AGH UNIVERSITY OF KRAKOW

FIELD OF SCIENCE: ENGINEERING AND TECHNOLOGY

SCIENTIFIC DISCIPLINE: AUTOMATION, ELECTRONICS, ELECTRICAL ENGINEERING AND SPACE TECHNOLOGIES

DOCTORAL THESIS

Freespace detection and examination based on surround occupancy grid

Author:	*Marek Szlachetka*
First supervisor:	*dr hab. inż. Jarosław Wąs, prof.AGH*
Second supervisor:	*dr hab. inż. Dariusz Borkowski*
Completed in:	*AGH University of Krakow* *Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering* *Department of Automatic Control and Robotics*

Kraków, 2023

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

DZIEDZINA NAUK INŻYNIERYJNO-TECHNICZNYCH

DYSCYPLINA AUTOMATYKA, ELEKTRONIKA, ELEKTROTECHNIKA
I TECHNOLOGIE KOSMICZNE

ROZPRAWA DOKTORSKA

Detekcja wolnej przestrzeni wokół samochodu w oparciu o siatkę zajętości

Autor:	*Marek Szlachetka*
Pierwszy promotor:	*dr hab. inż. Jarosław Wąs, prof.AGH*
Drugi promotor:	*dr hab. inż. Dariusz Borkowski*
Praca wykonana:	*Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie* *Wydział Elektrotechniki, Automatyki, Informatyki* *i Inżynierii Biomedycznej* *Katedra Automatyki i Robotyki*

Kraków, 2023

I would like to express my sincere gratitude to dr hab. inż. Jarosław Wąs and dr hab. inż. Dariusz Borkowski for their invaluable assistance and support in the preparation of my doctoral thesis. Their insights, expertise, and guidance were critical in shaping the direction and scope of my research, and their thoughtful feedback and constructive criticism helped me to refine my ideas.

Preamble

This research is a result of an industrial PhD program, funded by the Polish Ministry of Science and Higher Education (MNiSW), project number 0014/DW/2018/02, and carried out in cooperation with Aptiv Services Poland S.A., Technical Center Kraków and AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering.

Abstract

One of the most important issues in the development of Advanced Driver Assistance Systems (ADAS) is the perception of the vehicle surrounding. ADAS systems consist of several layers, from sensors layer, through the perception layer, to collision-free maneuvers planning layer.

Perception systems are used to describe the surrounding environment of the host vehicle, both dynamic and stationary. The stationary environment includes all stationary objects, that is, parked cars (which were not previously detected as moving by the system), buildings, safety barriers, curbs, signs, etc. Numerous mathematical models representing stationary environment can be found in the literature, starting from primitive shapes such as a rectangle or circle and ending on complex ones such as occupancy grid or parametric curve.

In this thesis, the algorithm is proposed to determine and track the boundary of the free space. The parametric curve in the form of the B-spline is used as the mathematical model, and the occupancy grid is an input. There are introduced several original improvements of the best at the time of writing state of the art algorithm for free space boundary tracking. The improvements cover topics like measurement downselection, measurement to spline association and dynamic adjustment of B-spline control points. The last improvement uses original indicators of local shape complexity of the B-spline proposed by the Author.

The advantage of the proposed algorithm over the reference state of the art algorithm is proven based on a wide set of artificial road scenarios and one real scenario. The metrics used in the comparison of the proposed and reference algorithms are defined and an in-depth analysis of the comparison results is conducted. The achieved results clearly show that the proposed algorithm offers a higher free space boundary approximation quality while reducing the algorithm execution time.

Keywords: parametric curve, B-Spline, automotive, perception system, approximation, stationary environment

Streszczenie

Jednym z najważniejszych zagadnień w rozwoju Zaawansowanych Systemów Wspomagania Kierowcy jest percepcja otoczenia pojazdu. Systemy składają się z kilku warstw zaczynając od warstwy czujników, przez warstwę percepcji, aż po warstwę planowania bezkolizyjnych manewrów.

Systemy percepcji służą do opisu otaczającego pojazd środowiska, zarówno dynamicznego, jak i stacjonarnego. Środowisko stacjonarne obejmuje wszystkie obiekty nieruchome, takie jak zaparkowane samochody (które system nie wykrył wcześniej jako poruszające się), budynki, bariery ochronne, krawężniki, znaki drogowe, itp. W literaturze można znaleźć wiele modeli matematycznych reprezentacji środowiska stacjonarnego, począwszy od prymitywnych kształtów, takich jak prostokąty czy koła, a kończąc na bardziej złożonych, takich jak siatki zajętości czy krzywe parametryczne.

W niniejszej pracy został zaproponowany algorytm wyznaczania i śledzenia granicy wolnej przestrzeni. Jako model matematyczny wykorzystano parametryczną krzywą sklejaną typu B-spline, a dane wejściowe stanowiła siatka zajętości. Wprowadzono kilka oryginalnych usprawnień najlepszego znanego w czasie pisania pracy algorytmu śledzenia granic wolnej przestrzeni. Uprawnienia te obejmują takie zagadnienia jak filtracja punktów pomiarowych, asocjacja punktów pomiarowych ze splajnem oraz dynamiczna modyfikacja punktów kontrolnych krzywej B-spline. Ostatnie usprawnienie wykorzystuje zaproponowane przez autora oryginalne wskaźniki lokalnej złożoności kształtu krzywej B-spline.

Algorytm proponowany w pracy został porównany z algorytmem referencyjnym na podstawie szerokiego zestawu sztucznych scenariuszy drogowych oraz jednego scenariusza rzeczywistego. Zdefiniowano metryki stosowane w porównaniu obu algorytmów, a następnie przeprowadzono dogłębną analizę wyników porównania. Uzyskane wyniki jednoznacznie wskazują, że zaproponowany algorytm oferuje lepszą jakość aproksymacji granic wolnej przestrzeni przy jednoczesnym skróceniu czasu wykonania algorytmu, co stanowi przewagę względem algorytmu referencyjnego.

Słowa kluczowe: krzywa parametryczna, B-Spline, przemysł samochodowy, systemy percepcji, aproksymacja, środowisko stacjonarne.

Contents

List of abbreviations and mathematical symbols

The following table describes the significance of various abbreviations and acronyms used throughout the thesis, in the alphabetical order. The page on which each one is defined or first used is also given. Common abbreviations or acronyms used once are not on this list.

Abbreviations	**Meaning**	**Page**
PFS	Parametric Free Space	21
WCS	World Coordinate System	56
GRCS	Grid Coordinate System	56
OCG	Occupancy Grid	101

Mathematical notation

The following table lists the mathematical symbols used throughout the thesis.

Symbol	**Meaning**
$\mathbf{r}(s)$	2D parametric spline curve
s	free variable correlated with spline s-domain
n	spline degree
N_q	number of control points
N_m	number of measurement points
N_s	number of sampled spline points
$\mathbf{q}$	state vector (control point positions)
$\mathbf{\Omega}$	measurement status
τ	number of cycles since the measurement status change
κ	number of cycles in which the variance of the control point position is high
Λ	control point supporting interval
$\mathbf{\Psi}$	local shape complexity
Δ	distance complexity indicator
Γ	corresponding spline point complexity indicator
Θ	angle complexity indicator
Ξ	host-dependent indicator
$\mathbf{\Phi}$	approximation error indicator
$B_i^{[n]}$	i-th basis function of the n-th degree corresponding to the i-th control point

1 Introduction

One of the most important problems to solve in the development of Advanced Driver Assistance Systems (ADAS) is how to accurately reflect the areas of the host car environment that are free from obstacles belonging to either dynamic or stationary environments.

The dynamic environment describes all moving objects, such as moving cars, pedestrians, etc. The stationary environment describes all stationary objects, that is, parked cars (which were not detected as movable before by the system), buildings, crash barriers, curbs, signs, etc.

Virtually all of the driver assistance systems in use today depend on the perception system, which affects directly such important features as, e.g., emergency braking. Thus, the system has to work under any circumstances (for instance, in bad weather conditions) without any assumptions regarding the shape of the environment. Moreover, both the perception system and the layer working above must respect the restrictions put on them that come from embedded systems. They are time and memory constraints. The system must react in the predetermined time and be "small" enough to be integrated within the embedded system.

Vehicle setup can vary from one car to another. It mostly depends on the scope of features offered by the vehicle. The vehicle can be equipped with sensors like RADAR, LIDAR, camera or ultrasonic. They need to be combined in different variants. Therefore, the perception system should be independent of the sensors used and capable of fusion of information provided by various sensors.

In the automotive industry and in other fields like, e.g. robotics, the surrounding environment may be described by a variety of mathematical models. There are several models used for describing the stationary environment in the literature, starting from primitive structures (rectangle, circle, etc.) and ending on the occupancy grid or the parametric curves. All models are used to define the boundary between free space and obstacle, but they differ from each other by flexibility (ability to reflect complex shapes), memory footprint, and application (road scenario for which they are suitable). Moreover, different models have different computing power demands.

In this thesis, an innovative algorithm for tracking the boundary of the free space boundary modeled by the parametric curve calculated from an input in the form of an occupancy grid is proposed.

1.1 Objectives

One way to describe the free space boundary is by using a parametric curve, which has been discussed in the literature. An example of this is the Parametric Free Space (PFS) approach, which utilizes the B-Spline as the boundary model with a fixed number of control points. However, such approach has some drawbacks. The PFS cannot adapt control points efficiently to changing shapes when the host vehicle is in motion. Using too few control points leads to poor approximation quality, while using too many control points significantly increases the computation time. And despite using a high number of control points, there is no guarantee of achieving an acceptable approximation quality. Furthermore, the PFS assumes that the measurement points are evenly distributed, which leads to a semi-equal distribution of control points. This reduces the flexibility to adjust the control points to areas of higher shape complexity, instead of being able to focus on areas of low shape complexity. Another aspect regarding measurement points is the utilization of all measurements without filtering out those that lack supplementary information, resulting in redundancy.

Free space boundary estimation and tracking has an application in automotive embedded systems, but these systems have limitations such as processing power, memory, and real-time constraints. Therefore, algorithms developed for embedded systems need to strike a balance between being fast with minimal memory consumption and achieving high approximation and tracking quality.

Based on described disadvantages of the Parametric Free Space approach and the requirements of embedded automotive systems, the following thesis is stated:

The downselection of measurement points combined with the dynamic adjustment of the control points defining the spline curve approximating the free space boundary around the host vehicle in free space boundary tracking algorithm allows to reduce the computation time and memory footprint simultaneously with improved quality of the approximation in relation to the algorithm using all available measurement points and a fixed number of uniformly spaced spline control points.

1.2 Author contributions

This work contains the following contribution of the Author to the field.

- **Stationary models comparison** — Section 3.3
  The comparison of stationary environment models with respect to the set of their features and the suitability of the models to road scenarios.
- **Novel free space boundary tracking algorithm** — Section 4
  The algorithm for tracking the free space boundary containing numerous improvements (listed below) with respect to the reference algorithm.
- **Measurement points downselection** — Section 4.4.2.2
  Set of novel algorithms that perform the downselection of redundant measurement points while still maintaining high approximation quality.
- **Local measurement status determination** — Section 4.4.4
  A novel determination of a measurement status of each control point of the spline curve based on the association of measurement data.
- **Local spline shape analysis** — Section 4.4.5.1
  A novel local spline shape analysis performed based on the introduced set of spline complexity indicators.
- **Solving local minima problem of approximation** — Section 4.4.5.2
  A new way of solving the issue when the approximator stucks in a local minima by introducing a approximation error indicator calculated for each control point.
- **Control point adjustment** — Section 4.4.6
  Set of decision rules to judge when and where a new control point is added or an existing control point is removed.
- **Curve approximation metrics** — Section 5.2.2.1
  Two new metrics used to determine the quality of the curve approximation that give an extended overview of the algorithm performance.

- **Detection error metrics** — Section 5.2.3
  A new way of evaluating the free space boundary approximation from the point of view of vehicle safety.
- **Artificial grid generators** — Section 5.1.2
  Two grid generators (binary grid and occupancy grid) were used for the development of the proposed algorithm and the comparison of algorithms.
- **Comparison results analysis** — Section 5.4
  The comparison of the proposed algorithm and the reference algorithm with respect to approximation quality, detection metrics and footprint.

