Freie Universität Berlin

Institut für Informatik

Bachelor-Thesis

A path following control architecture for autonomous vehicles

---

by: Daniel Krakowczyk

Supervisors: Prof. Dr. Raúl Rojas
Dr. Daniel Göhring

handed on: Wednesday 15th October, 2014

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, sind als solche gekennzeichnet. Die Zeichnungen oder Abbildungen sind von mir selbst erstellt worden oder mit entsprechenden Quellennachweisen versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner Prüfungsbehörde eingereicht worden.

Berlin, den 15. Oktober 2014

Daniel Krakowczyk

Abstract

After planning, adequate actions have to be taken. This thesis presents a control architecture for an autonomous model car to achieve a stable and accurate path following behavior. This encompasses a steering controller to drive desired curvatures computed by a pure pursuit algorithm. Additionally, an open-loop controller was implemented to determine a safe speed for driving curves and stopping on the path's end.

Contents

List of Figures

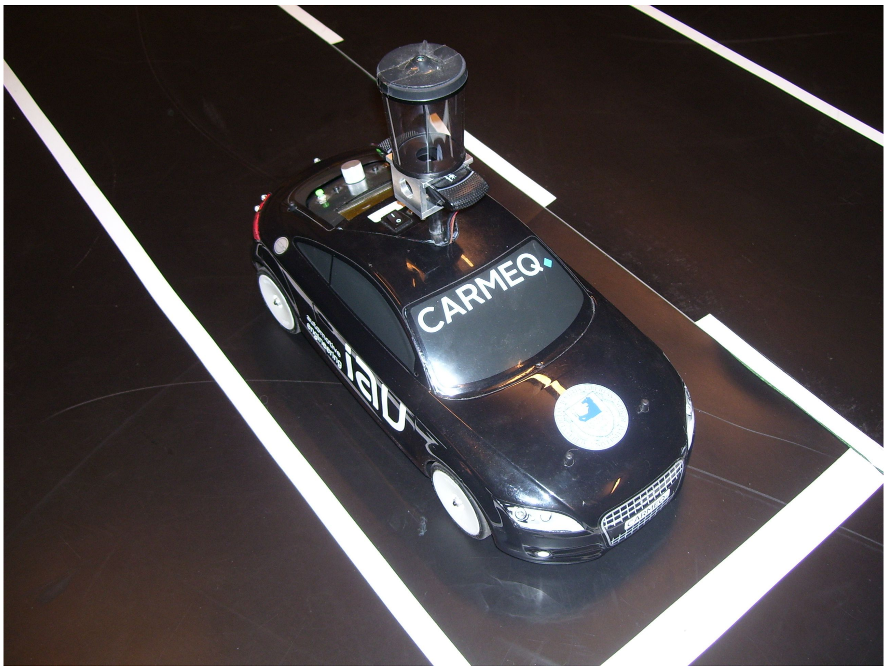


Figure 1.1: Model sized car named Able on laboratory track

1 Introduction

Small sized model cars pose similar challenges to developers as the larger equivalent. The environment has to be sensed, a goal has to planned dependent on defined rules and those plans have to be ultimately taken care of by a control strategy. This thesis' focus is on presenting a simple strategy for path following designed for autonomous car-like vehicles. The environment is in that case limited to a plain 2-dimensional workspace free of obstacles, shifting the field of obstacle avoidance to the path planning domain.

1.1 Specification

1.1.1 Functional requirements

Follow path A planned path is to be executed on the track by setting velocity and steering controls adequately.

Stop on path's end Furthermore the vehicle has to stop autonomously on the end of the path.

1.1.2 Design goals

Stability A stable controller is the primary objective in controller design. This means particularly, that it mustn't cause oscillations of actuators or state.

Accuracy The controller has to be accurate in path following. Clearly this is velocity dependent: when driving on the track with high velocity without obstacles, accuracy has not that high priority as it is while parallel parking with considerably lower velocity.

Safety Safety can be considered at different levels. First of all, this control architecture can not be hold responsible for avoiding harm to humans or obeying the traffic rules, as this belongs to the path planning domain. Nevertheless, it has to be able to react to sudden changes, like lane changing maneuvers due to late recognized obstacles, if it is ordered to.

Thus safety can in this scope only mean, to protect the cars own existence. This results in the requirement of a sane driving behavior with deceleration before curves that avoids uncontrollable states caused for example by tire-slip and lateral forces. Besides, this also implies that the vehicle is not allowed to move at all if no plan is evaluated.

Robustness It is not guaranteed that environmental parameters like the track's ground or the vehicle's mechanical configuration stay constant, hence the controller has to be sufficiently robust to overcome moderate parameter changes.

Response time This is a real-time application that is aimed to drive with high velocities, therefore the system's response has to be as fast as possible.

Maintenance Besides those apparent goals, maintenance issues cannot be disregarded. As this is primarily a research project under development, modifiability of the architecture mustn't be disrespected. Due to the team's varying members, the readability of code has to be ensured by providing a comprehensible documentation.

1.2 Document structure

Following this introduction, the preliminaries will be presented beginning with the underlying hardware and software platform. In search of knowledge about the vehicle's motion, a kinematic model will be introduced after taking account of its motion constraints. The preliminaries close by demonstrating the given path representation.

Moving on to the control domain, fundamentals to controller design will be laid out by outlining paradigms. Provided with this knowledge, the design and implementation of the control architecture and the particular controllers will be proposed.

After an evaluation of the driving performance on a test track, results are being discussed and future enhancements proposed.

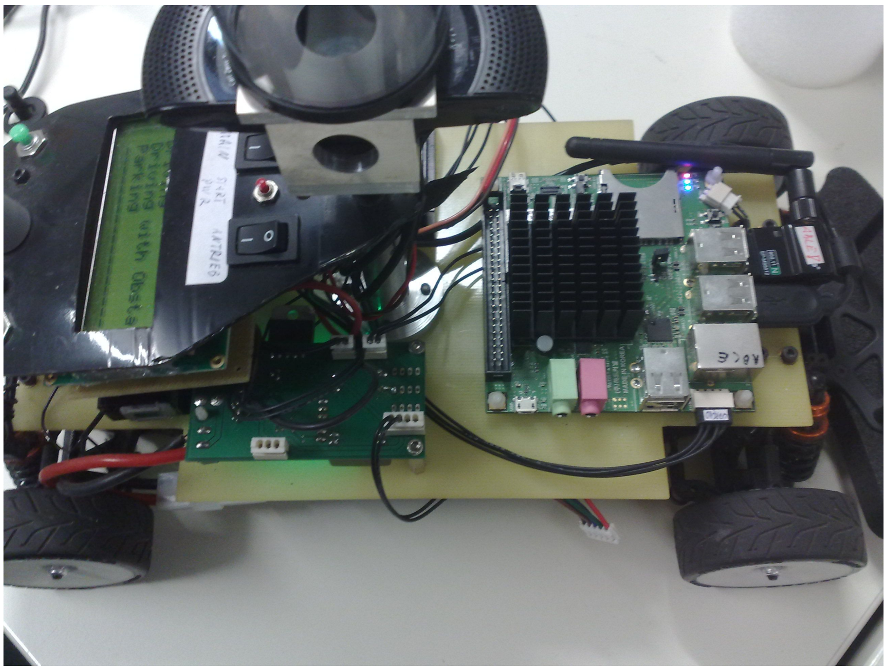


Figure 2.1: Model car without body

2 Preliminaries

2.1 Platform

2.1.1 Hardware

Several components are needed to build an autonomous driving car. As a detailed description of the current hardware is presented in [4], only short descriptions of components fundamental to the controller design are given.

Chassis The first to mention is the carbon-chassis *Xray T3 Spec. 2012*. The track width L_w is given by 186 mm, the gauge L_l by 263 mm. The chassis features an adjustable steering geometry, gear differential and springing.

Steering Servo The steering servo *Savöx SC1251MG Digital* is controlled by a PPM-signal. Unfortunately, no feedback channel is given, thus relying deeply on the integrated servo control to regulate the steering angle.

Engine The engine is the brushless DC motor *Faulhaber 3056 012 B-K1155* which is controlled by the *Faulhaber MCBL 3006 S RS* to maintain a desired velocity utilizing an internal PID-controller with bounded acceleration. The actual integration via RS-232 interface is laid out in [20].

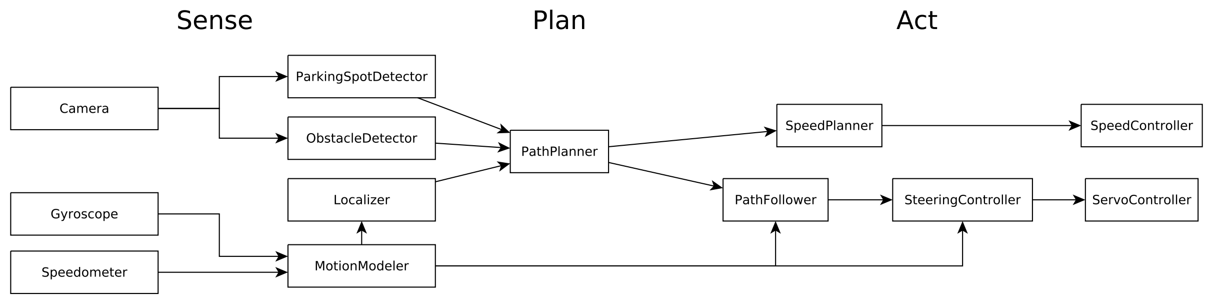


Figure 2.2: Software architecture

Gyroscope There is a 9DoF inertial measurement unit integrated on the microcontroller board. However, only the gyroscope data is handled. [5] Unfortunately, the measurement is subject to drift and noise which is not enough investigated at that time.

