UNIVERSITÀ DEGLI STUDI DI TRENTO

INTERNATIONAL DOCTORATE SCHOOL IN INFORMATION AND COMMUNICATION TECHNOLOGIES

XXIV CYCLE – 2013

Competitive Robotic Car: Sensing, Planning and Architecture Design

Tizar Rizano

UNIVERSITÀ DEGLI STUDI DI TRENTO

INTERNATIONAL DOCTORATE SCHOOL IN INFORMATION AND COMMUNICATION TECHNOLOGIES

XXIV CICLO - 2013

Tizar Rizano

Competitive Robotic Car: Sensing, Planning and Architecture Design

Luigi Palopoli (Advisor)

Thesis Committee

Lucia Pallottino

Antonio Giannitrapani

AUTHOR'S ADDRESS:

Tizar Rizano

Dipartimento di Ingineeria e Scienza dell'Informazione

Università degli Studi di Trento

via Sommarive 14, I-38050 Povo di Trento, Italy

E-MAIL: rizano@disi.unitn.it

WWW:

Abstract

Research towards a complete autonomous car has been pushed through by industries as it offers numerous advantages such as the improvement to traffic flow, vehicle and pedestrian safety, and car efficiency. One of the main challenges faced in this area is how to deal with different uncertainties perceived by the sensors on the current state of the car and the environment. An autonomous car needs to employ efficient planning algorithm that generates the vehicle trajectory based on the environmental sensing implemented in real-time. An complete motion planning algorithm is an algorithm that returns a valid solution if one exist in finite time and returns no path exist when none exist. The algorithm is optimal when it returns an optimal path based on some criteria. In this thesis we work on a special case of motion planning problem: to find an optimal trajectory for a robotic car in order to win a car race. We propose an efficient realtime vision based technique for localization and path reconstruction. For our purpose of winning a car race we identify a characterization of the alphabet of optimal maneuvers for the car, an optimal local planning strategy and an optimal graph-based global planning strategy with obstacle avoidance. We have also implemented the hardware and software of this approach on as a testbed of the planning strategy.

Keywords

[localization, path reconstruction, motion planning, trajectory tracking]

Acknowledgments

Bismillah hir Rahman nir Rahim, Alhamdulillah hi Rabbil Alameen, Praise be to Allah Almighty who always provide me with everything more than I have ever asked to bring this PhD research and dissertation into completion.

For this PhD research I would like to express my sincere gratitude to my supervisor, Luigi Palopoli, who always have his faith in me and my work and always been very supportive and helpful. Through various topics of my research he always provided me with ideas of new directions and knowledge I needed for every obstacles I found throughout the research.

During the various researches I've completed I have also been supported by Daniele Fontanelli, David Macii and Roberto Passerone. They have, in particular, provided significant helps in my publications. Together with my supervisor, they have also provided all the moral supports which play important role for me to conclude this work. I really appreciate it.

I would also like to thank my fellow PhD students, Federico Moro and Alessio Colombo who provided my with their friendship as well as being great Aula 5 laboratory mates.

All my love and forever gratitude to my beloved parents, Erawati and Chaizarno who had given me the encouragement and support all the time that has enabled me to reach this point in my life. I would also like to thank my wife, Yusi Ramadian, for her support all along this PhD program. I could not be anything I am now without them.

Last, I hope that the work in this thesis will be of value, could be a base of further researches and be my contribution for the knowledge in general.

Trento, April 29th 2013

Tizar Rizano

Contents

List of Tables

List of Figures

List of Symbols

α_i	tire slip angles of the i wheels (front or rear)
$\langle C \rangle$	Camera reference frame
$\langle I_c \rangle$	Bi-dimensional Image reference frame
$\langle P \rangle$	Path reference frame
$\langle R \rangle$	Robot reference frame
$\langle T \rangle$	Track reference frame
$\langle V \rangle$	Virtual reference frame
$\langle W \rangle$	Right–handed fixed world reference frame
l_f	Longitudinal distance from CoM to the front wheel
l_r	Longitudinal distance from CoM to the rear wheel
λ	Line of sight ambiguity
$\omega_{w,i}$	Wheel i angular velocity
$\overline{\varphi}$	Maximum steering angle
$\overline{a}$	Maximum acceleration
$\overline{v}$	Vehicle maximum velocity

ϕ_p Yaw angle of the path

Π Plane of motion

σ_i Tire slip ratios of the i wheels (front or rear)

θ Orientation of the vehicle with respect to X_w axis

$\underline{a}$ Minimum acceleration

φ Steering angle of the front wheel of the vehicle with respect to the vehicle

v_x Longitudinal velocity

v_y Lateral velocity

${}^c g_r$ A generic rigid motion transformation between the reference $\langle R \rangle$ and camera $\langle C \rangle$

${}^c R_r$ Rotation matrix between the reference $\langle R \rangle$ and the camera $\langle C \rangle$

a Acceleration

B_l Track left boundary

B_r Track right boundary

c'_a Aerodynamic drag coefficient

C_b Bend sector center position

c_x Tire longitudinal stiffness

c_y Tire cornering stiffness

CoM Vehicle center of mass

F_{aero} Aerodynamic force

Fxi Longitudinal forces which is parallel to the rolling direction of wheel i

Fyi Lateral forces which is perpendicular to the rolling direction of wheel i

h_c Distance between $\langle C \rangle$ and the plane of motion Π

H_p Distance vector from $\langle P \rangle$ to the motion plane Π

h_p Distance of O_c from the plane of motion Π

I_c Image from the camera

I_v Virtual image

I_z Moment of inertia

l The distance between the wheel axes

m The mass of the vehicle

O_w Origin of the world frame

R_b Bend sector radius

r_{eff} The effective radius of the wheel

v The velocity of the rear–wheel axis midpoint

x Position of the vehicle rear–wheel axis midpoint in X_w

X_I X axis of the reference frame I

y Position of the vehicle rear–wheel axis midpoint in Y_w

Y_I Y axis of the reference frame I

Z_I Z axis of the reference frame I

${}^T p_l$ A point located on the left border of the track

${}^T p_r$ A point located on the right border of the track

${}^{p}x_{p,c}$ Distance from the camera to the path

${}^{r}t_{r,c}$ The translation vector from O_r to O_c expressed in $\langle R \rangle$

Chapter 1

Introduction

Autonomous car is one of the main research topics for both academia and industries. Autonomous car offers numerous improvement to traffic flow, vehicle and pedestrian safety, and car efficiency due to a well planned driving techniques. This research topic spans across several different fields such as robotics, control design, artificial intelligence, etc.

Competition such as DARPA (Defense Advanced Research Projects Agency) Grand Challenge and DARPA urban challenge has sparked a great research interest in autonomous car. This competition served as a prominent showcase of state of the art self driving cars both in off-road and urban environment. In the competition, highly advanced cars are built by universities with great support from their industrial partners. However, these autonomous cars are not yet available for consumer purchase.

Nevertheless, some aspects of the achievements in autonomous car research have been implemented in most modern car in the form of driver assistance and crash avoidance systems. Automatic braking system helps drivers avoid or reduce the impact from accidents. Parking assistance system alerts the driver to obstacle or provides automatic parking. Adaptive cruise control system provides a safe distance by detecting the vehicles and pedestrian in front of the car. Lane departure warning and lane keeping system assist the driver to perform heading control. Many major automotive vendors such as Mercedes-Benz, Volkswagen, Toyota, Audi, BMW have implemented some of these systems into their consumer products and have started the push for a complete autonomous car, i.e., self driving car.

One of the main problems in building an autonomous car is dealing with different level of uncertainties [36]. *Sensors* used to perceive the current car state and the environment is one of the main source of the uncertainties. In order to reach a specified goal, an autonomous car relies on a *planning algorithm*, which uses the output of the environmental sensing system to generate vehicle trajectory. A successful autonomous driving strategy should deal with the uncertainties and is efficient enough to have a *real-time* implementation.

Autonomous cars obtain information for the environment with multitude of different sensors. Some of the common sensors are: radar, sonar, LIDAR (Light Detection and Ranging), GPS and camera. GPSs are widely available in consumer vehicles. However, previous researches have shown that it cannot be relied upon for high-accuracy localization [36]. LIDARs and cameras are mounted on most of the cars competing in DARPA challenges [36, 57, 58]. LIDARs not yet widely deployed in consumer cars due to cost, safety or legal issues. In contrast, cameras are widely available of most high end cars. This fact motivates for research in vision based techniques to deal with the sensing problem.

Car motion planning is a special case of the general motion planning problem, which in general is very difficult to solve due to its multidimensional nature. State-of-the-art algorithms operating on a three-dimensional subspace of this problem space are difficult to compute in real time. Moreover, Several simplifications of the general problem have been proven to be unsolvable in polynomial time [12]. A motion planning algorithm is *complete* if it returns a valid solution if one exist in finite time and returns no path exist when non exist. The algorithm is *optimal* when it returns an optimal path based on some criteria. Note that an optimal motion planning algorithm is also complete.

In this thesis we present a solution for trajectory planning i.e., how to decide which trajectory should an autonomous car follows in order to reach its desired goal. The problem is particularly challenging when the car moves along a crowded and partially known urban or extra-urban roads [32, 33], a situation frequently considered in different research activities. In this situation, motion planning has to be carried out frequently in order to make the car responsive to environmental changes and to secure high safety standards. The presence of other cars and of pedestrians in urban environments requires a huge intake of sensor data and a heavy

processing power. We address similar problem in a different scenario: a robotic racing car that runs on a track. The main distinction between this problem and the urban problem is that the environment is well–structured and the only external presence allowed are other racing cars. This introduce a significant simplification of the motion planning problem.

The goal of the robotic car in a competition scenario is easy to state: winning the competition. In order to do so, we need algorithmic solutions that allow the robotic pilot to effectively manage several situation that potentially occur in the competition. Significant examples are:

1. when other cars are far away and do not affect the pilot maneuvers, it should complete the lap in the minimum time,
2. when a slower car is in front of the robot, it should engage an overtake maneuver,
3. when a car closes in from behind, the robot should defend its current position, "covering" the best trajectories to the incoming car.

We emphasize that the overtake maneuvers have to be chosen in a game competition with the opponent car, which in its turn tries to "cover" the best trajectories.

1.1 Problem Description

The goal of this thesis is to provide a solution for a special case of motion planning problem that is to find an optimal trajectory for a robotic car in order to win a car race. This problem encompass four sub problems:

1. **vision based localization and path reconstruction** : This subproblem deals with the environment sensing problem i.e., localizing the robot over the path and reconstructing the sector using only information obtained from the two cameras mounted on the car
2. **planning** : This subproblem deals with finding the optimal trajectory to win the race while avoiding obstacle along the track
3. **control** : This subproblem deals with designing controllers that is able to execute the synthesized trajectory.

4. **architecture design** : This subproblem deals with the hardware and software architecture for the implementation of the sensing and planning algorithm in order to have an efficient real-time execution on a the robotic car.

In this thesis we provides a solution for problem 1, 2, and 4. Control design is well researched topic and is out of the scope of the thesis. Thus, we decided to use currently available controllers.

1.2 Relation to Previous Work

In this section, a non-exhaustive survey of motion planning techniques and a comparison with the technique detailed in this thesis is presented.

1.2.0.1 Potential Fields

In the potential field approach [6, 19, 31], a potential function is assigned to the configuration space. The robot is modeled as a particle which reacts to the function due to the potential fields. These potential functions have two parts: attraction and repulsion. The attractive potential function assigns the minima i.e., the lowest potential at the goal. Therefore, this function draws the particle towards the goal. Conversely, the repulsive potential repels the particle away from the obstacle.

The advantage of this approach is its low computational complexity and it enables the robot to compute the potential function using local information. Thus, this techniques is suitable for real-time implementation. However the main disadvantage of potential fields that it is incomplete. The robot can be trapped in a local minima which is not the minima associated with the goal.

1.2.0.2 Roadmap

Roadmap methods reduce the motion planning problem to that of a graph search problem by fitting a roadmap to the space. Using a roadmap the planner can construct a path from the start

to the goal configuration by finding a collision free path (accessibility), traversing the roadmap towards the goal (connectivity) and constructing the path from a point in the graph to the goal (departability).

Visibility graph [38] construct a graph using the start state, goal state and and all vertices of the obstacles. The edges are built by connecting each vertex with all vertices visible from its position. Since the approach leads to a path which arbitrarily close to obstacles, Obstacles are usually approximated by enlarged spaces around them to avoid collision.

In Voronoi roadmap [17] approach configuration space is mapped onto a one-dimensional subset of the space. It build a boundary that is maximally distant from the obstacle. A minimum distance path is between two configurations follows the path in these boundaries. It is designed for a two-dimensional motion planning. Voronoi roadmap are complete but not optimal.

1.2.0.3 Sampling Based Approach

Unlike roadmap and potential field approach, sampling-based approach [29] does not rely on explicit representation of obstacles in the configuration space. Sampling-based algorithm relies on collision-detection modules to verify the feasibility of a candidate trajectories. A sampled based roadmap is built using a set of points sampled from the obstacle-free space. Sampling-based approach is not complete. However this approach provides probabilistic completeness guarantees i.e., the probability of returning no path exist approaches zero as the number of samples approaches infinity [5].

Some of the most well-known sampling-based motion planning algorithm are Probabilistic RoadMaps (PRMs) [30] and Rapidly-exploring Random Trees [35]. The PRM algorithm starts by constructing a graph of the configuration space that represents the set of collision-free trajectories and then answer queries by computing a shortest path that connects the initial state with a final state in the generated graph. There are some applications where building the graph a priori is not desirable. This motivates an incremental sampling-based planning algorithms. The RRT algorithm is an example of such algorithm. The incremental nature of RRT algorithms enables on-line implementations.

1.2.0.4 Optimal Control Based Approach

This approach comes from controls community where the focus is on the system's dynamics where the physical properties play a major role. Planning problem for a dynamic system is cast as an *Optimal Control Problem* (OCP).

Model Predictive Control (MPC)/Receeding Horizon Control (RHC) [49] is one of the most well known OCP based approach. MPC/RHS is a feedback control system where the OCP is solved at each time t for a finite horizon $t + T$. At each time t, MPC algorithm computes prediction for future steps until time $t + T$ based on the current measurement and computes an optimal control strategy. The control strategy is only applied until the next sampling time where the process prediction is repeated. This approach requires an accurate a priori internal model of the system to make the prediction.

1.2.1 Comparison with Proposed Approach

Compared to classical approach such as potential fields and roadmap, our approach accounts for the car dynamics such as tire slip, steering servo and aerodynamic force. MPC/RHC techniques required a highly accurate internal model to compute the optimal control at each step for a finite horizon. Our local planning approach used a similar optimal control technique. However, it is performed once for a given sector of a track. Therefore, all maneuvers can be computed offline for a given track and the online global planning can be reduced to graph search problem. Similar to sampling based technique such as RRT/RRT*, our approach builds a graph for a given configuration space. However due to the well–structured nature of the environment in a competition scenario, our approach exploits the geometric description of the track in order to construct the graph.

1.3 Contribution and Thesis Organization

The main contribution of the thesis are:

1. An efficient real–time vision based technique for localization and path reconstruction

2. A characterization of the alphabet of optimal maneuvers

3. An optimal local planning strategy

4. An optimal graph-based global planning strategy with obstacle avoidance

5. A hardware + software implementation as a testbed of the localization and path reconstruction algorithm.

This thesis is organized into seven chapters. The first chapter introduces the reader to motion planning in general and the motivation of this work. Chapter 2 describes the formalization of the robotic car and track model.

Chapter 3 details our solution to the sensing problem in an autonomous car. In Section 3.1, we introduce the vision system configuration and the concept of virtual camera and virtual image. The localization techniques is presented in Section 3.3 while the path reconstruction is presented in Section 3.4.

Chapter 4 describes our local planning strategy. Formalization of the local planning problem is given in Section 4.1. Section 4.2 describes the optimal alphabet for the local planning problem. The optimal maneuver is presented in Section 4.3.

Chapter 5 describes our global planning strategy. Formalization of the global planning problem is given in Section 5.1 . Section 5.3 describes the graph–based planning strategy. Simulations results are presented in Section 5.5.

Finally, Chapter 6 describes the architecture design of our testbed platform and Chapter 7 is the conclusion and future works.

Chapter 2

Car and Track Model

In this section, we first introduce a race car model and a geometric characterization of the track. The car model consist of the car kinematic model and the car dynamic model that considers several car dynamic effect such the tire slip, the steering servo motor and the aerodynamic effect. The track model reduces the track into a collection of sectors that can be of type straight sector or bend sector.

2.1 Car Model

The robotic car platform considered in this thesis is a four wheel drive (4WD) robotic cars with two cameras mounted on the front and on the side of the car. A model for both the kinematic and the dynamic of the car is detailed in the following sections.

Let $\langle W \rangle = \{O_w, X_w, Y_w, Z_w\}$ be a right–handed fixed reference frame (see Figure 2.1). The configuration of the vehicle is described by $q(t) = (x(t), y(t), \theta(t), v(t))$, where $p(t) = (x(t), y(t))$ is the position in $\langle W \rangle$ of the rear–wheel axis midpoint, $\theta(t)$ is the orientation of the vehicle with respect to the X_w axis, $\varphi(t)$ is the steering angle of the front wheel with respect to the vehicle and $v(t)$ be the velocity of rear–wheel axis midpoint (Fig. 2.1).

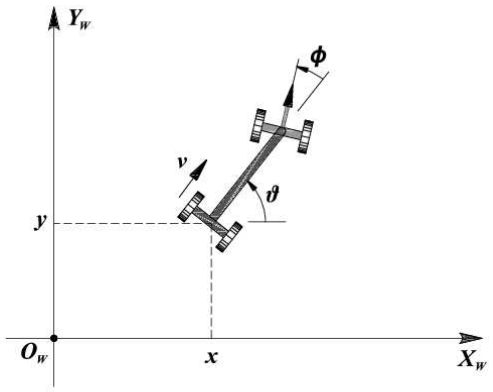


Figure 2.1: Mobile robot and system coordinates

2.1.1 Kinematic Model.

The kinematic model described in [26] uses a *forward driving* model (front traction). The main difference of our kinematic model is that we opted for a rear traction model for the vehicle. , i.e.

$$\begin{aligned} \dot{x} &= v_x \cos(\theta) \\ \dot{y} &= v_x \sin(\theta) \\ \dot{\theta} &= \frac{v_x \tan(\varphi)}{l} \end{aligned} \tag{2.1}$$

where l is the distance between the wheel axes. The steering angle φ and the traction acceleration a is chosen as control inputs.

The control inputs are constrained in the sets $\varphi \in [-\overline{\varphi}, \overline{\varphi}]$ and $a \in [\underline{a}, \overline{a}]$ (with $\underline{a} < 0 < \overline{a}$, i.e. maximum braking and acceleration applicable actions), respectively. Since in this thesis we are not interested in parking or docking maneuvers we assume the velocity $v \in [\underline{v}, \overline{v}]$ with $\underline{v} > 0$.

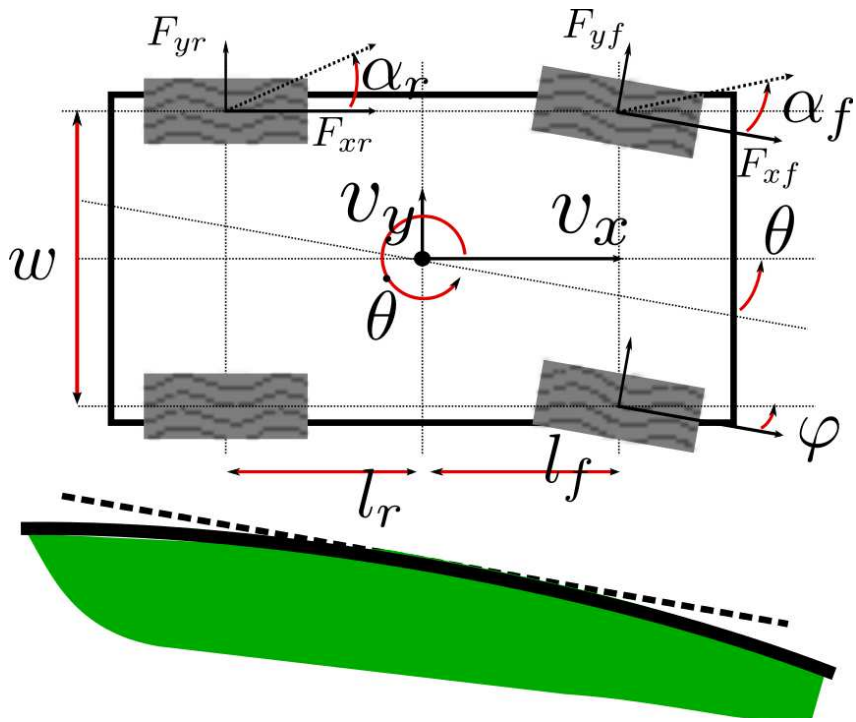


Figure 2.2: Vehicle dynamic model

2.1.2 Dynamic Model

The dynamic model is the *bicycle model* of an automobile [56], shown in figure 2.2. The model considers the effects of tires slip and steering servo motor, i.e.,

$$\begin{aligned}
\dot{v}_x &= \frac{F_{xr} + F_{xf}\cos(\varphi) - F_{yf}\sin(\varphi)}{m} + \dot{\theta}\, v_y - \frac{F_{aero}}{m}, \\
\dot{v}_y &= \frac{F_{yr} + F_{xf}\sin(\varphi) + F_{yf}\cos(\varphi)}{m} - \dot{\theta}\, v_x, \\
\ddot{\theta} &= \frac{l_f\,(F_{xf}\sin(\varphi) + F_{yf}\cos(\varphi)) - l_r\,F_{yr}}{I_z}, \\
|\varphi| &\leq \overline{\varphi},
\end{aligned}$$

where v_x and v_y are the longitudinal and lateral velocity at the Center of Mass (CoM) of the vehicle, φ is the steering angle, $\overline{\varphi}$ is the maximum steering angle and θ is the vehicle heading w.r.t. X_w axis. $l = l_f + l_r$ is the total wheel base, where l_f, l_r are the longitudinal distance from the CoM to the front and rear tires respectively (see Fig. 2.2). Each front and rear tire provides a lateral force F_{yf}, F_{yr} which is perpendicular to the rolling direction of the wheel

and a longitudinal force F_{xf}, F_{xr} which is parallel to the rolling direction (the *single track model* [46] is adopted here). The lateral forces are approximated as

$$F_{yf} \approx -c_y\,\alpha_f,$$
$$F_{yr} \approx -c_y\,\alpha_r,$$

where c_y is the tire cornering stiffness, α_f and α_r are the tire slip angles on the front and the rear wheels, respectively, and are given by

$$\alpha_f = \arctan\left(\frac{v_y + \dot{\theta}\,l_f}{v_x}\right) + \varphi,$$
$$\alpha_r = \arctan\left(\frac{v_y - \dot{\theta}\,l_r}{v_x}\right).$$

Similarly, the longitudinal forces are approximated as

$$\begin{aligned} F_{xf} &\approx c_x\,\sigma_f, \\ F_{xr} &\approx c_x\,\sigma_r, \end{aligned} \tag{2.2}$$

where c_x is the tires longitudinal stiffness. σ_f and σ_r are the tire slip ratios on the front and the rear wheels, respectively, and are given by

$$\sigma_i = \begin{cases} \dfrac{r_{eff}\,\omega_{w,i} - v_x}{v_x} & \text{when breaking,} \\ \dfrac{r_{eff}\,\omega_{w,i} - v_x}{r_{eff}\omega_{w,i}} & \text{when accelerating,} \end{cases}$$

where $i \in \{r, f\}$, r_{eff} is the effective radius of the wheel and $\omega_{w,i}$ is the wheel i angular velocity.

Notice that c_y and c_x can have different values if each wheel is driven independently. Moreover, for a 4WD vehicle (as the car considered in this thesis), the same ω_{w_i} is applied to each tire, hence we will use ω_w henceforth. Additionally, $\sigma_f = \sigma_r = \sigma$ for a 4WD.