1.3 Outline

Chapter 2 provides a brief overview of the autonomous driving systems available in the automotive industry. It sheds light on a history of ADAS systems development and provides information about the autonomous driving stack starting from sensors and ending up with motion planning.

Chapter 3 is an overview of existing stationary environment models and the way in which they are identified. It provides a comparison of the models described based on two different indicators. It also describes the typical flow of the free space determination algorithm. The chapter ends with the choice of an algorithm used in the thesis as the reference one.

Chapter 4 contains a detailed description of the algorithm proposed by the author. The description starts with the algorithm interface. Then the fundamentals of the system are defined, including the main assumptions. Next, two major stages of the proposed algorithm are presented: occupancy grid processing and closed curve estimation, where the B-spline model is incorporated. In the description of the second stage, the most important inventions proposed by the author are presented.

Chapter 5 contains the performance evaluation proposal in the form of a comparison between the proposed algorithm and the reference algorithm. The entire evaluation methodology is described, starting from metrics determination, through artificial data preparation and generation, testing and results analysis (incorporating real data as well). Furthermore, some additional analyses of different factors that may influence the quality of the spline estimation are presented.

Chapter 6 includes the thesis summary together with potential further improvements of the proposed algorithm.

In Appendix, the definition of a B-spline curve may be found. Appendix also contains extended details of performed tests such as scenarios visualization or detailed data.

2 Towards autonomous driving

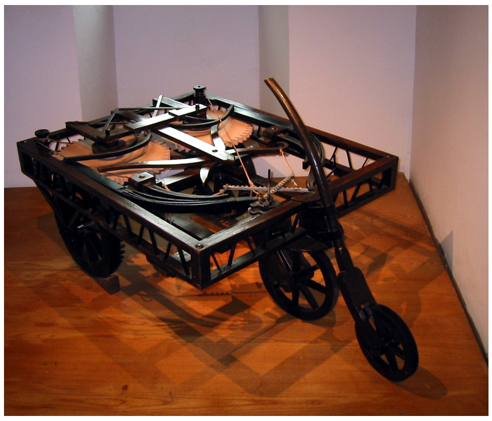
The automotive industry is one of the most dynamically developing fields in the world in recent years. However, its history goes back several centuries. The first part of this section introduces the history of cars viewed from the perspective of automated driving development (section 2.1), levels of autonomous driving defined in J3016 standard (sections 2.2) and automotive standardization requirements (section 2.3). Second part (section 2.4) shows autonomous driving stack starting from the lowest layer i.e. sensors (section 2.4.1) to the highest one, where most critical decisions are made (sections 2.4.4 and 2.4.5).

2.1 History at a glance

The first ideas of automated mobile date back to 1500s. Leonardo da Vinci invented the self-propelled cart [Hooper, 2004b] that could move without being pushed or pulled thanks to high-tension springs. The real pioneer move was taken in 1925. Houdina Radio Control firm developed a radio-operated automobile [Engelking, 2017], called the American Wonder. A car operator, staying out of the car, could control via radio small electric motors that directed every movement of the car. They allowed the operator for cornering, speeding up, slowing down and honking its horn. The car demonstration finished with crashing into an another car. Despite that, the car industry continued its dream about remote-controlled car.

In 1939, at World's Fair, Futurama exhibition [Kröger, 2016] showed the first self-driving car. They presented electric cars that were radio-controlled through electromagnetic fields provided by magnetized metal spikes embedded in the roadway. This model, more specifically the self-driving car, was finally deployed in 1958 [pete, 2004]. The car was guided by changing the electromagnetic fields on the spikes to keep the car in the designated lane.

Figure 2.1. A self-propelled cart replica at museum Clos Lucé.



At the beginning of the 1960s, cameras were used for the first time in an autonomous vehicle called Stanford Cart [Earnest, 2012] created by James Adams. Cameras were used to detect and follow a white line on the ground and to avoid obstacles placed in cart's path. After 16 years, in 1977, the University of Tsukuba's Mechanical Engineering Lab [Weber, 2014] improved James's idea with a camera system. Their first self-driving passenger vehicle could drive through Japanese roads at speeds up to 20 miles per hour.

Starting 1984 [Navlab, 2022], the Carnegie Mellon University (The CMU Navlab) began building autonomous cars. They used neural networks for image processing and steering control. Their car, named NavLab 5, traveled 2797 miles from Pittsburgh to San Diego in 1995 [Todd, 2022]. The Navlab project achieved 98.2% of autonomous driving.

There were many international and national projects conducted at the same time. The Prometheus project, conducted by EUREKA, spent over 1 billion US dollars from 1987 to 1995. In the 2000s, DARPA (The Defense Advanced Research Projects Agency) started to organize a series of challenges to expedite development of autonomous cars. The "Grand Challenge" competition is the one dedicated to autonomous cars. There were 3 organized competitions, in 2004 [Hooper, 2004a], 2005 [Buehler, 2007], and 2007 [Buehler, 2009], called "Urban Challenge". Then VisLab conceived VIAC, the VisLab Intercontinental Autonomous Challenge. Within the challenge, between 20.07.2010 and 28.10.2010, four vehicles were driving with no human intervention almost 16,000 kilometres (9,900 mi) [VisLab S.r.l., 2022][Bertozzi et al., 2011] trip from Parma (Italy) to Shanghai (China). By the mid-2010s, major car companies, as well as rideshare programs, began projects focusing on self-driving technology. However, true autonomy turned out to be more difficult to achieve than it was initially thought, and many of these companies eventually went out of business. It is worth to note that in 2020, Uber announced it was withdrawing from self-driving attempts as a result of safety, lawsuits, and money loss.

Nowadays, there are a lot of cars that reach some autonomous level maturity like 2020 Tesla Model S, 2020 Cadillac CT6, 2020 Nissan Rogue, 2020 BMW X7, 2020 Infiniti QX50, 2020 Volvo XC60, 2020 Mercedes-Benz S-Class, 2021 Toyota RAV4, 2021 Subaru Outback and more that are ongoing in time of writing of this thesis.

2.2 Levels of autonomous driving

In 2014, SAE organization issued standard J3016 [International, 2021a] that defines 6 levels of autonomous driving. Since that J3016 started to be the basic one used by wide number of automotive companies to describe their products. Following autonomy levels have been defined:

2.2.1 Level 0 - no driving automation

It is identified as full manual control. All dynamic driving tasks (DDT) are performed by driver. It means that the driver is solely responsible for vehicle maneuvering, including accelerating/braking, steering and any other maneuver necessary to move a car in any direction. Nevertheless, if a car is equipped with supporting systems such as collision warning, automatic emergency braking, blind-spot warning or lane-keeping assistance, it is still considered level 0. They don’t control a car and only offer alerts or mandatory reactions in some cases.

According to the April 2021 revision of SAE J3016, in order to attain a rating above L0 the automated control must be sustained (not transient/temporary).

2.2.2 Level 1 - driver assistance

Level 1 is distinguished from level 0 by having at least one system that provides steering or brake/acceleration feature. The driver is still responsible for driving and must be ready to take control at any time. An example system could be lane centering or adaptive cruise control. The last one is responsible for maintaining stable (safe) distance to a car ahead .

2.2.3 Level 2 - partial driving automation

At level 2, two or more systems work together in order to provide each feature mentioned in level 1 i.e. steering, acceleration and braking. The difference, with respect to level 1, a requirement to have a car equipped with both systems instead of only one of it.

2.2.4 Level 3 - conditional driving automation

Starting from level 3 it is assumed that a driver does not drive when automated driving features are engaged. It is still required to take control by the driver on the system demand. It can happen in such emergency situation like system failure. Traffic jam pilot (chauffeur) is an example of level 3.

2.2.5 Level 4 - high driving automation

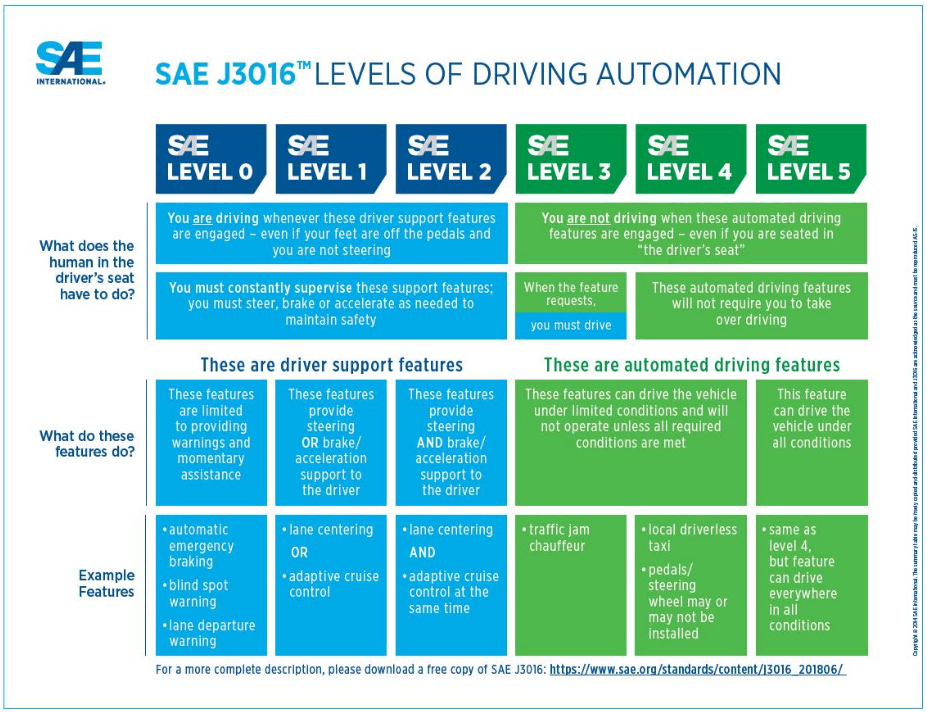
In comparison to level 3, there is no requirement to take control by a driver at level 4. Even in the case of system failure, a car must stop itself in a safe way. The example can be local driverless taxi.

2.2.6 Level 5 - full driving automation

Level 5 is the highest level of automation. It means that a car can drive itself from point A to point B without a driver and under any conditions.

2.2.7 Overview

All described levels of driving automation are shortly summarized in below graphic



SAE J3016™ LEVELS OF DRIVING AUTOMATION

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	**You <u>are</u> driving** whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			**You <u>are not</u> driving** when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			These are automated driving features		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering **OR** brake/ acceleration support to the driver	These features provide steering **AND** brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering **OR** • adaptive cruise control	• lane centering **AND** • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

For a more complete description, please download a free copy of SAE J3016: https://www.sae.org/standards/content/j3016_201806/

Copyright © 2014 SAE International. The summary table may be freely copied and distributed provided SAE International and J3016 are acknowledged as the source and must be reproduced AS-IS.

Figure 2.2. Levels of driving automation. Ref.[International, 2021b]

2.3 Standardization

The area where autonomous cars operate is very demanding as it interferes with human life. To minimize the probability of failure, a set of standards has been established. The most common are the following.

- **ASPICE (ISO/IEC 15504)** [VDA QMC Working Group 13 / Automotive SIG, 2017]
  ASPICE is the industry standard for evaluating software development processes. It helps automotive suppliers implement best practices to point out defects earlier and ensure OEMs' requirements are met.
- **Road vehicles — Safety of the intended functionality (SOTIF/ISO 21448)** [International Organization for Standardization, 2022]
  SOTIF is defined as "The absence of unreasonable risk due to hazards resulting from functional insufficiencies of the intended functionality or by reasonably foreseeable misuse by persons".
- **Functional Safety (ISO 26262)**[International Organization for Standardization, 2018]
  ISO 26262 is an international standard that specifies requirements and provides guidelines for the functional safety of road vehicles.
- **IEC 61508** [Commission, 2022]
  IEC 61508 covers methods for the application, design, implementation and maintenance of automatic safety-related systems.
- **AUTOSAR** [AUTOSAR, 2022]
  AUTOSAR stands for "AUTomotive Open System ARchitecture". It standardizes the interfaces between the software application and the basic functions of the vehicle.
- **MISRA** [MISRA, 2022]
  MISRA provides best practice guidelines for the safe use of both embedded control systems and standalone software.
- **TISAX** [TISAX, 2022]
  It is an information security assessment and information exchange mechanism in the automotive industry.