Omnidirectional camera A combination of a parabolic lens and a standard web-cam was used to get a 360°-view. The camera has a framerate of 30Hz. The downside is, that its reliable data is limited to distances around 3 m. To overcome this, self-localization is utilized to plan paths with more anticipation.

Micro-controller To communicate with actuators and the IMU, the in-house-developed micro-controller board OttoV2 is used. It is a *32bit ARM-Cortex-4* operated at a frequency of 168 MHz.

Host computer The responsibility of the host computer is computer vision, cognition and control. For this use an *Odroid-X2* with an *1.7 GHz ARM Cortex A9 Quad Core processor* was equipped.

2.1.2 Software

BerlinUnited-Framework The underlying BerlinUnited-Framework was developed at Berlin's Universities *Freie Universität* and *Humboldt-Universität* for different robotic projects (FUmanoids, NeuroCopter, NaoTH, BerlinUnited Racing-Team). This framework is a blackboard architecture implemented with the classical sense-plan-act paradigm in mind. The source code is partitioned in modules and representations. Modules implement computing blocks and include a declaration of required and provided representations to interface with. As cycles are prohibited in this data-flow, a recycle declaration provides respective data from the last execution cycle.

In the current state, only one single execution chain is used which is synchronized with the camera frame-rate and therefore every module is re-executed at 30 *Hz*.

Software-Architecture Consequently the high-level software architecture can be represented as a sensor-, plan-, and control-part. Figure 2.2 illustrates the data-flow without considering the actual communication implementation.

Sense The sensing of the environment provides a motion model by interpreting inertial measurements, detecting obstacles and parking spots by computer vision and eventually localizing itself on the track. [19]

Plan The behaviors are currently limited to loading static paths and forwarding them in car-local coordinates.

Act Finally, the planned path has to be executed on the track. The design of this subsystem is the focus of the underlying thesis.

2.2 Vehicle model

To design a well-suited control law, it is advisable to model the controlled system. This subsection introduces a basic kinematic model after taking account of kinematic constraints that affect the possible motions of car-like vehicles. Finally, friction is introduced as a limiting factor in acceleration and velocity.

2.2.1 Configuration space

A general nonlinear system with the n-dimensional state vector $q(t)$ and the m-dimensional control vector $u(t)$ can be described as a differential equation

$$\dot{q}(t) = f(\, q(t),\, u(t),\, t) \tag{2.1}$$

In the case of a linear time-invariant system, the evolution of the system state can be described as

$$\dot{q}(t) = A\, q(t) + B\, u(t) \tag{2.2}$$

with constant matrices A and B. [3, pp. 83] The robot's configuration space q is usually defined by its position and orientation with added steering angle. This simplifies several real world phenomena. Since the vehicle is assumed to drive on a plane track, the 3rd dimension is completely ignored which reduces the vehicles position two a 2-dimensional vector (x, y) and the orientation to a single heading angle θ. The position is defined as the midpoint of the rear axe, the orientation θ is the angle between the x-axis and the heading of the car. The steering angle ϕ is defined as the heading of the front wheels respective to θ. For the sake of simplicity, this thesis is restricted to a single-track bicycle-model illustrated in figure 2.3, which implies that the wheels of both axes collapse

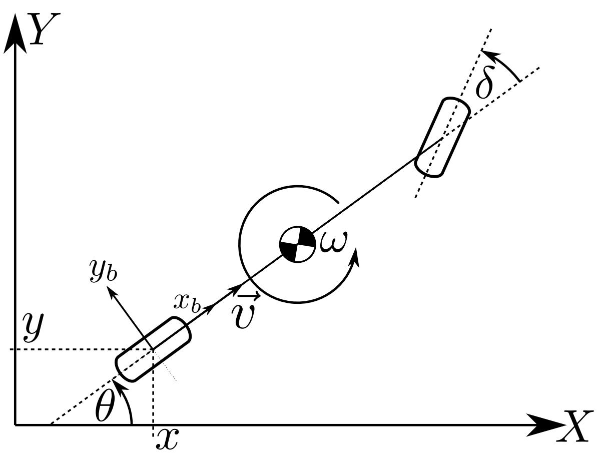


Figure 2.3: Bicycle model

on the midpoint of the respective axe. Joining this data, we get the 4-dimensional state vector

$$q(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \\ \phi(t) \end{bmatrix} \tag{2.3}$$

There are indeed more forces acting upon the car than taken into account for this configuration state. Although the car indisputably has a roll velocity while turning and a pitch velocity while accelerating, including these motions only becomes necessary if the limits of handling characteristics have to be thoroughly analyzed. [31, pp. 335] As a pure 2-dimensional world model is assumed in this thesis, those velocities will be neglected.

2.2.2 Nonholonomic constraints

The basic assumption for the presented vehicle model is, that the wheels are in a pure rolling state, hence neglecting lateral sliding. Although this won't be satisfied completely in reality, we can derive adequate basic models. This assumption implies that no side slip is allowed. By defining the Pfaffian constraint matrix $C(q)$ with

$$C(q)\dot{q} = 0 \tag{2.4}$$

these constraints can be expressed formally for a single wheel [18, pp. 180]:

$$\begin{bmatrix} sin\,\theta & -cos\,\theta & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0 \tag{2.5}$$

This formula just holds for the rear wheel. As a bicycle has a steerable front wheel specified by the coordinates (x_f, y_f), its constraints can be expressed dependant on ϕ:

$$\begin{bmatrix} sin(\theta+\phi) & -cos(\theta+\phi) & 0 \end{bmatrix} \begin{bmatrix} \dot{x_f} \\ \dot{y_f} \\ \dot{\theta} \end{bmatrix} = 0 \tag{2.6}$$

Supplied with the rigid body constraint

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} l\cos\theta \\ l\,sin\,\theta \end{bmatrix} \tag{2.7}$$

the front wheel kinematic constraint becomes by differentiation

$$\begin{bmatrix} sin(\theta+\phi) & -cos(\theta+\phi) & -l\cos\phi \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0 \tag{2.8}$$

This can be summed up in a single Pfaffian constraint matrix [18, pp. 181]:

$$C(q)\,\dot{q} = \begin{bmatrix} sin(\theta+\phi) & -cos(\theta+\phi) & -l\cos\phi & 0 \\ sin\,\theta & -\cos\theta & 0 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \\ \dot{\phi} \end{bmatrix} = 0 \tag{2.9}$$

An equality constraint of this form is called *nonholonomic*, if C is non-integrable. The proof applied to the vehicle model is left out but presented in [17, pp. 417] for the interested reader. Additionally to that, the vehicle's steering mechanism is subject to saturation which causes inequality constraints [17, pp. 409] of the form

$$C(q)\dot{q} \leq 0 \quad or \quad C(q)\dot{q} < 0 \tag{2.10}$$

As the steering angle is lower-bounded by

$$|\phi| \leq \phi_{max} \tag{2.11}$$

the curvature becomes upper-bounded by

$$c_{min} = \frac{tan(\phi_{max})}{L_b} \tag{2.12}$$

This constraint can be rewritten as

$$|\dot{\theta}| \leq \frac{|v|}{c_{min}} \tag{2.13}$$

Given these constraints, it is questionable if the vehicle can be controlled in a way to reach an arbitrary configuration q from any configuration q_0. Visually it is easy to

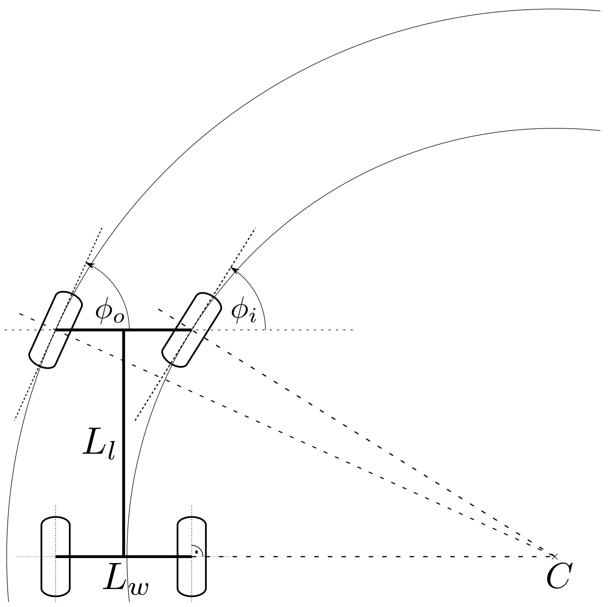


Figure 2.4: Ackermann steering geometry

imagine simple maneuvers that connect both configurations. By building a Lie Algebra L generated by the set of m vector fields this can also be proofed formally. If and only if L has a rank of m, this means L has maximal dimension, the robot is fully controllable. This is only the case for a car-like robot without upper-bounded curvature constraint. Apart from that, the system under design would be fully controllable, but is in fact not. As this is more of interest in the path planning domain, the derivation and exemplification is omitted and the reader is referred to [17, pp. 420] [18, pp. 183]. Anyway, the conclusion is that effective path planning methods are required to reach any configuration.