Based on the dynamic model found in [46], we further modified the dynamic model by adding an aerodynamic force effect to the vehicle (F_{aero}) given by

$$F_{aero} = c'_a\,v_x^i$$

where c'_a is the aerodynamic drag coefficient and $i = 1$ for air laminar motion, while $i = 2$ for turbulent motion. For the sake of simplicity, this thesis considers the laminar motion. The

laminar regime is accurate enough for scale car models used in robotics laboratories. However, since the computation of the extremals is not affected by the regime of the fluid motion (see section 4.2), we believe that the adaptation of the approach to turbulent regime can be made with little effort. Thus, defining the state variable

$$z = \begin{bmatrix} x \\ y \\ \theta \\ v_x \\ v_y \\ \dot{\theta} \end{bmatrix}$$

the complete model is given by:

$$\dot{z} = \begin{bmatrix} z_4 \cos(z_3) - z_5 \sin(z_3) \\ z_4 \sin(z_3) + z_5 \cos(z_3) \\ z_6 \\ \frac{F_{xr} + F_{xf}\cos(\varphi) - F_{yf}\sin(\varphi)}{m} + \dot{z_3}\, z_5 - \frac{F_{aero}}{m} \\ \frac{F_{yr} + F_{xf}\sin(\varphi) + F_{yf}\cos(\varphi)}{m} - \dot{z_3}\, z_4 \\ \frac{l_f\,(F_{xf}\sin(\varphi) + F_{yf}\cos(\varphi)) - l_r\, F_{yr}}{I_z} \end{bmatrix}$$

where the inputs are the steering angle φ and the wheel angular velocity ω_w resulting from the application of the forces F_{xf} and F_{xr}, see (2.2). Notice that x, y and θ are expressed in the global reference frame, while v_x and v_y are expressed in the car moving frame attached to the CoM (see Figure 2.1). In particular, x and y are the coordinates of the midpoint of the rear axle (see Figure 2.3), m is the mass of the vehicle and I_z is the moment of inertia about the vertical axis. The dynamic model thus presented can be further broadened adding, for example, the wind effect or the rolling resistance of the tires. However, such extensions are postponed to future works.

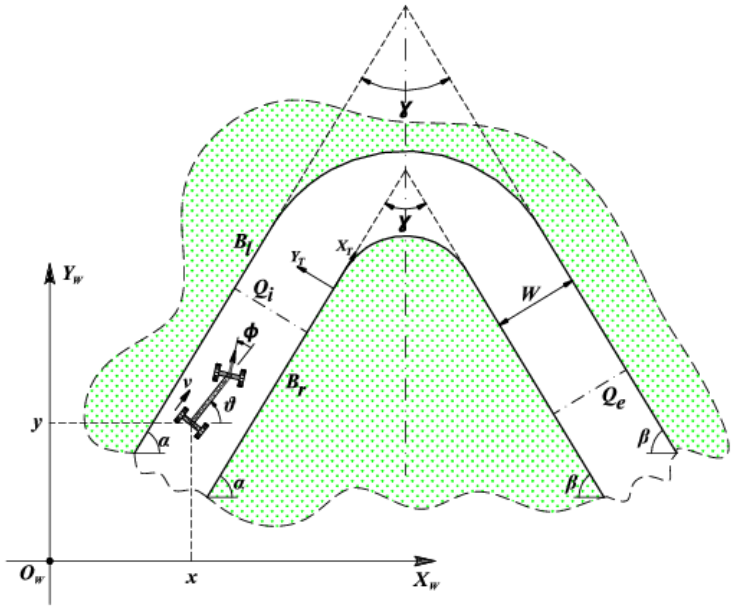


Figure 2.3: Mobile robot and system coordinates, with an example of a track.

2.2 Track Description

The track is defined as a sequence of straight sectors or of sectors comprising a bend on the right (or, equivalently, on the left), as shown in Figure 2.3. The track is characterized by two boundaries, left B_l and right B_r, which are also formed by a sequence of a straight line, arc of circle and another straight line. We make the simplifying assumption that the distance between B_l and B_r is constant along the track with width $W \geq 0$.

For the sectors containing a bend the two boundaries are parametrized as show in Figure 2.3. The angle γ is the characteristic curve angle, i.e. the angle between the two straight line sectors of each boundary. The bend boundaries are characterized by the centers C_b and the radii R_b. In particular, R_b^i refers to the inner boundary, while R_b^o to the outer. Trivially, $R_b^i < R_b^o$ for widths $W > 0$.

Without loss of generality we can assume the bend to be oriented such that the axis Y_w is parallel to the the bisector of γ. The angles α and $\pi - \beta$ are the orientations of the line sectors w.r.t. X_w.

2.2.1 Reference frames

Let $\langle T\rangle = \{O_T,\, X_T,\, Y_T,\, Z_T\}$ be the *track reference frame* with O_T a point on B_r, X_T tangent to B_r and Y_T pointing towards the left boundary B_l (see Fig. 2.3). This frame is a Frenet frame attached on the the right boundary of the track. Consider the *track initial reference frame* $\langle I\rangle$ as $\langle T\rangle$ with O_T placed on B_r at the beginning of the track and the *track final reference frame* $\langle E\rangle$ as $\langle T\rangle$ with O_T placed on B_r at the end of the track.

A point ${}^T p_r$ on X_T has a corresponding point on B_l given by ${}^T p_l = {}^T p_r + [0,\, W]^T$. In particular, any point inside the track at the same distance of ${}^T p_r$ from the curve can be expressed as ${}^T p = {}^T p_r + k[0,\, W]^T$, where $0 \leq k \leq 1$. We will denote by $\mathcal{P}$ the set of all points lying inside the track, for which there exist a Frenet frame in which their X coordinate is 0 and their Y coordinate is smaller than W: $\mathcal{P} = \big\{p | \exists \langle T\rangle \text{ s.t. } {}^T p = k[0,\, W]^T, 0 \leq k \leq 1\big\}$.

Chapter 3

Vision Based Localization and Path Reconstruction

Many autonomous robot application require the ability to navigate across an a environment which might be unknown or known, for example to follow a road (e.g., for automatic driving) [10] or to move across a factory or house floor [34]. The presence of markers in the environment that suggest a possible path or delimit the borders of the lane greatly simplify this navigation task. There are a selection of sensors which enables autonomous robots to detect these markers such as LIDARs (Light Detection and Ranging), sonar sensors or cameras.

Camera, mounted on an autonomous robot, is a versatile sensor which provides a rich information about the environment. This information can be use for robotic vehicle control, lane-departure warning, parking assistance, etc. Cameras have been widely deployed by manufacturers in their car with different technologies. Hence, computer vision becomes one of the critical technology for autonomous cars.

Moving the robot with a specified distance from road markers is known in the literature as *path following* [1, 54] and it requires the relative *localization* of the robot [60]: i.e., the reconstruction of its distance and its bearing with respect to the marker. An autonomous robot requires an ability to move (change its configurations) to reach some goals while bounded by some constraints. There can be one or more possible paths from one configuration to another.

The problem of choosing the path that optimizes some metric of interest is known as *path planning*. The computation of an adequate plan requires the knowledge of the shape of the road for some distance ahead. We refer to this problem by the term *path reconstruction*. In this thesis we advocate the use of camera system both for localization and path reconstruction.

From a computational perspective, the most efficient way to visually detect and track a path on a surface is to place the image plane parallel to the surface. This way, image distortions are completely removed by design and the image processing algorithms can be simplified. However, such solution might not be feasible due to the robot construction or could limit the image measurements to a small portion of the scene: the range of situations in which the position of the robot can be estimated is very small.

In Section 2.1 we have described the system overview of the robot, especially the position of the two cameras on the robot. There are two cameras mounted on our robot which serve different purposes. The lateral camera, mounted on the right side of the robot, is used for the localization of the robot. Conversely, the front camera which is mounted in front of the robot is used for the path reconstruction. Both cameras are mounted so that they are at an angle to the surface plane. To carry out the computation of the position and of the path as in the previous situation where the cameras are parallel to the ground, we use the notion of virtual cameras. The virtual cameras are placed such that their image planes are always parallel to the surface. In order to reconstruct the scene seen by the virtual camera, we use the *Inverse Perspective Mapping* (IPM) technique, which allows us to remove the distortions introduced by the perspective projection [39].

3.1 System configuration

Let $\langle R \rangle = \{O_r, X_r, Y_r, Z_r\}$[1] be the right-handed reference frame attached to the vehicle, whose plane $\Pi = X_r \times Y_r$ is the plane of motion and axis Z_r pointing upwards and the origin O_r is attached in the mid–point of the rear wheel axle. (Fig. 3.1). Let $\langle C \rangle = \{O_c, X_c, Y_c, Z_c\}$ be the actual camera reference frame, i.e., the frame attached to the vision system, which is

[1] O_r stands for "the origin of the reference frame".

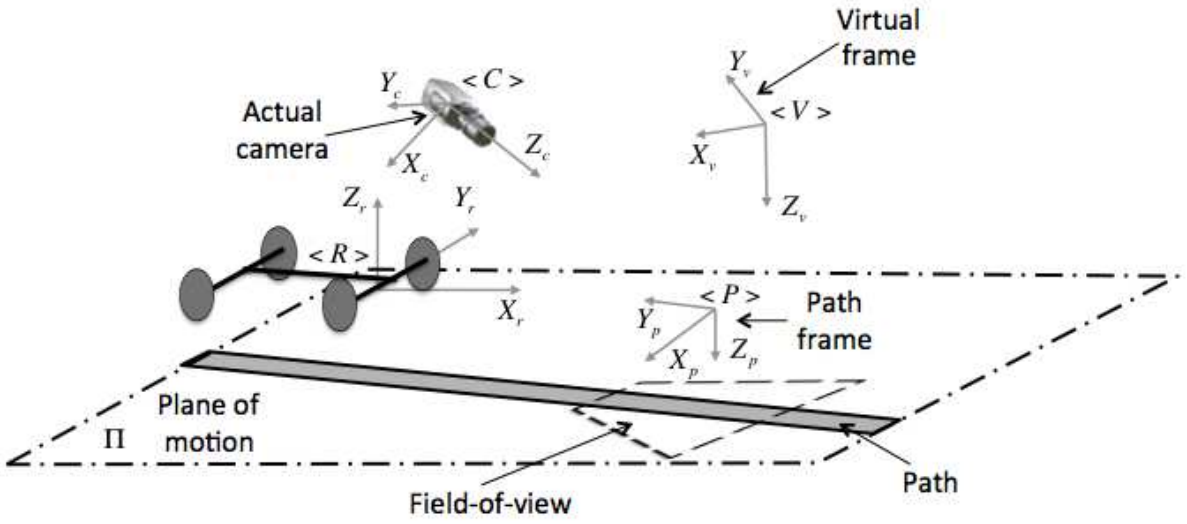


Figure 3.1: Geometric description of the problem and reference frames: robot frame $\langle R \rangle$, actual camera frame $\langle C \rangle$, virtual camera reference frame $\langle V \rangle$ and path reference frame $\langle P \rangle$.

mounted on the vehicle chassis. Notice that Z_c is chosen orthogonal to the image plane, and it intersects it in the *principal point*, i.e., the origin of the current bi-dimensional image reference frame $\langle I_c \rangle = \{O_{i_c}, X_{i_c}, Y_{i_c}\}$. The path to follow is defined by a stripe placed on the plane of motion, which is represented in the image by two line edges.

The relation between the reference $\langle R \rangle$ and camera $\langle C \rangle$ frame is given by a generic rigid motion transformation ${}^c g_r \in se(3)$, where $se(3)$ is the Special Euclidean Group. The ${}^c g_r$ is expressed by means of a rotation matrix ${}^c R_r \in SO(3)$, where $SO(3)$ is the Special Orthogonal Group, and of a translation vector ${}^c t_{c,r} \in \mathbb{R}^3$, where ${}^c t_{c,r}$ stands for "The vector starting from O_c and pointing to O_r expressed in $\langle C \rangle$". More precisely, the rotation matrix ${}^c R_r$ is given by the composition of three basic rotation matrices, each one expressing a rotation w.r.t. the fixed axis of $\langle R \rangle$. In particular,

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & \sin(\psi) \\ 0 & -\sin(\psi) & \cos(\psi) \end{bmatrix},$$

$$R_y(\gamma) = \begin{bmatrix} \cos(\gamma) & 0 & -\sin(\gamma) \\ 0 & 1 & 0 \\ \sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix},$$

$$R_z(\phi) = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where ψ is the *roll* angle, γ is the *pitch* angle and ϕ is the *yaw* angle, i.e., *Roll–Pitch–Yaw* (RPY) notation. With this choice, we can write

$$^{c}R_r = R_x(\psi_r)\, R_y(\gamma_r)\, R_z(\phi_r). \tag{3.1}$$

In particular, the configuration reported in Fig. 3.1, in which the camera points towards the plane of motion in front of the vehicle, is obtained by imposing $\psi_r \in [-\pi,\, -\pi/2]$, $\gamma_r = 0$ and ϕ_r generic.

For a generic point $^{r}p_j = [^{r}x_j,\, ^{r}y_j,\, ^{r}z_j]^T$ expressed in $\langle R \rangle$, we have

$$^{c}p_j = {}^{c}R_r\, {}^{r}p_j - {}^{c}R_r\, {}^{r}t_{r,c} = {}^{c}g_r(^{r}p_j),$$

where $^{r}t_{r,c}$ is the translation vector from O_r to O_c expressed in $\langle R \rangle$. The relation between $^{r}p_j$ and its image $^{i_c}p_j$ in the image plane is given by

$$\lambda\, {}^{i_c}p_j = {}^{c}p_j = {}^{c}g_r(^{r}p_j), \tag{3.2}$$

where $\lambda \in \mathbb{R}$ expresses the *line of sight* ambiguity, according to the *pinhole camera model* [25] (see Fig. 3.2). Notice that $^{i_c}p_j = [^{i_c}x_j,\, ^{i_c}y_j,\, f]$, where $(^{i_c}x_j,\, ^{i_c}y_j)$ is the pixel position in the image plane (hence expressed in $\langle I_c \rangle$), while f is the focal length of the camera (see Fig. 3.2). The position in the image plane is given by the *perspective projection*

$$\begin{cases} {}^{i_c}x_j = f\dfrac{^{c}x_j}{^{c}z_j} \\ {}^{i_c}y_j = f\dfrac{^{c}y_j}{^{c}z_j} \end{cases}, \tag{3.3}$$

where $\lambda = {}^{c}z_j / f$.

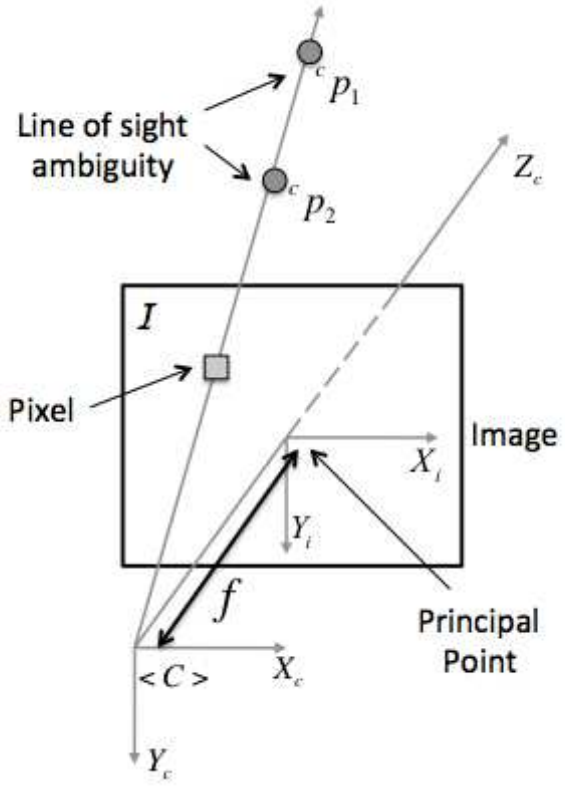


Figure 3.2: Pictorial description of the image formation using the standard pinhole camera model.

Since the motion of the vehicle is (approximately) planar, only three quantities are needed to fully describe its pose: its position in the plane (two coordinates) and its orientation with respect to a reference axis (one angle). The localization and path reconstruction can be exactly solved when the pose of the camera reference frame $\langle C \rangle$ is known w.r.t. the path frame (in Section 3.2 we will see how to release this assumption and deal with a partial knowledge of the camera pose). More precisely, let us define $\langle P \rangle = \{O_p, X_p, Y_p, Z_p\}$ as the *path reference frame*, i.e., the reference frame whose axis X_p is orthogonal to the path edge, axis Y_p is parallel to the line edges, while axis Z_p points towards the plane of motion (see Fig. 3.1). Notice that the plane the $X_p \times Y_p$ is parallel to Π and that frame $\langle P \rangle$. We restrict, for the moment, to a path consisting of a straight line. This assumption will be released in Section 3.4.1. A camera

placed in O_p would have a path image reference frame $\langle I_p \rangle = \{O_{i_p}, X_{i_p}, Y_{i_p}\}$ that would grab an image in which the path edges are parallel and symmetric to Y_{i_p} axis. With this choice of $\langle P \rangle$ and defining

$$
\begin{aligned}
{}^{c}R_p &= R_x(\psi_p)R_y(\gamma_p)R_z(\phi_p), \\
{}^{p}t_{p,c} &= [{}^{p}x_{p,c}, {}^{p}y_{p,c}, {}^{p}z_{p,c}]^T, \\
{}^{c}p_i &= {}^{c}R_p {}^{p}p_i - {}^{c}R_p {}^{p}t_{p,c} = {}^{c}g_p({}^{p}p_i),
\end{aligned}
$$

the position of O_c in the plane of motion is given by the first two entries of ${}^{p}t_{p,c}$ (the third entry ${}^{p}z_{p,c}$ playing no role for localization in the Π plane), and the orientation of $\langle C \rangle$ w.r.t. $\langle P \rangle$, expressed by the *yaw* angle ϕ_p along the Z_p axis. The position of the camera along the path, i.e., along the Y_p axis, is *unobservable* from the camera measurements. This is because the only measurable image features are the edges, which are determined using the image gradient. Such a gradient is orthogonal to Y_p; therefore only the distance along X_p is can be evaluated.

3.1.1 Virtual Camera and Virtual Image

The rigid transformation ${}^{c}g_p$ has to be estimated from the knowledge of the path pose in the image I_c. In what follows we assume that the path edges, even though affected by outliers and noises, are gathered in the grabbed image using standard image processing tools [11] and, hence, are assumed known. One way to estimate ${}^{c}g_p$ is to exploit the prior knowledge that the line edges are equidistant. However, since the rotation matrix ${}^{c}R_p$ is generic even with perfectly planar motion, parallel lines in the 3D space are not mapped into parallel lines in the image space for the presence of the perspective projection (3.3) (see [25] for reference). On the other hand, the image processing algorithm would be greatly simplified if parallelism were preserved in the images, paving the way for the application of computationally light and robust algorithms (e.g., [22]). The idea is then to synthesize an image from actual measures that retains all the quantities needed for ${}^{c}g_p$ estimation and, at the same time, preserves line parallelism.

To this end, we make use of a *virtual* reference frame $\langle V \rangle = \{O_v, X_v, Y_v, Z_v\}$, with plane $X_v \times Y_v$ parallel to the plane Π and with Z_v pointing towards the plane of motion (see Fig. 3.1 for reference). From a geometric viewpoint and assuming a prefect knowledge of ${}^{c}g_r$, i.e., no

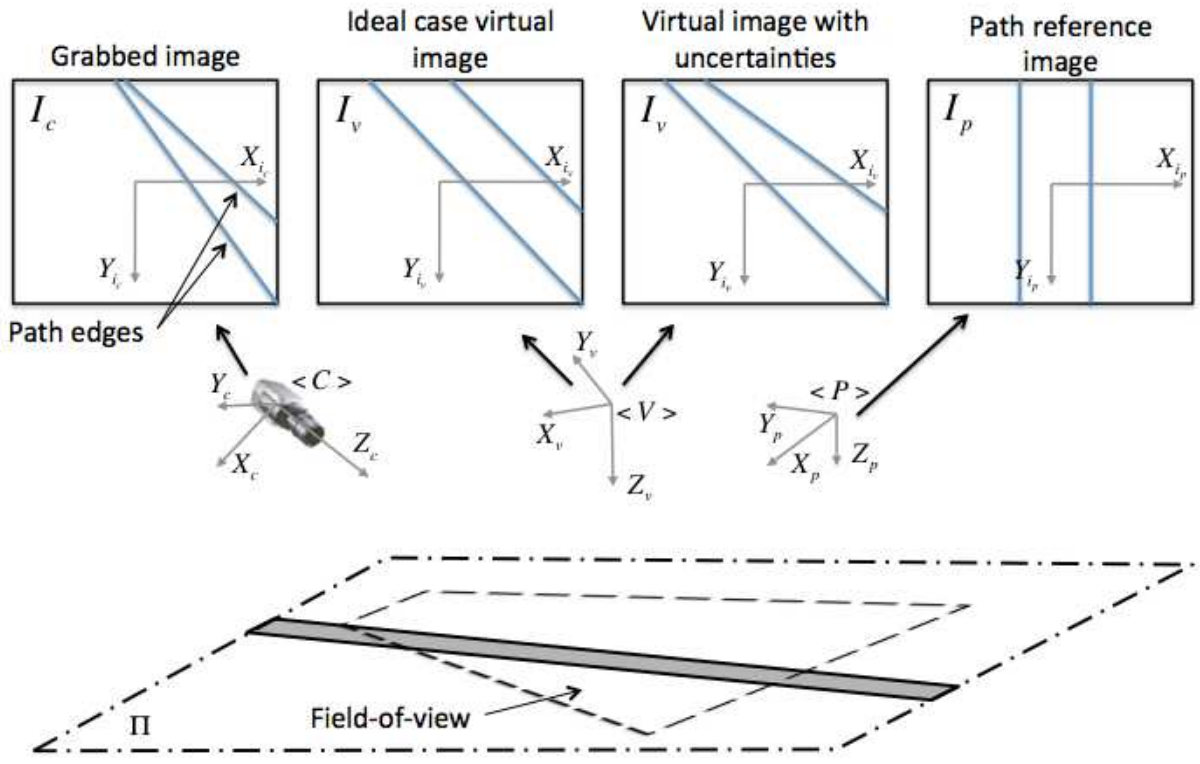


Figure 3.3: Images grabbed from each position: actual grabbed image I_c, virtual image I_v, both in the ideal case or in the presence of uncertainties, and path reference image I_p.

unknown roll and pitch angles, the rigid transformation ${}^{c}g_v \in se(3)$ that maps $\langle V \rangle$ to $\langle C \rangle$ is defined by means of the rotation matrix ${}^{c}R_v = R_x(\psi_v)\, R_y(\gamma_v)\, R_z(\phi_v)$ and the translation vector ${}^{v}t_{v,c} = [{}^{v}x_{v,c},\, {}^{v}y_{v,c},\, {}^{v}z_{v,c}]^T$. For example, we have $\psi_v = \pi + \psi_r$, $\gamma_v = 0$ and ϕ_v generic if the configuration of Fig. 3.1 with a front camera is considered. The *virtual image* grabbed from the virtual reference frame $\langle V \rangle$ generate the bird's–eye view, in which the path edges in the virtual image reference frame $\langle I_v \rangle = \{O_{i_v},\, X_{i_v},\, Y_{i_v}\}$ are parallel, as depicted in Fig. 3.3. Notice that ${}^{v}z_{v,c}$ is the height of the virtual camera w.r.t. Π, hence it will have a zooming effect for the virtual image.

3.2 Problem Formulation

In the geometric framework introduced in Section 3.1, vehicle–to–path localization amounts to the reconstruction of ${}^{r}g_p \in se(3)$, which is the rigid transformation between $\langle R \rangle$ and $\langle P \rangle$. Since the quantities defining ${}^{c}g_r \in se(3)$ are assumed known while ${}^{c}g_v \in se(3)$ is controllable, i.e., the virtual camera can be placed freely in the Π plan, vehicle–to–path localization is solved whenever $\langle V \rangle \equiv \langle P \rangle$. Indeed, in such a situation, we have

$$ {}^{r}g_p = {}^{r}g_c \cdot {}^{c}g_p = {}^{r}g_c \cdot {}^{c}g_v = {}^{c}g_r^{-1} \cdot {}^{c}g_v. $$

In other words, ${}^{c}g_v$ can be considered as an estimate of ${}^{c}g_p$, which is unknown. Recalling the discussion about the frame $\langle P \rangle$, the problem we want to address can be precisely defined as:

Problem 1 (Localization with known camera pose)**.** *Assuming* full *knowledge of* ${}^{c}g_r$ *and given the path edge measures in the image* $\langle I_c \rangle$ *taken from the actual camera position* $\langle C \rangle$*, determine the distance from the path* ${}^{p}x_{p,c}$ *and rotation matrix* ${}^{c}R_p$*.*

One easy way to find the transformation relating $\langle I_v \rangle$ and $\langle I_p \rangle$ could be found in the image space by applying, for example, image convolutions. Unfortunately, this is not applicable if one is interested in 3D localization, which calls for a different solution, as detailed in Section 3.3.