2.4 Autonomous driving stack

Behind the scene, the self-driving car can be divided into several layers, as is shown in Figure. 2.3. Each upper layer uses information provided by the lower layers. The most bottom one layer is *Sensors*, the host car's eyes and ears. It is responsible for producing raw information about the host and surrounding environment. The *vehicle state estimation* (VSE) utilizes the raw data from the sensors in order to get most accurate information about the host car, such as its position or speed. The *environment perception* provides information about the surrounding environment, both the dynamic world (e.g. moving cars and pedestrians) and the stationary world (e.g. guardrails). The topmost layers are *features* and *motion planning*. The *features* are functions, well known for drivers, that supports daily driving like e.g. lane keeping assistant. The *motion planning* is responsible for finding a safe (non collision) path from point A to point B.

2.4.1 Sensors

The basis of autonomous car are sensors. They are equivalent to human eyes and ears (Figure 2.4). This section is an overview of sensors most commonly used in automotive industry.

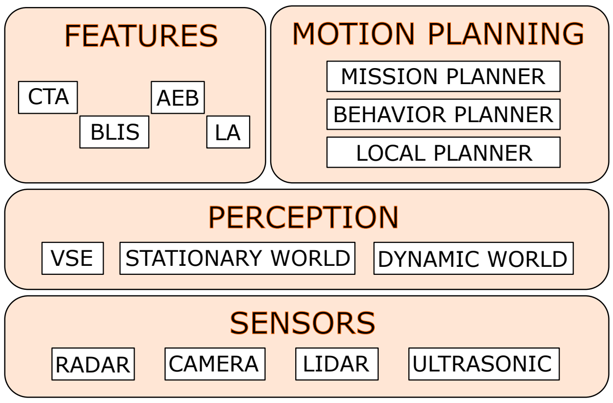


Figure 2.3. Autonomous driving stack

2.4.1.1 RADAR

It stands for Radio Detection and Ranging and consists of at least radio frequency generator transmitter (76-77GHz range band) and antenna receiver. A transmitter sends scattered waves. They fly out of an antenna, bouncing off every target they hit. Part of radiations come back to a receiver which, in most configurations, is located in the same or similar place as the transmitter [Kok and Fu, 2005]. The fundamental information provided by radar about a target is [Rahman, 2019]:

- range - obtained by recording the round trip time of the pulse
- radial velocity - obtained by utilizing the Doppler Effect
- azimuth (angular direction) - obtained by antenna rotation or in modern systems by utilizing a patch antenna with DSP-based pattern beam-forming methodology.

2.4.1.2 LIDAR

LIDAR, or light detection and ranging, uses a light beam, usually generated by an infrared laser diode, which is reflected from a rotating mirror [Rablau, 2019]. When the light comes across an object that does not absorb the light, then the light is reflected back to the sensor, which creates a map similar to the radar. LIDAR's important parameter is the detection range to an obstacle. The range can be degraded by wheather conditions (i.e. humidity) and object reflectivity [Ilas, 2013].

2.4.1.3 Camera

Camera absorbs the light as it reflects off an object, just like the human eye. The light beam is split into 3 components red, blue and green and fed to a metal oxide semiconductor or charge coupled sensor. The light is then converted into an electric charge. The most common way of evaluating distance to an object is the usage of two cameras called stereo vision. Stereo vision system uses two or more cameras to provide the depth of objects. A typical configuration uses two cameras separated by a horizontal distance. There is a correlation between separation distance and the desired depth of field and resolution. It is similar to human eyes. By giving two different perspectives of the same object, depth can be distinguished [Ozguner U, 2011] Camera is the sensor, with respect to the others, that has unique abilities like recognition of flat structures differing only by color or lightness like traffic signs content or road signs, lanes separating lines and traffic lights. In general the camera image is better suited for object class determination (distinguish between car, pedestrian, etc.) than e.g. radar data.

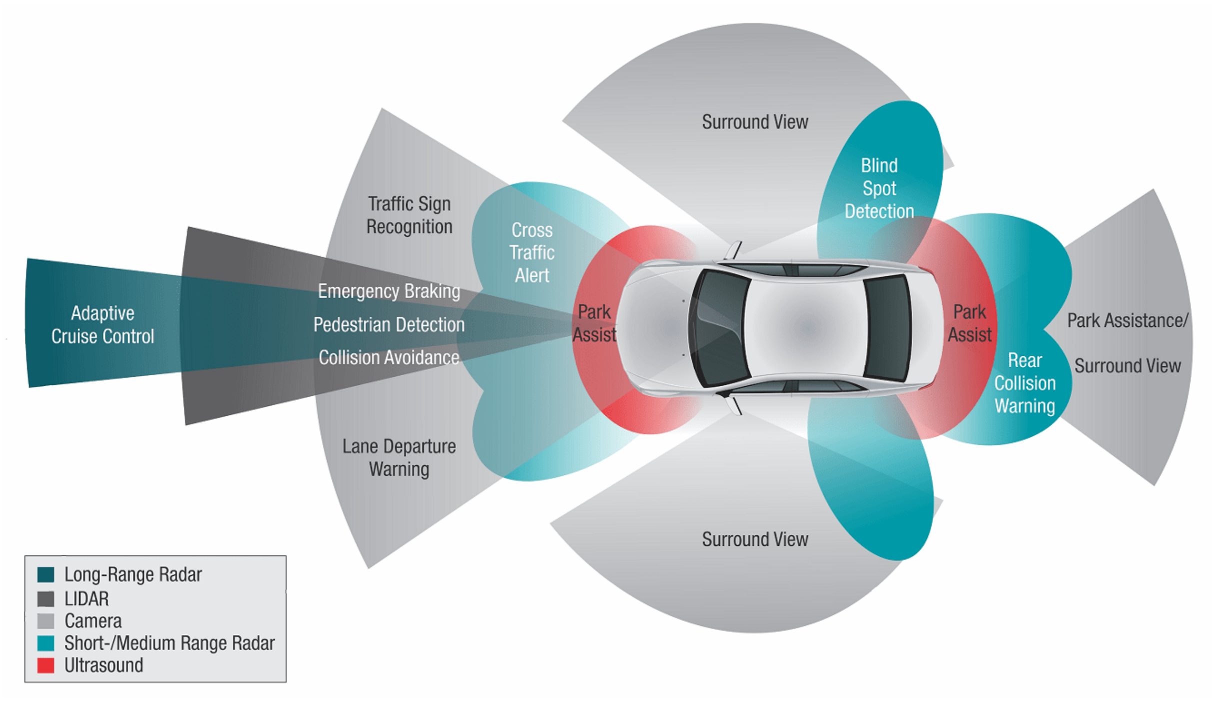


Figure 2.4. An example of car sensing setup. Ref.[Z et al., 2013]

2.4.1.4 Ultrasonic

Ultrasonic utilizes the sound wave that is reflected from various objects within range, and the frequency of the return pulses is used to estimate the distance. The sound wave is generated by piezoelectric material that is charged with an alternating electric current, causing its vibration. When it passes through the air, the sound wave causes the differential pressure and transfers the energy until the wave is dispersed or reflected [J et al., 2013].

2.4.2 Vehicle State Estimation

Vehicle State Estimation, shortly VSE, is a module responsibly for providing information about the host car such as position, velocity vector, heading, sideslip and acceleration.

The dead-reckoning is the process of calculating the current position and heading of a moving vehicle based on previously determined position, speed and direction. It may also be referred as odometry (with some simplifications). Using only previous position and speed data is not enough to reach high quality estimation. It is because of things such as slips or sideways movement (sideslip). For this reason, the Inertial Measurement Units (IMU) are incorporated. The IMU consists of gyroscopes, magnetometers and accelerometers from which additional information can be used for estimation corrections.

Global Navigation Satellite System (GNSS) utilizes a receiver that gets information from different satellites in order to triangulate its absolute (world) position. The system measures propagation time of radio signals traveling from the satellites to the receiver. Knowing the speed of the electromagnetic wave and the exact propagation time, the receiver can calculate the distance from the satellites. The GNSS signal contains also an information about the placement of satellites and information about their theoretical path and deviations. The GNSS receiver processes received information and uses it to determine its distance from individual satellites for which the receiver is within range. Thus, it is possible to estimate position of the GNSS receiver by having distances and positions of satellites using triangulation

[Schmid, 1974]. The most popular GNSS is GPS (Global Positioning System) [Raju, 2003] developed in the 1970s. Others well-known GNSS systems are Russian's GLONASS [Karutin et al., 2021], European Union's GALILEO [Rodríguez et al., 2021] or China's BEIDOU [Betz, 2016a]. Because of estimation errors caused by many reasons (e.g. satellites' atomic clock synchronization or receiver noises) followed improvement of basic GNSS systems was designed: Differential GNSS (DGNSS) [Grewal et al., 2020], Satellite Based Augmentation System (SBAS) [Betz, 2016b] and Real-Time Kinematic (RTK) [Langley, 1998].

Current vehicle positioning systems are hybrid ones that combine data from various sources such as odometry, GNSS, LiDAR, RADAR and cameras to obtain best results. Data fusion is usually done by Kalman Filters [Bersani et al., 2019] or Particle Filter [Zhu et al., 2019][Chu et al., 2015].

2.4.3 Environment perception

The estimation of car surrounding environment is a challenging part of the autonomous driving stack. The car has to deal with dynamically changing environment, partially occluded road objects or bad weather conditions. Moreover, environment estimators have to provide feasible world representation, in a form of mathematical models, based on which upper layers decide for example to stop the host car or not. Thus, it can save human life.

The environment can be split into two complementary categories: dynamic environment and stationary environment. The most significant difference between those two is object movability assumption. In other words, the dynamic environment models provide information only about objects that are currently moving or have been seen moving in the past.

2.4.3.1 Stationary world

Stationary world estimator provides information about the boundaries of the space within which the host can maneuver. Stationary obstacles are those that have a constant shape and position on the ground. There are several models that can represent stationary environment (Figure. 3.2).

Grid maps

Introduced by Elfes in [Elfes, 1989] and Moravec in [Moravec, 1989]. Grid map represents a surrounding environment in the form of the set of cells of the same size, defined on/over the ground. Each cell holds information about a part of the surrounding environment. And maps can be discretized in 2 or 3 dimensions (e.g. Octomap [Wurm et al., 2010], multi-level surface [Triebel et al., 2006]). There are also 2.5D maps which hold information about overhanging obstacles or just about the height of obstacles (e.g. [Gutmann et al., 2005], elevation map [Nam et al., 2017]). The most common grid map is occupancy grid map [Guerra et al., 2018].

Interval maps

This model can be treated as a one-dimensional special case of a 2D grid map [Weiherer et al., 2013]. It discretizes surrounding environment along the host car's longitudinal direction as opposed to grid maps.

Primitive structures

Primitive structures, as the name suggests, are the simplest models. A surrounding environment can be represented by primitive structures [Peng et al., 2015], such as circles and rectangles (or quadrilaterals [T.R. and M., 2019]), or by simple boundaries [Gex and Campbell, 1987a] e.g. lines.

Contours

Contours are yet another way to describe stationary environment. They define a border which is not drivable/traversable. Contours can be open or closed. An open contour is usually used to describe such structures like a highway guardrail. Thus, in this case, the open contour is represented by a low degree polynomial (up to 3rd) or by a clothoid [Danescu et al., 2006] (first time introduced to transportation by Talbot for railways [Talbot, 1901]). More complex are closed contours. If contours are closed, then a space surrounded by contours defines free space within the host car can move without making collision with stationary obstacles. There are several ways to model closed contours such as polygons [Gex and Campbell, 1987b] or closed splines of a degree higher than 1st [Schreier et al., 2016]. The last one is called "Parametric Free Space" (PFS) in [Schreier et al., 2016].

2.4.3.2 Dynamic world

Knowledge of the stationary environment is not enough to move safely through the world. Stationary objects do not move at all. In the dynamic environment all objects are movable. Thus, they are more dangerous and because it is not possible to predict their trajectories over longer time frames. There are several approaches to work with dynamic objects [Llamazares et al., 2020] called DATMO (Detection and Tracking of Moving Obstacles). DATMO observes a state of all moving (and temporally stopped) objects around the host car. Object's state contains such parameters as position, speed vector, size, class (car, pedestrian, etc.). State parameters are provided with respect to or compensated by the host state.