2.2.3 Steering model

As the used vehicle has two steerable front wheels, the orientation of those wheels has to be determined by the turning point C to hold the pure rolling assumption while turning. Figure 2.4 illustrates that while turning around C, the radius of the inner and outer wheels have to be in a certain relationship, as a wheel can only be in a pure rolling state if its heading is perpendicular to the vector pointing to C. This concludes in following relationship between the inside angle ϕ_i and the outside angle ϕ_o

$$cot\,\phi_o - cot\,\phi_i = \frac{L_w}{L_l} \tag{2.14}$$

This steering geometry is usually called *Ackermann steering*. This simple relation between the motion of the vehicle and its steering angle holds at low speeds. [31, pp. 336] Nevertheless, a single-track model with a single steering angle ϕ was chosen due to simplicity. A mapping from single-track steering angle to double-track steering angle is layed out in [6, pp. 64].

2.2.4 Kinematic model

The issue of a kinematic model is to determine the motion of a body without taking account of the forces that act on it. Therefore, an expression for $\dot{q}$ is needed. Derived from the kinematic constraints, the translational velocities are defined as

$$v_x = V\,cos\,\theta \tag{2.15}$$

$$v_y = V\,sin\,\theta \tag{2.16}$$

with V being the summed velocity of the rear axe.

$$V = v_x + v_y \tag{2.17}$$

Assuming that the vehicle follows an arc-shaped path, its rotation is described by

$$\omega = V\,\frac{tan\,\phi}{L_b} \tag{2.18}$$

By neglecting slip angles, the rear axe's velocity is approximated as the summed velocity V. The kinematic model can be stated dependant on its control inputs [6, p. 62]

$$\dot{q}(t) = \begin{bmatrix} v_x(t) \\ v_y(t) \\ \omega(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} cos\,\theta & 0 \\ sin\,\theta & 0 \\ \frac{tan\,\phi}{L_b} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \dot{\phi}(t) \end{bmatrix} \tag{2.19}$$

There is a singularity at $\phi = \pm\pi/2$, but as the steering angle is upper-bounded to a degree less than $\pi/2$, that singularity becomes in our case less important. [18, pp. 181]

2.2.5 Friction

As our model depends on the assumption that no tire slip takes place, a maximum speed for certain curvatures has to be considered. [26, pp. 10] presents the physical foundations. A single tire's maximum acceleration is bounded by it's friction coefficient $\mu \leq 1$

$$a_{max} = \mu\,g \tag{2.20}$$

where g is the approximately constant gravitational acceleration. The maximum velocity v to drive a given curvature c safely is gained by combining the definition of the centrifugal force

$$F_Z = m\,a_Z = m\,\frac{v^2}{r} \tag{2.21}$$

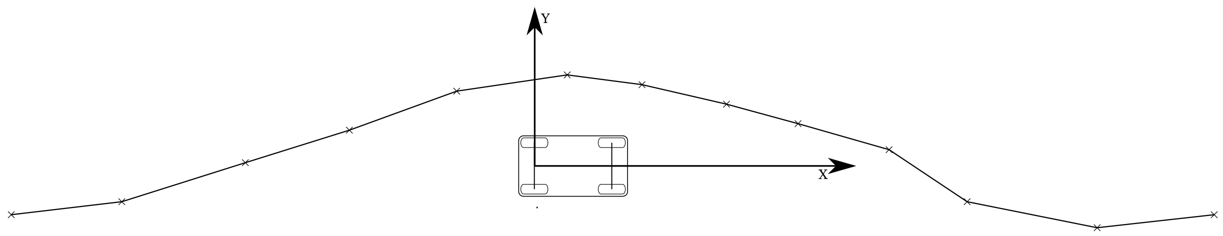


Figure 2.5: Exemplary path

with equation (2.20) to bound the centrifugal acceleration

$$v = \sqrt{\mu\, g\, r} = \sqrt{\frac{\mu\, g}{c}} \tag{2.22}$$

As the maximum friction does not bound longitudinal and lateral acceleration separately, both characteristics have to be summed up to get a correct bounded acceleration

$$a_{max} = \sqrt{(\mu\, g)^2 - (v^2\, c)} \tag{2.23}$$

This defines a circle called the *Kamm's circle* that separates states with sufficient friction inside the circle, from states outside the circle where the applied force is greater than the maximum friction $F_R = \mu\, m\, g$.

2.3 Path

This chapter presents the path and which of its parameters are taken into account.

2.3.1 Representation

The path is represented as a list of points in body frame coordinates. This simple representation suffices if it is smooth and dense enough, otherwise it will likely introduce errors.

2.3.2 Parameters

The control perspective takes following parameters into account:

Projected path-point The point on the path, which is closest to a given world coordinate.

Distance By taking the magnitude of the vector between the considered coordinates and the resultant projected path-point, the distance to the path can easily be acquired.

Path point after distance As it is reasonable to drive with anticipation, not just the closest path point is of interest, but also subsequent points in a certain distance.

Curvature As a single line segment doesn't define a curvature, the circumference of a triangle is computed. The triangle is defined by the path point of interest and both its neighboring path points.

Distance between two path points To determine in what distance the vehicle has to stop, it is essential to know the distance between two path points.

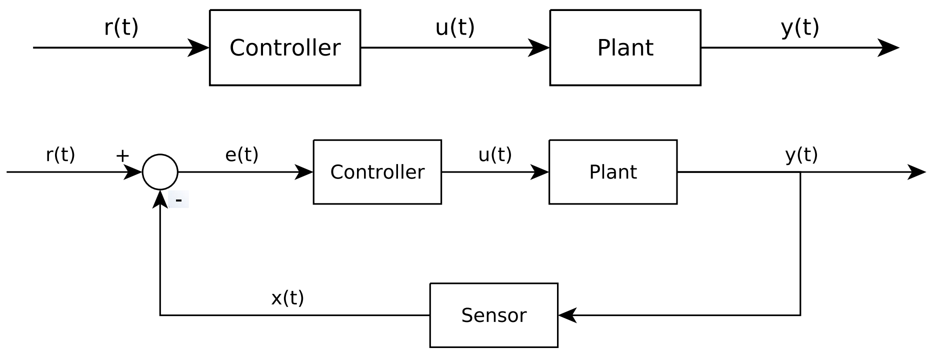


Figure 3.1: Schematic open-loop and closed-loop system

3 Controller

3.1 Paradigms and Terminology

The purpose of a controller is to regulate the output $y(t)$ of a underlying system, usually called the plant. In this application the plant is the model sized vehicle. Given the reference $r(t)$, a control law for the plant's control input $u(t)$ has to be defined to let $e(t) = r(t) - y(t)$ eventually go to zero.

3.1.1 Open-loop and closed-loop control

A simple guess to achieve a goal set by the reference is to define a function of $r(t)$. This function $f : r(t) \mapsto u(t)$ can be obtained by a model that maps the reference to the controller output. This doesn't take the actual measured state into account and thus requires a valid plant model. Therefore this approach is very sensitive to changes in plant configuration, as small changes can have drastic effects. [3, pp. 75] Apart from this, it is unlikely to model all possible disturbances that could occur. Assuming a simple constant gain, this renders following control law just depending on the reference :

$$u(t) = K_c\, r(t) \tag{3.1}$$

Taking account of the control error $e(t)$, a more robust controller can be rendered without exact model knowledge which is driven entirely by error. [15, pp. 63] As the system output is fed back into the controller, a circle in data flow is created, thus calling this approach closed-loop control. It's simplest form is a proportional gain controller:

$$u(t) = K_c(r(t) - y(t)) = K_c\, e(t) \tag{3.2}$$

The drawback is, that using feedback as a control input produces potentially unstable systems, which is not the case in open-loop controllers. [3, pp. 75]

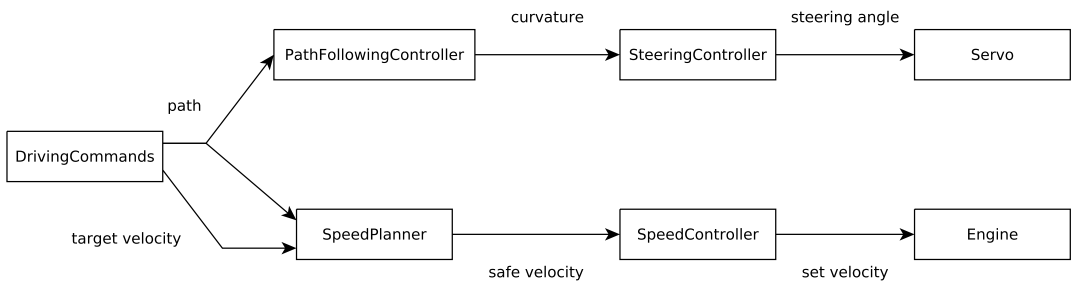


Figure 3.2: Control data-flow diagram

3.1.2 PID controller

A simple proportional gain won't suffice if the system is of higher order. This is the case if the effects of inertia or friction in a dynamic system are included. Taking the integral and difference of the error gives the advantage of taking these disturbances of higher order at least approximately into account. [3, pp. 78]

$$u(t) = K_P\, e(t) + K_I \int_0^\infty e(t)\, dt + K_D\, \dot{e(t)} \tag{3.3}$$

3.1.3 Adaptive control

The previous sections proposed time invariant control laws due to their constant parameter K_c. Dependent on the system's current state, control laws can be specified to adjust controller gains considering various operation points.