While the problems that we have introduced in Section 3.3 are on the reconstruction of the position of the robot with respect to a known path, another problem is on the reconstruction of the path in front of the robot (which is of the greatest importance for planning).

Problem 2 (Path reconstruction)**.** *Assuming* full *knowledge of* ${}^{c}g_r$ *and given the path edge measures in the image* $\langle I_c \rangle$ *taken from the actual camera position* $\langle C \rangle$*, reconstruct the path in front of the car (captured from the front camera).*

The solution to this problem will be offered in Section 3.4.

3.3 Localization

In this section, we show how to solve Problem 1 when the rigid transformation ${}^{c}g_r$ is known without uncertainties. As a consequence, ψ_p and γ_p are perfectly known, and the only unknowns

Algorithm 1 Algorithm for vision based localization

1: **while** true **do**
2: $I_c \leftarrow$ Capture()
3: $I_b \leftarrow$ Preprocess(I_c)
4: $I_r \leftarrow$ Rectify(I_b)
5: **if** first_image **then**
6: $K \leftarrow$ KalmanInitialize()
7: first_image $\leftarrow$ false
8: **else**
9: $K \leftarrow$ KalmanPredict(K);
10: ${}^{p}x_{p,c}, \phi_p \leftarrow$ RANSACwithPrediction(I_r, K)
11: $K \leftarrow$ KalmanUpdate($K, {}^{p}x_{p,c}, \phi_p$)
12: **end if**
13: **end while**

are ${}^{p}x_{p,c}$ and the yaw angle ϕ_p. The solution to Problem 1 is based on the application of the IPM, which allows us to synthesize the virtual image I_v, and on the subsequent application of an estimation algorithm to derive the quantities of interest.

An overview of the algorithm is shown in Algorithm 1. The preprocess step performs the required cropping, and edge detection [11] to convert the captured image into a binary image. The localization algorithm required three main components which are IPM/rectification (see Section 3.3.1), RANSAC (see Section 3.3.2) and Kalman filter (see Section 3.3.3).

3.3.1 Recover the Virtual Image from Measurements

In this section we describe how to "rectify" the image grabbed from position $\langle C \rangle$ through an IPM and then we set the theoretical basis for the estimation algorithm. The rectification algorithm is shown in the 4-th line of Algorithm 1.

Consider the *distance vector* from the path frame $\langle P \rangle$ to the motion plane defined by $H_p = h_p N_p$, where $N_p = [0,\ 0,\ 1]^T$ is the unit vector normal to Π expressed in $\langle P \rangle$ (see Fig. 3.1) and,

hence, h_p is the distance of O_p from the ground. The following holds true,

$$N_c = {}^cR_p N_p, \tag{3.4}$$

where N_c is the unit vector normal to Π expressed in $\langle C \rangle$. The distance h_p is the distance of O_c from the ground is given by: $h_p = {}^pz_{p,c} + h_c$.

Proposition 1. *The distance h_c between $\langle C \rangle$ and the plane Π is not affected by a translation on the plane Π nor by ${}^cR_p$.*

Even though Proposition 1 is straightforward from a geometric view–point, it will prove fundamental in the presence of uncertainties, since it reveals, for instance, that a reduction in the distance of the parallel lines is devoted only to translation along the Z_p axis, i.e., when the vehicle carrying the camera bounces on the road plane.

Consider the line of sight ambiguity defined by λ in (3.2). By construction, its value is constant for $\langle P \rangle$, while it is not constant in general for each point ${}^cp_j \in \pi$ in the frame $\langle C \rangle$.

Proposition 2. *The value of λ_{c_j} is not affected by ${}^px_{p,c}$ nor by the yaw angle ϕ_p, i.e.,*

$$\lambda_{c_j} = \frac{h_c}{N_p^T R_y(\gamma_p)^T R_x(\psi_p)^T {}^ic p_j}.$$

The previous results apply in the same way to the frame $\langle V \rangle$. We are now in condition to show that the unknown parameters ${}^px_{p,c}$ and ϕ_p are preserved in the rectified image and, hence, that Problem 1 can be solved in the rectified image space.

First, from (3.3) it is straightforward that if the image plane of a camera is parallel to the ground, parallelism between line on the ground and their image is preserved. Indeed, the distance from the camera pin–hole and the ground does not change. Therefore, this holds true for the image I_p. Let us consider the transformation defined from the virtual camera frame $\langle V \rangle$ to the actual camera frame $\langle C \rangle$. In the ideal case, the rotation matrix ${}^cR_v$ is partially known, since only the quantities $\psi_v = \psi_p$ and $\gamma_v = \gamma_p$ are known, while ϕ_v is the estimate of the yaw angle ϕ_p. Similarly, ${}^pt_{p,c} = [{}^px_{p,c} - {}^vx_{v,c},\, 0,\, 0]^T + {}^vt_{v,c}$, where ${}^vt_{v,c} = [{}^vx_{v,c},\, {}^vy_{v,c},\, {}^vz_{v,c}]^T$ is the current estimate of ${}^pt_{v,c}$, with ${}^vy_{v,c}$ is unobservable and useless for the problem at hand. Moreover, ${}^vz_{v,c} = {}^pz_{p,c}$ by construction.

Proposition 3. *The points ${}^{i_v}p_j$ of the virtual image I_v, obtained from the image points ${}^{i_c}p_j$ of the current grabbed image I_c by means of the IPM*

$$ {}^{i_v}p_j = \frac{f}{h_v}\left(\lambda_{c_j} R_y(\gamma_v)^T R_x(\psi_v)^T {}^{i_c}p_j + {}^{v}t_{v,c}\right), \tag{3.5} $$

preserve the 3D parallelism between the lines and restrain the unknown quantities ${}^{p}x_{p,c}$ and ϕ_p.

A byproduct result of Proposition 3 is the connection between points in I_v and the points of the reference image I_p, that is:

$$ {}^{i_v}p_j = R_z(\phi_p)\left({}^{i_p}p_j - \frac{{}^{p}t_{p,c}}{\lambda_v}\right) + \frac{{}^{v}t_{v,c}}{\lambda_v}, \tag{3.6} $$

It has to be noted that ${}^{i_p}p_j$ are the reference points of the parallel lines in I_p. Notice that, by definition, line edge points in the image I_p are parallel and symmetric to the Y_p axis of $\langle I_p \rangle$ (see Fig. 3.3). Moreover, they represent an invariant, i.e., from every possible configuration ${}^{r}g_p$, the measured image points ${}^{i_c}p_j$ map into the same image points ${}^{i_p}p_j$. Therefore, the reference image I_p can be determined once and for all during an initial calibration phase or easily estimated as reported in Section 3.4.2, since only the image distance among the lines is of interest. Therefore, it will be assumed known thereafter.

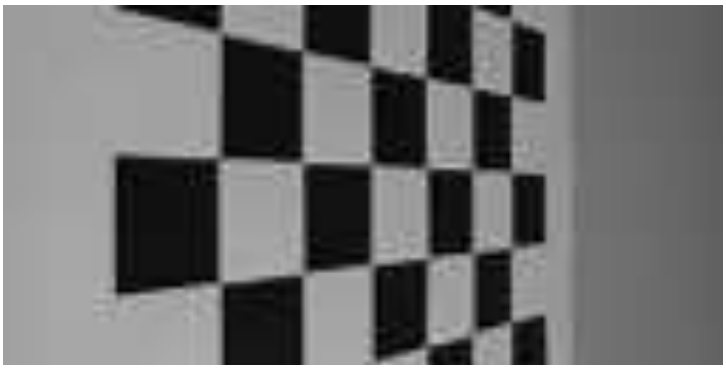
Example 3.1. *Consider an image of a checkerboard captured with the side camera of the car as shown in Figure 3.4 a. The image is preprocessed by converting it to a binary image. The rectified image (shown in Figure 3.4 b) is computed by placing a virtual camera such that the image plane of the virtual camera is parallel to the plane of motion* Π.

3.3.2 Estimating ${}^{c}g_p$

Using Proposition 3 and Equation (3.6) it is now possible to derive an estimation algorithm for the unknown rigid transformation ${}^{c}g_p$. A description of the estimation algorithm is offered next.

1. Using an image processing algorithm (e.g., the RANSAC based algorithm defined in [22]), detect the position and orientation of the parallel lines in the virtual image using the points ${}^{i_v}p_j$ obtained using (3.5);

a)



b)

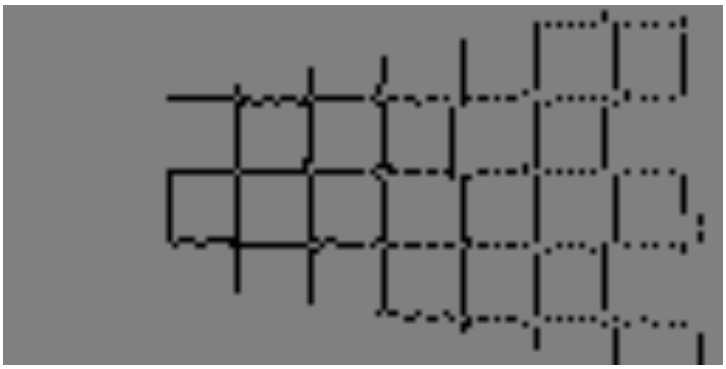


Figure 3.4: Example of rectification: a) original image, b) image after preprocess and rectification

2. Compute the angle ϕ_v between the parallel lines defined by ${}^{i_v}p_j$ and the Y_v axis of $\langle I_v \rangle$. In fact, this angle corresponds to the angle between line edge points ${}^{i_v}p_j$ and ${}^{i_p}p_j$. Since (3.6) holds, $\phi_v = \phi_p$;

3. Using ϕ_p, the value of ${}^{p}x_{p,c}$ in ${}^{p}t_{p,c}$ is readily available by inverting (3.6), i.e.,

$$ {}^{p}t_{p,c} = -\lambda_v R_z(\phi_p)^T \left({}^{i_v}p_j - \frac{{}^{v}\hat{t}_{v,c}}{\lambda_v} \right) + \lambda_v {}^{i_p}p_j. \tag{3.7} $$

Equation (3.7) in Step 3 of the procedure above can be used only if the point correspondence between ${}^{i_v}p_j$ and ${}^{i_p}p_j$ is available. The correspondence can be enforced using intersections with X_v axis in $\langle I_v \rangle$. In what follows, all the quantities refer to the left line, since the analysis for the right line can be derived similarly. Consider a point intersecting the X_{i_p} axis in $\langle I_p \rangle$, i.e.,

$$ {}^{i_p}p_j - \frac{{}^{p}t_{p,c}}{\lambda_v} = \begin{bmatrix} \bar{x}_l \\ 0 \\ 0 \end{bmatrix}, \tag{3.8} $$

where $\bar{x}_l$ is the intersection point. It has to be noted that this choice corresponds in selecting a point in the left line that has a coordinate along the Y_{i_p} axis ${}^{i_p}y_j = \frac{{}^{p}y_j}{\lambda_v}$, which has no effect on the estimation algorithm since all the points of the left line has the same value of ${}^{i_p}x_j$. From Equation (3.6), the intersection of the left line of the path with X_{i_v} can be expressed with

$$ R_z(\phi_p)^T \left({}^{i_v}p_j - \frac{{}^{v}t_{v,c}}{\lambda_v} \right) = \begin{bmatrix} \bar{x}_l \\ 0 \\ 0 \end{bmatrix}. $$

Assuming that a description of the left line is given, for example, in terms of its slope a and the offset b, i.e., ${}^{i_v}y_j = a\,{}^{i_v}x_j + b$, one gets

$$ \begin{aligned} \begin{bmatrix} {}^{i_v}x_j \\ a\,{}^{i_v}x_j + b \\ f \end{bmatrix} = {}^{i_v}p_j &= R_z(\phi_p) \begin{bmatrix} \bar{x}_l \\ 0 \\ 0 \end{bmatrix} + \frac{{}^{v}t_{v,c}}{\lambda_v} \\ &= \begin{bmatrix} \cos(\phi_p)\bar{x}_l \\ -\sin(\phi_p)\bar{x}_l \\ 0 \end{bmatrix} + \frac{1}{\lambda_v} \begin{bmatrix} {}^{v}x_{v,c} \\ {}^{v}y_{v,c} \\ {}^{v}z_{v,c} \end{bmatrix}, \end{aligned} $$

in the unknown $\bar{x}_l$. This condition leads to

$$\bar{x}_l = \frac{{}^{v}y_{v,c} - a^{v}x_{v,c} - \lambda_v b}{\lambda_v(a\cos(\phi_p) + \sin(\phi_p))}.$$

It has to be noted that the rectified images may have vertical lines. If this is the case, the angle $\phi_p \approx 0$, which finally yields to

$$\bar{x}_l = {}^{i_v}x_j - \frac{{}^{v}x_{v,c}}{\lambda_v}.$$

In any case, we can make use of Equation (3.8) for the first component

$${}^{p}x_{p,c} = \lambda_v({}^{i_p}x_l - \bar{x}_l),$$

where ${}^{i_p}x_l$ is the intersection of the left line with the X_{i_p} axis for the reference path image I_p, which is known. Repeating the same computations for the right line

$${}^{p}x_{p,c} = \lambda_v({}^{i_p}x_r - \bar{x}_r),$$

that, to increase the accuracy, finally yields to

$${}^{p}x_{p,c} = \lambda_v \frac{{}^{i_p}x_l + {}^{i_p}x_r}{2} - \lambda_v \bar{x},$$

where $\bar{x} = \frac{\bar{x}_l + \bar{x}_r}{2}$. Since the path is symmetric with respect to the Y_{i_p} axis in I_p, we finally get to

$${}^{p}x_{p,c} = \lambda_v \bar{x}.$$

Example 3.2. *Figure 3.5 shows two different set of images with different noise level. The first set (Figure 3.5 a,c,e) represents the case where no noise appear after the preprocesing phase while in the second set, structural and random noise still appears. In both cases, the RANSAC algorithm is able to correctly identify the two parallel lines (Figure 3.5 e & f)*

3.3.3 Prediction with Kalman

Extended kalman filter is used in order to improve the performance of the localization algorithm. The values of ${}^{p}x_{p,c}$ and ϕ_p associated with the current image is used as prior knowledge for the

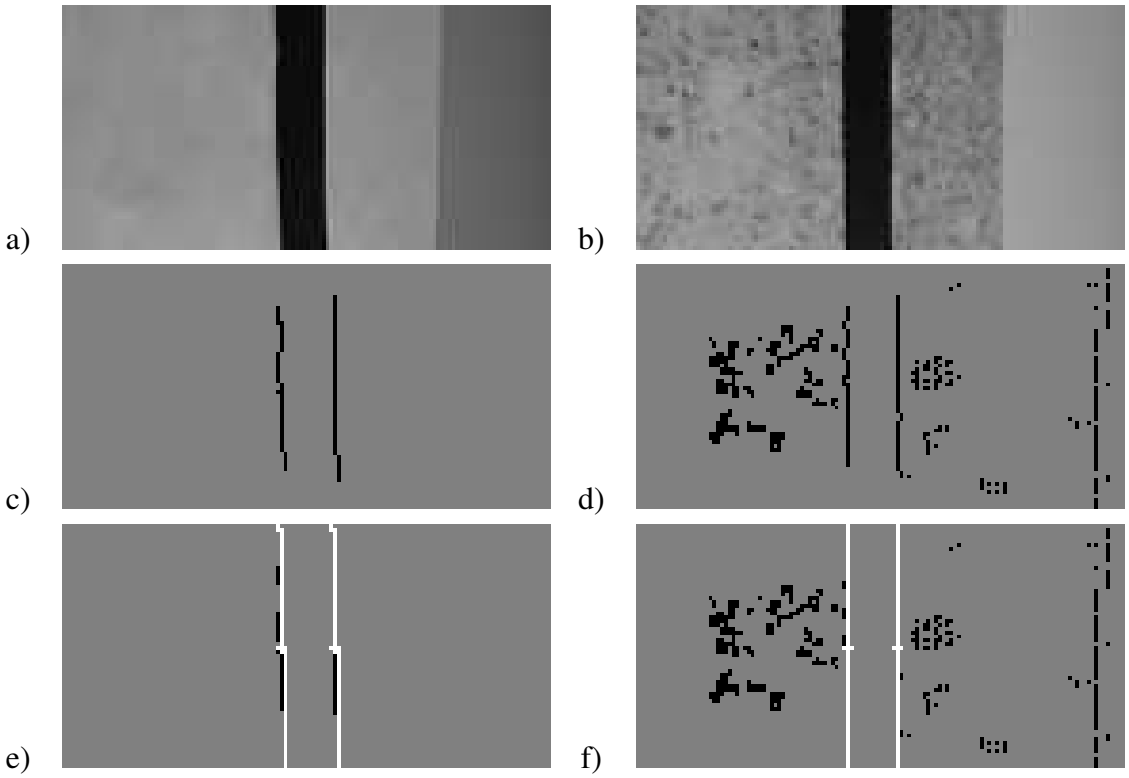


Figure 3.5: Example of rectification: a & b) original image, c & d) image after preprocess and rectification, e & f) parallel lines found by ransac

road localization in the following image. This way, the computation times will be reduced by constraining the line search in a subset of the image. The dynamics of these three parameters is a function of both the line to be tracked and the motion of the camera. By denoting with $(h_p,\ \alpha_p,\ d_p)$ the path quantities in the 3D coordinates and recalling that $h_p = {}^p x_{p,c}$ and $\alpha_p = \phi_p$ by construction, one gets

$$\dot{q} = \begin{bmatrix} \dot{h}_p \\ \dot{\alpha}_p \\ \dot{d}_p \end{bmatrix} = \begin{bmatrix} v \sin \alpha_p \\ \frac{v}{l} \tan \phi_r + \frac{v}{r} \cos \alpha_p \\ 0 \end{bmatrix} = f(q, v, r), \tag{3.9}$$

where r is the curve radius, which is $+\infty$ for straight paths. The meaning of $\dot{d}_p = 0$ is that the path width is constant. It is now evident that by means of (3.9) and (3.3), the time evolution of the quantities $(h,\ \alpha,\ d)$ in the image plane can be derived.

For robust estimation of path lines in the images, an extended Kalman filter (EKF) is applied to (3.9), in order to return refined values of h, α and d to be used as prior to the RANSAC–based algorithm presented above. With this choice, computational burden is dramatically reduced. Indeed, the unknown probability density functions related to the estimation processes of h and α can be considered multimodal for the presence of both outliers and noise in every grabbed image, which drains a relatively high computing power to get a correct estimate from an embedded system. However, the camera can not move instantaneously in multiple different directions, hence the uncertainty distribution associated with the motion of the camera is definitely unimodal and model–based.

At the beginning, the EKF is initialized as soon as a road line is clearly detected in the image plane. In this preliminary phase, a longer execution time is tolerated in order to have an accurate first guess. Then, the estimation algorithm performs iteratively the following three steps: a) the Kalman filter predicts the future position of the parallel lines using (3.9); b) the position and orientation of the parallel lines (i.e., the values of h, α and d) are computed by RANSAC using the position prediction given by the Kalman filter; c) the estimation of the state q updated using the measures coming from the RANSAC–based algorithm.

The duration of each iteration must be shorter than the frame period of the camera. If an image is not sufficiently informative due to structured or unstructured outliers (e.g., illumina-

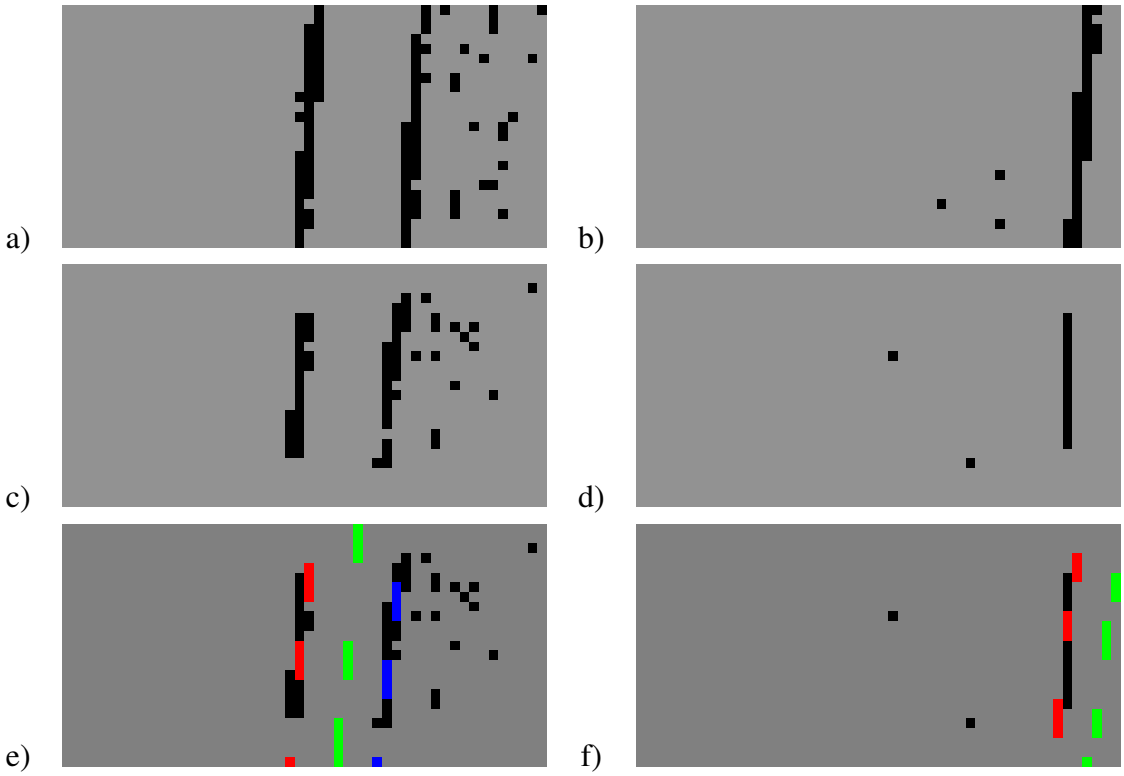


Figure 3.6: Example of kalman filter: a & b) original image, c & d) image after preprocess and rectification, e & f) parallel lines found by RANSAC and Kalman filter

tion problems, shadows, small potholes or faded paint), RANSAC does not return a valid line measurement. However, this kind of situations can be tolerated to a certain extent using the information retained by the Kalman filter. In such a case, the filter works in open loop, which leads to an increasing uncertainty of the estimates. If, after the EKF update, the covariance exceeds a certain threshold (whose maximum value is bounded by the image size), the EKF is reinitialized and the procedure starts over.