There are two main approaches used for determining dynamic world: *object-based* and *grid-based* approaches. There are also other approaches, which are not described in the thesis like deep learning techniques [Pavitha et al., 2021] or more general machine learning [Kraus et al., 2020] (approach supporting role).

Object-based approach

Object-based approach consists of independent tracking of multiple objects state (properties) in time. The approach can be decomposed to data association and object tracking.

In association stage new measurements from sensors are assigned to objects, to be used later in measurement update stage. Depending on the method either direct measurements or clustered measurements may be associated to objects. Association is done based on chosen metric, which describes distance of a single measurement or cluster of measurements from an object which is already tracked. Various association and clustering methods have been proposed in literature, like e.g. Nearest Neighbor Rule (NNR) [Hart, 1966][Cover and Hart, 1967], Global Neighbor Rule (GNR) [Konstantinova et al., 2003], Joint Probabilistic Data Association Filter (JPDAF) [Fortmann et al., 1983][Cox, 1993], Multiple Hypothesis Tracking (MHT) [Reid, 1979], Density-based spatial clustering of applications with noise (DBSCAN) [Lim et al., 2018] and more.

Regardless of the choice of association method, a kind of measurements segmentation (e.g. azimuth-range based clustering) is needed for new objects initialization. Measurements which are free (not associated to any object thus not used in measurement update) are clustered to distinguish groups of consistent measurement which can indicate the existence of a real object which is not yet tracked. After sufficient information is collected in a cluster of measurements then new object can be created out of this cluster and added to the list of tracked objects.

Object tracking is a stage of the *object-based* approach that deals with object's mathematical kinematic model and utilizes measurements associated to the object to update its state. The tracking is usually performed by a Kalman Filter [Kalman, 1960] or its variants like Extended Kalman Filter [Ribeiro, 2004], Unscented Kalman Filter [Chen et al., 2018] or information filter [Assimakis et al., 2012]. Another way is to use non-parametric filter like Particle Filter [Doucet et al., 2000].

Grid-based approach

The *grid-based* approach uses occupancy grids. Each cell of the grid is tracked independently. Bayesian Occupancy Filter [Coué et al., 2006][Saval-Calvo et al., 2017] and Dempster-Shafer Theory [Dempster, 2008] are examples of methods that can be employed for this task.

Cellular automaton

A cellular automaton is a mathematical model that can be used to simulate complex systems [Neumann, 1966][Wolfram et al., 2002]. It consists of a grid of cells, where each cell has a finite number of states. The state of each cell is determined by the states of its neighboring cells and its current state, according to a set of predefined rules. These rules specify how the state of each cell should change based on the states of its neighbors. Cellular automaton is used in the study of car traffic flow and congestion [Nagel and Schreckenberg, 1992][Małecki et al., 2022] as well as pedestrian dynamics simulation [Burstedde et al., 2001][Wąs and Lubaś, 2014].

2.4.4 Active safety features

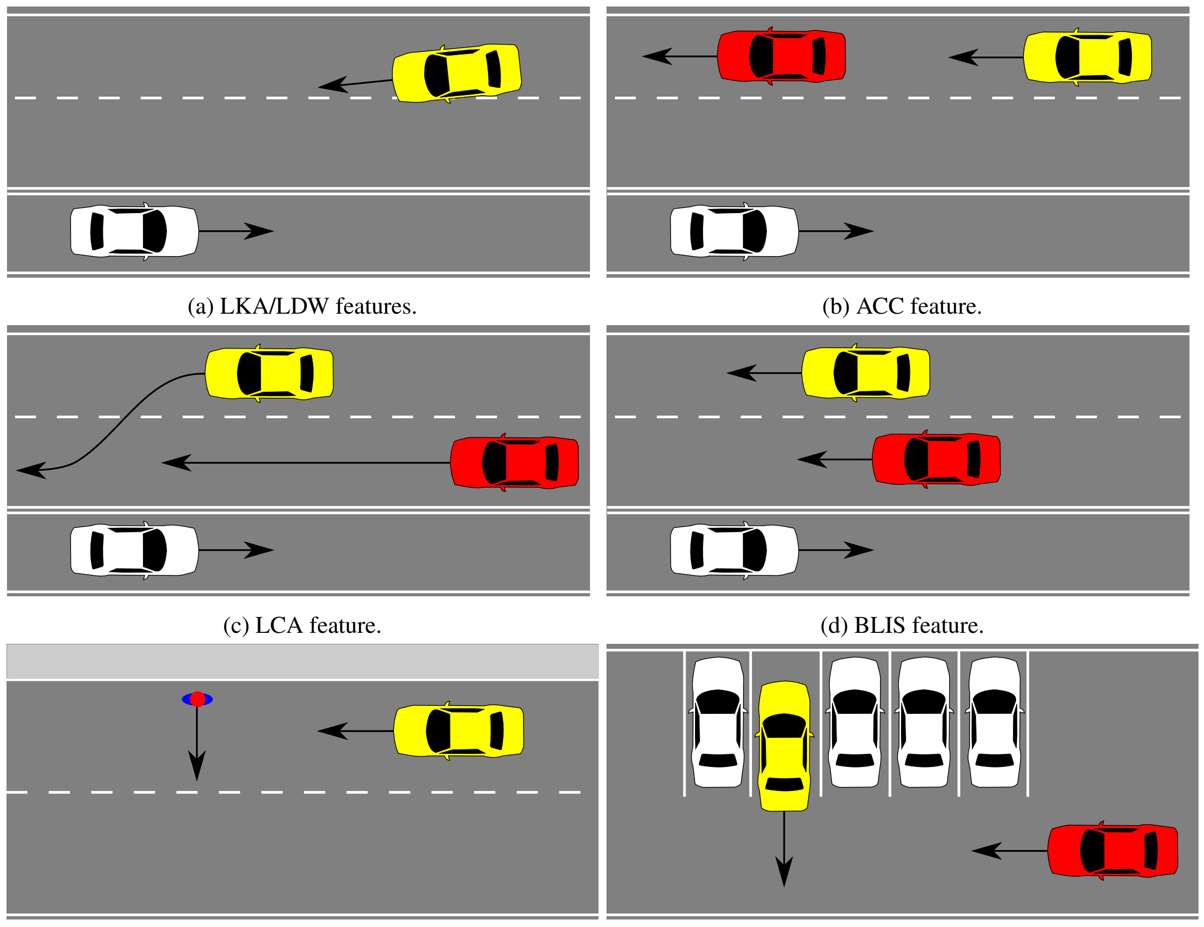
This section contains a brief overview of selected active safety features. All described features can differ between cars providers e.g. in conditions under which the system is active or not. Figure 2.5 presents usecase example for each described, in the following section, feature.

2.4.4.1 CTA - Cross Traffic Alert

Mostly used in reversing scenarios (Rear CTA). The CTA is intended to support the driver by detecting vehicles crossing the host car path [Spampinato et al., 2018], for example while leaving parking slot surrounded by other cars limiting road visibility. The support is done by alarming the driver via sound, light or on a screen (showing approaching vehicle).

2.4.4.2 AEB - Automatic Emergency Braking

The AEB rule is to stop the host car in order to avoid a collision. Once AEB discovers danger situation, it tries to warn driver and other road actors. If a driver does not react on time then AEB decides to stop a car on it own [Guo et al., 2022]



(a) LKA/LDW features. (b) ACC feature.

(c) LCA feature. (d) BLIS feature.

(e) AEB feature. (f) CTA feature.

Figure 2.5. Examples of features usecase scenarios. Legend: yellow - the host car, red - interesting object(car or pedestrian), white - extra objects

2.4.4.3 BLIS - Blind Spot Information System

It is a system that detects the presence of another vehicle in so-called blind spot area, which is not covered by the door mirrors [Liu et al., 2017]. BLIS is also called Blind-Spot Warning (BSW).

2.4.4.4 Lane Assistants system

LA - Lane Assist

Lane Assist is an advanced safety system that warns against unintended changes in lane while driving in a traffic and can cope with driving for a short time on its own.

LCA - Lane Change Assistant

Lane Change Assistant supports the driver once he wants to change a lane [Roelofsen et al., 2010]. The LCA looks for potential dangerous obstacles or other road actors approaching in adjacent lanes. There are two kinds of LCA. One just informs the driver about potential danger. More advanced one can perform lane change maneuver on its own.

LDW - Lane-Departure Warning

An active system designed to increase driving safety by warning against unintentionally departing from its own lane [Chen et al., 2020].

LKA - Lane-Keeping Assist

A system that can be treated as extension of LDW. It warns the driver and, in case of no response, takes control to ensure the vehicle stays in its lane [Sentouh et al., 2018].

2.4.4.5 ACC - Adaptive Cruise Control

Adaptive cruise keeps a constant distance between the host car and the car in the front, automatically maintaining a safe distance [Rajamani, 2021]. If traffic slows down, the system stops or slows down the host car and automatically restores speed when traffic resumes, allowing the driver to focus on lateral steering only.

2.4.5 Motion planning

The top of the autonomous driving stack is occupied by motion planning. Its responsibility is generating collision-free trajectory from a start position to a destination position. In details, it consists of three modules: mission planner, behaviour planner and local planner. Each module incorporates information provided by previous module (except the mission planner) in its calculations.

2.4.5.1 Mission planner

Mission planner is a first step in motion planning. Its main task is to construct a global path from a start to destination position. It is similar to well-known google maps or similar navigation systems [Rathnayake, 2018]. Mission planner finds the best possible path based on a cost function. It can be for example the fastest road (in terms of time) or the shortest (in terms of distance). Most of the algorithms used in mission planer are graph/tree-based methods. Examples of used algorithms are: Breadth-First-Search (BFS) [Cormen et al., 2022], Dijkstra's algorithm [Dijkstra, 1959], [Wenzheng et al., 2019], A* algorithm ("A" star) [Hart et al., 1968], D* algorithm ("D" star) [Stentz, 1997], Aggressive Heuristic Search (AHS) [Liu and Li, 2018], Jump Point Search (JPS) [Harabor and Grastien, 2011] and more.

2.4.5.2 Behavior planner

The Behavior planner plans a set of high-level actions/maneuvers to safely complete a driving mission under various road conditions. It takes into account such constraints as road rules (e.g. speed limit, stop locations), static and dynamic objects around the vehicle [Sadat et al., 2019]. As a result, the behavior planner provides a driving maneuver to be executed with a set of constraints including among other the host's speed.

2.4.5.3 Local planner

The local planner is the last step in motion planning system. It consists of a path planner and a velocity profile generator. The path planner computes the trajectory of the collision-free path [Oliveira et al., 2018], mostly in short distance horizon. The velocity profile generator is responsible for computing goal velocity including, among others, comfort constraints.

3 Stationary environment modeling and estimation

Modeling the stationary environment is a crucial part of the autonomous driving stack. There are several stationary environment models described in the literature. Each model describes the stationary environment in a different way. Different implementations are constrained among others by use case, i.e. driving scenario (e.g. highway, city, parking), type and number of available sensors, or by memory footprint and computing power demand.

This chapter presents existing models grouped in two categories i.e. parametric (section 3.2) and non-parametric (section 3.1) ones. The described models are compared at the end of the chapter (Section 3.3). Figure 3.1 is a kind of legend explaining common objects used in Figure 3.2 that presents different representations of stationary environments described in this chapter.