The main approach in adaptive control called *model reference adaptive control*, implements an adjustment mechanism that computes the error between the underlying reference model and the actual system input to determine control parameters. [16, pp. 313] [3, pp. 89]

3.2 Design

Starting with recalling the most important subsystems the controller interfaces with, there are two actuators, namely an engine and a steering servo, which control v and ϕ respectively. This implies the control output vector

$$u(t) = \begin{bmatrix} v(t) \\ \phi(t) \end{bmatrix} \tag{3.4}$$

The sensors output information about v given from engine control, ω from the gyroscope and pose p is computed by localization. As neither translational velocity nor the set steering angle is interpreted, just the constricted state $\dot{q}_r$ is available.

$$q(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \\ \theta(t) \\ \phi(t) \end{bmatrix} \tag{3.5}$$

$$\dot{q}_r(t) = \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3.6}$$

The underlying assumption with both equations is, that v and ϕ are set instantaneously. The planning stage releases DrivingCommands, that define the reference

$$r(t) = \begin{bmatrix} path(t) \\ s(t) \\ v_{ref}(t) \end{bmatrix} \tag{3.7}$$

Assuming that velocity and rotation can be handled separately at this stage of control, they will be handled individually. Two modules are employed to handle rotation: a SteeringController, that is responsible for holding a desired curvature $c(t)$, and a PathFollower, that employs the pure pursuit algorithm to determine curvatures to follow the reference path.

Therefore the speeds have to be chosen wisely to have negligibly small slip angles. The main purpose of the SpeedPlanner is to set velocities that guarantee to hold this precondition. As the engine controller already takes care of limiting acceleration and maintaining a desired velocity, there is just one module required that takes care of determining this velocity. This includes a position controller for $s(t)$ and a maximum curvature velocity calculator respectively.

3.2.1 PathFollower

The path follower is responsible to compute a turning curvature to stay on a desired path. The basic principles of the taken approach called pure pursuit control are depicted in [7]. This is a geometric approach that takes a certain point G on the path and calculates an arc which has the midpoint of the rear axe as a tangent and passes through G.

The basic steps of the algorithm are given by

1. Find the path point closest to the vehicle.
2. Find the goal point a lookahead distance away.
3. Transform the goal point to vehicle coordinates.
4. Calculate the curvature and request the vehicle to set the steering to that curvature.

As the path is given in the vehicle's body frame, the configuration is assumed as

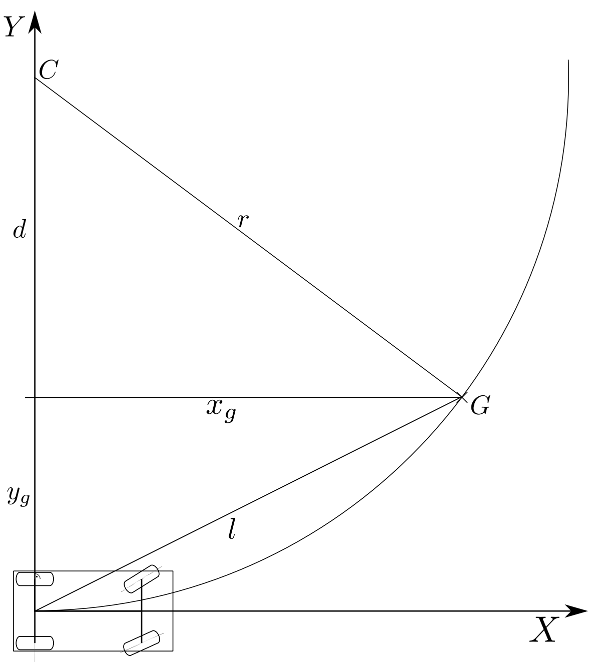


Figure 3.3: Geometry of arc calculation

$$q(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} 0m \\ 0m \\ 0^\circ \end{bmatrix} \tag{3.8}$$

Find goal point The goal point is defined by its coordinates

$$G(t) = \begin{bmatrix} x_g(t) \\ y_g(t) \end{bmatrix} \tag{3.9}$$

The parameter $l > 0$ defines the distance between the vehicle's position and the goal point, also called the lookahead distance. It is assumed, that l is a time invariant constant.

$$x_g^2 + y_g^2 = l^2 \tag{3.10}$$

With this constraint, the goal point is chosen by projecting the vehicle's position on the path and returning the first path point in distance l.

Calculating the curvature Given the goal point G in body frame coordinates, a geometric method can be applied to get an arc that is passing the goal point G and the rear axe tangent to its centerline. [7, p. 11] Figure 3.3 depicts following equations. C is the center of the arc and as the arc is tangent to the vehicle on the rear axe, C lies on the y-axis with coordinates $(0, r)$.

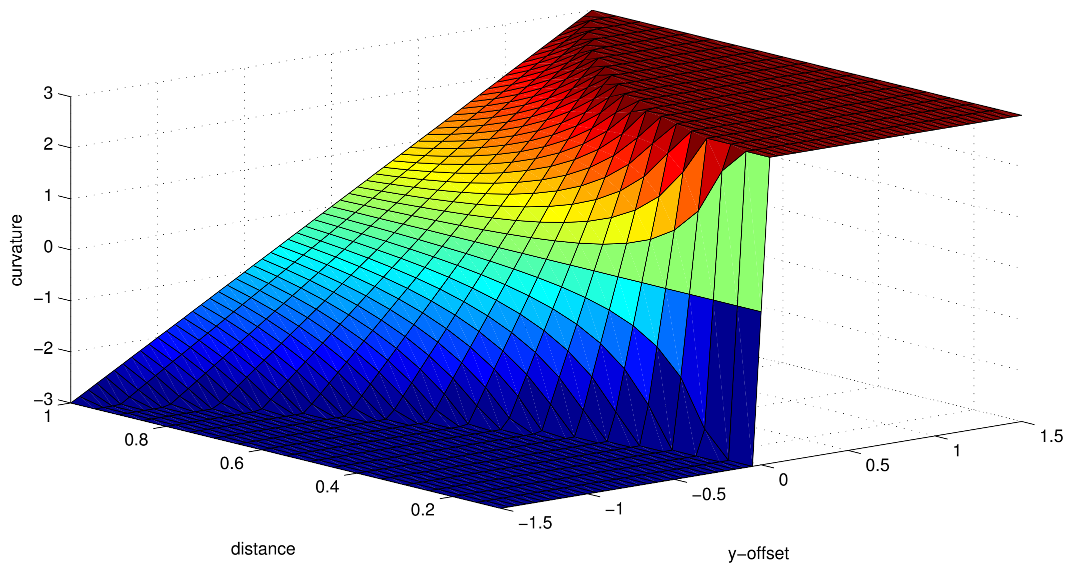


Figure 3.4: Curvature calculation

$$y_g + d = r \tag{3.11}$$

r, y_g and d span a right-angled triangle.

$$x_g^2 + d^2 = r^2 \tag{3.12}$$

Combining equations 3.10, 3.11 and 3.12, simple algebra transformations yield the arc-radius r.

$$r^2 = x_g^2 + (r - y_g)^2 = x_g^2 + r^2 + y_g^2 - 2ry_g \tag{3.13}$$

$$r = \frac{x_g^2 + y_g^2}{2y_g} = \frac{l^2}{2y_g} \tag{3.14}$$

Taking the reciprocal of r gains the sought curvature [1, pp. 22]

$$c = \frac{2\,y_g}{l^2} \tag{3.15}$$

Assuming a constant lookahead distance, the curvature is just dependant on y_g. Thus the error and the control law is defined by

$$e_{pp}(t) = y_g(t) \tag{3.16}$$

$$u(t) = \frac{2\,e_{pp}(t)}{l^2} \tag{3.17}$$

To define a transfer function G_{pp} for the pure pursuit algorithm, $u(t)$ has to be transformed into the frequency domain. This can be achieved with a Laplace or Fourier transformation, although this thesis' analysis will be restricted to Laplace transformation. The following transformations assume time-independent constant distance l.

$$U(s) = \mathcal{L}\{u(t)\} = \int_0^\infty \frac{2\,e_{pp}(t)}{l^2}\,e^{-st}dt = \frac{2}{l^2}\int_0^\infty e_{pp}(t)\,e^{-st}dt = \frac{2}{l^2}\,E_{pp}(s) \tag{3.18}$$

Thus, the transfer function is given simply by

$$G_{pp}(s) = \frac{U(s)}{E_{pp}(s)} = \frac{2}{l^2} \tag{3.19}$$

This represents a proportional gain controller with the adjustable parameter l.

3.2.2 SteeringController

This module's purpose is to maintain a desired curvature by setting an appropriate steering angle ϕ. Although this task is velocity dependant, a constant velocity $v \neq 0$ is assumed. There are two approaches employed: using a bilinear map out of actual curvature measurements and using a reactive PID-controller which gets its feedback from the motion model. In its current implementation, the weight of each approach is determined by a constant gain.

The reference is calculated by the PathSteeringController and is defined by the desired curvature c.