Example 3.3. *Figure 3.6 shows an example of the improvement provided by kalman filter. In the case where one or more of the line goes out of the image frame. the internal model is still able to track the lines. The red lines show the detected left line, the blue lines show the detected left line. Figure 3.6 f shows an example where the algorithm can still correctly tracked the center line even when the right line goes out of the frame.*

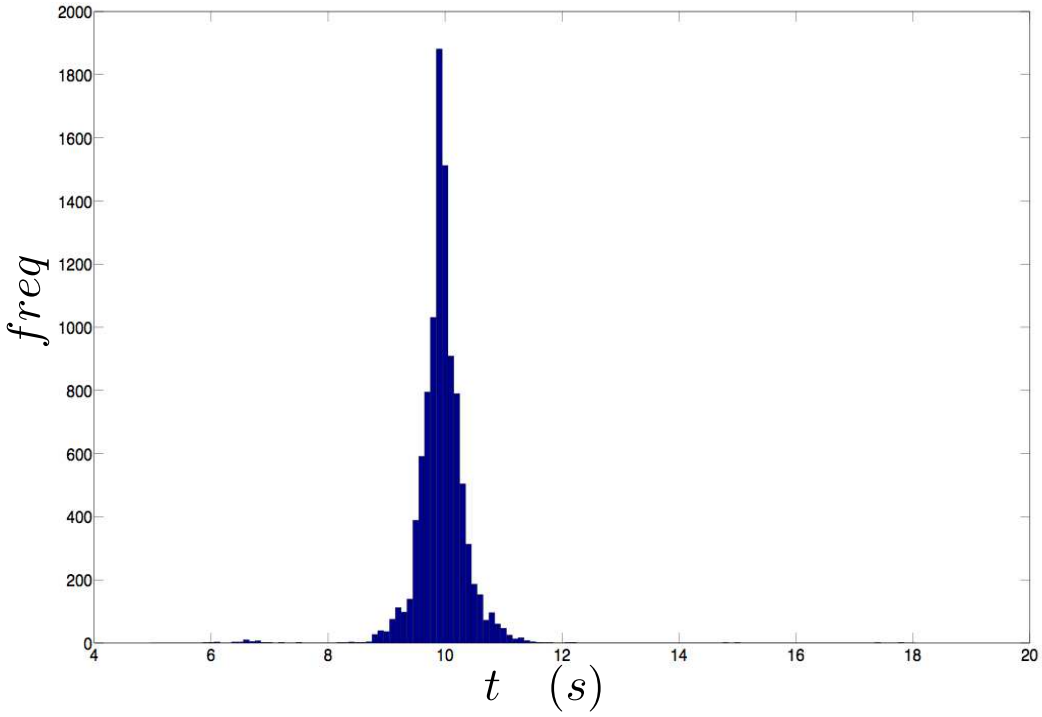


Figure 3.7: Time histogram of the localization algorithm

Example 3.4. *Figure 3.7 shows a histogram of the time to perform the localization algorithm for 5000 images. In this experiment more that 95% of the image is processed below 10 ms. Note that the RANSAC is an anytime algorithm. Thus, the algorithm can be stop at anytime to produce a guess of the two parallel lines. In this experiment there is no timeout set for the RANSAC algorithm. However, in the real-time implementation the localization process is forced to produce a result every 10 ms.*

3.4 Path Reconstruction

This section presents a solution to Problem 2: estimating the path in robot frame $\langle R \rangle$. The intuition underlying our approach is very simple: we let the virtual camera fly over the path and recover its position. In each position we estimate the rigid transformations ${}^{c}g_{v}$ and ${}^{c}g_{r}$ by means of the robust algorithm presented in the previous section. This way, we come up

Algorithm 2 Algorithm for vision based path reconstruction

```
1: I_c ← Capture()
2: I_b ← Preprocess(I_c)
3: I_r ← Rectify(I_b)
4: V ← ComputeCoveredRegion(I_r, nHeight)
5: for i = 0..nHeight do
6:    while valid do
7:       valid,model ← FindPath(V(h_v^i));
8:       if valid then
9:          ^p x_{p,c}, φ_p ← EstimatePositionOrientation(model)
10:         save current ^p x_{p,c}, φ_p in hypothesis[i]
11:         move virtual camera
12:      else
13:         break
14:      end if
15:   end while
16: end for
    path ← FindBestHypothesis(hypotesis)
```

with the reconstructed path. A crucial enabler for this technique is the ability to localize the virtual camera with respect to generic paths, compounded by straight lines and curves (in the previous sections we have restricted to straight lines). The unavoidable problem of parallax is also discussed in this section.

3.4.1 Paths with Curves

So far we have treated the problem of robust estimation for the rigid transformation ${}^{c}g_{p}$ in the presence of straight paths. However, a generic path for wheeled vehicles is in general make up of a set of straight lines and curves. In such a situation, it is necessary to estimate the bending path in the image space I_v and, then, estimating its curvature. Although solutions have

been presented in literature to solve this problem, the processing time may be prohibitive for an embedded implementation. Indeed, it is now becoming a commonplace to utilize probabilistic or statistical tools in the edge map of the image, for example in combination with the Hough transform [15, 37]. More efficient in this respect is the application of RANSAC for curves, as presented among the others in [7, 14]. However, all the proposed methods asks for a parametrization of the curve in the image space, with well defined maximum curvature. Our solution instead does not rely on a particular curve model but only assumes the knowledge of the maximum path curvature. It is worthwhile to point out that such an assumption is obviously common for vehicle moving on road, due to the limited curvature radius of cars, as well as by path in factory floors, for which too sharp bends may require moving platform velocity adjustments with unavoidable loss of mechanical energy.

Therefore, by using the inlier threshold of built in the RANSAC algorithm, i.e., the threshold that discriminates between outliers, noise and inliers, we simply choose to approximate the curve with a straight line. In fact, by acting on the ${}^{v}z_{v,c}$ coordinates of the virtual camera, a digital zoom is obtained: the more the image is zoomed, the better is the approximation.

3.4.2 Reconstruction

We are now in a position to present the generic path reconstruction algorithm, which can be used for a variety of purposes, e.g., mapping, efficient vehicle path planning, predictive control, etc., which builds upon the robust solution presented in Section 3.3. The path is reconstructed in the robot reference frame $\langle R \rangle$ and it is parametrized by a set of points in $\mathbb{R}^2 \times S$, i.e., by triplets $p_j = [x_{p_j}, y_{p_j}, \theta_{p_j}]^T$ where (x_{p_j}, y_{p_j}) are the Cartesian coordinates in the Π plane, while θ_{p_j} is the orientation of the path w.r.t. Y_r axis in position (x_{p_j}, y_{p_j}). Strictly speaking, the path is a sequence of straight line segments with different orientation describing the path. The reconstruction algorithm (shown in Algorithm 2) comprises of 4 main steps: computing covered region(Section 3.4.2.1), finding the path in current virtual camera (Section 3.4.2.2), moving the virtual camera position along the path (Section 3.4.2.3) and finally reconstructing the path from the best hypothesis(Section 3.4.2.4)

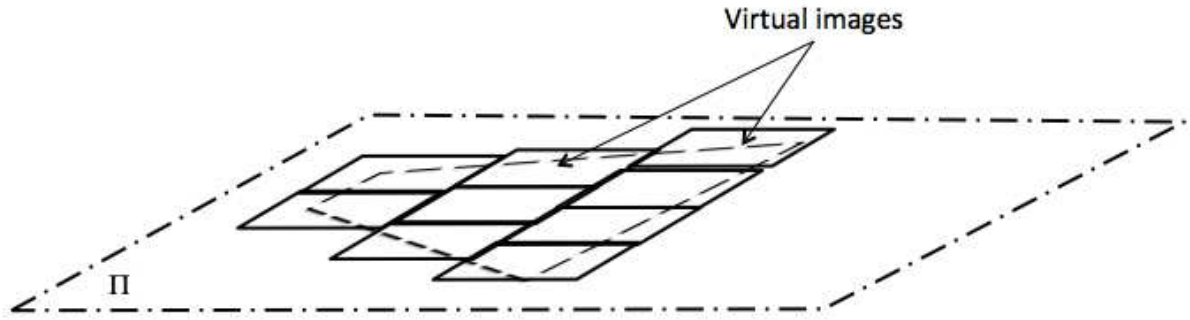


Figure 3.8: Field-of-view of the real camera reprojected onto the Π plane and set of virtual images taken from various positions, but fixed height, to cover the entire scene in view.

3.4.2.1 Virtual camera covered region

The first issue to address is related to the feasible transformation ${}^{r}g_{v}$ between $\langle R \rangle$ and $\langle V \rangle$ in order to let the virtual image cover the image I_c (see Fig. 3.8). To this end, we first compute the projection of the field–of–view (FOV) of the actual camera onto the plane of motion Π by computing the intersection between the line starting from the pin–hole and passing through each of the image corners with the plane Π. Hence, by defining the maximum ${}^{v}z_{v,c}^{\max}$ and minimum ${}^{v}z_{v,c}^{\min}$ virtual camera heights, it is possible to define all the virtual camera position sets $\mathcal{V}(h_v^j)$ as a function of the current j–th virtual camera height h_v^j, in order to cover the entire reprojected FOV of the actual camera for all the different heights. Although this operation it is not strictly needed, it speed up the path starting point detection.

3.4.2.2 Path starting point

Path reconstruction algorithm starts with the detection of the line edges in the virtual image edge map. To this end, the grabbed image I_c is scanned, starting from the lowest camera height h_v^0, and letting the virtual camera move along the position set $\mathcal{V}(h_v^0)$. The search for straight path edges is based on the Algorithm 1 presented previously. Once a valid solution has been found, i.e., an estimate of ${}^{c}g_{p}^{0}$ is derived, we get ${}^{r}g_{p}^{0} = {}^{r}g_{c} \cdot {}^{c}g_{p}^{0}$ and, finally, the first path parameter

$p_0 = [x_{p_0},\, y_{p_0},\, \theta_{p_0}]^T$ is extracted from the translation vector and the yaw angle collected in ${}^r g_p^0$. It has to be noted that the position of the reference path frame $\langle P \rangle$ is referred to the current virtual camera pose.

If two virtual images shares the same path edges, i.e., they produce the same virtual camera position after correction, the two hypotheses are fused together and then passed to the next step. If n distinct line edge pairs are found, i.e., if more than one starting path point $p_{0,1},\, p_{0,2},\, \ldots,\, p_{0,n}$, all the solutions are considered as valid hypotheses and propagated to the next step. If no path edges are found, the grabbed image scan is performed with an increased value of the height. If at the maximum feasible height no valid path data can be retrieved, the process stops and goes to Section 3.4.2.4 of this procedure.

3.4.2.3 Moving the bird's–eye view

As far as the path position p_j is available, the virtual camera moves along the path considering fixed orientation and fixed forward velocity, generating a predicted virtual camera position hypothesis, i.e.,

$$\hat{p}_{j+1} = \begin{bmatrix} x_{p_j} + \cos(\theta_{p_j})\Delta_s \\ y_{p_j} + \sin(\theta_{p_j})\Delta_s \\ \theta_{p_j} \end{bmatrix},$$

where Δ_s is the elementary translation vector, a design parameter dependent from the actual camera configuration and resolution. From $\hat{p}_{j+1}$, the estimate of ${}^c g_p^{j+1}$ is then derived if line edges can be found in the image, otherwise the process terminates and goes to Section 3.4.2.4 of this procedure. Therefore the pose of $\hat{p}_{j+1}$ is updated by

$$\hat{p}_{j+1} = \begin{bmatrix} \hat{x}_{p_{j+1}} - {}^p x_{p,c}^{j+1} \\ \hat{y}_{p_{j+1}} \\ \hat{\theta}_{p_{j+1}} - \phi_p^{j+1} \end{bmatrix}.$$

As previously recalled, the estimates of ${}^p x_{p,c}^{j+1}$ can be very noisy due to the presence of vibrations or plane of motion imperfections. In this case, the process is iterated by computing a new estimate of ${}^c g_p^{j+1}$ in the new corrected estimated position $\hat{p}_{j+1}$. The iteration stops whenever the

values of ${}^{c}g_{p}^{j+1}$ are below a predefined target uncertainty. Hence this step is started over from position $p_{j+1}x$.

This step is iterated for each hypothesis found in Section 3.4.2.2.

3.4.2.4 Reconstruction and the parallax problem

If n distinct path hypotheses have been found, the outputs of Step 3 are n sequences of triplets. We assume that only the valid path hypothesis is presented in all the virtual image, that is the valid path is the one with the longest set of triplets. Therefore, the longest sequence of triplets represent the best estimate of the path.

Unfortunately, even though the algorithm shows good performance using the idea of the bird's–eye view and it offers also a set of design parameters, e.g., the height of the virtual camera h_v^j or the translation step Δ_s, that can be adaptively tuned to trade–off between accuracy and processing time, the algorithm suffers of the problem of parallax. Indeed, the more the virtual camera is further away from the actual camera position, the less is its path reconstruction accuracy. The problem thus described comes directly from the perspective projection (3.3) and the limited resolution of the imager, which is quantized by the pixel. More precisely, let $\delta_x\delta_y$ be the surface of the pixel: the 3D portion of the scene that maps on the same pixel surface is given by

$$\Delta_{cx} = \frac{\delta_x {}^{c}z}{f} \text{ and } \Delta_{cy} = \frac{\delta_y {}^{c}z}{f},$$

i.e., the further is the portion of the scene in view (that is with increasing value of ${}^{c}z$, see also Fig. 3.2), the larger is the portion of the scene that maps onto the same pixel. Since for the virtual camera image I_v the distance from Π is constant by definition, the pixel coming from I_c becomes more and more scattered as the virtual camera moves away from the current camera position.

Although this problem is unavoidable and mainly limits the horizon of the vehicle maneuver prediction that can be drawn, it does not impair the effectiveness of the localization algorithm in the presence of generic paths.

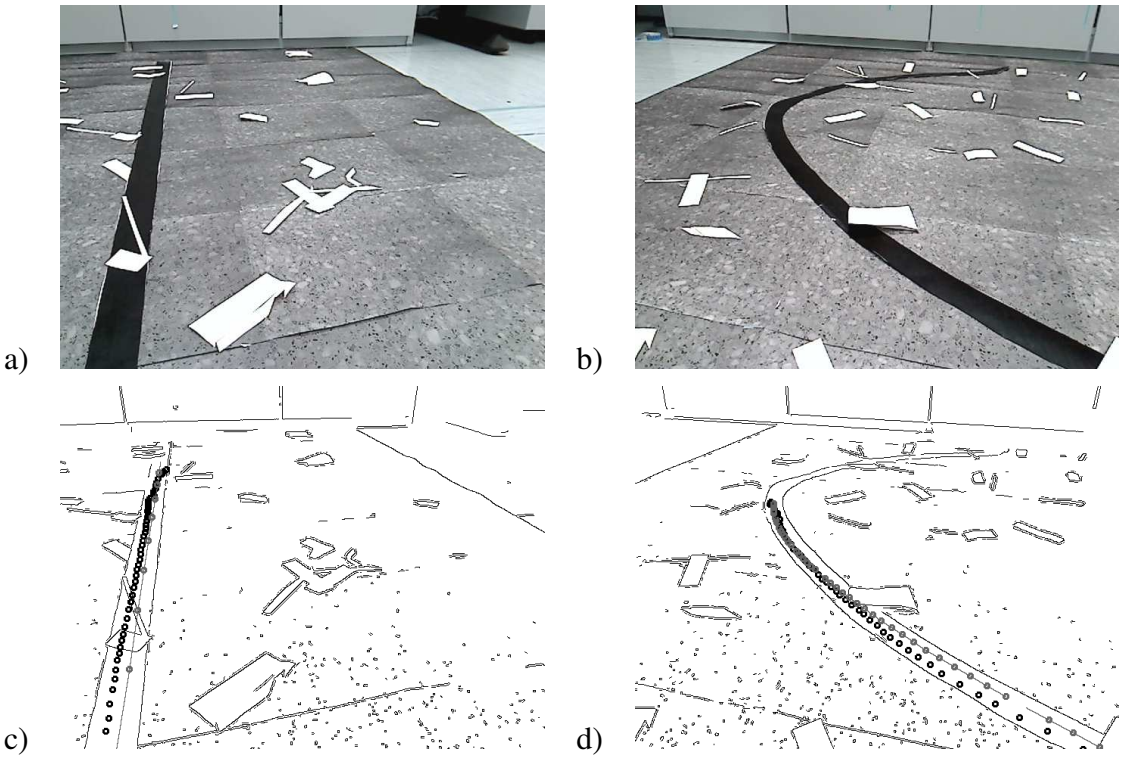


Figure 3.9: Example of reconstruction: a) & b) original image, c) & d) reconstructed points

Example 3.5. *Consider two example of images captured by the front camera (shown in Figure 3.9 a)& b). The result of the path reconstruction is show in Figure 3.9 c) & d).*

Chapter 4

Local Planning

In this chapter we focus on the local planning problem [51]. The problem addressed can be summarized in the following terms: *find the set of maneuvers that steer the car in minimum time between two configurations, each one identified by position on the track and velocity, respecting the dynamic constraints of the car and the geometric constraint of the track.*

The local planning problem is a sub problem of trajectory tracking problem. A local planning algorithm synthesizes a plan that is the optimal trajectory for a given car and track to reach the desired goal. Control algorithms enable the car to execute the synthesized plan. There are a lot of techniques and algorithms to design such controllers however it is out of the scope of this thesis.

The car model adopted in this thesis is inspired to the one proposed for Stanford's Stanley autonomous car [26] which offers a sufficient coverage of the most important physical phenomena that are usually considered. It comprises two different components: kinematic and dynamic model as detailed in Section 2.1. The authors approach the trajectory tracking problem by decomposing it into two sub-problems: a steering controller, acting on the steering wheels angle, for path following; a cruise controller, acting on the engine power, to track desired velocity profiles. While the cruise controller has been synthesized in a standard way adopting PID control, the steering angle is determined on the kinematic model, afterwards it is modified to take into account dynamic effects described by the dynamic model. We follow the same rationale, although the steering controller is slightly modified to derive a kinematic model that can be

effectively used to synthesize the optimal trajectories. Hence, the motion planning problem can be approached restricting to the kinematics of the vehicle as far as a controller is applied to the dynamics. Under the constraint that the car does not slip away (i.e. the lateral acceleration remains bounded below a given bound), we show that the optimal plan is composed of a concatenation of elementary maneuvers.

We offer a precise characterization of the alphabet of elementary maneuvers and we provide a preliminary characterization of the optimal sequences of maneuvers on the track sectors. Due to the complexity of the problem, we consider a simplified sequence of maneuvers that is easier to implement on a low cost vehicle and that remains reasonably close to the optimal solution. The simplified maneuvers allow us to use geometrical considerations to characterized the sub–optimal sequences of maneuvers on both straight and bend sectors.

In the next section the formalization of the local planning problem is described in Section 4.1. Section 4.2 details the optimal alphabet that will be used to construct the set of optimal maneuvers that is explained in Section 4.3.

The literature of optimal (shortest) paths stems mainly from the seminal works on unicycle vehicles with a bounded turning radius by Dubins [21] and on the car moving both forward and backward by Reeds and Shepp [47]. Other optimization cost functions has been considered such as the minimum wheel rotation paths for differential-drive robots ([16]), minimum time trajectory ([59] for differential drive robots, [4] for omnidirectional vehicles, [18] for a mobile robot with a trailer subject to limited control inputs, and [48] for robots with two independently driven wheels), minimum path length ([52] and [53] for differential drive robot with limited Field-of-View and [2] for a car–like robot).

4.1 Problem Formulation

The kinematic and dynamic model of the car have been presented in Section 2.1. Assuming that the steering and thrust controller proposed by Hoffman et al. [26] are used, the extended

kinematic model adopted for optimal path synthesis is given by

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_x \end{bmatrix} = \begin{bmatrix} v_x \cos(\theta) \\ v_x \sin(\theta) \\ v_x \frac{\tan(\varphi)}{l} \\ a - c_a v_x \end{bmatrix} \tag{4.1}$$

where φ is the steering angle, one input of the model, constrained by $|\varphi| \leq \overline{\varphi}$, $c_a = \frac{c'_a}{m}$ and a is the acceleration input, given by

$$a = \frac{F_{xr} + F_{xf} \cos(\varphi) - F_{yf} \sin(\varphi)}{m},$$

that, due to the limited steering angle, simplifies to

$$a \approx \frac{F_{xr} + F_{xf}}{m},$$

while the term $\dot{\theta} v_y$ is neglected since it is supposed to be compensated by the traction controller.

Definition 4.1. A *possible* maneuver is a maneuver that moves the vehicle from configuration q_I to q_E where $q_I, q_E \in \mathcal{Q}$. A *feasible* maneuver is a possible maneuver where all intermediate configuration $q \in \mathcal{Q}$. An *optimal* maneuver is a member of the feasible maneuver set.

The problem addressed in this section is to find an optimal path that steers the car from a configuration q_I (associated with a configuration at the beginning of a sector) to a configuration q_E (associated with a configuration at the end of the sector) in minimum time. Let t_I represent the instant when the planned motion starts and t_E the instant when the motion ends. A configuration q is associated to the state variables $q = [x,\, y,\, \theta,\, v]$. The solution of the motion planning problem aims to identify the acceleration function $a(t)$, the steering angle function $\varphi(t)$ and the final instant t_E. In mathematical terms the problem can be formulated as the following optimal control problem:

Problem 3. Optimal Control Problem for Local Planning

$$\min_{a(t),\varphi(t)} \int_{t_I}^{t_E} L(q,a,\varphi)dt, \text{ } \textit{subject to}$$

$$(1) \quad q(t) \textit{ solution of } (4.1),$$

$$(2) \quad q(t_I) = q_I, q(t_E) = q_E \, , \; \forall t \in [t_I, t_E]$$

$$(3) \quad q(t) \in \mathcal{Q}$$

$$(4) \quad v_x^2(t)|\tan\varphi| \leq a_l l$$

$$(5) \quad a(t) \in [\underline{a}, \overline{a}], \varphi(t) \in [-\overline{\varphi}, \overline{\varphi}].$$

The two constraint in (2) are defined to set exactly the desired initial and final configurations. The constraint (3) requires that all configurations throughout the interval $[t_i, t_E]$ remain feasible. To elaborate this notion, for a given configuration q define by $p_r(q)$ the subvector $p_r = [x, y]$ associated with the mid point of the rear axle. The position of the midpoint of the front axle is given by $p_f(q) = [x + l\cos\theta, y + l\sin\theta]$. The set of feasible configurations $\mathcal{Q}$ is made of all configurations such that both p_r and p_f are inside the track: $\mathcal{Q} = \{q|p_f(q), \, p_r(q) \in \mathcal{P}\}$. Clearly both the initial and the final configuration q_I and q_E are required to be inside $\mathcal{Q}$ and so have to be the intermediate configurations. The constraint (4) requires that the car never exceeds the maximum allowed lateral acceleration a_l. This gives rise to the following constraints for the intermediate configurations of the system:

$$v_x \leq \sqrt{a_l R} \tag{4.2}$$

where $R = \frac{l}{|\tan\varphi|}$. Notice that a_l is a function of the tires grip, which depends on the ground characteristics, e.g., dry or wet asphalt, off road, etc., and generates constraint depending on the state variable v_x and the control input φ. The constraint (5) is on the physical limitation of the vehicle (maximum and minimum acceleration and steering angle). The cost function is in this case the time to complete the motion:

$$\int_{t_I}^{t_E} L(q,a,\varphi)dt, \tag{4.3}$$

where $L(q,a,\varphi) = a_1$, with $a_1 > 0$.

4.2 Maneuver Extremals

In order to find a solution to the problem 3, we first need to identify the set of extremals that verify the necessary condition for optimality based on the Pontryagin Minimum Principle (see e.g. [43] and [9] for the constraints on the state and control variables). For this purpose we study the problem disregarding the constraints on the configurations q that impose that the vehicle is on the track (constraint (3)). We will recover such geometric constraints later on.

Proposition 4. *Optimal paths consists of concatenation of*

1. *Straight line $\mathcal{S}$, traveled with any velocity profile, compatible with the maximum and minimum accelerations, i.e. $\overline{a}$ and $\underline{a}$, respectively;*

2. *Circular curve $\mathcal{C}_{\overline{r}}$ traveled with constant maximum velocity $\overline{v_x} = \frac{\overline{a}}{c_a}$; the radius is fixed to the maximum value $\overline{r}$ that is compatible with the constraint on the lateral acceleration: $\overline{r} = \overline{v_x}^2/a_l$, where a_l is the maximum lateral acceleration;*

3. *Circular curve $\mathcal{C}_{\underline{r}}$ traveled at (possibly time–varying) velocity $0 \leq v_x \leq \sqrt{\frac{a_l l}{|\tan(\overline{\varphi})|}} = v_{x\varphi}$; the radius is fixed to the minimum possible value allowed by the vehicle : $\underline{r} = \frac{l}{|\tan(\overline{\varphi})|}$;*

4. *Variable radius curves $\mathcal{V}_{\overline{a}}$ and $\mathcal{V}_{\underline{a}}$ executed with maximum or with minimum acceleration respectively, which always verify the relation $v_x^2|\tan(\varphi)| = a_l\, l$.*

Proof. The state-control constraint $v_x^2|\tan\varphi| \leq a_\ell l$ generates two constraints

$$\begin{aligned} C_1(q,a,\varphi) &= v_x^2 \tan\varphi - a_\ell l \leq 0, \quad \varphi > 0 \\ C_2(q,a,\varphi) &= -v_x^2 \tan\varphi - a_\ell l \leq 0, \quad \varphi < 0\,. \end{aligned}$$

Hence, the Hamiltonian function associated to the optimal control problem 3 is

$$\begin{aligned} H = a_1 &+ \lambda_1 v_x \cos\theta + \lambda_2 v_x \sin\theta + \lambda_3 \tan\varphi \frac{v_x}{l} + \lambda_4(a - c_a v_x) \\ &+ \mu_1(v_x^2 \tan\varphi - a_l l) - \mu_2(v_x^2 \tan\varphi + a_l l) \end{aligned} \tag{4.4}$$

where $(\lambda_1,\, \lambda_2,\, \lambda_3,\, \lambda_4)^T$ are the co-state variables $\mu_1 = 0$ ($\mu_2 = 0$) when $v_x^2 \tan\varphi < a_l l$ ($-v_x^2 \tan\varphi < a_l l$).