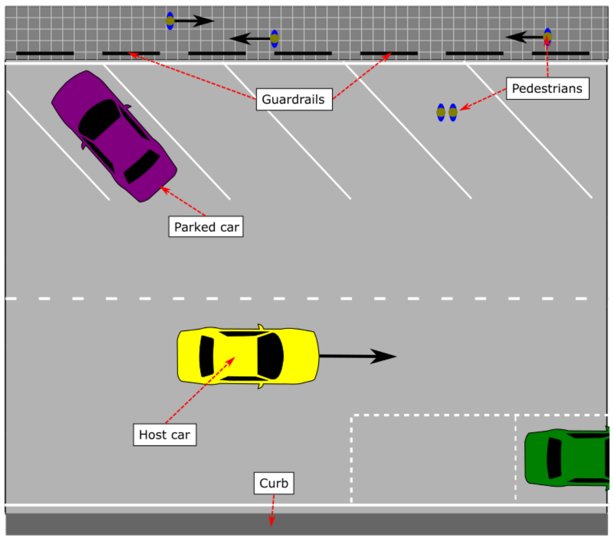
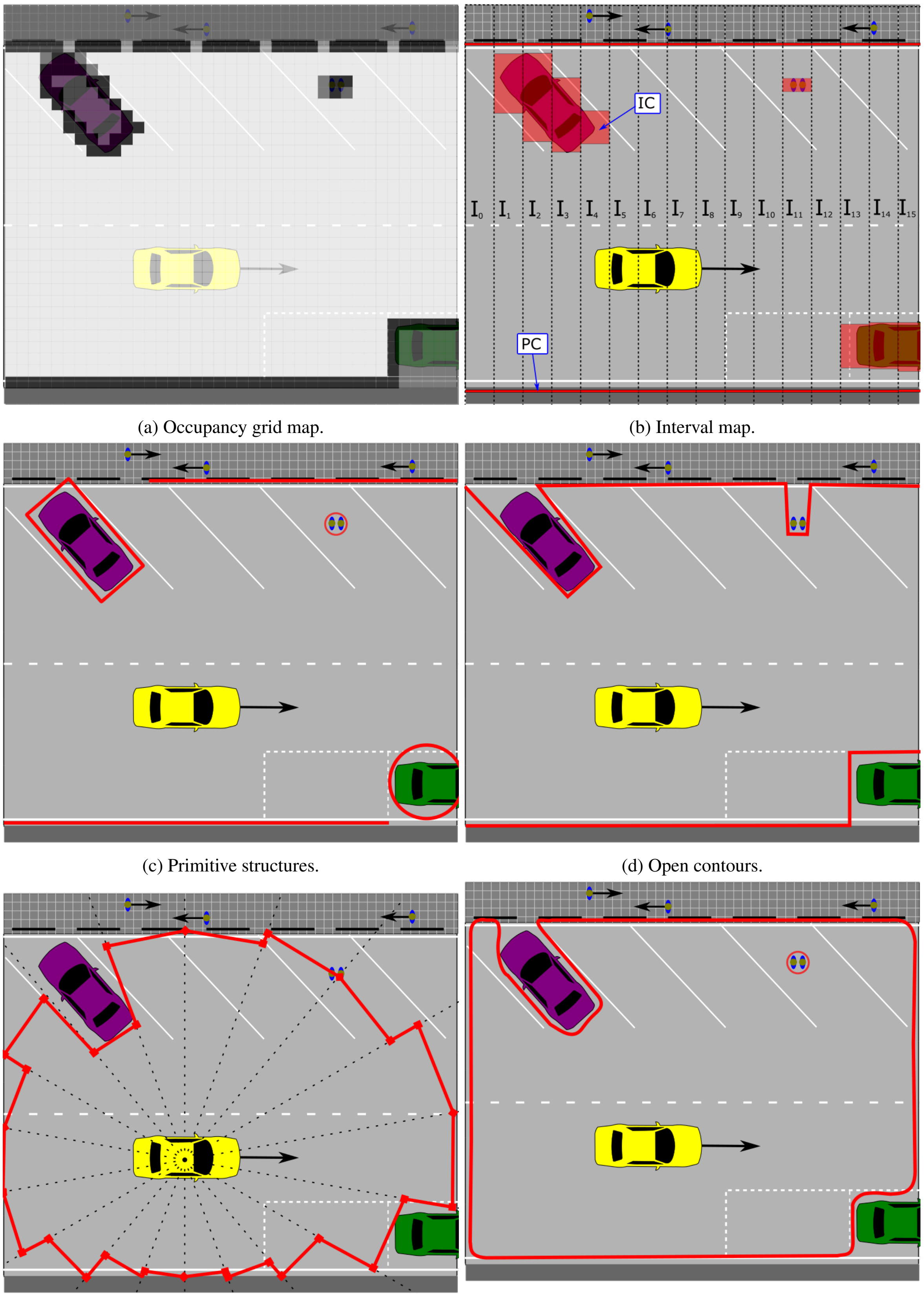


Figure 3.1. Legend for stationary environment representations shown in Figure. 3.2.


IC
PC
I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9 I_{10} I_{11} I_{12} I_{13} I_{14} I_{15}

(a) Occupancy grid map.

(b) Interval map.

(c) Primitive structures.

(d) Open contours.

(e) Closed contours (polygonal chain)

(f) Closed contours (high degree spline) as outer boundary and primitives for inner obstacles.

Figure 3.2. Examples of 2D stationary environment representations (idealized)[Szlachetka et al., 2020].

3.1 Non parametric models

Typical feature of non-parametric models is a relatively dense discretization of the model domain. Each cell of this discretized domain stores its own parameters that describe only the matching portion/part of the model domain. It is quite common that parameter sets are decoupled, that is, the parameters of a particular cell are independent of those of other cells. Then each cell parameter can be estimated independently of the others. The number of domain cells does not change with time (that is, in subsequent identification of the model or its refinements). An example of a parametric model of a dynamic system can be its frequency response obtained by means of the discrete Fourier transform. In the case of stationary environment modeling, this domain can be either a longitudinal position along the host path (projected and past), a ground plane, a volume, or an azimuth angle anchored in the middle of the host vehicle.

3.1.1 Grid maps

Grid maps come from robotics [Urmson et al., 2008] and their basic idea is a dense and uniform discretization of the environment. In other words, grid maps represent the world in the form of square cells in 2D grids or cuboids in 3D grids.

3.1.1.1 Occupancy grid

The occupancy grid is the most widely used type of grid map. It can handle a wide spectrum of sensor types and can support obstacles detection, objects tracking, and the host vehicle localization. Each cell of the occupancy grid contains information about the probability that the cell is occupied, that is, not drivable for the host vehicle. A high probability in a given cell indicates that the system is confident about the existence of a real obstacle in this cell, and thus the host should not move there.

In general, a 2D occupancy grid (Figure 3.2a) can be modeled as follows:

$$\mathbf{M}(t) = \left\{m_{i,j}^{t}\right\} \tag{3.1.1}$$

$$m_{i,j}^{t} = P_{obstacle}\left(m_{i,j} | (x_i, y_i)\right) \tag{3.1.2}$$

$$\bigvee_{i \neq a, j \neq b} : m_{i,j} \cap m_{a,b} = \emptyset \tag{3.1.3}$$

t	– time
$\mathbf{M}(t)$	– occupancy grid matrix in time t
$m_{i,j}^{t}$	– nondrivable obstacle probability in (i, j) grid cell in time t
x, y	– directions in cartesian grid
i, j, a, b	– grid cell indexes

Putting it differently, each cell is independent and describes a small part of the environment. A single cell (as well as the entire grid) is immobilized to the ground. In this way, information about specific portions of the stationary environment is well accumulated within the cells over time. In practice, working with probabilities (3.1.2) is mathematically inconvenient. Therefore, the odds[1] are used instead. At each

[1]The odds are defined as the probability that the event occurs divided by the probability that the event does not occur

iteration of the estimation of the occupancy grid parameters, new sensor measurement data is utilized to obtain the best possible reflection of the real world. A new measurement $z(t_n)$, at iteration t_n, is fused with the prior cell state $m_{i,j}^{t_{n-1}}$ to obtain the posterior cell state $m_{i,j}^{t_n}$. Fusion of measurement data with the occupancy grid is done with the help of sensor modeling, which transfers as much information as possible from the detection model to the occupancy grid. There are two families of sensor models known in the literature, namely forward sensor models [Carvalho and Ventura, 2013] and inverse sensor models [Andriamahefa, 2017].

The integration of data from various sensors is vital in occupancy grid processing. Through data fusion, the information obtained from each sensor is combined to create a detailed and precise representation of the environment. Prior to fusion, inverse sensor modeling is applied to each sensor detection to ensure its accuracy. This results in the creation of a reliable and accurate occupancy grid, providing autonomous vehicles with a reliable tool for safe navigation and operation.

The occupancy grid can be defined using one of the three main probability frameworks. They are the Bayesian probability framework [Parsons, 2006], Dempster-Shafer Theory (DST) [Challa and Koks, 2004] and Dezert-Smarandache Theory [Smarandache and Dezert, 2005]. The Bayesian probability assumes only two states of a cell, i.e. "occupied" and "free". These states are mutually exclusive. This means that the sum of probables is equal to 1. Once the probability of being "occupied" is decreased, then the probability of being "free" is increased at the same time. The Dempster-Shafer model extends the Bayesian probability framework by introducing the "unknown" hypothesis. Thus, there is no direct connection between the "occupied" and "free" states. The last framework, Dezert-Smarandache Theory (DSmT), goes even further and introduces the intersection of states, i.e. "occupied and free". The DSmT is a generalization of the DST.

It is important to note that the utilization of the occupancy grid maps is not done in the same way as is done in the robotic field. There are several assumptions specific to the automotive field only [Porebski, 2022]:

- each cell is an independent Markov stochastic process,
- the host vehicle position on the gird is known,
- the occupancy grid is a square (and each cell too) map limited only to the host local surrounding environment,
- the occupancy grid map can only be shifted relatively to the fixed world frame without rotation.

3.1.1.2 Multi-dimensional 2.5/3D grids

Two-dimensional (2D) grids naturally expand by adding a dimension in the vertical direction. Early work on 3D grids can be found in [Moravec, 1996] and [Roth-Tabak and Jain, 1989]. Three-dimensional (3D) grids are usually composed of cubes (or cuboids) and can contain information similar to that of 2D grids. The advantage of a 3D grid over a 2D is the ability to see the height of ground obstacles and height of overhanging obstacles such as bridges or road signalization. It is crucial information for determining traversability. Cars can drive under bridges and over some low road structures like curbs (if the curb height is low enough). 3D grid maps are not so popular in ADAS due to significant memory size increase in comparison to 2D grids. Nevertheless, in the literature, some improvements in optimizing 3D grid memory size can be found as an octomap [Wurm et al., 2010]. The Octomap model is based on the octree data structure. Each cell in the octree can be recursively divided into eight subvolumes. This enables lossless compression, which significantly reduces memory size. Another way to have an applicable ADAS model that allows determining traversability is to extend the 2D grid by only height/overhang information. An example can be the "elevation map" [Nam et al., 2017] or [Gutmann et al., 2005]. This approach is good enough to describe an environment that is composed only of grounded obstacles (curbs, stopped cars, guardrails, dots, etc.). It cannot handle overhangs or vertical structures that are not connected (not

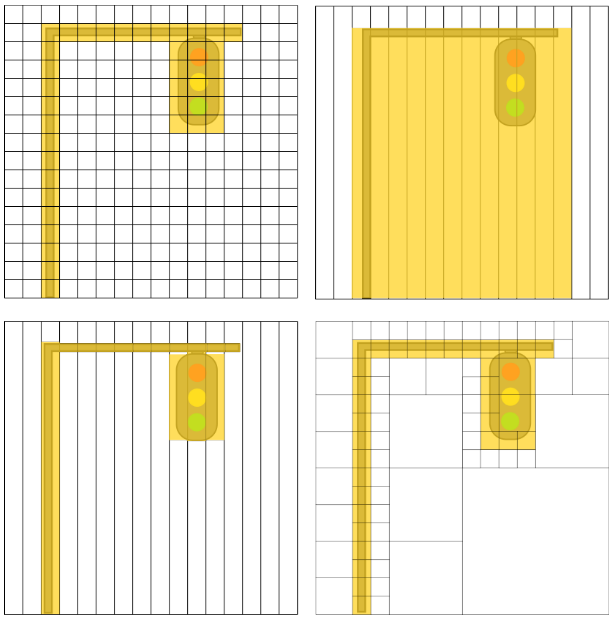


Figure 3.3. Example of vertical cross-section of a traffic light represented in 2.5D/3D grid maps. Top left: 3D grid, top right: elevation map (2.5D), bottom left: multi-level surface map (2.5D), bottom right: Octomap (3D). The occupied area is shown in yellow. Ref. [Szlachetka et al., 2020]

touching) to the ground. This drawback can be overcome by approaches such as the "multi-level surface map" [Triebel et al., 2006]. It allows storage of multiple surfaces in each cell of the grid. It has the ability to describe overhanging and vertical obstacles with a high level of accuracy while having a low memory requirement. All grid maps models are illustrated in Figure 3.3.