Model driven control By manual measuring the actual radii of driven curves with a fixed steering angle and velocity, a point wise mapping was built. The measurements were taken for the velocities $1.0\frac{m}{s}$, $1.5\frac{m}{s}$, $2.0\frac{m}{s}$, $3.0\frac{m}{s}$ and steering angles 20°, 15°, 10°, 7.5°, 5° and 4°. For the implementation of the model, a simple mapping with a poly-line-function was used, as it is quite simple and insures a monotonically increasing function.

This results in four functions $f_{1.0}(c), f_{1.5}(c), f_{2.0}(c), f_{3.0}(c)$, one for each velocity. As most of the time the actual velocity isn't exactly the one associated to the respective function, a linear function between inner and outer boundary was employed. This renders the bilinear map $f(v,c)$ illustrated in figure 3.5.

Reactive control As this control task is very prone to the mechanical configuration and other changes in environment, an additional closed-loop control system was taken as a basis to render a more robust controller.

$$c(t) = \frac{\omega(t)}{v} \tag{3.20}$$

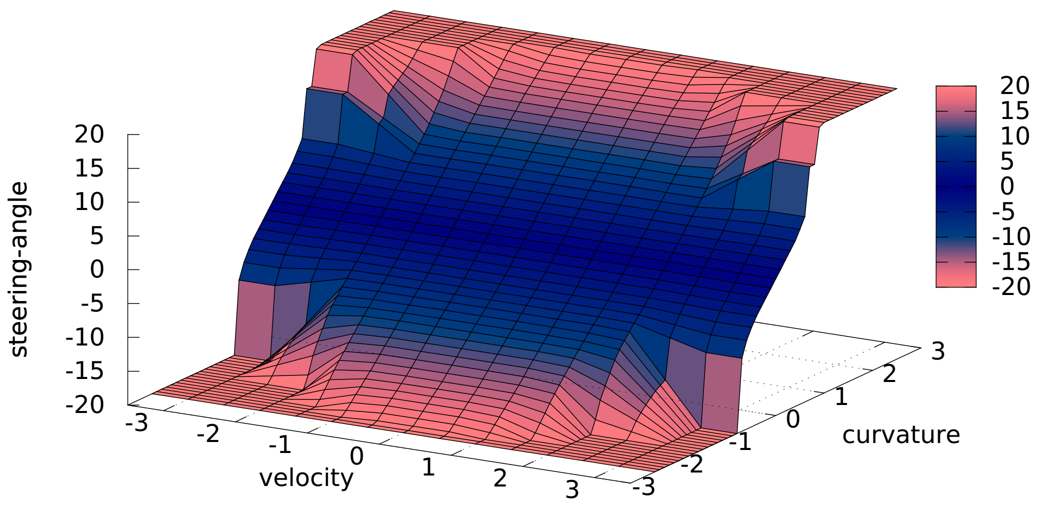


Figure 3.5: Steering map

As the curvature is proportional to the rotational velocity, $\omega(t)$ can be used as a reference input

$$r(t) = \omega_{ref}(t) = v\, c(t) \tag{3.21}$$

which is useful, as this is provided by the motion model

$$y(t) = \omega_{out}(t) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \dot{q}_r(t) \tag{3.22}$$

The rotational velocity error can be defined by

$$e_c(t) = r(t) - y(t) \tag{3.23}$$

Including the control output

$$u(t) = \phi(t) \tag{3.24}$$

a closed-loop proportional gain control law can be stated as

$$u(t) = K_P\, e_c(t) = K_P\, (r(t) - y(t)) \tag{3.25}$$

which can be expanded to a PID controller

$$u(t) = K_P\, e_c(t) + K_I \int_0^t e_c(t)dt + K_D \frac{d}{dt} e_c(t) \tag{3.26}$$

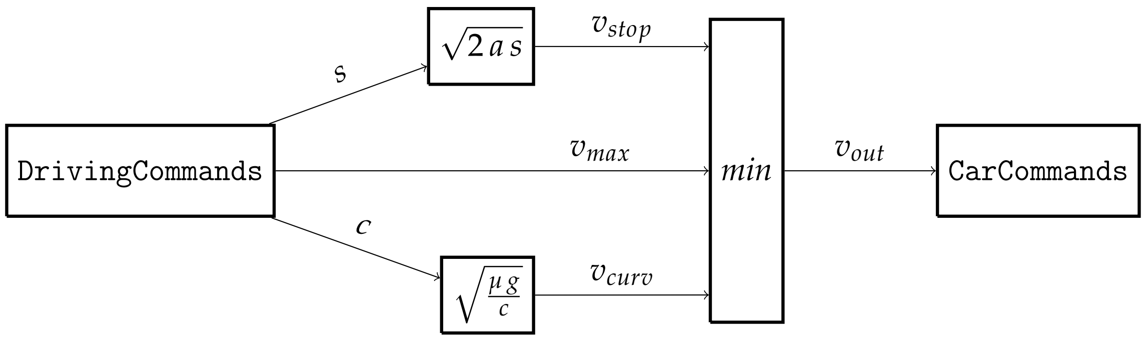


Figure 3.6: Data-flow diagram of SpeedPlanner

The Laplace-transformed output is obtained by

$$U(s) = \mathcal{L}\{u(t)\} = \int_0^\infty (K_P\, e_c(t) + K_I \int_0^t e_c(t)dt + K_D \frac{d}{dt} e_c(t))\, e^{-st}\, dt \tag{3.27}$$

which can be expressed in relation to the Laplace-transformed $E_c(s)$

$$U(s) = (K_P + \frac{K_I}{s} + K_D s)\; E_c(s) \tag{3.28}$$

The curvature-control transfer function can then be defined by

$$G_{PID}(s) = \frac{U(s)}{E_c(s)} = K_P + \frac{K_I}{s} + K_D s \tag{3.29}$$

Tuning of the parameters was done with the Ziegler-Nichols method [28, pp. 243]. The p-gain K_{Krit} that introduces constant oscillation is given by $K_{Krit} = 2.5$. After some additional manual tuning involving dropping the differential part as it was prone to input noise and lowering the integral part to prevent oscillations these parameters were identified:

$$\begin{bmatrix} K_P \\ K_I \\ K_D \end{bmatrix} = \begin{bmatrix} 1.2 \\ 0.008 \\ 0.0 \end{bmatrix} \tag{3.30}$$

The integral was limited to 2.0 to prevent integral windup.

3.2.3 SpeedPlanner

The SpeedPlanner is responsible for setting a speed that is safe enough to drive on the given path. Furthermore it stops on the path's end.

The approach is based on open-loop control utilizing two functions: a velocity v_{stop} to derive a speed to stop in a certain distance, and a velocity v_{curv} to get a safe speed to drive a desired curvature. Additionally there is an input velocity v_{max} which is the desired velocity by the path planner. Evidently the minimum of the computed speeds will be forwarded down to the actuator controllers. Thus its output is determined by

$$u(t) = min\{v_{stop}(s),\ v_{curv}(c),\ v_{desired}\} \tag{3.31}$$

Stop velocity As mentioned in [15, p.131], it is advised to take just the forward distance to a stopping point in account and not the vector-magnitude, because if the car has a bit of an y-offset to the goal, it won't be able to decrease the distance to zero and therefore generate possibly unstable behavior.

Assuming constant acceleration, the velocity v_{stop} to stop in distance s is defined by:

$$v_{stop} = \sqrt{2\,a\,s} \tag{3.32}$$

As noted in [12, pp. 16], it is also possible to have other than constant deceleration. As this is a model car and therefore the comfort doesn't have as a high priority as in passenger cars, this was omitted.

Curvature velocity Given the friction coefficient and the highest curvature in a certain range, v_{curve} can be computed by

$$v_{curve} = \sqrt{\frac{\mu\, g}{c}} \tag{3.33}$$

4 Evaluation

This section is dedicated to the performance evaluation of the mentioned control approaches. It is divided into two parts, one reviews the circle stability of the steering controller, and the other evaluates the path follower for stability and cornering behavior.

The tests were executed on a laboratory track of 4x6 m^2 using motion and localization input and control output for displaying the results. Looking at the respective figures, there are notable jumps in the tracked pose that are caused by small localization errors and do not represent the vehicle behavior in reality.

4.1 Circle stability

To check the stability of driving a circle, a simple testing method was employed: the vehicle is set to a constant speed of $1\frac{m}{s}$ and after one second of driving straight forward, it is commanded to drive a left curvature of $1\frac{1}{m}$. This results in a goal rotational velocity of $1\frac{rad}{s}$.

The two proposed methods are evaluated separately, using the open-loop steering model gained from manual measurements and using the PID controller with input from the measured motion.