From the dynamic of the co-state $(\lambda_1,\, \lambda_2,\, \lambda_3,\, \lambda_4)^T$ we have $\dot{\lambda}_1 = -\frac{\partial H}{\partial x} = 0$ and $\dot{\lambda}_2 = -\frac{\partial H}{\partial y} = 0$ hence we can define $\lambda_1 = d\cos\gamma$ and $\lambda_2 = d\sin\gamma$ obtaining

$$\begin{aligned} H = a_1 + d\, v_x \cos(\theta-\gamma) + \lambda_3 \tan\varphi \frac{v_x}{l} + \\ + \mu_1(v_x^2 \tan\varphi - a_l l) - \mu_2(v_x^2 \tan\varphi + a_l l) \end{aligned} \tag{4.5}$$

the dynamics of the rest of the co-state are

$$\begin{aligned} \dot{\lambda}_3 &= -\frac{\partial H}{\partial \theta} = d\, v_x \sin(\theta-\gamma) \\ \dot{\lambda}_4 &= -\frac{\partial H}{\partial v_x} = -d\, \cos(\theta-\gamma) - \lambda_3 \frac{\tan\varphi}{l} + \\ &\qquad - c_a\lambda_4 - 2(\mu_1-\mu_2) v_x \tan\varphi \end{aligned} \tag{4.6}$$

We first analyze the extremal arcs when the state-control constraint is active, i.e. we assume $v_x^2|\tan\varphi| = a_l l$. Without loss of generality, we consider only the case in which $C_1 = 0$ and hence $C_2 < 0$. The opposite case gives a similar result. Notice that constraints C_1 and C_2 are mutually exclusive. With this choice, $\mu_1 \geq 0$ and $\mu_2 = 0$. Moreover, if $C_1 = 0$ the control variable φ can be obtained from the state variable v. However, φ could be less than $\bar{\varphi}$. The remaining control variable is $a \in [\underline{a}, \overline{a}]$.

If $-\bar{\varphi} < \varphi < \bar{\varphi}$ and $\underline{a} < a < \overline{a}$ than, for optimality, we have $\frac{\partial H}{\partial \varphi} = 0$, i.e. $v(\lambda_3/l + \mu_1\, v_x)(1+\tan\varphi^2) = 0$, and hence, as $v_x > 0$, $\mu_1 = -\lambda_3/(l v_x)$ which implies $\lambda_3 \leq 0$. Moreover, we have $\frac{\partial H}{\partial a} = 0$, i.e. $\lambda_4 = 0$, and hence $\dot{\lambda}_4 = 0$. From the second equation in (4.6), $\lambda_3 = -\frac{dv_x^2}{a_l}\cos(\theta-\gamma)$. Deriving λ_3 and considering the first equation in (4.6) we obtain that $a = c_a v_x + a_l \tan(\theta-\gamma)$. However, for minimum time problem, a necessary condition for optimality implies that $\dot{H} = 0$ along the optimal trajectories. By substituting all the above results, the Hamiltonian becomes $H = 2dv_x\cos(\theta-\gamma)$ and its derivative is $\dot{H} = da_l \sin(\theta-\gamma) = 0$ which implies $\theta = \gamma = const.$. This conclusion violates the assumption, i.e. $C_1 = 0$. As a consequence, at least one of controls a and φ must to be on the boundary.

If $0 < v_x < \overline{v_x}$ and $a = \overline{a}$ ($a = \underline{a}$) v_x increases up to $\overline{v_x}$ (decreases down to 0) and φ changes with v_x according to $C_1 = 0$, i.e. $v_x^2 \tan\varphi = a_l l$. Notice that the constraint on the maximum lateral acceleration can be written in terms of the curvature radius R as $v_x^2 = R a_l$, hence the extremal is a curve with a radius that increases proportionally to v_x^2, i.e. an arc of type $\mathcal{V}_{\overline{a}}$ (or $\mathcal{V}_{\underline{a}}$).

If $v_x = \overline{v_x}$ we have $\dot{v}_x = 0$ and hence $a = c_a\overline{v_x} = \overline{a}$. In this case the extremal arc is an arc of circle with radius $\frac{l}{\tan\varphi}$ followed at constant velocity $\overline{v_x}$ with an angle φ that is solution of $v_x^2 \tan\varphi = a_l l$, i.e. an arc of type $\mathcal{C}_{\bar{r}}$. This extremal exists only if $\overline{\varphi} \geq \arctan\frac{a_l l}{\overline{v_x}^2}$. In other words if the curves of the track are not too sharp this extremal may be an arc of the optimal path. In case of sharp turns it implies that the velocity must be decreased before the turn to avoid a lateral acceleration larger than a_l.

If $-\overline{\varphi} \leq \varphi \leq \overline{\varphi}$ we have $\frac{\partial H}{\partial \varphi} = \lambda_3(1+\tan^2\varphi)\frac{v_x}{l} = 0$. This implies $\lambda_3 = \dot{\lambda}_3 = 0$ and, from the first equation in (4.6) we have $\theta = \gamma$, $\dot{\theta} = 0$ and hence $\varphi = 0$. The obtained extremal is hence a straight line that is part of an optimal path only if is followed at the maximum speed based on the initial and final values of the velocity, i.e. an arc of type S.

If $\varphi = \pm\overline{\varphi}$ the vehicle proceeds along a arc of circle of constant radius $\underline{r} = \frac{l}{|\tan\varphi|}$. The arc is part of an optimal path only if is followed at the velocity $0 \leq v_x \leq \sqrt{\frac{a_l l}{|\tan\varphi|}}$, i.e. an arc of type $C_{\underline{r}}$.

□

4.2.1 Maneuver Analysis

For all maneuvers as described above, given the model (4.1), the velocity at the end of sector with constant acceleration a for the duration $\delta t = t_b - t_a$, is given as follows:

$$v_x(t_b) = \left(v_x(t_a) - \frac{a}{c_a}\right) e^{-c_a\delta t} + \frac{a}{c_a}. \tag{4.7}$$

Straight Line Maneuver $\mathcal{S}$ In the straight line maneuver we can have any velocity profile. The velocity at the end of the maneuver is given by:

$$\begin{aligned} v_x(t'_a) &= \min\left(\overline{v_x},\, \left(v_x(t_a) - \frac{\overline{a}}{c_a}\right) e^{-c_a\delta t_1} + \frac{\overline{a}}{c_a}\right), \\ v_x(t_b) &= \max\left(\underline{v_x},\, \left(v_x(t'_a) - \frac{\underline{a}}{c_a}\right) e^{-c_a\delta t_2} + \frac{\underline{a}}{c_a}\right), \end{aligned} \tag{4.8}$$

where $\delta t_1 = t'_a - t_a$ and $\delta t_2 = t_b - t'_a$. Moreover, for a given $v(t_a)$, $v_x(t_b)$ is in the set

$$\Gamma_{v_x} = \left\{ \max\left(\underline{v_x},\, \frac{\underline{a}}{c_a} + \left(v_x(t_a) - \frac{\underline{a}}{c_a}\right) e^{-c_a\delta t}\right),\, \min\left(\overline{v_x},\, \frac{\overline{a}}{c_a} + \left(v_x(t_a) - \frac{\overline{a}}{c_a}\right) e^{-c_a\delta t}\right) \right\}. \tag{4.9}$$

In particular, for each velocity in Γ_{v_x} there is *exactly one* solution for the switching point t'_a in order to have the optimal maneuver. Defining with t^0_a and t^0_b the times in which the vehicle reaches the maximum $\overline{v_x}$ and minimum $\underline{v_x}$ velocities respectively, and the time intervals $\delta t_1 = t^0_a - t_a$, $\delta t_2 = t'_a - t^0_a$, $\delta t_3 = t^0_b - t'_a$ and $\delta t_4 = t_b - t^0_b$, the distance traveled along the straight path is then given by

$$d_s = \frac{1}{c_a}\left(\overline{a}\delta t_1 + \underline{a}\delta t_3 - \left(v(t_a) - \frac{\overline{a}}{c_a}\right)e^{-c_a\delta t_1} - \left(v(t'_a) - \frac{\underline{a}}{c_a}\right)e^{-c_a\delta t_2}\right) + \overline{v_x}\delta t_2 + \underline{v_x}\delta t_4. \tag{4.10}$$

Circular curve $\mathcal{C}_{\underline{r}}$ In the curve with minimum radius, the velocity at the end of the maneuver can take any value $\underline{v_x} \leq v_x \leq v_{x\phi}$. Since we are interested in minimum time trajectories, we can assume that:

$$v_x(t_a) = v_x(t_b) = v_{x\phi}, \tag{4.11}$$

i.e, this maneuver does not modify the velocity profile, nor it can be taken if the velocity is less than $v_{x\phi}$. Thus for a case where $v_x < v_{x\phi}$, the vehicle has to take $\mathcal{S}$ maneuver. The overall angle $\Delta\theta_{\mathcal{C}_{\underline{r}}}$ covered during this maneuver, i.e., the overall change in the vehicle heading, corresponds to the total arc travelled on the maneuver, hence

$$\Delta\theta_{\mathcal{C}_{\underline{r}}} \in [0,\, 2\pi). \tag{4.12}$$

The time to execute the maneuver is then given by

$$\delta t = t_b - t_a = \frac{\underline{r}\Delta\theta_{\mathcal{C}_{\underline{r}}}}{v_{x\phi}}. \tag{4.13}$$

Variable radius curves $\mathcal{V}_{\underline{a}}$ For this type of curve, the velocity at the end of the maneuver is given by:

$$\begin{aligned} v_x(t'_a) &= v_x(t_a)\begin{cases} = \overline{v_x}, \text{ hence } t'_a \geq t_a \\ < \overline{v_x}, \text{ hence } t'_a = t_a \end{cases} \\ v_x(t_b) &= \left(v_x(t'_a) - \frac{\underline{a}}{c_a}\right)e^{-c_a\delta t_2} + \frac{\underline{a}}{c_a} \geq v_{x\phi}, \end{aligned} \tag{4.14}$$

where $\delta t_1 = t'_a - t_a$ and $\delta t_2 = t_b - t'_a$. Notice that as soon as $v_x(t_b) = v_{x\phi}$, the maneuver ends and switch to $\mathcal{C}_{\underline{r}}$ maneuver due to the previous assumption. Moreover,

$$\delta t_2 = -\frac{1}{c_a} \log \left(\frac{v_x(t_b) - \frac{\underline{a}}{c_a}}{v(t'_a) - \frac{\underline{a}}{c_a}} \right). \tag{4.15}$$

The overall angle covered by the maneuver is given by

$$\Delta\theta_{\mathcal{T}_{\underline{a}}} = \frac{\overline{v_x}}{\overline{r}} \delta t_1 + \frac{a_l}{\underline{a}} \left(\log(v_x(t_b)) + c_a \delta t_2 - \log(v_x(t'_a)) \right), \tag{4.16}$$

for $\mathcal{T}^-_{\underline{a}}$ (counter-clockwise maneuvers), while it is

$$\Delta\theta_{\mathcal{T}_{\underline{a}}} = \frac{\overline{v_x}}{\overline{r}} \delta t_1 - \frac{a_l}{\underline{a}} \left(\log(v_x(t_b)) + c_a \delta t_2 - \log(v_x(t'_a)) \right),$$

for $\mathcal{T}^+_{\underline{a}}$ (clockwise maneuvers).

Variable radius curves Maneuver $\mathcal{V}_{\overline{a}}$ Similar to the previous case, the velocity at the end of the maneuver is given by:

$$\begin{aligned} v_x(t'_a) &= \left(v_x(t_a) - \frac{\underline{a}}{c_a} \right) e^{-c_a \delta t_1} + \frac{\underline{a}}{c_a} \leq \overline{v_x}, \\ v_x(t_b) = v_x(t'_a) &\begin{cases} = \overline{v_x}, \text{ hence } t_b \geq t'_a \\ < \overline{v_x}, \text{ hence } t_b = t'_a \end{cases} \end{aligned} \tag{4.17}$$

where $\delta t_1 = t'_a - t_a$ and $\delta t_2 = t_b - t'_a$. Notice that $v_x(t_a) \geq v_{x\phi}$. Moreover,

$$\delta t_1 = -\frac{1}{c_a} \log \left(\frac{v_x(t'_a) - \frac{\overline{a}}{c_a}}{v_x(t_a) - \frac{\overline{a}}{c_a}} \right). \tag{4.18}$$

The overall angle covered by the maneuver is given by

$$\Delta\theta_{\mathcal{T}_{\overline{a}}} = \frac{\overline{v_x}}{\overline{r}} \delta t_2 + \frac{a_l}{\overline{a}} \left(\log(v_x(t'_a)) + c_a \delta t_1 - \log(v_x(t_a)) \right), \tag{4.19}$$

for $\mathcal{T}^-_{\overline{a}}$ (counter-clockwise maneuvers), while it is

$$\Delta\theta_{\mathcal{T}_{\overline{a}}} = \frac{\overline{v_x}}{\overline{r}} \delta t_2 - \frac{a_l}{\overline{a}} \left(\log(v_x(t'_a)) + c_a \delta t_1 - \log(v_x(t_a)) \right),$$

for $\mathcal{T}^+_{\overline{a}}$ (clockwise maneuvers).

4.2.2 Extremals with Geometric Constraints

For the principle of optimality any subpath of an optimal path is optimal itself. When taking into account the physical borders of the track the optimal solution will consist of subpaths along such borders (named as constrained subpaths) and subpaths strictly verify the physical border constraints (named as unconstrained subpaths).

In a straight sector, the track border constraints generates straight constrained subpaths that are equivalent to the unconstrained ones. On the other hand, along a curve the only two constrained paths are the arcs of circle with radius R_o and R_i. Along the optimal path those constrained paths are concatenated with the extremal unconstrained paths. To be optimal the constrained subpath must be followed at the maximum allowed acceleration without violating the lateral acceleration constraint.

4.3 Optimal Maneuvers

After introducing the alphabet of optimal maneuvers (extremals) that compose the motion plan, we will now discuss the optimal sequence of maneuvers to be used for the solution of Problem 3. We will discuss two different cases. In the first case, the sector of the track between the two end-points of the path is a straight line (straight sector), while in the second case it contains a curve (turn sector).

Generally speaking, the solution to this problem is given by a concatenation (a *word*) of extremals. Each extremal is associated with some free parameters. For instance parameters of the straight line are initial and final velocity and length. When two extremals are interconnected some of the free parameters are constrained (for instance the initial velocity of the second extremal has to be equal to the final velocity of the first one). Other constraints are obviously given by the two end-points that have to be interconnected by the sequence. Nevertheless, some of the parameters in the word remain free choice. So, in general, to find the optimal sequence of maneuvers that steers the vehicle between two configurations, one has to identify the optimal sequence of extremals and the correct choice of parameters that produce a minimum time

transition between the two configurations.

The solution of this problem is very challenging. However, by approximating the extremals with circular arcs, geometrical properties can be used to reduce the number of parameters to be identified. We will analyze in the depth this simplified case and show how it can be a useful source of inspiration for a solution heuristic that applies to the general case.

4.3.1 Sub–optimal sequences in a simplified case

In this section, we consider a simplified scenario in which the circular arcs, followed at constant speed with $v_x^2|\tan(\varphi)| = a_l l$, instead of variable radius maneuver $\mathcal{V}_{\overline{a}}$ and $\mathcal{V}_{\underline{a}}$ arcs. This leads to a sub–optimal path whose cost remains reasonably close to the optimal solution. We will analyze in depth this simplified case and show how it can be a useful source of inspiration for a heuristic solution that applies to the general case.

Each sector is delimited by two lines sl and al which are perpendicular to the lane. So the initial configuration $q(t_I)$ is such that the sub-vector $p_f(q(t_I)) \in sl$ (i.e., the front axle is on the start line) and the final configuration $q(t_E)$ is such that $p_f(q(t_E)) \in al$ (i.e. the front axle is on the finish line of the sector). For the sake of simplicity, we further assume that $\varphi(t_I) = \varphi(t_E) = 0$, i.e. the car is parallel to the lane at the beginning and the end of the sequence.

4.3.1.1 Trajectories for straight sectors

Consider a straight sector as represented in Figure 4.1. We are interested in analyzing all position pairs $p(t_I) = [x(t_I),\, y(t_I)]$ and $p(t_E) = [x(t_E),\, y(t_E)]$ where $x(t_I) = 0$ and $x(t_E) = L_S$ (L_S is the length of the sector) and $0 \leq y(t_I),\, y(t_E) \leq W$. Given the optimal path from $p(t_I)$ to $p(t_E)$ with velocities $v(t_I) = v_I$ and $v(t_E) = v_E$, for translation invariance, it is also the optimal path from $(0,\, y(t_I)+h)$ to $(L_{S_i},\, y(t_E)+h)$ with $0 \leq h \leq W - \max\{y(t_I),\, y(t_E)\}$ and with the same velocities. Furthermore, the symmetry with respect to lines parallel to the sector borders provide the optimal path from $[0,\, y(t_I)]$ to $[L_S,\, 2y(t_I) - y(t_E)]$ with $0 \leq y(t_I) \leq \frac{W+y(t_E)}{2}$ and with $v(t_I)$ and $v(t_E)$. Hence, without loss of generality, it is sufficient to consider $y(t_I) = W$

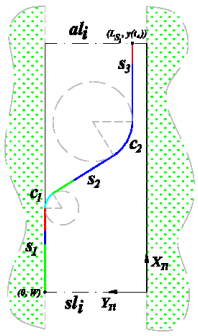


Figure 4.1: Optimal solutions on a straight sector.

and $0 \leq y(t_E) \leq W$.

The analysis can be carried out using the classical arguments of Dubins [21]. In particular, excluding the case of L_S much smaller than the minimum radius of curvature $R = \frac{L}{\tan \bar{\varphi}}$ (which gives rise to more complex sequence), the optimal word is given by $\mathcal{SCSCS}$, where the turns $\mathcal{C}$ can be a circle with generic radius traveled at maximum allowed velocity and some of the extremals of the sequence can be missing (or equivalently have zero duration).

In the sequence $\mathcal{SCSCS}$ the free parameters are the length s_1 and s_3 of the first and of the third straight sectors and the velocities v_1 and v_2 of the two curves. By varying all the four parameters a path from $p(t_I)$ to $p(t_E)$ is found. Additional constraints come from geometric considerations: for example, it can be shown that the quantity $s_1 + s_3$ can never exceed L_S.

The parameters are strongly affected by the initial and the final velocities. For example, if $v(t_I) = v(t_E) = \bar{v}$ (and if L_S is sufficiently large with respect to the radius of the admissible curve at maximum speed $R = \frac{\bar{v}_x^2}{a_l}$) the solution is a Dubins path $\mathcal{CSC}$ with $s_1 = s_3 = 0$ and $v_1 = v_2 = \bar{v}_x$. It can be shown that, for small values of c_a and for sufficiently large L_S, the behavior along the solution is to use maximum acceleration along the first two straight lines while braking (with minimum negative acceleration), if needed, along the third straight line to reach the desired final speed.

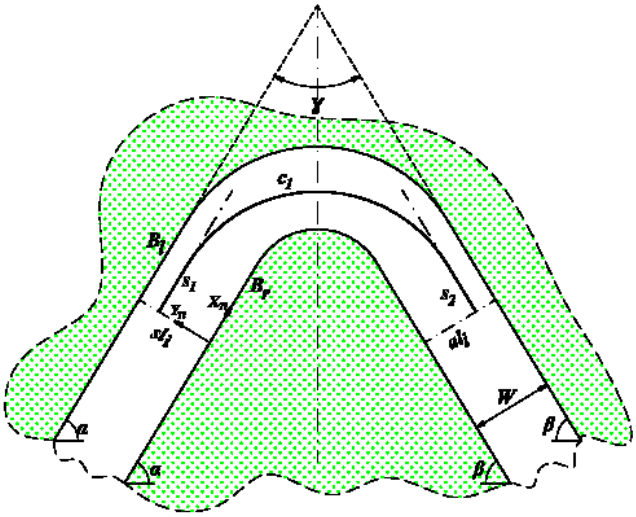


Figure 4.2: Sub–optimal path on a turn sector.

4.3.1.2 Trajectories for turn sectors

Consider a turn sector S_i as represented in Figure 4.2. We are interested in analyzing all position pairs $p(t_I) = [x(t_I),\, y(t_I)]$ and $p(t_E) = [x(t_E),\, y(t_E)]$ where $x(t_I) = 0 \in sl_i$ and $x(t_E) \in al_i$ and $0 \leq y(t_I),\, y(t_E) \leq W$. Contrary to the straight sectors, in this case invariance properties of the optimal solution do not help restrict the initial of final points to be considered.

Based on the geometric characteristics of a turn sector and using arguments *à la* Dubins, the considered word is in this case given by $\mathcal{SCS}$. By using this word to move the car from one configuration to another, the only free parameter is s_1, which is the length of the first (short) straight line. The radius and hence the velocity v_1 associated to the curve is determined univocally by s_1. Indeed, there exists only a circle tangent to both the straight lines of $\mathcal{SCS}$ at distance s_1 from the initial point. Obviously, for the existence of an admissible velocity v_1 and for the sector borders constraints, the values of s_1 are limited. Finally, the length of the second

(short) straight line follows from the position of the arrival line al,form s_1 and v_1. By varying the parameter l_1 the path from $p(t_I)$ to $p(t_E)$ can be found.

As for the case of the straight line, the solution is strongly affected by the initial and final velocities, as well as by the values of R_b^i and R_b^o. It can be shown that, for sufficiently large values of R_b^i, the behavior along the solution is to touch the inner border of the turn.

4.3.2 Optimal sequences in the general case

In the general case, the family of the extremals is much richer and any two extremals can potentially be interleaved by a straight line. Therefore, we propose here a heuristic solution (rather than the exact one) that is mathematically tractable and is closely inspired to the simplified case discussed above.

First of all, we can regroup the maneuvers that move along the constraint imposed by the lateral acceleration, and the other that will not. Hence, we will define the maneuvers

$$\mathcal{T}_{\underline{a}} = \mathcal{C}_{\overline{r}} \circ \mathcal{V}_{\underline{a}}$$
$$\mathcal{T}_{\overline{a}} = \mathcal{V}_{\overline{a}} \circ \mathcal{C}_{\overline{r}}$$

Let us first focus on the straight sector and assume that the velocity at the beginning and at the end is higher than the one that can be held in the curve. In the simplified case, the maneuver (which is essentially a change of lane) clearly requires two circular curve in the opposite sense interleaved by a straight line. Besides, we have an initial straight and a final straight. The initial straight can be used to reduce the speed before starting to turn (so as to reduce the radius of the curve that can be taken). Likewise, the final straight can be used to accelerate the car until the target velocity is reached. With addition of the extremal $\mathcal{T}$, we have an important advantage: the car can start turning while changing the speed. Likewise, when the curve finishes the car uses the $\mathcal{T}$ to anticipate the opening of the throttle and by gradually opening the curve until it turns into a straight. The resulting sequence is the following

$$\mathcal{S}\mathcal{T}_{\underline{a}}\mathcal{C}_{\underline{r}}\mathcal{S}\mathcal{C}_{\underline{r}}\mathcal{T}_{\overline{a}}\mathcal{S}.$$

Similar arguments apply to a turn sector. In this case, a potentially good sequence that

generalizes the $\mathcal{SCS}$ sequence can be:

$$\mathcal{ST}_{\underline{a}}\mathcal{C}_{\underline{r}}\mathcal{S}[\mathcal{C}_{B_r}|\mathcal{C}_{B_l}]\mathcal{SC}_{\underline{r}}\mathcal{T}_{\overline{a}}\mathcal{S}.$$

In this case we make the same use of the $\mathcal{T}$ maneuvers as in the straight line. In addition, we can have either a curve with maximum or with minimum radius $[\mathcal{C}_{B_r}|\mathcal{C}_{B_l}]$ of the sector to account for the geometry of the track (see section 4.2.2).

Additionally, each curved maneuver has a superscript equal to $-$ or $+$ depending on the fact that the curve turns in the clockwise or counterclockwise direction (thus decreasing or increasing the value of the angle θ according to the right–hand rule).