3.1.2 Interval maps

This model can be treated as a special one-dimensional case of a 2D grid map [Weiherer et al., 2013][Weiherer et al., 2012]. An example of such a model is shown in (Figure 3.2b). It discretizes the surrounding environment along the longitudinal direction of the host car. Instead of square cells, there are intervals (I) of given length and width. This implies that the lateral information is continuous (not discrete) and thus more precise than the longitudinal information, which is discrete. It is more profitable for a highway scenario where lateral information is more important than longitudinal information. Within each interval, information regarding the lateral span of obstacles is stored within this interval. It can use following forms: a point structure (PC) describing obstacles which vertical span is low (e.g. fence or guardrail) or an interval structure (IC) describing obstacles which vertical span is significant (e.g. parked car or building).

$$\mathbf{I}_i^{(t)} = \left[PC_1^{(t)}, ..., PC_k^{(t)}, IC_1^{(t)}, ..., IC_m^{(t)}\right] \tag{3.1.4}$$

$$\mathbf{PC}^{(t)} = \left[y^{(t)}, \phi^{(t)}\right] \tag{3.1.5}$$

$$\mathbf{IC}^{(t)} = \left[y_{start}^{(t)}, \phi_{start}^{(t)}, y_{end}^{(t)}, \phi_{end}^{(t)}\right] \tag{3.1.6}$$

t – time

$\mathbf{I}_i^{(t)}$	– i-th interval at time t
$\mathbf{PC}_k^{(t)}$	– k-th point structure at time t
$\mathbf{IC}_m^{(t)}$	– m-th interval structure at time t
y	– obstacle lateral position
ϕ	– obstacle orientation
x	– longitudinal position (used in case of an associated moving object to a cell)
$P(o\|z)$	– posterior occupancy probability
n_{age}	– the age of the cell
ID_{object}	– the ID of an associated dynamic object

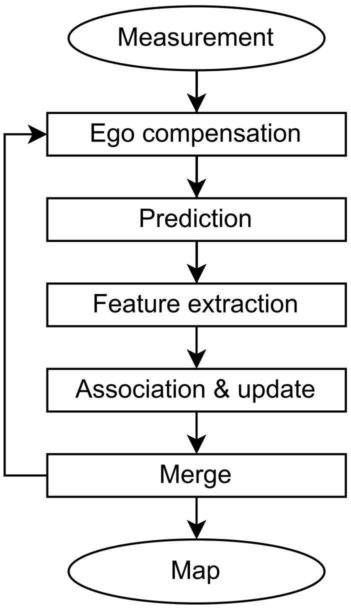


Figure 3.4. The interval map parameters estimation flow diagram.

The authors then extended the basic method by a probabilistic approach that can track the probability of occupancy in each interval instead of just binary information about occupancy. They introduce an occupancy interval structure (IC_{oc}).

$$\mathbf{IC}_{oc}^{(t)} = \left[y_{end}^{(t)}, \sigma^2(y_{end})^{(t)}, x^{(t)}, P(o|z^{(1:t)}), n_{age}^{(t)}, ID_{object}^{(t)}\right] \tag{3.1.7}$$

x	– longitudinal position (used in case of an associated moving object to a cell)
$P(o\|z)$	– posterior occupancy probability
n_{age}	– the age of the cell
ID_{object}	– the ID of an associated dynamic object

Each interval is divided into cells as shown in Figure 3.2b. A cell is a lateral span that has some probability of being occupied. The lateral span of a cell is defined by its two borders represented by continuous (floating) values. An estimation algorithm for the parameters of the interval map is shown in Figure 3.4. The estimation starts with the host motion compensation, which compensates for: longitudinal movement, lateral movement, and rotation. In each iteration, if the longitudinal forward host position increase is greater than the longitudinal span of a single interval, the intervals' information is shifted toward the host as in a shift register. A new interval then appears at the front far end, whereas the interval behind the host is discarded.

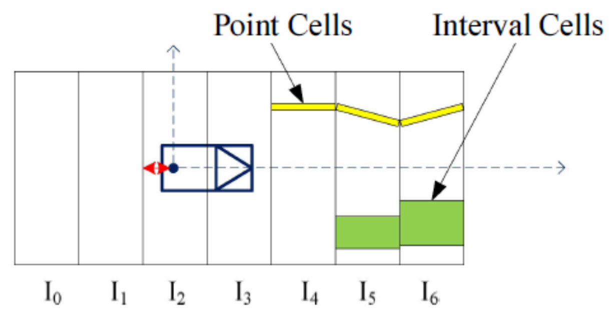


Figure 3.5. Visualization of the PC rotation component [Weiherer et al., 2013].

These cells borders make some state, which is tracked by the Kalman filter. The prediction stage is done on the basis of cell borders from previous iteration and host movement kinematic model. The predicted cell borders in a given interval are then updated by new measurements detected in this interval (if any, Figure 3.6). To avoid too much fragmentation of cells within a single interval, a merging of cells having

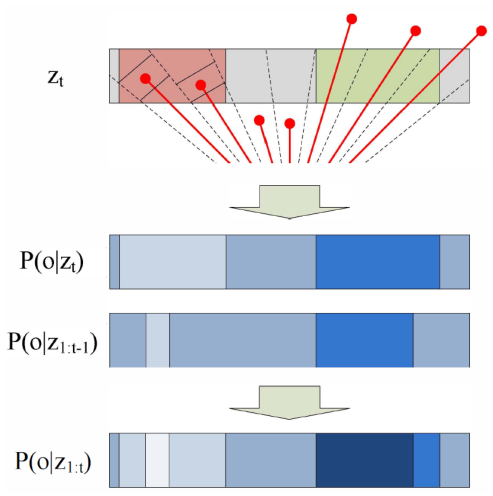


Figure 3.6. The extraction of measurement data from raw sensor data into IC representation [Weiherer et al., 2013].

similar occupancy probability is performed. The interval map can also utilize information about moving objects (provided by the moving object tracking system). Once a cell is associated with a moving object, the object ID is written to the cell.

The interval map model can be extended even further by dynamically bending the entire interval map according to the curvature of the host trajectory [Li et al., 2018]. This extension makes the model more suitable for curvy roads.

3.2 Parametric models

A parametric model is characterized by having a limited set of parameters that define the model in its entire operational domain. Usually, parameters cannot be estimated independently of others.

3.2.1 Primitive structures

Primitive structures are the simplest models. Structures are geometric shapes such as circles and quadrilateral or simple boundaries, for example, lines (Figure. 3.2c). Taking [T.R. and M., 2019] as an example, an obstacle detection algorithm process is shown in Figure 3.7. LIDAR detections are used as input.

Despite the fact that most modern lidars, installed on car's roof, provide in each scan a 3D point cloud describing the environment around the car (full 360 degree view seen at several fixed elevation angles), the one used in this example is a forward looking only planar lidar. It scans the environment in front of the vehicle in a single horizontal plane perpendicular to the ground. The lidar is mounted on the front of the vehicle and provides a single line of detections. Each detection carries information about the azimuth angle and range r, i.e., the distance to the closest non-drivable obstacle on a given azimuth.

The algorithm starts with temporal-spatial filtering of lidar points by applying a 3x3 median filter:

$$\begin{bmatrix} r_{i-1}^{t-1} & r_{i}^{t-1} & r_{i+1}^{t-1} \\ r_{i-1}^{t} & r_{i}^{t} & r_{i+1}^{t} \\ r_{i-1}^{t+1} & r_{i}^{t+1} & r_{i+1}^{t+1} \end{bmatrix} \tag{3.2.1}$$

t	–	current time
i	–	lidar scan azimuth index
r_i^t	–	distance to the closest obstacle on the i-th azimuth

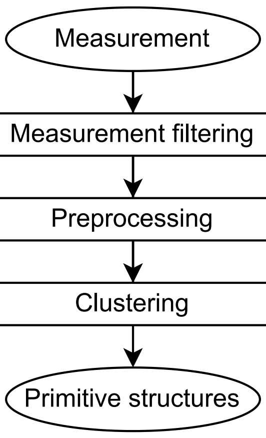


Figure 3.7. The interval map parameters estimation flow diagram.

Points, filtered in the preprocessing step, are then grouped into blocks based on the consistency rule, i.e., the blocks are separated by gaps in the points. These blocks are merged or split afterwords based on the distance between blocks (i.e., the distance between the border points of neighboring blocks). In the clustering step, all created blocks are evaluated into one of the following shapes:

- **circle** - if a block has five or fewer points. The center of the circle is the midpoint of the line that connects the first and last points with a radius equal to the maximum distance between the midpoint and all points within the block.
- **line** - if a block has more than five points. The line is created at the first and last points. The distance between the line and each point of the block must be less than 0.1% of the line length.
- **quadrilateral** - same requirements as for a line, but the distance between the line and each point of the block must be greater than 0.1%. The block endpoints are two of the quadrilateral vertices, and the two points with the longest distance from the line are the other two points of the quadrilateral.

3.2.2 Contours

As mentioned in section 2.4.3.1, contours define a non drivable border. They can be open or closed. The following subsections describe them on the basis of examples.

3.2.2.1 Open contours

Open contours are used to describe road structures distributed/located along the road such as guardrails, tunnel boundary or anything that limits a road on one or both sides (left and right) (Figure 3.1). Open contours can be represented by polynomials, a clothoid, or a spline.

In mathematics, a polynomial is an expression consisting of indeterminates (also called variables) and coefficients as follows:

$$f(x) = c_k x^k + c_{k-1} x^{k-1} + ... + c_1 x + c_0 \tag{3.2.2}$$

x	–	variable
c_i	–	coefficients
k	–	polynomial degree

Usually, the 1D polynomial domain is oriented parallel to the direction of the movement of the host car to estimate the parameters of road structures such as highway guardrails. From a mathematical point of view, a polynomial has no limits. In automotive, it is important to specify obstacle size such as guardrail length. Thus, models like 3.2.2 are limited by x_{min} (lower bound) and x_{max} (upper bound). Figure 3.8 shows these limits.

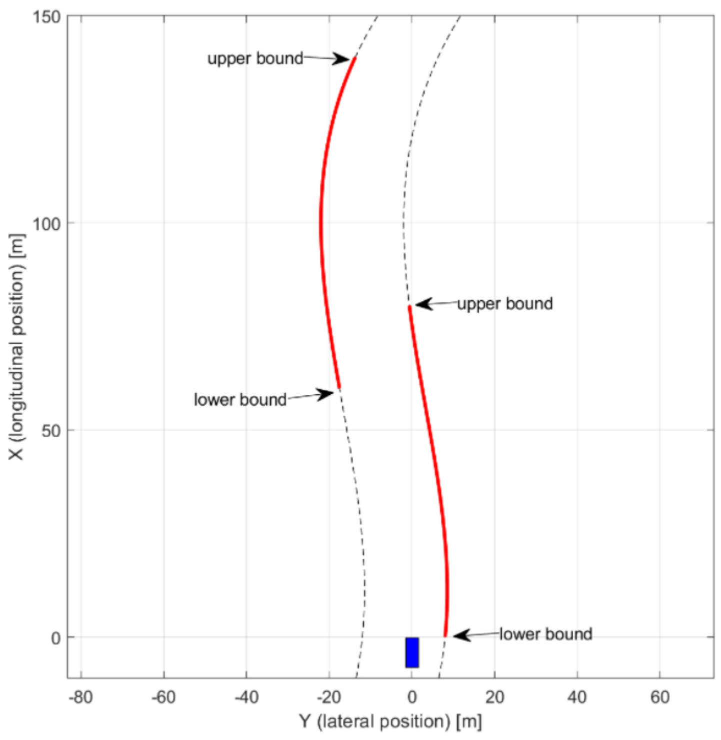


Figure 3.8. Polynomials that approximate the road corridor with their longitudinal limits.

A clothoid, also called an Euler spiral, is a curve which curvature changes linearly with its curve length:

$$
\begin{aligned}
x &= a\sqrt{\pi}\int_0^t cos\left(\frac{\pi t^2}{2}\right)dt \\
y &= a\sqrt{\pi}\int_0^t sin\left(\frac{\pi t^2}{2}\right)dt
\end{aligned}
\tag{3.2.3}
$$

t	–	parameter $t = \frac{s}{a\sqrt{\pi}}$
a	–	coefficient from the equation expressing the proportionality of the curvature κ to the length of the arc $\kappa = \frac{s}{a^2}$
s	–	arc length

A vehicle moving along a clothoid with constant linear velocity has uniform angular acceleration and uniformly increasing centrifugal force. Therefore, clothoid functions are often used in the design of highway interchanges and lane markers.