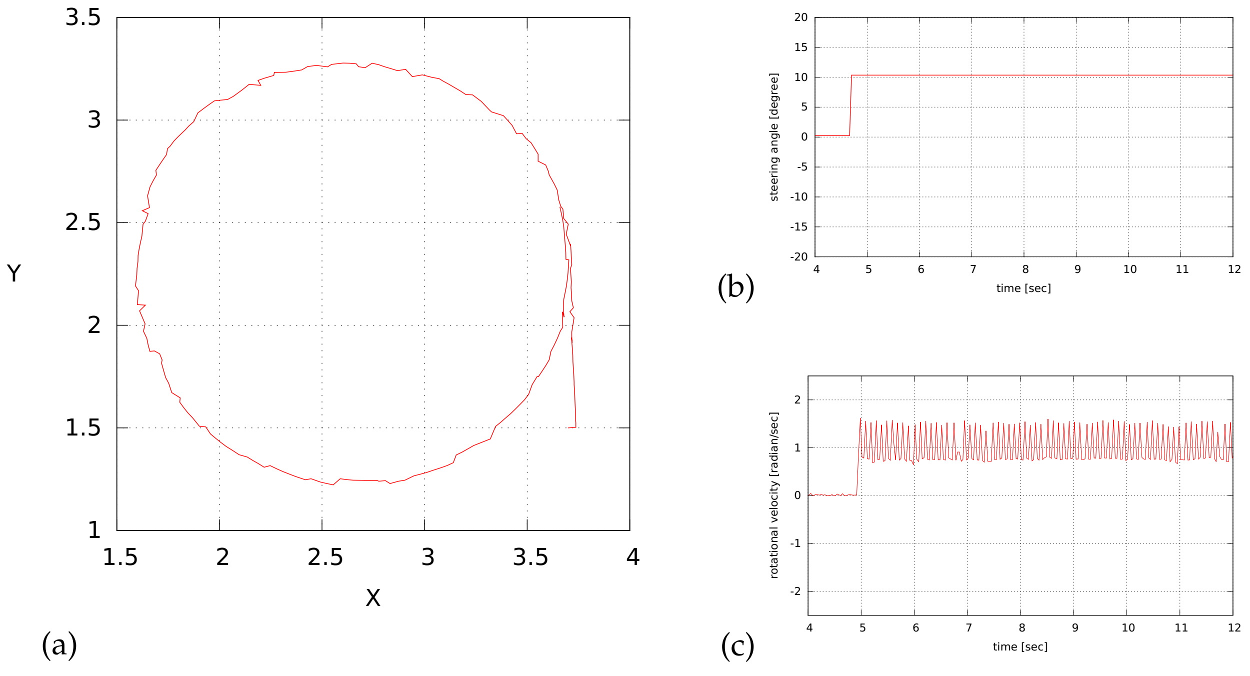


Figure 4.1: Driving a circle with steering model
(a) Tracked pose (b) set steering angle (c) measured rotational velocity

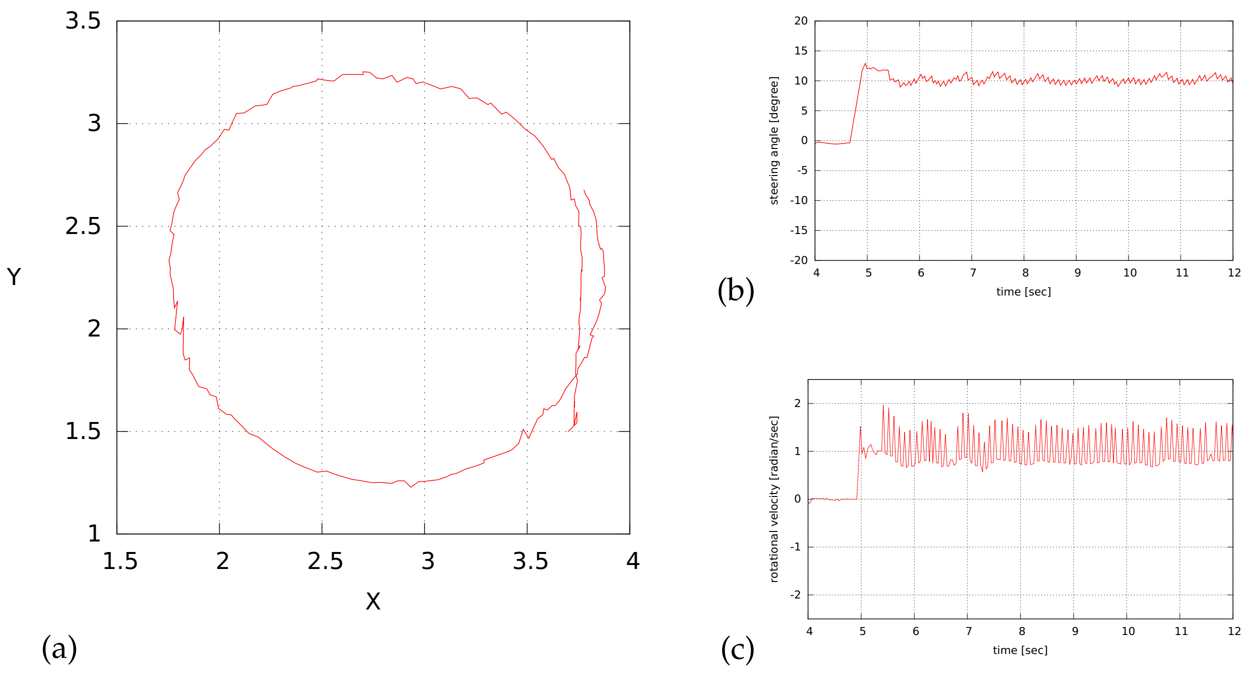


Figure 4.2: Driving a circle with PID-controller
(a) Tracked pose (b) set steering angle (c) measured rotational velocity

4.1.1 Steering model

Figure 4.1 illustrates the controllers performance. As this is a pure open-loop controller, it is dependant just on the reference and as the curvature stays constant after a step input, the output steering angle stays constant as well. Thus bounded-input bounded-output stability can be argued to be true.

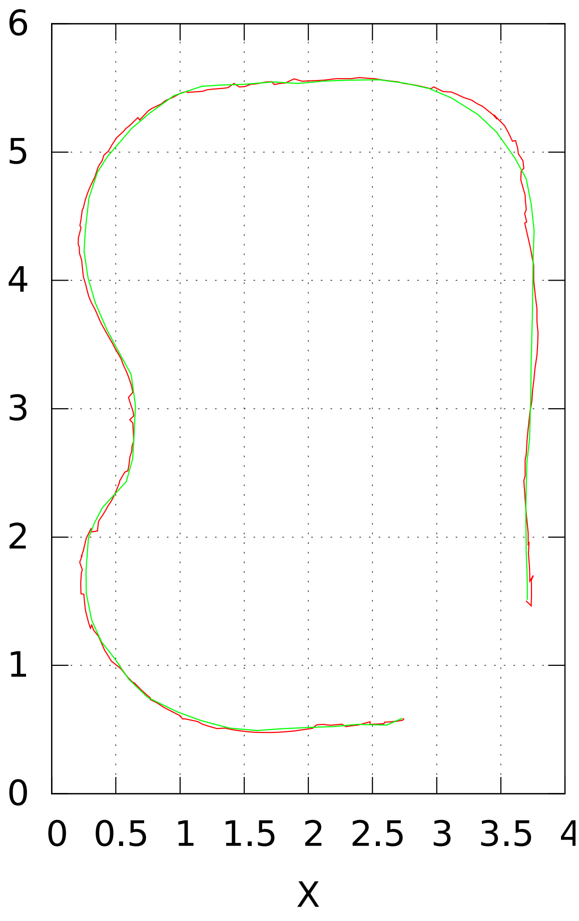
4.1.2 PID controller

Figure 4.2 illustrates the PID controllers performance. Stability is more of an issue in this closed-loop controller. Although the measured input is inaccurate, the steering output is robust enough to overcome this difficulty. However, an overshoot and a slight oscillation can be registered with the used parameters.

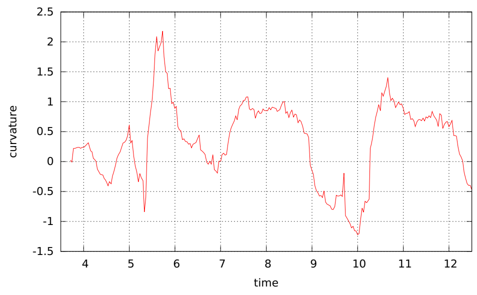
4.2 Path following

To study the performance of the controller, a simple static path on the laboratory track was set as the desired path which is represented by the green line in respective figures. The laboratory track's size is given by 4x6 meters.

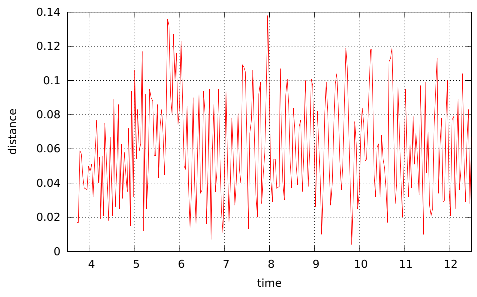
The evaluation goal is to follow the given path and decelerate before curves and finally stop on the last given path point.