4.3.3 Optimizing Parameters

The parameters used in the optimization for the simplified scenario are the length and the velocity at the end of the first straight line. An example of the result of the optimization for a turn sector can be observed in figure 4.3. The car starts and ends with equal velocities $v_i = v_e = 58.31$ m/s. It reaches 62.23 m/s at the end of the first line. Afterwards the car decelerates (shown as the blue curve) until it can accelerate to reach the correct orientation and a final velocity feasible to reach v_e with a straight line (shown as the red curve).

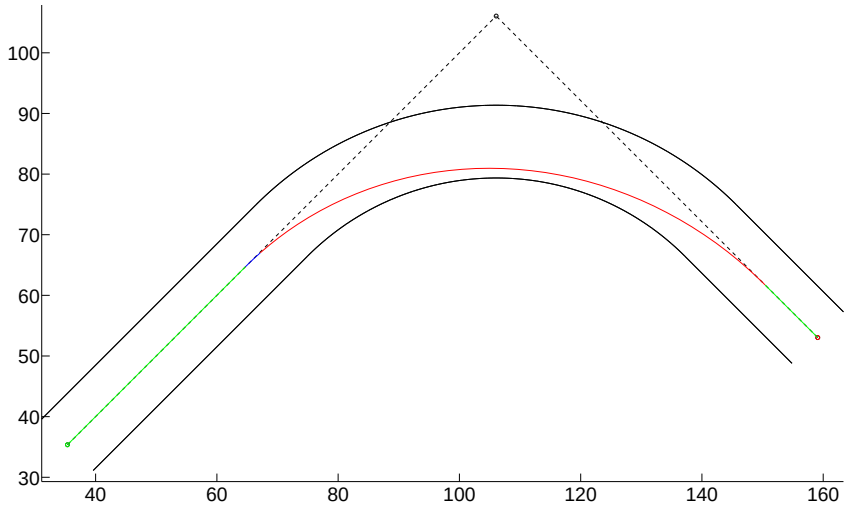


Figure 4.3: An example of an optimal maneuver for a given initial and final configuration.

Chapter 5

Global Planning

acThe global planning roblem addressed in this thesis can be summarized in the following terms: *given an autonomous car–like vehicle that runs on a known track along with other vehicles, plan a trajectory that allows the vehicle to complete a given number of laps on the track, in minimum time while avoiding collisions with other vehicles* [50].

Path planning in a competition track is significantly different from path planning in a urban or extra urban road [32]. First, in competition scenario the environment is strongly structured and well known upfront; thus, reducing the effect of uncertainty from the environment. Second, the track is accessible only to a small and predetermined number of cars. Third, all the cars in the track have very similar spatial footprint and dynamic characteristics. This reduces the amount of real–time data that the planner requires. Hence, the amount of data that the robot has to collect and process in real–time is relatively small compared to cases of the absence (or lack) of *a-priori* information. On the other hand, the high speed of the car requires the information acquisition, the optimal planning with collision avoidance and the control phases to be executed within tight real–time constraints.

Autonomous driving for cars is a very popular theme amongst a multidisciplinary research community [27, 28]. In particular, trajectory optimization for race car simulators has received a constant attention. The idea of decomposing the track into segments has been explored in [8], the authors apply genetic algorithm to find the best trade–off between length and curvature of the racing line. The authors in [13] solve numerical optimization problems taking into account

the inertia of the vehicle. role of yaw inertia or the mass of the vehicles.

The approach we advocate in this chapter builds on the extremal maneuvers, i.e. that verify the Pontryagin Minimum Principle necessary conditions for optimality [43], to steer the car from one configuration to another in minimum time as described in Chapter 4.

Finally, a very important inspiration for this work is the technique usually referred to as "discrete abstraction" [3, 24, 40, 41, 42, 44]. whereby a system with dense state is translated into a discrete system (essentially a state machine) to simplify planning and verification of properties.

This chapter is organized as follows: In Section 4.1 we introduce the most important concepts this chapter revolves around and propose a formal statement for the global planning problem. In Section 5.3 we discuss our graph based discrete abstraction of the problem. In Section 5.5, we support our technique by a large set of simulation results.

5.1 Problem Definition

The goal of this chapter is to find the sequence of maneuvers that allow the car to complete a generic number of laps in minimum time. Hence, the cost function to be minimized over the track is

$$\int_0^T L(q, a, \varphi)dt, \tag{5.1}$$

where $L(q, a, \varphi) = 1.$, with $a_1 > 0$ is a weight for the maneuver total time.

Given a configuration $q = [x,\, y,\, \theta,\, v]$, $p_r = [x, y]$ denotes the position of the midpoint of the rear axle, while the midpoint of the front axle is $p_f = (x + L\cos\theta, y + L\sin\theta)$. Let Σ be the sequence of sectors to be traversed. In other words, Σ is the region of configurations such that the vehicle is inside the track; such configuration requires that both p_f and p_r are inside the track: $\Sigma = \{q|p,\, p_f \in \mathcal{P}\}$. Moreover, at time t_i the configuration is supposed to lie on $sl_i = \{q|^T p_f = k[0,\, W]^T, 0 \leq k \leq 1,\ p \in \mathcal{P}\}$, i.e. the starting region of sector S_i. If the circuit has n_s sectors and the number of laps is n_l, Σ comprises n_l ordered sequences of n_s sectors. We can now state the *Track Optimal Problem*:

Problem 4. Track Optimal Problem

$$\min_{a(t),\varphi(t)} \sum_{i=0}^{n_l n_s - 1} \int_{t_i}^{t_{i+1}} dt, \text{ subject to}$$

$$q(t) \text{ solution of } (4.1),$$
$$q(t) \in \Sigma, q(t_i) \in sl_i, \forall i = 1, \ldots, n_l n_s$$
$$v(t) \in [\underline{v}, \overline{v}]$$
$$v^2(t) \tan\varphi \leq a_l L$$
$$a(t) \in [\underline{a}, \overline{a}]$$
$$\varphi(t) \in [-\overline{\varphi}, \overline{\varphi}].$$

At time $t_1 = 0$ the initial configuration is supposed to lie on $sl_1 = \{q|^I p_f = k[0,\ W]^T, 0 \leq k \leq 1,\ p \in \mathcal{P}\}$, with $sl_1 = sl_{n_s}$ while $sl_{n_l n_s} = \{q|^E p_f = k[0,\ W]^T, 0 \leq k \leq 1,\ p \in \mathcal{P}\}$. Such problem refers to a single car requiring that the kinematic model, and the different geometric and dynamic constraints are respected. An additional requirement (addressed in the final part of the chapter) is that no collision happen with the other vehicles, assuming that they also adopt a time optimal strategy.

5.2 Overview of The Approach

The approach for the global planning problem is represented in Figure 5.1, where squares denote information, ovals denote steps and squares with rounded corners denote results of the steps. Part of the steps are carried out offline and part are carried out online. The backbone of our approach is an abstraction that allows us to reformulate the path planning problem in a discrete graph–based setting. The approach comprises of the following steps:

1. **Graph Construction**: in this step, a graph for each car is constructed. It can be divided into sub steps as follows: 1) partitioning the track into sectors, 2) building the vertices by quantizing the position and velocities of the car into finite cell in each partition, 3) connecting a pair of vertices using the cost of the optimal maneuver between the vertices as weight.

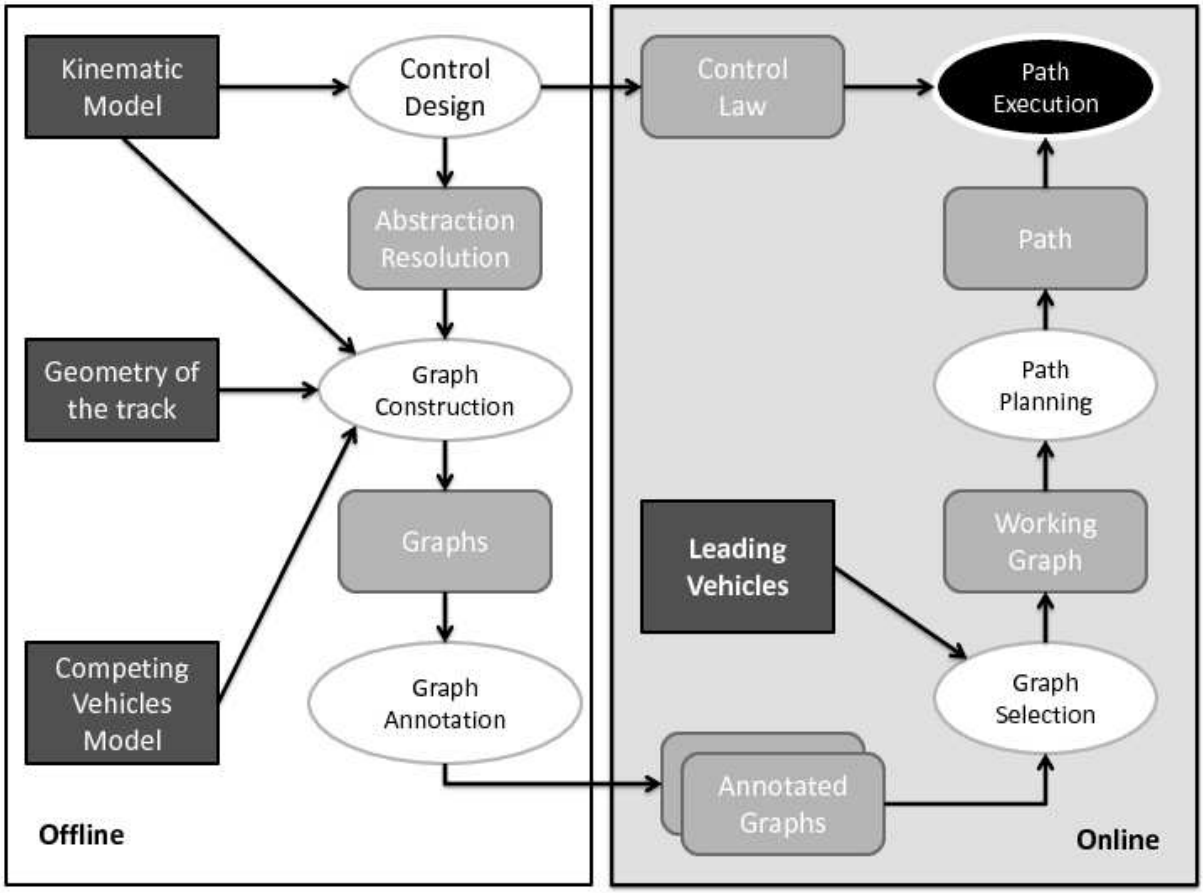


Figure 5.1: Overview of the approach

2. **Graph Annotation**: in this step two or more graphs are analyzed to detect collisions.

3. **Graph Selection**: to execute an avoidance or overtake maneuver graphs of leading cars are selected while graphs of following cars is omitted from the selection

4. **Path Planning**: a shortest path algorithm is perform to find the optimal path to be followed

5.3 Graph–based Planning

A solution to Problem 4 can be found using nonlinear optimal control theory or MPC–like tools. However, a complete characterization of the solution as well as the high computational cost

required for such solutions (in particular, if the algorithm has to be executed on an autonomous robotic vehicle endowed with limited computing resources) leads to the necessity of a more manageable solution. In this thesis, we decide to represent the track using a discrete abstractions of its possible configurations. More precisely, a possible representation of the track can be given in terms of a graph.

5.3.1 Track Partitioning

The first step in the graph construction is track partitioning. The complete track is partitioned into sectors where each sectors can be characterized a straight sector or a turn sector (see Section 4.3). *Way lines* (sl_i, sl_{i+1}) are the start and end line of sector i which are orthogonal to the track boundaries. Two subsequent sectors shared a way line, i.e., the way line at the end of sector i is the start way line for sector $i + 1$.

Example 5.1. *Figure 5.2 shows an example of the track partitioning algorithm. The track is partitioned into 18 partition that consists of 1 straight sector and 17 bend sectors.*

5.3.2 Graph Construction

Consider a discretization of dimension d_{w} of the width W of the track. The d_{w} points laying on the orthogonal lines sl_i are the starting points of S_i and final points of S_{i-1}, while the points laying on the orthogonal lines $al_i = sl_{i+1}$ are considered as starting points of S_{i+1} and ending points of S_i. Considering a circuit that is partitioned into n_s sectors. The total number of nodes in the graph is $(n_s + 1)d_{\text{w}}$ where we assume $S_{n_s+1} = S_1$. In other words, to close the circuit, nodes of S_1 are considered twice: one as initial nodes of S_1 and one as final nodes of S_{n_s}.

The edges of the graph represent a maneuver inside the sectors. Two nodes are connected with an edge iff: 1) they are located in the same sector 2) they belong to different way lines 3) there exist a maneuver that moves the car from the initial node (associated with a car configuration) to the finale node. Hence, every node on sl_i can be connected through an edge to every node on $sl_{i+1} = al_i$. Therefore, each sector consist of d_{w}^2 arcs and the graph consists of $n_s d_{\text{w}}^2$

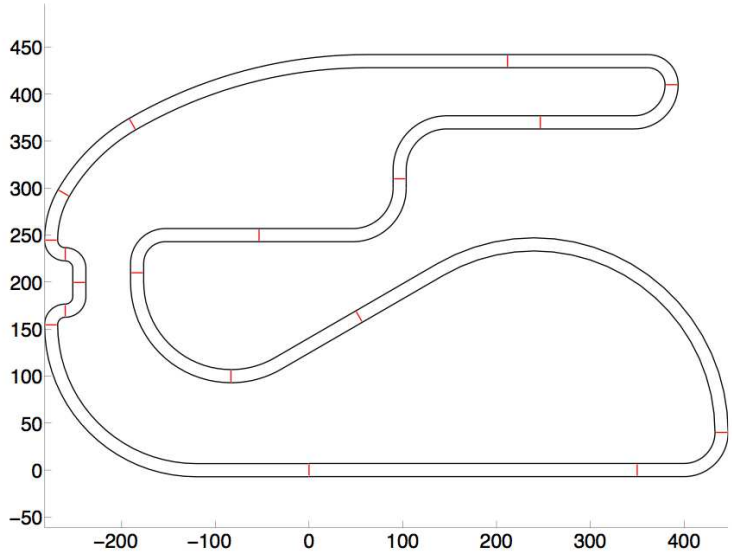


Figure 5.2: Example of track partition. The red lines are the separator between each partition

arcs. A cost $c_{i,j}$ equal to the minimum time required to go from a node i to j is associated to arc (i, j). However, combination of minimum time sectors does not results in a minimum time lap. Nevertheless, it is sufficient to reach al_i from sl_i along S_i in the minimum time *and* with the maximum velocity in al_i. In other terms, it is sufficient for sector S_i to minimize the following cost index

$$\int_{t_i}^{t_{i+1}} L(q, a, \varphi)dt, \tag{5.2}$$

where $L(q, a, \varphi) = 1$ in each track sector.

Therefore, for a sector S_i, the initial and final configurations assumed by the vehicle are constrained on two lines, defined by $sl_i = \{q|^{I_m}p_f = k[0, W]^T, 0 \leq k \leq 1,\ p \in \mathcal{P}\}$ and $al_i = \{q|^{E_m}p_f = k[0, W]^T, 0 \leq k \leq 1,\ p \in \mathcal{P}\}$, respectively. Hence, in place of Problem 4 the following set of optimal control problems can be defined

Problem 5. Optimal Control Problem

$$\min_{a(t),\varphi(t)} t_{i+1} - t_i, \textit{ subject to}$$

$$\begin{aligned}
&q(t) \textit{ solution of } (4.1),\\
&q(t) \in \Sigma, q(t_i) \in sl_i,\ q(t_{i+1}) \in al_i\\
&v(t_i) = v_i^i,\ v(t_{i+1}) = v_i^f\\
&v(t) \in [\underline{v},\ \overline{v}]\\
&v^2(t)\tan\varphi \leq a_l L\\
&a(t) \in [\underline{a},\ \overline{a}]\\
&\varphi(t) \in [-\overline{\varphi},\ \overline{\varphi}].
\end{aligned}$$

It is worth noting that, with respect to Problem 4, the optimal solution in each sector must be determined given the speeds v_i^i and v_i^f at the beginning and at the end of the sector S_i. Based on the principle of optimality, those constraints are introduced to obtain the concatenation of optimal solutions of Problem 5 for each sector that is the optimal solution of Problem 4.

To solve this problem, a discretization of dimension d_v of the speed space v is also provided, so that each point on sl_i can be crossed at d_v different speed values. Hence, $d_{\mathrm{w}}d_v$ nodes are associated to any initial segment $sl_i, \forall i$. With n_s sectors, the total number of nodes in the graph is now $(n_s + 1)d_{\mathrm{w}}d_v$, while the number of arcs turns to $n_s d_{\mathrm{w}}^2 d_v^2$. More formally, a node k is represented by a triplet $k = (S_k, p_k, v_k)$ where S_k is the sector, p_k is one of the d_{w} position of the point represented by k on sl_k, and v_k is one of the d_v speeds pertaining to a point p_k. Given nodes i and j the arc (i, j) belongs to the graph if and only if $S_j = S_{i+1}$. The cost $c_{i,j}$ associated to arc (i, j) is equal to the minimum time required to go from a node i to j (with the corresponding speeds). Hence, the cost $c_{i,j}$ equal to the solution of the minimum time Problem 5 with $v(0) = v_i,\ v(T) = v_j$ from p_i to p_j.

Example 5.2. *Figure 5.2 shows an example of the discretization of a sector where each line is discretized into three positions ($d_w = 3$) and the velocity space is also discretized into 3 values ($d_v = 3$). The different colored circles represent different velocity values.*

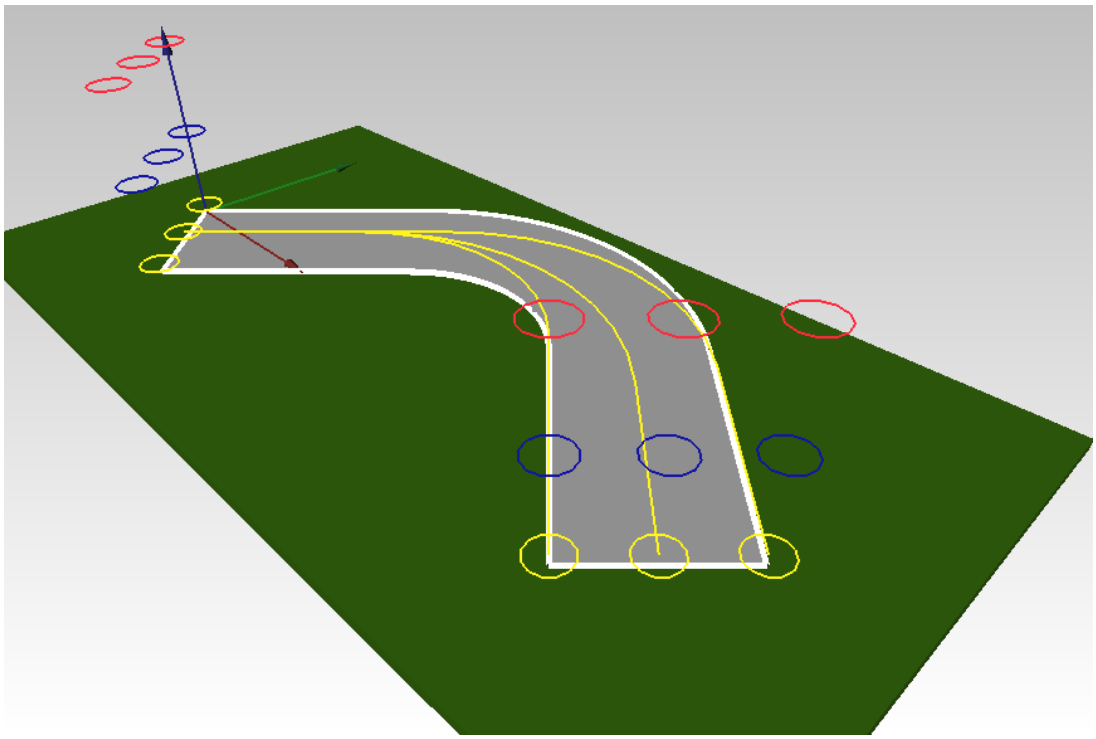


Figure 5.3: Example of a discretization of a bend sector.

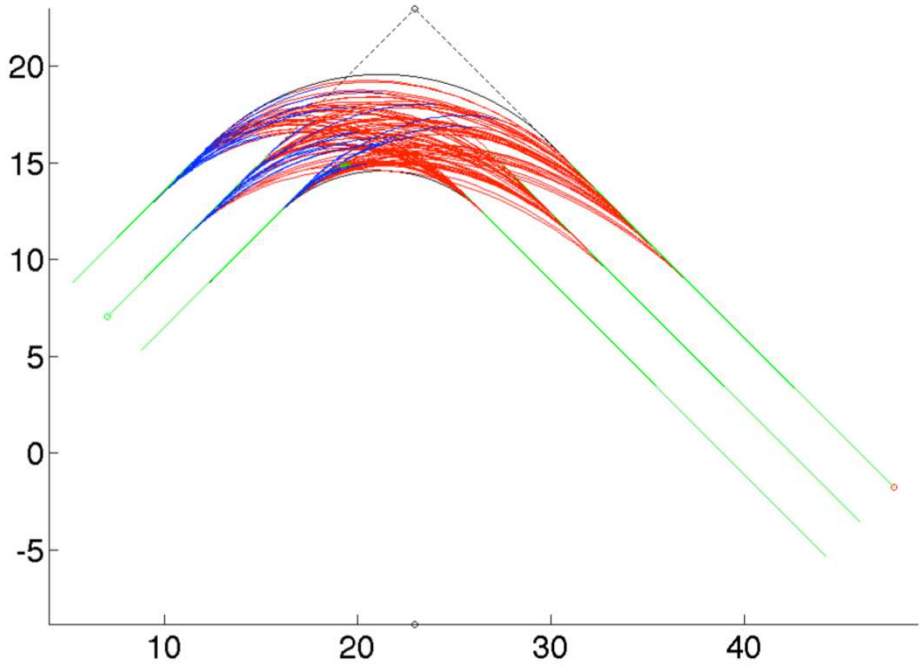


Figure 5.4: Example of all feasible maneuver for a given sector.

Example 5.3. *Figure 5.4 shows an example all possible maneuver constructed for a given bend sector. The waylines in this sector is discretized into 3 positions and 6 velocity values. The green lines in the figure show that the car is traveling in a straight line with a given velocity profile, the blue lines represent the car decelerating while turning while the red lines represent the car is accelerating while turning. If there exist a maneuver from an initial configuration to a final configuration, an edge is added to the graph between the nodes associated with with the configuration and the weight of the edge is the time to complete the maneuver.*

5.3.3 The Optimal Path

In order to apply standard shortest paths algorithms such as Dijkstra [20] it is necessary to introduce two nodes. An initial node I and a final node F and all arcs (I, i) and (j, F) where $S_i = S_1$ and $S_j = S_{n_s+1}$. The associated costs are null, i.e. $c_{Ii} = c_{jF} = 0$. With the introduction of such nodes algorithm such as Dijkstra provide shortest path from any node of the graph to

F. In particular, from any point on S_1 to S_{n_s+1}.

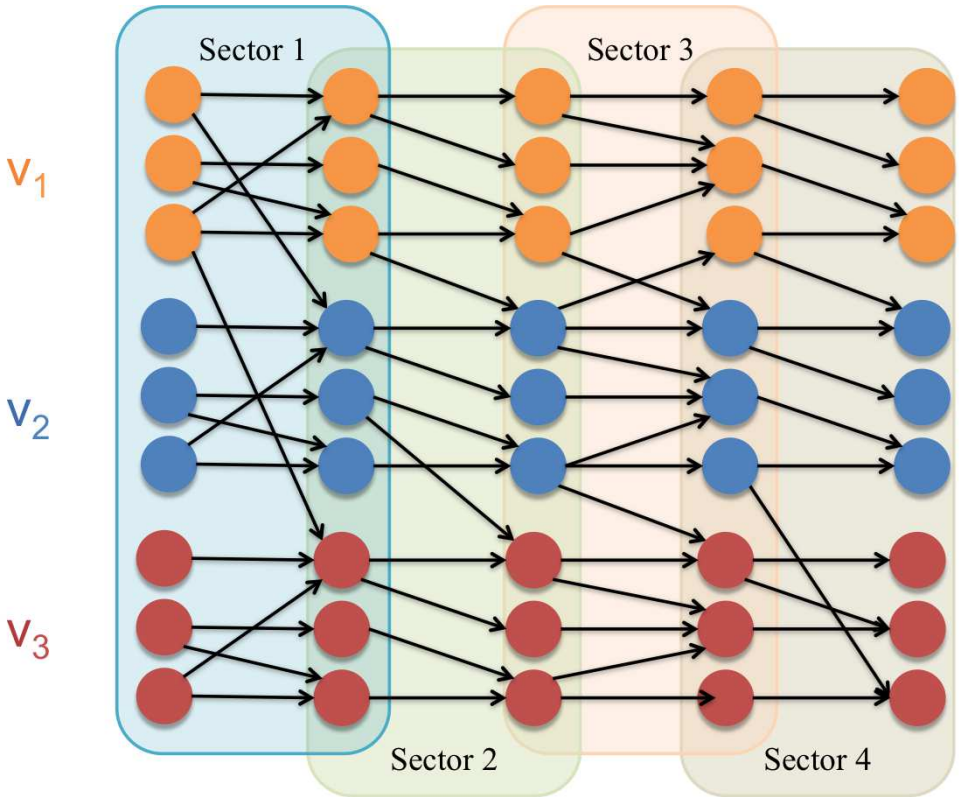
Determining the minimum path from the nodes associated to the circuit starting line to the same set of nodes considered on the arrival line the minimum time lap can be determined with the associated sequence of maneuvers described in previous section. It is worth noting that with this approach the best trajectory for the qualifying lap is determined, where there is only one car on the circuit. If we are interesting in 2 or more minimum time laps the graph must be extended duplicating the nodes and the arcs. Indeed, a graph associated to two laps on the circuit must be taken into account considering S_i of the first lap different from S_{n_s+i} of the second lap. A graph with $2n_s d_{\mathrm{w}} d_v$ nodes and $2(n_s+1)d_{\mathrm{w}}^2 d_v^2$ arcs is hence considered. The same construction of nodes I and F with associated arcs and cost can be followed and the optimal trajectory for a generic lap of the race can be found.