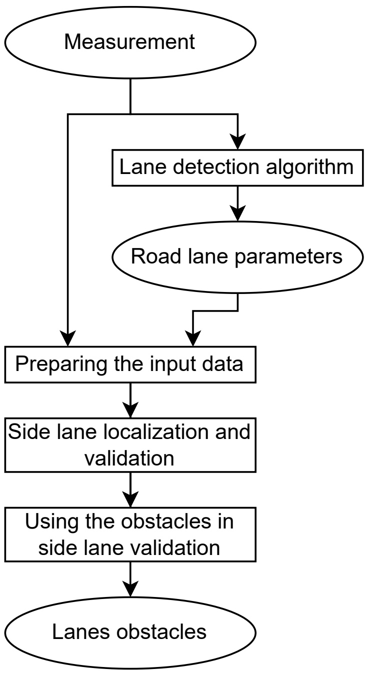


Figure 3.9. The clothoid parameters estimation flow diagram

Taking as an example [Danescu et al., 2006], an estimation of open contour boundaries parameters, in the form of a clothoid, is shown in Figure 3.10.

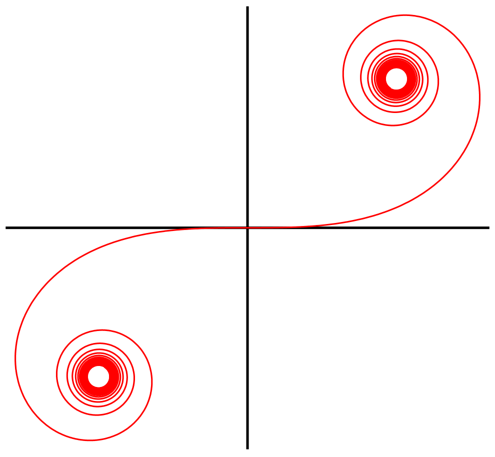


Figure 3.10. The shape of a clothoid.

The algorithm consists of two steps: lane detection where the host car is currently and side lanes (boundaries) parameters estimation. Lane detection is done using *3D lane detection system* [Nedevschi et al., 2004b]. The boundaries parameters estimation starts with preparing of 3D points input data. First, all points that do not belong to the road surface are filtered out. For the remaining points, their lateral parameters (in the curvilinear coordinate system) are estimated and transformed into the car coordinate system. This step can be treated as straightening the set of curved points.

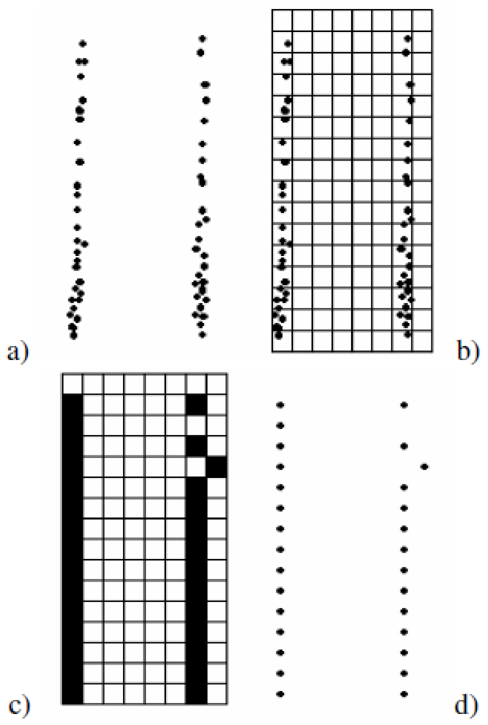


Figure 3.11. Point uniformization: a) Original set of points, b) Added grid on points, c) Evaluated grid based on points, d) Uniformed set of points, Ref: [Danescu et al., 2006]

The algorithm then obtains uniform points using a simple grid (Figure 3.11). All points are collected in cells on the basis of the straighten position. Each cell, which contains some points, generates a uniform point. Uniform points that fall within the search areas of the left and right lane (search areas generated by prediction using past results and associated uncertainties) are used to construct two histograms. A histogram (count histogram) represents the distribution of points versus the lateral position. Another histogram is a weighted one, where each entry represents the sum of weights of points corresponding to the lateral position. On the basis of the mentioned histograms, a validation of the side lines is done. A valid side line can be invalidated by the presence of an obstacle within a 15-30 m longitudinal interval afterward. An obstacle is detected based on [Nedevschi et al., 2004a].

Another suitable model to describe a contour is a spline curve. It is made up of connected polynomials of fixed and low degrees (usually up to the third degree). The smoothness of a spline curve depends on the continuity of its higher derivatives at knots (spline domain points at which two adjacent polynomials connect). The first degree spline (consisting of polynomials of the first degree, that is, a piecewise linear spline) is just a polyline (Figure 3.2d). Its segments are continuous, while its first derivative is discontinuous at knots (it is piecewise constant), so it is not smooth. The smoothness of the spline increases with the degree of the spline because a higher order of spline derivatives can be continuous.

The spline is a special type of curve, which is defined in piecewise polynomials. Each polynomial can have a very simple form, yet a whole spline is flexible and can reflect complex shapes at the same time. The spline *r* in one variable $s \in [a, b]$ is defined as [Micula and Micula, 2012]:

$$r(s) : [a, b] \to \mathbb{R} \tag{3.2.4}$$

The interval $[a, b]$ can be defined as *k* ordered, disjoint subintervals:

$$[t_i, t_{i+1}] \quad i = 0, ..., k-1 \tag{3.2.5}$$

t_i – i-th knot

Thus

$$\begin{gathered} [a, b] = [t_0, t_1) \cup [t_1, t_2) \cup ... \cup [t_{k-1}, t_k) \cup [t_k] \\ a = t_0 \leq t_1 \leq ... \leq t_{k-1} \leq t_k = b \end{gathered} \tag{3.2.6}$$

Each piece-wise polynomial *P* can be defined as:

$$P_i : [t_i, t_{t+1}] \to \mathbb{R} \tag{3.2.7}$$

By using equation 3.2.7 in equation 3.2.4, we get the following.

$$r(s) = \begin{cases} P_0(s), & t_0 \leq s < t_1 \\ P_1(s), & t_1 \leq s < t_2 \\ \quad ... & \\ P_{k-1}(s) & t_{k-1} \leq s \leq t_k \end{cases} \tag{3.2.8}$$

Additionally:

- The spline $r(s)$ is said to be of degree n when each $P_i(s)$ is at most degree n.
- The vector **t** $= (t_0, t_1, ..., t_k)$ is known as the knot vector. If the values (knots) in the vector **t** are equally distributed in the whole s—-interval, then the spline is uniform, otherwise non-uniform.

3.2.2.2 Closed contours

Closed contours are used to describe the free space boundary enclosing the host car, within the host car can move without making a collision with stationary obstacles. They can also represent complex structures mapped around (of known ground-plane intersection shape).

The polygon, also known as the closed polygonal chain, is the basic example of a closed contour (Figure 3.2e). It is just a series of connected line segments. More segments are needed to achieve better accuracy of complex shape approximation. In most cases, the number of segments is limited, which affects the accuracy of the approximation. This problem can be solved using a more flexible contour representation, such as the spline of a second or third degree (Figure 3.2f).

An example of the determination of the polygonal chain algorithm described in [Gex and Campbell, 1987a] and its flow is shown in Figure 3.12.

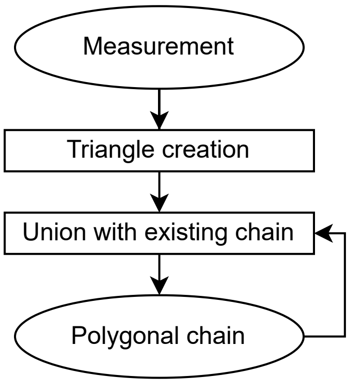


Figure 3.12. The polygonal chain parameters estimation flow diagram

The algorithm starts with the creation of triangles. The triangle represents the acoustical cone dispersion in two dimensions. One of the triangle vertex is located at the sensor position. The space inside the triangle is free from obstacles. The size of the triangle is the distance to the detected obstacle or the maximum detection range of the sensor (distance) if no obstacle is detected (distance between point A and segment CB in Figure 3.13). The triangle is fully determined in absolute coordinates using the position and heading of the vehicle, the position of the sensor, the direction, and the return. Once the triangle is ready, it is combined with an existing chain as a union of both. The figure. 3.13 shows the union example. The ABC triangle has intersections with existing chains 0-14 at points Q and P. As a result, the boundary is extended between points 6 and 7 by the PBCQ chain.

A more complex approach to describe the surrounding environment, called the "Parametric Free Space" (shortly PFS), is proposed in [Schreier et al., 2016]. It uses a 2D parameteric uniform periodic B-Spline (a special case of a spline curve) to model outer boundary of the free space and primitives for interior obstacles.

The parametric curve $r(s)$ used by the PFS algorithm is defined as follows:

$$\mathbf{r(s)} = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} \mathbf{B}(s)^T & \mathbf{0} \\ \mathbf{0} & \mathbf{B}(s)^T \end{bmatrix} q_P \tag{3.2.9}$$

$$\mathbf{B}(s) = [B_0(s)\ B_1(s)\ ...\ B_{N_b-1}(s)]^T \in \mathbb{R}^{N_b} \tag{3.2.10}$$

$$\mathbf{q_P} = [q_{x,0} \cdots q_{x,N_b-1}\ q_{y,0} \cdots q_{y,N_b-1}]^T \in \mathbb{R}^{2N_b} \tag{3.2.11}$$

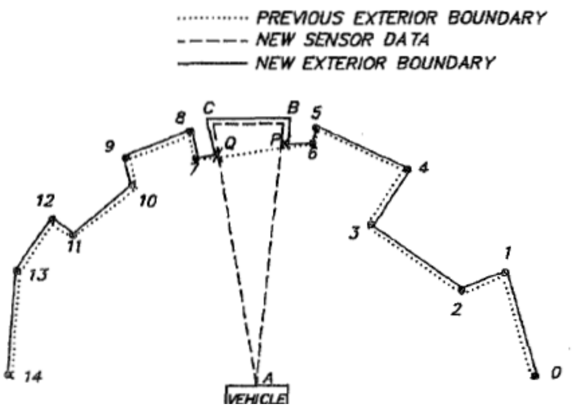


Figure 3.13. Polygonal chain union with the sensor triangle. Ref: [Gex and Campbell, 1987a].

$x(s)$	–	longitudinal component of the parametric curve
$y(s)$	–	lateral component of the parametric curve
$B(s)$	–	quadratic basis function used to generate the B-spline curve
$\mathbf{q}_P$	–	control points vector
N_b	–	number of basis functions (control points)

B-spline periodicity is a feature that allows the curve to be closed while the curve endpoints are joined as smoothly as all polynomial segments of the curve. The flow of the PFS algorithm is shown in Figure 3.14.

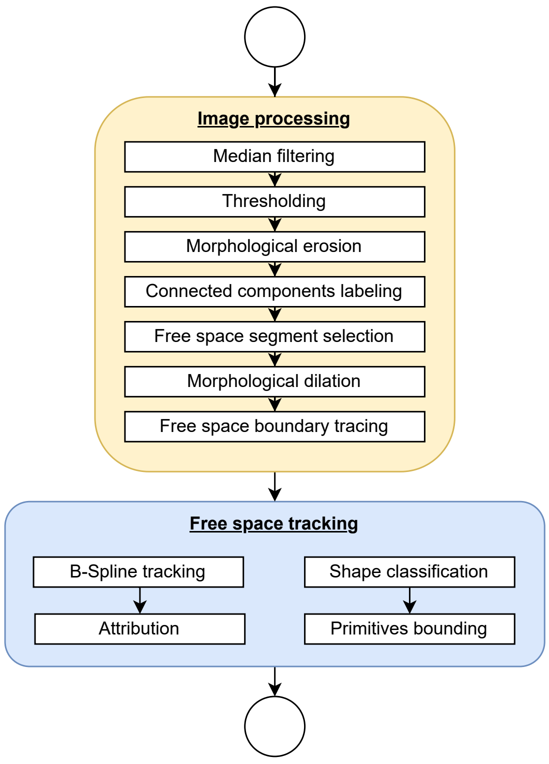


Figure 3.14. The flow diagram of the PFS algorithm.