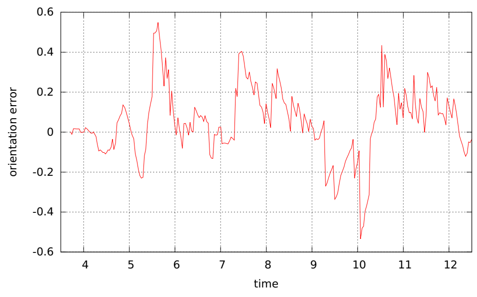
(a)



(b)



(c)



(d)

Figure 4.3: Pure pursuit with l = 0.4m
(a) Tracked pose (b) desired curvature (c) path distance (d) orientation error

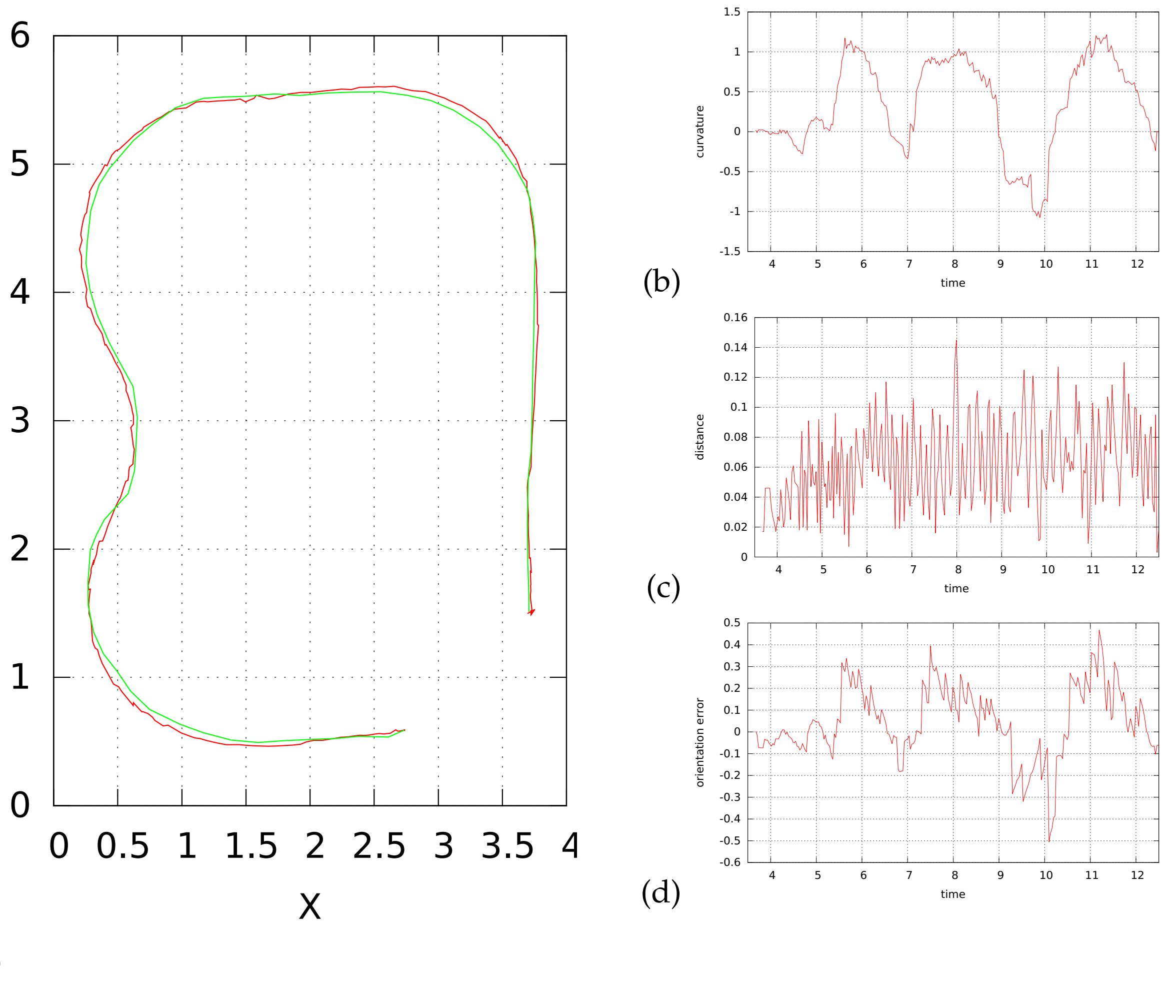
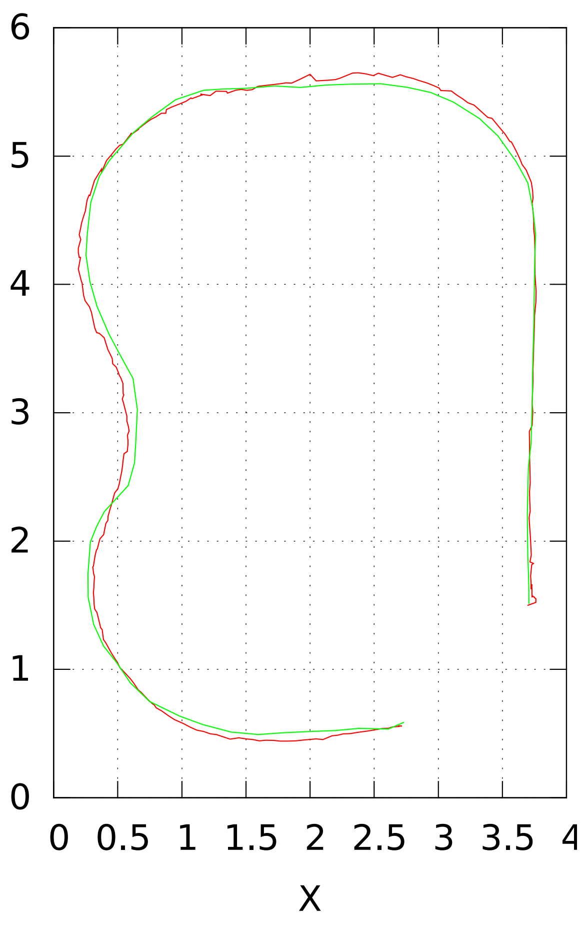
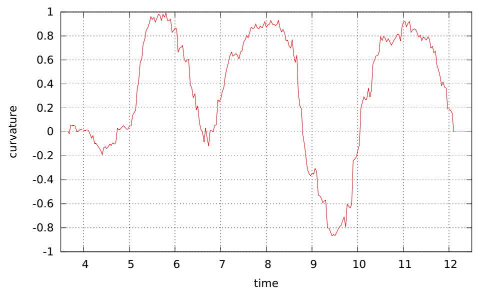


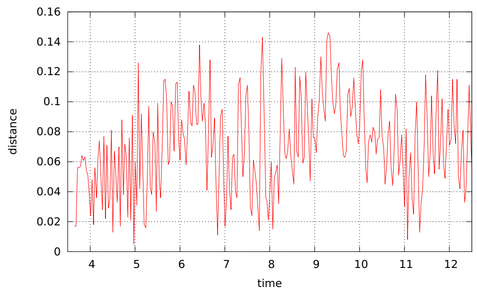
Figure 4.4: Pure pursuit with l = 0.55m
(a) Tracked pose (b) desired curvature (c) path distance (d) orientation error



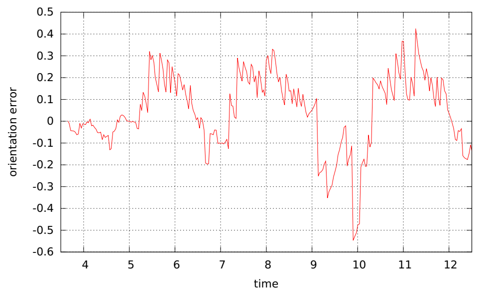
(a)



(b)



(c)



(d)

Figure 4.5: Pure pursuit with l = 0.7m
(a) Tracked pose (b) desired curvature (c) path distance (d) orientation error

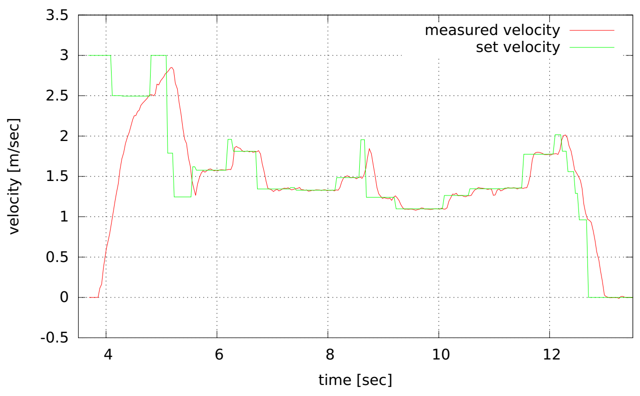


Figure 4.6: Measured and set velocity while path following

The path was followed with three different lookahead distances 0.4m, 0.55m and 0.7m. 0.4m lead to instability during the first long straight. 0.7m had shortcomings in cornering performance. This is especially notable in the s-curve. A lookahead distance of 0.55m rendered best results.

Figure 4.6 depicts the set velocity and the measured velocity with l=0.55m. The vehicle drove up to $2.85m/s$ on the straight before alternately decelerating and accelerating on the given curves. Finally it decelerated to $0m/s$.

5 Discussion

After proposing several possible enhancements for multiple subsystems which stand in relation with the presented path following solution, this section closes with a conclusion.

System modeling This thesis assumes a simple kinematic model, nevertheless there exist other more sophisticated models with fewer assumptions. The main problem with the presented model is, that it neglects tire slip and therefore probably decreases the controller's performance. Dynamic models instead, take the cornering stiffness and tire slip angles [31, pp. 340] [10, pp. 198] as well as roll dynamics [8] into account. Apart from that, also the actuators can be modeled to predict future states and render more stable results [12]. This improvement comes often with a cost of more powerful sensor information like an additional lateral displacement sensor in [14]. Using a kalman filter, it is also possible to learn a model from sensor data [27]. Besides, reinforcement learning can be utilized with Neural Fitted Q Iteration [25] [23].