The discretization of the width W and of the speed v will obviously provide a suboptimal solution. However, a finer quantization provides a better solution but with the drawback of having a huge graph and hence a higher computational costs.

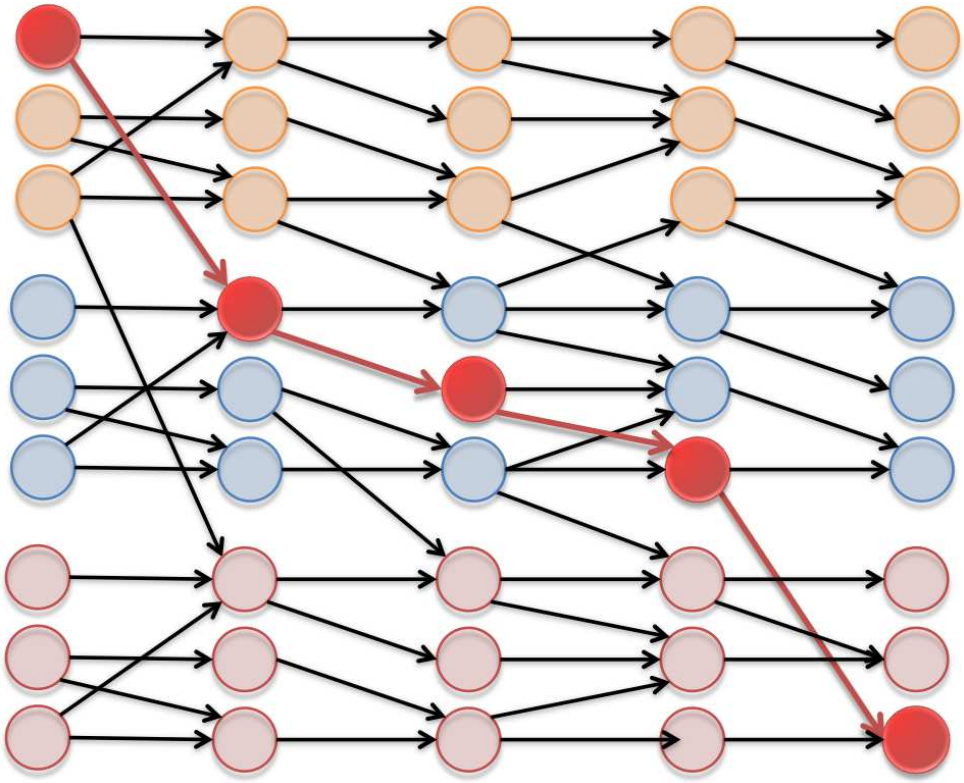
Example 5.4. *Figure 5.5 shows an example a graph constructed for a track with 4 sectors where each sector is discretized into 3 positions and 3 velocity values and the optimal path as a result of the Dijkstra's algorithm*

5.3.4 Avoiding Obstacles

The graph abstraction can easily be applied to account for the presence of other (slower) cars in front of the vehicle that are not cooperative (i.e., do not facilitate the overtake). This is done working with two graphs (one for each vehicle). The first step is to create a relations between the arcs of the two graphs. A pair of arcs belongs to the relation if it is possible to have a collision when the arcs are taken with a wrong timing. Suppose that vehicle A follows and vehicle B leads, and assume that a couple of arcs $(a_A,\, b_A)$ (belonging to the graph of A), $(a_B,\, b_B)$ (belonging to the graph of B) potentially lead to a collision. Using simple kinematic considerations (which we do not detail for the sake of brevity) it is possible to find a minimum inter



a)



b)

Figure 5.5: a) Example of a graph constructed for a track with 4 sectors b) the optimal maneuver found with Dijkstra's algorithm

arrival time $\tau_{(a_A,b_A)\to(a_B,b_B)}$ such that if A enters the arc $(a_A,\ b_A)$ after a time $\tau_{(a_A,b_A)\to(a_B,b_B)}$ elapsed since B entered $(a_B,\ b_B)$, then the collision is avoided.

The algorithm for path planning can be modified as follows. At the beginning A detects the position of B and assumes that it will use its graph for a minimum time path planning. As a consequence it knows the position of B in its graph for any time in the future. As a first step A finds the shortest path using its graph and annotates it with the time each node is reached. If the path thus found contains an arc (a_A, b_A) that is in relation with another arc (a_B, b_B) that B used, the algorithm checks if the minimum inter arrival time is respected. If not the arc (a_A, b_A) is removed from the graph and the Dijkstra algorithm is repeated on the updated graph. These steps are repeated until a "clear" path is found. The algorithm also checks when A becomes the leader. From that point on the arcs that are possibly removed during the algorithm execution are reinserted and A can use all of its graph.

5.4 Towards an optimal solution

Instrumental to the construction of the graph is the solution of the local planning problem: how to steer the car from a configuration $q(t_I)$ where each configuration is define by position and velocity. This problem has been addressed in Chapter 4.

Each extremal is characterized by a set of parameters. For instance a straight line $\mathcal{S}$ is associate with its length, initial velocity, final velocity. When the extremals are concatenated together some of the choices become bound by the previous extremal in the sequence. For instance, if a straight line is followed by a bend, the initial velocity of the bend will have to be equal to the final velocity of the straight line. Similar constraints holds for the trajectory tangents in the concatenation point. Additional constraint are obviously imposed by the initial and the final configuration. Nevertheless some of the parameters remain open and can be considered as decision variables in an optimization procedure aiming for the minimum time solution. Efficient solution strategies for this challenging problem are still under investigation. In the simulation section below we report solutions obtained with a combination of simulated annealing and gradient descent.

The optimal solution of this problem is the weight of the arc connecting the two nodes of the graph.

5.5 Simulations

The simulations are performed with two cars running on a track for 5 laps. Both cars have equal size ($3.5m$ in length and $1.8m$ wide) and same maximum and minimum accelerations $\overline{a} = 34.5m/s^2$ and $\underline{a} = -20m/s^2$ respectively. A minimum curvature radius of $15m$ is imposed on both cars to account for their minimum turning radius.

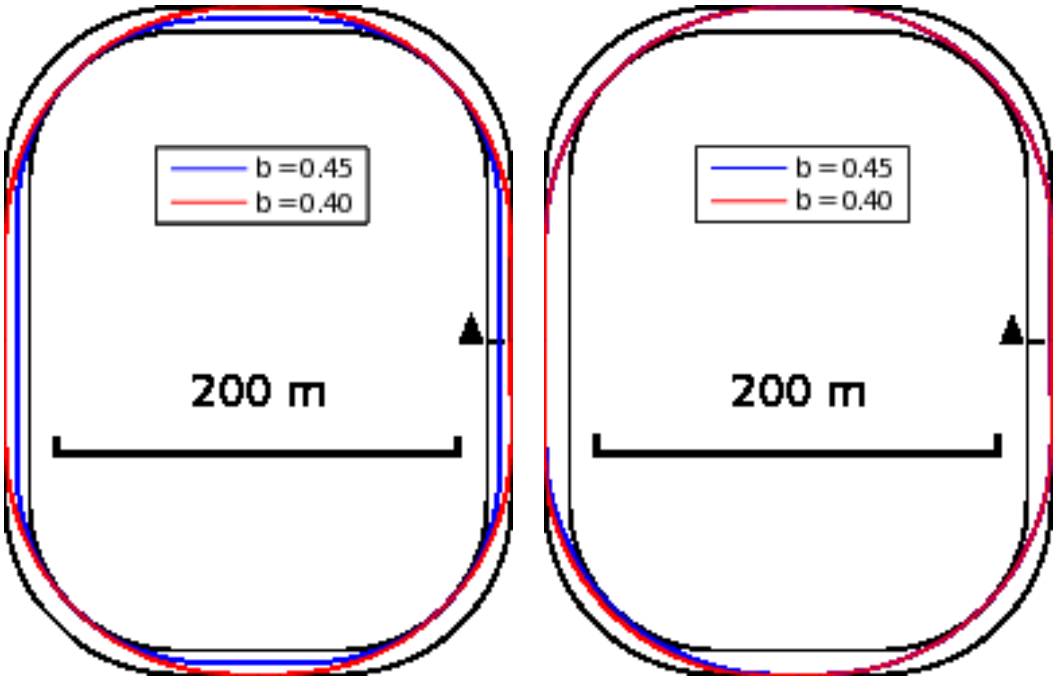
The cars have different viscous friction parameter (b) which in turn affects their maximum speed ($\overline{v}$) and different maximum lateral acceleration (a_l). For the first car: $b = 0.45$, $\overline{v} = 0.95\frac{\overline{a}}{b} = 72.89m/s$, and $a_l = 67.5m/s^2$, while for the second car: $b = 0.4$, $\overline{v} = 82m/s$, and $a_l = 60m/s^2$,

5.5.1 Static Graph

The optimal trajectory of a vehicle is affected by the choice of parameter values e.g. maximum velocity, maximum acceleration, length of the track, maximum track curvature radius, etc. Therefore, the produced graph differs for different set of parameter values. Figure 5.6.a shows the result of two independent simulations where the viscous friction of the car differs. Car 1 (blue) is able to reach its maximum velocity and hold this velocity along the blue line, thus trajectories with longer path are suboptimal. Conversely, car 2 (red) is able to reach its maximum velocity and hold this velocity when traveling along the outer boundary of the straight line, hence it will take more time than the Car 1.

Figure 5.6.b shows the result of a similar simulation with a reduced (half) value of a_l. Due to the reduced maximum lateral acceleration, both cars cannot find a trajectory where they can reach the maximum velocity along the curve.

Table 5.1 and Table 5.2 show the results of simulations of different choices of quantization and different road conditions. The wet road condition is simulated by reducing the values of



a b

Figure 5.6: a) Trajectories from two independent simulations of a car with different viscous frictions (b) b) trajectories for reduced lateral acceleration $a_l^b = a_l^a/2$

$\overline{a}$, $\underline{a}$ parameters with respect to the dry road condition. Table 5.1 is related to the track shown in Figure 5.6 where the length of rectilinear paths are $160m$ and $80m$ while the curvature radii are $80m$. Table 5.2 shows results for a similar track with a smaller dimension (rectilinear paths length are $100m$ and $50m$, curvature radii are $50m$).

As shown in Table 5.1, the time to complete a lap of Car 1 does not change by increasing the dimension d_v of the velocity space. This is due to the fact that both cars can reach their maximum velocity and hold the velocity until the end of the lap. However, increasing the dimension d_{w} enables Car 1 to reduce the lap time by having a trajectory with shorter length. Conversely, the time to complete a lap of Car 2 on a dry road condition is unaffected by changing d_{w} or d_v. This is due to the fact that there is only one trajectory that enables Car 1 to reach highest feasible velocity. On a wet road condition, where the maximum acceleration is lower, Car 2 reduces its lap time by increasing d_v because the finer discretization increases the number of feasible maneuvers.

car id	road	d_{w}	d_v	lap time (s)		
				1	2 - 4	5
1	dry	3	6	14.20	13.14	13.14
			12	14.20	13.14	13.14
		6	6	14.12	13.04	13.04
			12	14.12	13.04	13.04
	wet	3	6	15.05	13.32	13.32
			12	15.05	13.32	13.32
		6	6	15.03	13.28	13.28
			12	15.03	13.28	13.28
2	dry	3	6	13.27	11.84	11.84
			12	13.27	11.84	11.84
		6	6	13.27	11.84	11.84
			12	13.27	11.84	11.84
	wet	3	6	15.65	13.90	13.90
			12	15.49	13.72	13.72
		6	6	15.65	13.90	13.90
			12	15.49	13.72	13.72

Table 5.1: Time to complete each one of the 5 laps with different choices of quantization and road conditions for track in Figure 5.6.

Table 5.2 shows the effect of different track parameters on the time to complete the lap. On a track with smaller dimension, increasing d_{w} or d_v reduces the time to complete the lap. However, increasing the dimension of the quantization increases the computation time of the simulation. There are a few simulations shown in Table 5.2 where the time to complete the last lap is the smallest among all laps. This is due to the fact that in the last lap the car is able to reach a higher final velocity. This velocity is not available in the previous laps because it will result in a configuration with no possible next maneuvers.

Finally, from both tables we can observe that there are at most three different times to complete a lap. Thus, we can distinguish the laps into three types: the initial lap, the steady

car id	road	d_{w}	d_v	lap time (s)		
				1	2 - 4	5
1	dry	3	6	9.68	8.40	8.40
			12	9.63	8.40	8.40
		6	6	9.57	8.23	8.23
			12	9.52	8.23	8.23
	wet	3	6	11.57	9.97	9.97
			12	11.43	9.93	9.86
		6	6	11.45	9.76	9.76
			12	11.24	9.66	9.63
2	dry	3	6	10.11	8.88	8.88
			12	10.02	8.83	8.83
		6	6	10.07	8.74	8.74
			12	9.88	8.56	8.56
	wet	3	6	12.26	10.87	10.77
			12	11.95	10.58	10.58
		6	6	12.08	10.65	10.55
			12	11.78	10.28	10.26

Table 5.2: Time to complete each one of the 5 laps with different choices of quantization and road conditions for track in Figure 5.6 with smaller dimension.

state lap and the final lap. The path planning algorithm produces a solution for the optimal path that reaches a steady state. Due to the space limitation, the analytic proof that the algorithm always produces a steady state solution is postponed to future work.

5.5.2 Obstacle Avoidance

For this simulation, we constructed a track that is similar to Autódromo José Carlos Pace (aka Interlagos). The track is 4305.3 m in length and 12 m wide. It consist of 14 curves (circle arcs) and 14 straight lines. The track is partitioned into 14 segments by dividing each straight lines

into two parts of equal length. Therefore each partition contains a sequence of a straight line, a curve, and another straight line. A more interesting examples is the one where we have more than one car on the track. Similarly to the previous example a graph of feasible trajectories is built for each car with the parameters given in the beginning of this section. Two optimal trajectories are produced for each graph.

Figure 5.7 shows the two sets of optimal trajectories. The blue line represents the trajectory of car 1 and the red line represents the trajectory of car 2 before applying the collision avoidance algorithm described in Section 5.3.4, while the green line represents the new optimal trajectory of car 2 with collision avoidance. The inset shows the configurations of the two cars at time t_i before and after the path planning. A more detailed view is depicted in Figure 5.8 where we took a snapshot of the cars configurations at three different times.

We applied the path planning algorithm on the two graphs and removed edges from the graph of the car at the rear if the two cars collide along the trajectories represented by the edges. The algorithm produces two new optimal paths where the two cars do not collide as shown in Figure 5.8.

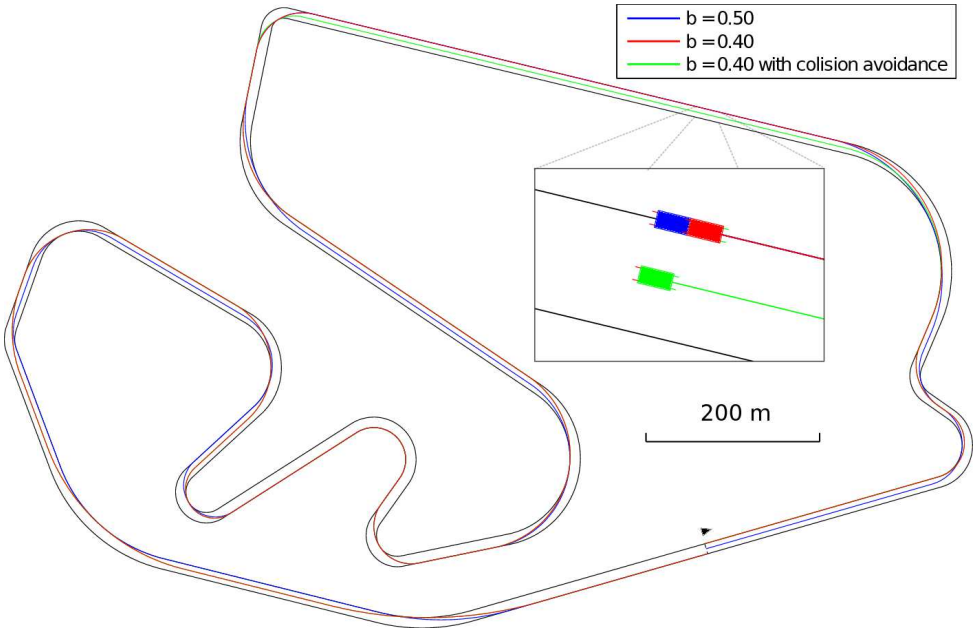


Figure 5.7: Trajectories of two cars, on the Interlagos circuit, with different viscous friction b parameters before graph pruning (red) and after pruning (green)

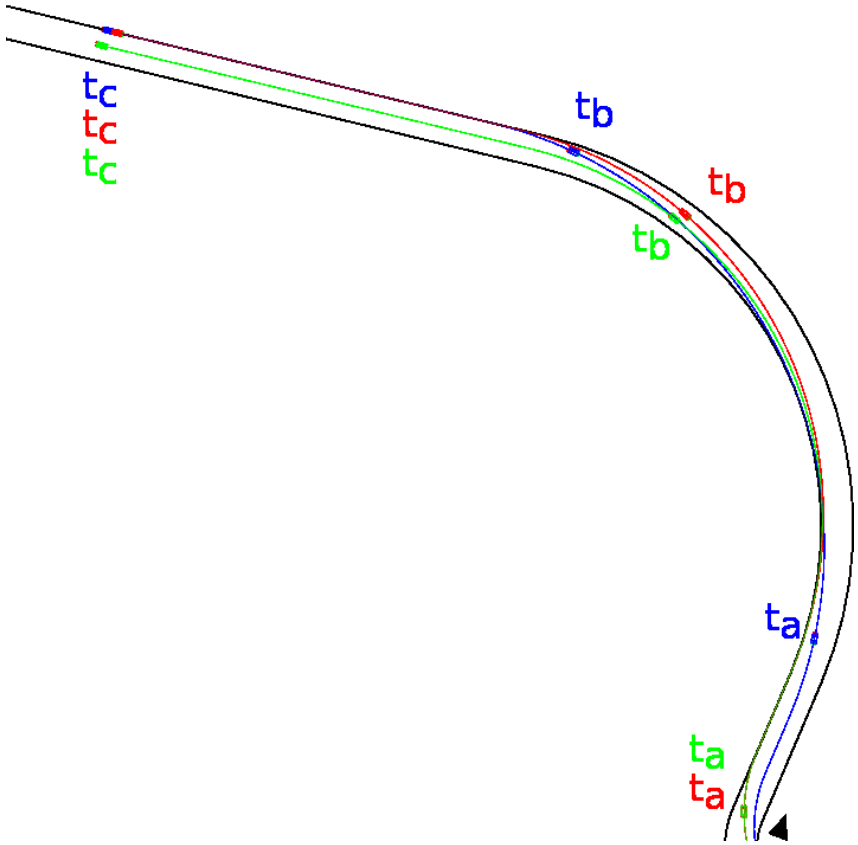


Figure 5.8: Particular of the trajectories of two cars, on the Interlagos circuit, with different viscous friction b parameters before graph pruning (red) and after pruning (green)

Chapter 6

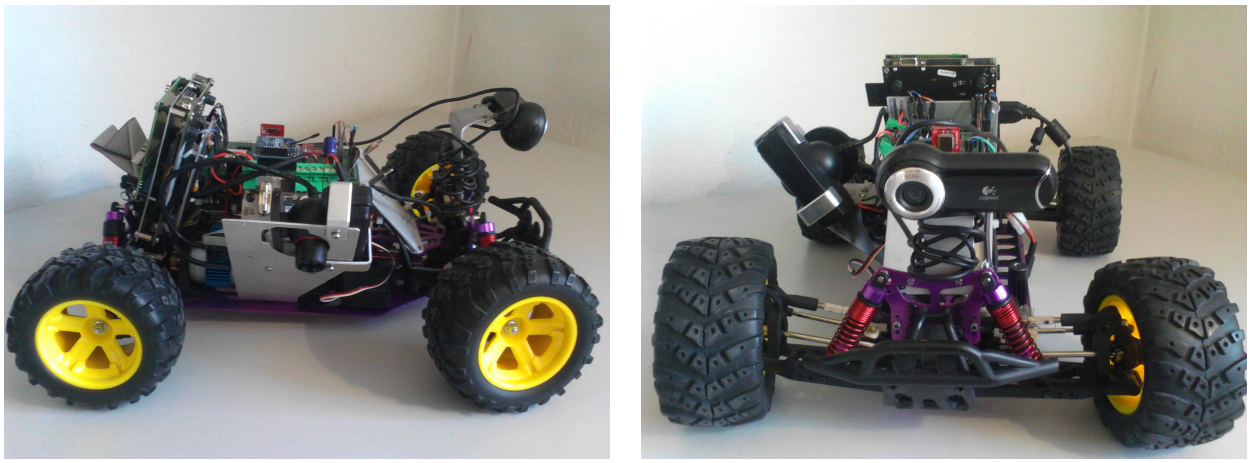
Architecture Design and Implementation

An autonomous car benefits from an efficient real-time implementation of the sensing and planning algorithms. As an example, the vision based localization use the side camera in order to produce the output that will be used in the lateral control. Consequently, the camera is required to capture new frame within the sampling period of of the control task. Moreover, the processing units need to able to guarantee that all of the tasks required for the control tasks meet their respective deadlines. This motivate a hardware and software architectures that are able to satisfy these real-time requirements.

As a proof of concept we build a high performance robotic system using low cost hardware as a testbed of our algorithm (shown in Figure 6.1) [23].

The starting point was a 1/8 scale radio controlled competition car model. The model is a 4WD buggy car and is powered by an electric DC motor that thrusts it up to a maximum speed of about 25 km/h, which makes driving demanding for an unexperienced user on a scaled down track. Indeed, if the vehicle runs on a 1/8 scale track and sensors are placed very close to the ground (as per obvious physical constraints), the time constants required for kinematic control are comparable to those in a vehicle of natural size running at $200Km/h$ on a real track (clearly, the same does not apply to control of the system dynamics, which is outside of the scope of this thesis).

As a first step, we have removed the original electronic components used for remote driving, using the bare mechanical components of the vehicle and the driver for the motor, which is



a) b)

Figure 6.1: The robotic car platform: a) side view, b) front view

controlled by a PWM signal digital signal. We have also modified the chassis to accommodate the controller, which consists of a set of computing boards and of sensors.

