environment and evaluate the performance in easy conditions, namely when the tracking is near the center of the road, or when the car is sliding left and right -we should evaluate the continuity of the output as a performance measurement -also we should distinguish the performance metric of the road estimation, from the whole estimation+tracking performance, in this paper we are interested only in the first one. -check papers: `https://ieeexplore.ieee.org/document/6728473`, `https://ieeexplore.ieee.org/document/9398517`, which are more about the deep learning aspect -computational efficiency -different performance metrics for different sections (curves, straight lines)

useful papers: `https://ieeexplore.ieee.org/abstract/document/6977166`,`https://link.springer.com/article/10.1007/s00138-011-0404-2`,`https://ieeexplore.ieee.org/document/1603550`,`https://ieeexplore.ieee.org/document/6728473`,`https://ieeexplore.ieee.org/document/9398517`

8 Neural Network Estimation

read code not pdf here...

SECTIONS: -Data collection -explaining data collection in simulator -how all the pieces come together -reasons behind the choices -(explaining data collection in real world, better explained in results sections) -dependency on possible parameters: -noise level in steering, can be evaluated both a priori(type of noise injection) and a posteriori(variance between optimal path and training path), -noise level in speed -number of frames -fps of the Video -length of Video -length of simulation path -(variability of the path, choosing a very repetitive path or a variable one) -(camera orientation, requires redoing all videos) -(camera position, requires redoing all videos)

-Neural Network Estimation -objectives of the network -estimating heading error -fast estimation -accuracy -continuity -(estimating the road) -explainability -architecture -explain all the layers and what they do -layers, activations, weights initialization, biases initialization, and reasons for everything -pytorch implementation -Data flow -io of the network -image dimension -data augmentation -explain how it works, reasons for doing it, and why it is necessary

-data normalization -data splitting -data loading -data batching -data shuffling -training -optimizer -loss function -learning rate schedule -early stopping -validation -training loop -epochs -losses, and loss plots -Network analysis -visualizations of conv filters -visualizations of fully connected layers -layer activations -layer weights -layer biases -(Extensions) -(multiple frames) -(time coherence) -(continuity) -(multiple points estimation, path ahead) -variable speed -state estimation -reducing dataset size

Poi discussione sulle estensioni:

- multiple frames

- time coherence
- continuity
- multiple points estimation, path ahead
- variable speed
- state estimation
- reducing dataset size

9 Results

Analisi dei risultati in relazione a hyperparamters, optimal path, quantita' di dati, randomicita', metriche di prestazione differenti, facilita' di implementazione e di distribuzione, etc...

10 Conclusion and Future Work

end

11 Appendix

consider adding full derivation of the bicycle model https://www.youtube.com/watch?v=HqNdBiej23I&ab_channel=Prof.GeorgSchildbach%2CUniversityofLuebeck https://thomasfermi.github.io/Algorithms-for-Automated-Driving/Control/BicycleModel.html http://code.eng.buffalo.edu/dat/sites/model/bicycle.html#:~:text=motion%20under%20the%20assumption%20of,hard%20braking%20and%20steering%20maneuvers. https://ieeexplore.ieee.org/document/9311987

consider adding derivation of the pure pursuit https://ieeexplore.ieee.org/document/9368694

References

[1] Dirk E. Smith and John M. Starkey. Effects of model complexity on the performance of automated vehicle steering controllers: Model development, validation and comparison. *Vehicle System Dynamics*, 24(2):163–181, 1995.

[2] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017.

[3] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[4] Hiroki Ohta, Naoki Akai, Eijiro Takeuchi, Shinpei Kato, and Masato Edahiro. Pure pursuit revisited: Field testing of autonomous vehicles in urban areas. In *2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, pages 7–12, 2016.

[5] Yaofu Huang, Zengshan Tian, Qing Jiang, and Junxing Xu. Path tracking based on improved pure pursuit model and pid. In *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT*, pages 359–364, 2020.

[6] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004.

[7] Y Kanayama and B Hartman. Smooth local path planning for autonomous vehicles (part ii. cubic spirals). In *Proc. Annual Conference of Robotics Society*

of Japan, pages 93–96, 1988.

[8] Hervé Delingette, Martial Hebert, and Katsushi Ikeuchi. Trajectory generation with curvature constraint based on energy minimization. 1991.

[9] A Segovia, M Rombaut, A Preciado, and D Meizel. Comparative study of the different methods of path generation for a mobile robot in a free environment. In *Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments*, pages 1667–1670. IEEE, 1991.

[10] Boris Lau, Christoph Sprunk, and Wolfram Burgard. Kinodynamic motion planning for mobile robots using splines. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2427–2433. IEEE, 2009.

[11] Madhavan Shanmugavel, Antonios Tsourdos, Brian White, and Rafał Żbikowski. Co-operative path planning of multiple uavs using dubins paths with clothoid arcs. *Control engineering practice*, 18(9):1084–1092, 2010.

[12] S. Fleury, P. Soueres, J.-P. Laumond, and R. Chatila. Primitives for smoothing mobile robot trajectories. *IEEE Transactions on Robotics and Automation*, 11(3):441–448, 1995.

[13] Jean-Daniel Boissonnat, André Cérézo, Elena V. Degtiariova-Kostova, Vladimir P. Kostov, and Juliette Leblond. Shortest plane paths with bounded derivative of the curvature. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 329(7):613–618, 1999.

[14] T. Fraichard and A. Scheuer. From reeds and shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, 20(6):1025–1035, 2004.

[15] Mišel Brezak and Ivan Petrović. Real-time approximation of clothoids with

bounded error for path planning applications. *IEEE Transactions on Robotics*, 30(2):507–515, 2014.

[16] Yong Chen, Yiyu Cai, Jianmin Zheng, and Daniel Thalmann. Accurate and efficient approximation of clothoids using bézier curves for path planning. *IEEE Transactions on Robotics*, 33(5):1242–1247, 2017.

[17] AW Nutbourne, PM McLellan, and RML Kensit. Curvature profiles for plane curves. *Computer-aided design*, 4(4):176–184, 1972.

[18] B.A. Barsky and T.D. DeRose. Geometric continuity of parametric curves: three equivalent characterizations. *IEEE Computer Graphics and Applications*, 9(6):60–69, 1989.

[19] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[20] Nicolas Montes, Alvaro Herraez, Leopoldo Armesto, and Josep Tornero. Real-time clothoid approximation by rational bezier curves. In *2008 IEEE International Conference on Robotics and Automation*, pages 2246–2251, 2008.

[21] Johannes Schmitz and phillipd94. pyclothoids. `https://github.com/phillipd94/pyclothoids`, 2020. [Online; accessed 2022].

[22] Enrico Bertolazzi and Marco Frego. G1 fitting with clothoids. *Mathematical Methods in the Applied Sciences*, 38(5):881–897, 2015.

[23] Enrico Bertolazzi and Marco Frego. On the g2 hermite interpolation problem with clothoids. *Journal of Computational and Applied Mathematics*, 341:99–116, 2018.

[24] Enrico Bertolazzi and Marco Frego. Interpolating clothoid splines with curva-

ture continuity. *Mathematical Methods in the Applied Sciences*, 41(4):1723–1737, 2018.

[25] Marco Frego and Enrico Bertolazzi. Point-clothoid distance and projection computation. *SIAM Journal on Scientific Computing*, 41(5):A3326–A3353, 2019.