Path representation Due to the simplicity of the used path representation, naturally some limitations are implied. It is not guaranteed that the car can even drive on the desired path, as no guard exists. As the car won't be able to follow curvatures exceeding its mechanical constraints, the path needs a boundary on the maximum allowed curvature. Furthermore, a safe driving behavior can only be achieved if the path is also continuous in its curvature. This is due to the fact, that the car cannot instantaneously change its steering angle and therefore its rotation.

Among others, there exist two approaches to generate continuously differentiable curves. As shown in [9][11] a path can be constructed out of line segments, circular arcs and clothoids (these are curves with a linear change in curvature). This is basically an enhanced versions of the Reeds-Shepp-path [29]. In contrary to joining segments, a single continuously differentiable function can be utilized. A common method is using cubic spline paths, which are piece-wise polynomial functions with continuous derivatives, as presented in [13][22].

Instead of representing just a 2-dimensional path, a trajectory can be planned and subsequently used by the controller. As actions like light control also have to be timed, those can be annotated on that trajectory [30].

Path following A possible improvement as pointed out in [6, pp. 92] would be, to take the current splip angle into account when calculating the arc. This effectively renders a PD controller as shown analytically in [21, pp. 4].

Another enhancement could be to add an orientation controller, as the algorithm implicitly assumes that by following the path, the vehicle's orientation will stay similar to the path's orientation.

Taking an adaptive control approach to regulate the lookahead distance could enhance performance. There are several parameters conceivable: taking the maximum curvature on the path to define lookahead distances to improve cornering behavior, taking the target velocity as a gaining factor or taking the lateral path error to achieve better performance when the vehicle is far off the path. [24, pp. 59]

Nevertheless, if using the pure pursuit approach, the path planner should be designed to plan paths nearby the vehicle.

Steering control The most noticeable drawback with the proposed steering controller is, that it is based on several manual measurements for system identification. Small changes in mechanical configuration will eventually render the made measurements useless.

Apart from this, a better way to gain parameters, especially those of the PID chain and the model chain has to be implemented by using adaptive control techniques. A suitable way would be to use the steering model as a reference to determine the weight of steering model against PID control.

Furthermore, taking account of system delay [2] and disturbances [3, pp. 92]

Speed planning The acceleration can be made smoother by implementing an acceleration function which uses the distance to determine the critical curvature. The speed planner unfortunately does not take account of required translational acceleration on the path. A proper back-tracking method is presented in [12, pp. 16].

Conclusion

A simple but effective control architecture for path following tasks with nonholonomic vehicles was proposed and evaluated. Apart from little shortcomings in cornering behavior, the employed pure pursuit algorithm rendered stable results with just one adjustable parameter.

References

[1] Amidi, Omead: *Integrated Mobile Robot Control*. Carnegie Mellon University, Pittsburgh, 1990

[2] Behnke, Sven; Egerova, Anna; Gloye, Alexander; Rojas, Raúl; Simon, Mark: *Predicting away Robot Control Latency*. In: RoboCup-2003: Robot Soccer World Cup VII, Springer, 2004, pp. 712 - 719.

[3] Bekey, George A.: *Autonomous Robots. From Biological Inspiration to Implementation and Control*. The MIT Press, Cambridge, 2005.

[4] Boldt, Jan Frederik; Junker, Severin: *Evaluation von Methoden zur Umfelderkennung mit Hilfe omnidirektionaler Kameras am Beispiel eines Modellfahrzeugs*. Technical Documentation, Freie Universität Berlin, Berlin, 2012.

[5] Borrmann, Daniel: *Integration einer inertialen Messeinheit in die Mikrocontroller Plattform eines autonomen Modellfahrzeugs*. Bachelor Thesis, Freie Universität Berlin, Berlin, 2012.

[6] Campbell, Stefan F.: *Steering Control of an Autonomous Ground Vehicle with Application to the DARPA Urban Challenge*. Master Thesis, Massachusetts Institute of Technology, Cambridge, 2007.

[7] Coulter, Craig R.: *Implementation of the Pure Pursuit Path Tracking Algorithm*. Carnegie Mellon University, Pittsburgh, 1992.

[8] Feng, Kai-ten; Tan, Han-Shue; Tomizuka, M.: *Automatic steering control of vehicle lateral motion with the effect of roll dynamics* In: Proceedings of the 1998 American Control Conference, Volume 4, 1998, pp. 2248 - 2252.

[9] Fraichard, Thierry; Scheuer, Alexis: *From Reeds and Shepp's to Continuous-Curvature Paths* In: IEEE Transactions on Robotics and Automation, Vol. 20, No. 6, 2004, pp. 1025 - 1035.

[10] Gillespie, Thomas D.: *Fundamentals of Vehicle Dynamics*. Society of Automotive Engineers, Warrendale, 1992.

[11] Girbés, Vincent; Armesto, Leopoldo; Tornero, Josep: *Continous-Curvature Control of Mobile Robots with Constrained Kinematics* In: Proceedings of the 18th IFAC World Congress, Milano, 2011, pp. 3503 - 3508.

[12] Göhring, Daniel: *Controller Architecture for the Autonomous Cars: MadeInGermany and e-Instein*. Technical Report, Freie Universität Berlin, Berlin, 2012.

[13] Gómez-Bravo, Fernando; Cuesta, Frederico; Ollero, Aníbal; Viguria, Antidio: *Continuous curvature path generation based on ß-spline curves for parking manoeuvres*. In: Robotics and Autonomous Systems Vol. 56 Issue 4, April, 2008, pp. 360 - 372.

[14] Guldner, Jürgen; Sienel, Wolfgang; Tan, Han-Shue; Ackermann, Jürgen; Patwardhan, Satyajit and Bünte, Tilman: *Robust Automatic Steering Control for Look-Down Reference Systems with Front and Rear Sensors*. In: IEEE Transactions on Control Systems Technology, Volume 7, Number 1, 1999, pp. 2 - 11.

[15] Holland, John M.: *Designing Autonomous Mobile Robots. Inside the Mind of an Intelligent Machine*. Elsevier, Burlington, 2004.

[16] Ioannou, Petros; Sun, Jing: *Robust Adaptive Control*. Prentice-Hall Inc., New Jersey, 1996.

[17] Latombe, Jean-Claude: *Robot Motion Planning*. 3rd printing, Kluwer Acedemic Publishers, Norwell, 1993.

[18] Laumond, Jean-Paul: *Robot Motion Planning and Control*. Springer-Verlag, London, 1998.

[19] Maischak, Lukas: *Lane Localization for Autonomous Model Cars*. Master Thesis, Freie Universität Berlin, Berlin, 2014.

[20] Meier, Thorben: *Integration eines elektrischen Antriebs in ein autonomes Modellfahrzeug*. Bachelor Thesis, Freie Universität Berlin, Berlin, 2012.

[21] Murphy, Karl N.: *Analysis of Robotic Vehicle Steering and Controller Delay*. In: Fith International Symposium on Robotics and Manufacturing, 1994, pp. 631 - 636.

[22] Nagy, Bryan ; Kelly, Alonzo: *Trajectory Generation for Car-Like Robots Using Cubic Curvature Polynomials*. In: Field and Service Robots 2001, Helsinki, 2001.

[23] Nikolov, Ivo: *Maschinelles Lernen zur Optimierung einer autonomen Fahrspurführung*. Master Thesis, Hamburg University of Applied Sciences, Hamburg, 2011.

[24] Rankin, Arturo L.: *Development of Path Tracking Software for an Autonomous Steered-Wheel Robotic Vehicle and its Trailer*. Ph.D. Dissertation, University of Florida, 1997.

[25] Riedmiller, Martin; Montemerlo, Mike; Dahlkamp, Hendrik: *Learning to Drive a Real Car in 20 Minutes*. In: Frontiers in the Convergence of Bioscience and Information Technologies, 2007, pp. 645 - 650.

[26] Risch, Matthias: *Der Kamm'sche Kreis - Wie stark kann man beim Kurvenfahren Bremsen* In: Praxis der Naturwissenschaften - Physik in der Schule. Themenheft: Fahrphysik und Verkehr, Nr. 5/51, Aulis Deubner, Köln, 2002, pp. 7 - 11.

[27] Roumeliotis, Stergios I.; Sukhatme, Gaurav S.; Bekey, George A.: *Circumventing Dynamic Modeling: Evaluation of the Error-State Kalman-Filter applied to Mobile Robot Localization*. In: Proceedings of the 1999 IEEE International Conference on Robotics and Automation, 1999, pp. 1656 - 1663.

[28] Schulz, Gerd: *Regelungstechnik 1*. 3rd edition, Oldenbourg Wissenschaftsverlag GmbH, München, 2007.

[29] Shepp, L. A.; Reeds, J.A.: *Optimal Paths for a car that goes both forwards and backwards*. In: Pacific Journal of Mathematics, Vol. 145, No. 2, 1990, pp. 367-393.

[30] Wang, Miao; Ganjineh, Tinosch; Rojas, Raúl: *Action Annotated Trajectory Generation for Autonomous Maneuvers on Structured Road Networks*. In: Proceedings of the 5th IEEE International Conference on Automation, Robotics and Application (ICARA), 2011, pp. 67 - 72.

[31] Wong, Jo Yung: *Theory of ground vehicles*. 3rd edition, John Wiley & Sons Inc., New York, 2001.