In our second step, we have selected an appropriate set of sensors to be used for the control tasks. Our sensing scheme is organized in two layers. The lower layer contains such sensors as encoders, gyros and accelerometers which are used to estimate speed and accelerations. The application of these is to control the vehicle in the execution of specific set of maneuvers (accelerate up to a desired position, turn of a specified angle). The higher layer is used to perform high level tasks such as to localize the vehicle in the environment, to reconstruct the path from the image and to perform the global planning algorithm. For vision-based localization and path reconstruction we have selected two visual sensors, the first one facing headway and the second one sideways. This sensing scheme naturally induces an architecture design organized in two layers: the lower one is operated by a simple computing elements, which is easy to interface with the power electronic components and executes simple operations reliably and with a high rate. The higher layer requires more computation power and a sophisticated software infrastructure (to integrate legacy software components used for the vision algorithms), but operates on longer time scales.

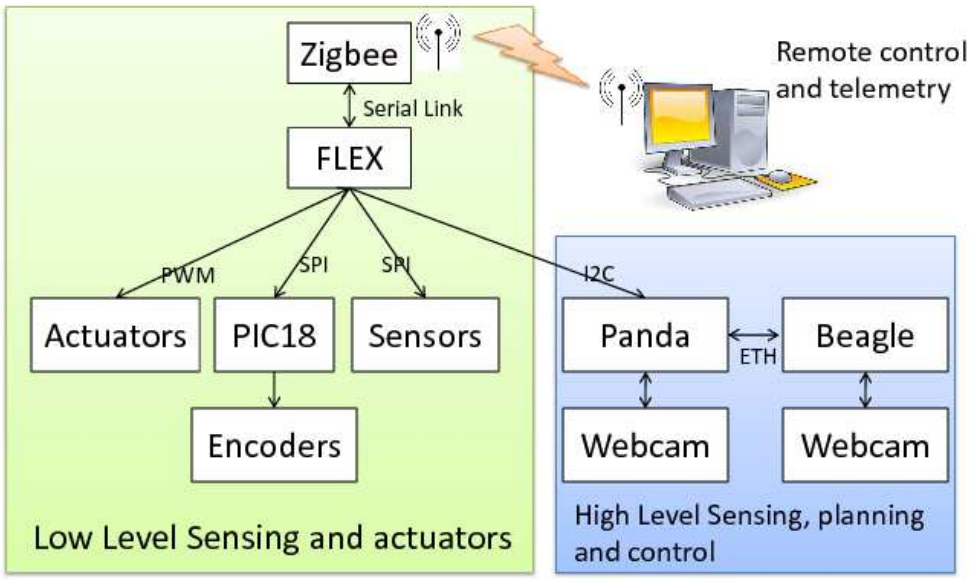


Figure 6.2: Architectural scheme of the robotic vehicle

6.1 Hardware Architecture

The scheme of the hardware architecture is shown in Figure 6.2. It consist of a collection of sensors, actuators, communication channels and 4 computation units. The low level task are managed by the Flex board and the PIC18 microcontroller. While the high level task are executed on the Pandaboard and the Beagleboard.

The Flex board Board[1] is a development board which is based on a microchip dsPIC33 controller. It is a 16-bit architecture with 40 MIPS cpu speed. The board can be power supplied with a variety of possible voltages (in the range $9 - 36$V) and it exposes a set of connectors that can be used to piggyback expansion boards. For this project, we have developed an ex-

[1]The Flex board is produced and sold by Evidence S.R.L.: `http://www.evidence.eu.com/node/55`.

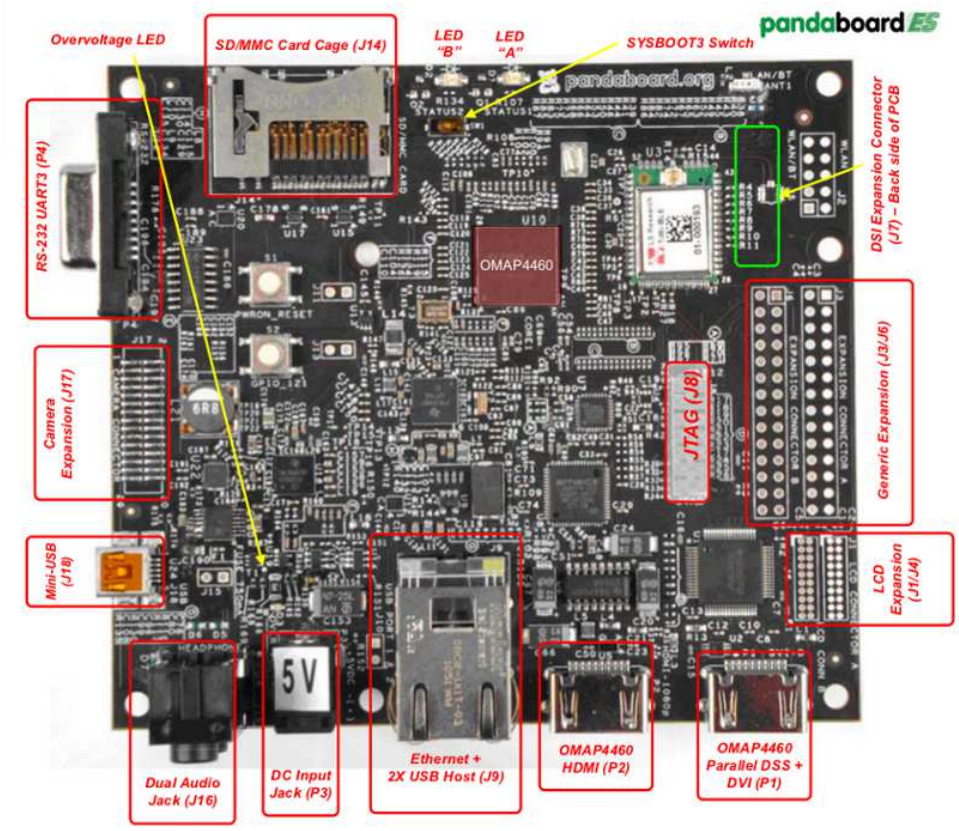


Figure 6.3: Pandaboard dual-core development board. Courtesy of http://pandaboard.org

pansion board that integrates several peripherals: accelerometers, gyros, encoders (controlled by a dedicated PIC18 micro-controller) and a Digi xBee Pro Zigbee module[2] for wireless communication, which is used for remote emergency control and telemetry. These components are connected to the Flex Board using different serial channels (UART and SPI). The Flex Board is also connected to the motor driver and to the steering motor by means of its PWM channels.

The high level sensing, planning and control functionalities are implemented a Pandaboard[3] and a BeagleBoards XM[4]. The Pandaboard (Figure 6.3) is a low cost development platform based on the Texas Instruments OMAP4460 system on a chip (SoC). It features a dual-core 1.2 Ghz Cortex-A9 CPU, a 384 MHz PowerVR SGX540 GPU, 1 GiB of DDR2 SDRAM, a collection of peripherals and network ports. It supports linux operating system installed on an

[2] `http://www.digi.com/`

[3] `http://pandaboard.org/`

[4] `http://www.beagleboard.org`

Figure 6.4: Beagleboard XM single board computer. Courtesy of http://beagleboard.org

SD card as the primary persistant storage. Its size and computing power makes it a suitable processing unit for the localization algorithm.

The Pandaboard is connected to a PlayStation Eye, which is a commercial camera from Sony. The camera is connected via a USB connection. It is able to capture an image with a resolution of 640x480 pixels at 60 frames per second. However, it can be configure to capture at a lower resolution 320x240 pixels at a higher frame rate of 120 frames per second. It is easily available and affordable makes the camera interesting for high speed vision algorithm. The PlayStation Eye camera is mounted on the side of the robotic car and is mainly used for the localization algorithm. The high frame rate of the camera (120 fps) enables a lateral control task to be executed at most every 8.33 second.

The Beagleboard XM is the second generation of Beagleboard. It is a single board computer from Texas Instrument with a single DM3730 system on a chip (SOC). It features a single-core 1 Ghz Cortex-A8 CPU, TMS320C64x+ core and is equipped with 512MB of low power DDR RAM. The Beagleboard is connected to Logitech Pro 9000 camera, which is a high resolution

camera which supports 720p resolution. This camera is also connected via a USB connection to the Beagleboard. It provides a high resolution image for the path reconstruction algorithm.

6.2 Software Architecture

The different software functionalities were distributed as follows between the different boards. Overall, the solution proposed optimizes the efficiency in utilizing the available resources, in the face of other solutions such as the ROS execution environment [45]. The price to pay is the development software development effort requested by the direct manipulation of low level mechanisms.

An overview of the software architecture is shown in Figure 6.5.

6.2.1 Flex board

The Flex board is running OSEK compliant Erika kernel [5]. The Erika kernel is suitable for real–time programming, for it features: 1) a clear tasking scheme, 2) fixed and dynamic priority scheduling, 3) real–time resource sharing using the Priority Inheritance Protocol [55], 4) support for the most common serial buses, which makes for an easy integration of new peripherals. There are 6 real time tasks running on the Flex board as shown in Figure 6.5. The tasks are :

1. **Sensor Update**: this task integrates all the information coming from the sensors connected via the two SPI buses. The sensors are: accelerometers, gyros and encoders. This task is synchronous and executed every 150 ms.

2. **Zigbee Send**: this task collects all sensors data and other information required for debugging purposes. The information is encoded and packed as a communication package to be transmitted to a PC via Zigbee protocol (IEEE 802.15.4). The Zigbee uses a serial communication channel that is configured to 9600 bps. The Zigbee Send task is synchronous and executed every 1000 ms.

[5] `http://erika.tuxfamily.org/`

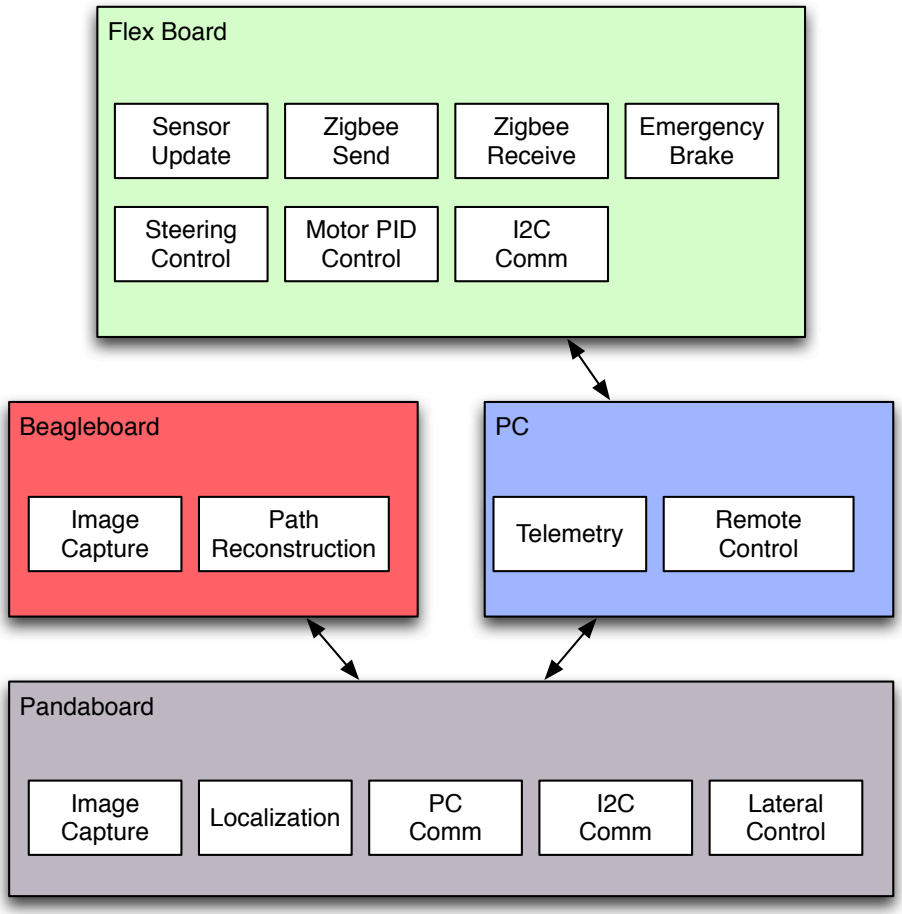


Figure 6.5: Overview of software architecture

3. **Zigbee Receive**: this task handle the communication from the PC. This task receives command packets from the PC via Zigbee, interprets and execute the command. This task is mainly used to start / stop the controllers. This task is asynchronous and is activated when a packet arrives from the PC.

4. **Emergency Brake**: It is a simple safety feature that stops the robotic car in the case of communication error between the boards. Emergency Brake task is asynchronous and is activated when within 100 ms no packets is received from the PC via Zigbee of the Pandaboard via I2C.

5. **Steering Control**: This task reads values from the potentiometer connected to the wheels, which measures the steering angle. This task also performs a PI control to control the steering angle. This task is executed every 1 ms.

6. **Motor PID Control**: This task implements a PID controller to control the speed of the DC motor and is executed every 2 ms.

7. **I2C Comm**: This is an asynchronous task that activates when a packet arrives in the I2C bus from the Pandaboard. The packet contains the set point for the velocity and steering and an output for the lateral control running on the Pandaboard.

6.2.2 Pandaboard

The Pandaboard is running Linux operating system with kernel 2.6.32. The main functions running on the Pandaboard are:

1. **Image Capture**: This function captures an image from the PlayStation Eye camera at 100 frames per second and performs the preprocessing algorithm (crop and edge detection)

2. **Localization**: This function receives the image and performs the localization algorithm as described in Section 3.3.

3. **Lateral Control**: This function receives the output of the localization function and computes the velocity and steering set point.

4. **PC Comm**: This function is a publisher of a pub-sub communication channel between the Pandaboard and the PC. It sends debugging data to the PC and receives commands to start or stop the localization algorithm.

5. **I2C Comm**: This function handles the communication between the Pandaboard and the Flex board.

6.2.3 PC

The PC software is built to do basic control of the robotic car and to show the telemetry data from the robotic car. The PC software is able to directly communicates to the Flex board via Zigbee and to the Pandaboard via WLAN (IEEE 802.11).

Chapter 7

Conclusion

In this thesis we have described problems encountered in designing an autonomous car in the context of a car race. We also provide solutions for the uncertainty in sensing by presenting a vision based localization and and path reconstruction techniques. These techniques are based on RANSAC and Kalman filter which enables an efficient real-time implementation. We also presented our solution for local planning problem for a robotic car racing on a track. The problem of local planning has been considered first identifying the optimality of the maneuver and then giving a complete geometric analysis of the simplified scenario. In this case, sub-optimal maneuvers show to be largely dependent on the problem parameters: the weights of the time to complete the maneuver and the velocity at the end of the track as well as the dynamic properties of the vehicle completely change the maneuvers. Hence, the sub-optimal solution is given in terms of a set of functions to be minimized once the parameters of the problem are known. In this thesis, we also have proposed a graph-based global planning technique. By using a discrete abstraction, we are able to generate a motion plan that optimizes the completion time of the race in a short time, both when the car runs in isolation and when it has to overtake a slower car. A robotic platform has been developed as a testbed for the localization and the path reconstruction algorithm. The robotic car is able to perform a simple path following algorithm using the localization algorithm.

Future development will explicitly consider generic initial orientation and approaching maneuvers to the curve. On of the future direction is to implement advanced game theoretic strate-

gies that allows the car to overtake other opposing cars. Another important area for future work will be on the implementation of the global planning on the presented robotic vehicle.

Bibliography

[1] A.P. Aguiar and J.P. Hespanha. Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty. *IEEE Trans. on Automatic Control*, 52(8):1362–1379, August 2007.

[2] J.C. Alexander, J.H. Maddocks, and B.A. Michalowski. Shortest distance paths for wheeled mobile robots. *Robotics and Automation, IEEE Transactions on*, 14(5):657 – 662, oct 1998. ISSN 1042-296X. doi: 10.1109/70.720342.

[3] R. Asaula, D. Fontanelli, and L. Palopoli. Safety provisions for human/robot interactions using stochastic discrete abstractions. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2175–2180, 2010.

[4] D. Balkcom and M. Mason. Time-optimal trajectories for an omnidirectional vehicle. *The International Journal of Robotics Research*, 25(10):985–999, 2006.

[5] Jérôme Barraquand, Lydia Kavraki, Rajeev Motwani, Jean-Claude Latombe, Tsai-Yen Li, and Prabhakar Raghavan. A random sampling scheme for path planning. In *Robotics Research*, pages 249–264. Springer, 2000.

[6] Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288, 1991.

[7] Amol Borkar, Monson Hayes, and Mark Smith. A Template Matching and Ellipse Modeling Approach to Detecting Lane Markers. In Jacques Blanc-Talon, Don Bone, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelli-*

gent Vision Systems, volume 6475 of *Lecture Notes in Computer Science*, pages 179–190. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-17690-6.

[8] M. Botta, V. Gautieri, D. Loiacono, and P.L. Lanzi. Evolving the optimal racing line in a high–end racing game. In *Computational Intelligence and Games (CIG), 2012 IEEE Symposium on*, pages 108–115, 2012.

[9] A.E. Bryson and Y.C. Ho. *Applied optimal control*. Wiley New York, 1975.

[10] Martin Buehler, Karl Iagnemma, and Sanjiv Singh, editors. *The 2005 DARPA Grand Challenge - The Great Robot Race*, volume 36 of *Springer Tracts in Advanced Robotics*. Springer Berlin / Heidelberg, 2007.

[11] John Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679 –698, Nov. 1986. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851.

[12] John Canny. *The complexity of robot motion planning*. The MIT press, 1988.

[13] D. Casanova, R.S. Sharp, and P. Symonds. Minimum time manoeuvring: The significance of yaw inertia. *Vehicle System Dynamics*, 34(2):77–115, 2000. doi: 10.1076/0042-3114(200008)34:2;1-G;FT077.

[14] Y.C. Cheng and Y.-S. Liu. Polling an image for circles by random lines. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(1):126 –131, Jan. 2003.

[15] Yu Chin Cheng. The distinctiveness of a curve in a parameterized neighborhood: extraction and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(8):1215–1222, Aug. 2006.

[16] H. Chitsaz, S. M. LaValle, D. J. Balkcom, and M.T. Mason. Minimum wheel-rotation for differential-drive mobile robots. *The International Journal of Robotics Research*, pages 66–80, 2009.

[17] Howie Choset, Sean Walker, Kunnayut Eiamsa-Ard, and Joel Burdick. Sensor-based exploration: incremental construction of the hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, 19(2):126–148, 2000.

[18] M. Chyba and S. Sekhavat. Time optimal paths for a mobile robot with one trailer. In *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 3, pages 1669 –1674 vol.3, 1999. doi: 10.1109/IROS.1999.811718.

[19] Christopher I Connolly, JB Burns, and R Weiss. Path planning using laplace's equation. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 2102–2106. IEEE, 1990.

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

[21] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, pages 457–516, 1957.

[22] D. Fontanelli, M. Cappelletti, and D. Macii. A RANSAC-based fast road line detection system for high-speed wheeled vehicles. In *IEEE Int. Instrumentation and Measurement Technology Conference (I2MTC)*, pages 186–191, Hang Zhou, China, May 2011.

[23] Daniele Fontanelli, Luigi Palopoli, and Tizar Rizano. High speed robotics with low cost hardware. In *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*, pages 1–8. IEEE, 2012.

[24] A. Girard and G.J. Pappas. Approximate bisimulation relations for constrained linear systems. *Automatica*, 43(8):1307–1317, 2007.

[25] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[26] Gabriel M Hoffmann, Claire J Tomlin, Michael Montemerlo, and Sebastian Thrun. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In *American Control Conference, 2007. ACC'07*, pages 2296–2301. IEEE, 2007.

[27] Martin Buehler Issue edited by: Karl Iagnemma. Special issue: Special issue on the darpa grand challenge, part 1. *Journal of Field Robotic*, 23(8):461652, 2006.

[28] Sanjiv Singh Issue edited by: Martin Buehler, Karl Lagnemma. Special issue: Special issue on the 2007 darpa urban challenge, part i. *Journal of Field Robotic*, 25(8):423566, 2008.

[29] Lydia Kavraki and J-C Latombe. Randomized preprocessing of configuration for fast path planning. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 2138–2145. IEEE, 1994.

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

[31] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500–505. IEEE, 1985.

[32] D. Kogan and R.M. Murray. Optimization-based navigation for the darpa grand challenge. In *Conference on Decision and Control (CDC)*, 2006.

[33] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, JP How, and G. Fiore. Real-time motion planning with applications to autonomous urban driving. *Control Systems Technology, IEEE Transactions on*, 17(5):1105–1118, 2009.

[34] J.P. Laumond. *Robot motion planning and control*. Springer, 1998.

[35] Steven M LaValle. Rapidly-exploring random trees a ew tool for path planning. 1998.

[36] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[37] Zhi-Yong Liu and Hong Qiao. Multiple ellipses detection in noisy environments: A hierarchical approach. *Pattern Recognition*, 42(11):2421–2433, 2009.

[38] Tomás Lozano-Pérez and Michael A Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[39] H. A. Mallot, H. H. Bülthoff, J. J. Little, and Boher S. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Springer-Verlag Biological Cybernetics*, 64(3):177–185, 1991.

[40] S. Pancanti, L. Pallottino, D. Salvadorini, and A. Bicchi. Motion planning through symbols and lattices. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3914–3919. IEEE, 2004.

[41] Roberto Passerone, Jerry R. Burch, and Alberto L. Sangiovanni-Vincentelli. Refinement preserving approximations for the design and verification of heterogeneous systems. *Formal Methods in System Design*, 31(1):1–33, August 2007.

[42] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[43] L.S. Pontryagin, VG Boltyanskii, RV Gamkrelidze, and EF Mishchenko. *The mathematical theory of optimal processes*. Interscience Publishers New York, 1962.

[44] M. Prandini, J. Hu, J. Lygeros, and S. Sastry. A probabilistic approach to aircraft conflict detection. *Intelligent Transportation Systems, IEEE Transactions on*, 1(4):199–220, 2000.

[45] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, 2009.

[46] Rajesh Rajamani. *Vehicle dynamics and control*. Springer, 2011.

[47] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, pages 367–393, 1990.

[48] David B. Reister and Franois G. Pin. Time-optimal trajectories for mobile robots with two independently driven wheels. *The International Journal of Robotics Research*, 13(1): 38–54, 1994. doi: 10.1177/027836499401300103.

[49] J Richalet, A Rault, JL Testud, and J Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, 1978.

[50] T. Rizano, D. Fontanelli, L. Palopoli, L. Pallottino, and P. Salaris. Global path planning for competitive robotic cars. In *Conference on Decision and Control (CDC)*, 2013. URL `http://www.centropiaggio.unipi.it/sites/default/files/CDC13-GlobalPlanni` Submitted.

[51] T. Rizano, D. Fontanelli, L. Palopoli, L. Pallottino, and P. Salaris. Local motion planning for robotic race cars. In *Conference on Decision and Control (CDC)*, 2013. URL `http://www.centropiaggio.unipi.it/sites/default/files/CDC13-LocalPlannir` Submitted.

[52] P. Salaris, D. Fontanelli, L. Pallottino, and A. Bicchi. Shortest paths for a robot with nonholonomic and field-of-view constraints. *IEEE Transactions on Robotics*, 26(2):269–281, April 2010.

[53] P. Salaris, L. Pallottino, and A. Bicchi. Shortest paths for finned, winged, legged, and wheeled vehicles with side-looking sensors. *The International Journal of Robotics Research*, 31(8):997–1017, 2012.

[54] C. Samson and K. Ait-Abderrahim. Feedback control of a nonholonomic wheeled cart in Cartesian space. In *IEEE Intl Conf. on Robotics and Automation*, pages 1136–1141, April 1991.

[55] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *Computers, IEEE Transactions on*, 39(9):1175–1185, 1990.

[56] D Gillespie Thomas. Fundamentals of vehicle dynamics. *Society of Automotive Engineering Inc*, pages 168–193, 1992.

[57] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006. ISSN 1556-4967. doi: 10.1002/rob.20147. URL `http://dx.doi.org/10.1002/rob.20147`.

[58] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William Red Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. ISSN 1556-4967. doi: 10.1002/rob.20255. URL `http://dx.doi.org/10.1002/rob.20255`.

[59] H. Wang, Y. Chan, and P. Souères. A geometric algorithm to compute time-optimal trajectories for a bidirectional steered robot. *IEEE Transaction on Robotics*, pages –, 2009.

[60] Wuhong Wang. A digital-driving system for smart vehicles. *Intelligent Systems, IEEE*, 17 (5):81–83, Sep. 2002.