It can be noticed that PFS's goal is not to estimate free space area parameters (it is already done by the occupancy grid) but to limit the amount of data provided to a receiver, i.e., information compression. The size of control points vector defining fully the fee space boundary is significantly smaller than the size of an adequate occupancy grid. Thus, it is better suited to send over automotive interfaces of limited bandwidth. The PFS algorithm is described in detail in Section 4.

3.3 Models comparison

All models mentioned so far are not perfect (issues-free). They were invented to solve some specific problems or to fit specific scenarios and project constraints.

3.3.1 Provided features

Table 3.1 shows a summary of the models mentioned with respect to the features listed below.

- **SI** sensor independence — ability to work with data coming from different sensor types
- **G** granularity — the level of detail to model complex stationary environment
- **T** traversability — the ability to determine passability classes (e.g. underdrivability)
- **C** overall complexity — the memory consumption and computing power requirements of the model

Table 3.1. Models comparison. Ref. [Szlachetka et al., 2020]

Model	**SI**	**G**	**T**	**C**
Interval map	HH	L	NA	L
2D Occupancy grid	HH	HH	NA	M
Elevation map	M	H	L	H
Multi-level surface	M	H	M	H
Octomap	M	HH	HH	HH
3D Occupancy grid	M	HH	HH	HH
Primitives	H	L	NA	LL
Contours	H	M	NA	L

LL - very low, L - low, M - medium, H - high, HH - very high, NA - not available
SI - sensor independence, G - granularity, T - traversability, C - complexity

The most sensor-independent models are those based on spatial discretization, but only in 2D. 2.5D/3D models require sensors that provide height/elevation information. Grids are also the most granulated models. However, the level of detail depends on the resolution of the grid. In terms of traversability, only 2.5D/3D grids can provide obstacle height/overhand information. Grids are mostly the best in this comparison, but their complexity is the highest as well.

3.3.2 Fitting to scenario

Each of the models mentioned can work well in some scenarios, achieving good approximation quality while presenting bad quality in others. Table 3.2 contains summary information on the feasibility of the models for three common scenarios, i.e., highway, city and parking. In a highway scenario, the host car would require information about road limits (usually guardrails or curbstones), as well as information about lane. Simple geometric shapes which are distributed longitudinally as e.g. highway barriers can be sufficiently described by polynomials or clothoids. Thus, contours or interval maps are the most sufficient choices. In automated parking or urban driving situations, a model that can describe complex shapes consisting of road dividers, empty parking slots, parked cars, etc. is necessary. For such scenarios, grids fit the best because they do not make assumptions about the shape of the modeled obstacles. The models are compared using separate driving scenarios. The host car usually drives through all scenarios on a daily basis. Having all models integrated at a time is not sufficient approach, for example, due to memory limitation and model switching issues. Among all models contours seem to be the most universal choice, good for all considered scenarios if system is limited to handle only one model.

Table 3.2. Feasibility of models to different road scenarios. Ref. [Szlachetka et al., 2020]

Model	Highway	City	Parking
Interval map	HH	M	M
2D Occupancy grid	H	HH	HH
Elevation map	M	HH	H
Multi-level surface	M	HH	H
Octomap	M	HH	H
3D Occupancy grid	M	HH	H
Primitives	L	LL	H
Contours	HH	H	H

LL - very low, L - low, M - medium, H - high, HH - very high

The extended comparison of different models can be found here [Szlachetka et al., 2020].

3.4 Typical data flow in free space determination algorithms

All mentioned algorithms have similar information flow, i.e. steps required to go through. Four main steps can be identified (Figure 3.15): measurement preprocessing, host movement compensation, preprocessed measurement utilization, and post updates. The details of each step strongly depend on the

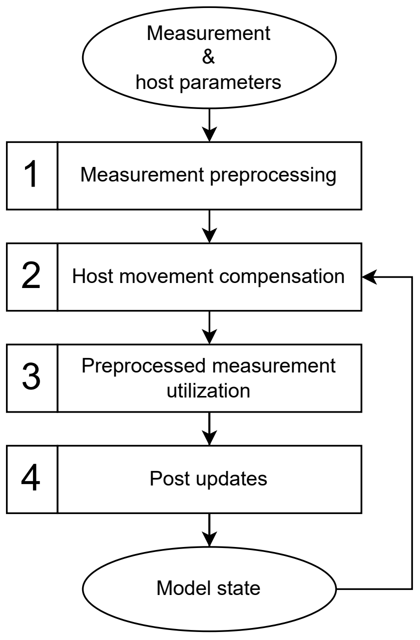

Measurement & host parameters
1
Measurement preprocessing
2
Host movement compensation
3
Preprocessed measurement utilization
4
Post updates
Model state

Figure 3.15. Typical data flow consisting of 4 steps.

type of measurement and the model of stationary environment used.

1. The measurement preprocessing step is characterized by data filtering and/or clustering. The initial number of measurement data is reduced by eliminating false positives and grouping the remainders into clusters.
2. Host movement compensation, also known as time update, is apparently not needed for stationary environment, which does not change in time. However, the coordinate system of stationary environment models is usually not anchored to the ground (fixed world coordinates), but is tied to the host car that is moving. Thus, stationary objects change their host relative position, so the movement of the host car needs to be compensated for. This step also includes clipping areas of the model that move out of the sensors' field of view (areas no longer visible for the sensors).
3. Preprocessed measurement utilization is the step in which preprocessed measurements are used to update the state of the model with the latest available information. This update is often weighted by probabilities of current state, and updating measurements using Bayesan Filter or Kalman Filter.
4. Post updates is a last step, in which different bookkeeping operations are usually done. It strongly depends on the model type. It can be, e.g. merging or splitting of contours, removal outdated parts of the model, or additional classification of traversability.

3.5 Reference choice

Yet despite the flexibility and advantages of the PFS approach, it has several drawbacks and room for improvement. Taking into consideration the entire chapter and the information contained therein on stationary environment models and algorithms used for their identification, the PFS has been chosen as a reference algorithm and a starting point for the development of an improved algorithm proposed in this thesis.

4 Free space boundary tracking by a 2D spline with dynamically adjusted control points

The algorithm described in this section is the main contribution of the author. The proposed algorithm reflects the free space boundary of the stationary world using a closed 2D B-spline as a mathematical model. It focuses on the dynamic control points adjustment, making the approximation of the stationary world more accurate. The form of the algorithm description is as follows. Section 4.1 contains the algorithm interface where the input and output data are described. Then the fundamentals of the system, such as the description of the mathematical model used, are mentioned in Section 4.2. Section 4.3 describes the first part of the algorithm, which is the processing of the occupancy grid. The next Section 4.4 describes an iterative estimation of the B-spline curve that includes elements such as the adjustment of control points. The high-level flow chart of the proposed algorithm is shown in Figure 4.1.

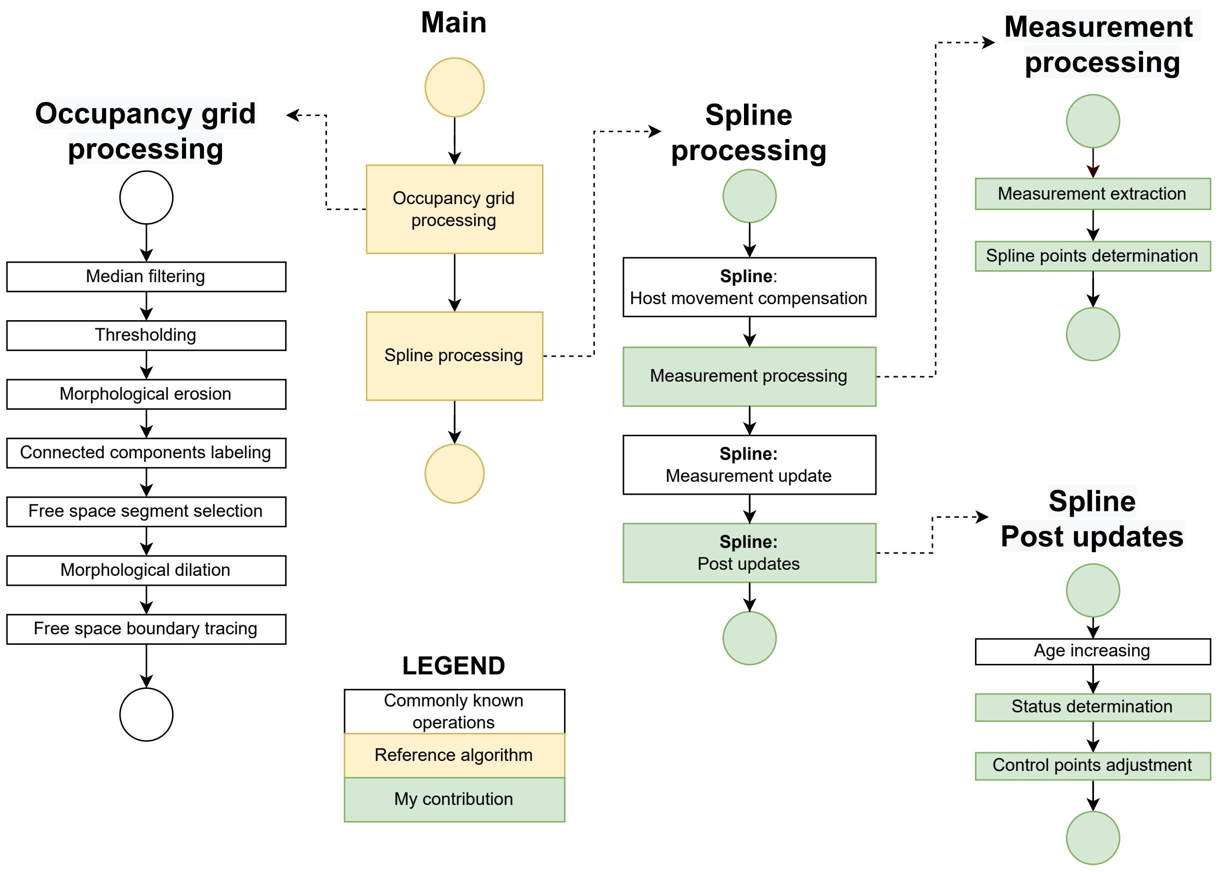


Figure 4.1. The flow chart of the proposed algorithm distinguishing my contribution (original ideas) from the prior art.

4.1 Interface

The interface of the proposed algorithm describes all input data required and defines the output data (Figure 4.2). In the following sections, both input and output data are described in detail.

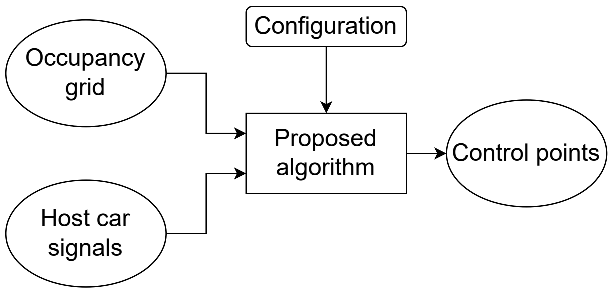


Figure 4.2. The high-level interface of the proposed algorithm.

4.1.1 Input

The proposed algorithm's input data consists of two parts: host signals and the occupancy grid. It has been decided to keep the same input signals and parameters used in the PFS to make the comparison of both algorithms fair. In detail, the input data consists of occupancy grid data and host car data.

4.1.1.1 Occupancy grid data

Configuration:

- **grid length** [m] — *longitudinal size of the grid*
- **grid width** [m] — *lateral size of the grid*
- **cell size** [m] — *length of a side of the square cell of the grid*

Run-time data:

- **occupancy probability matrix** [0-1] - *NxM matrix storing occupancy probability of each cell*

The input occupancy grid is a squared area that represents the surrounding environment around the host car. The grid can move only by the integer multiply of the cell size in either the lateral or longitudinal direction of the grid. The orientation of the grid in the World Coordinate System (WCS) is fixed regardless of the varying host orientation in the WCS. It settles during ignition and does not change afterwards (see Figure 4.3).

4.1.1.2 Host car data

Configuration:

- **host length** [m] - *host car length*
- **host width** [m] - *host car width*

Run-time information:

- **orientation** [rad] — *host car orientation with respect to the grid longitudinal axis*
- **velocity**
  - **longitudinal** [m/s] — *longitudinal component of host velocity (parallel to the grid longitudinal axis)*

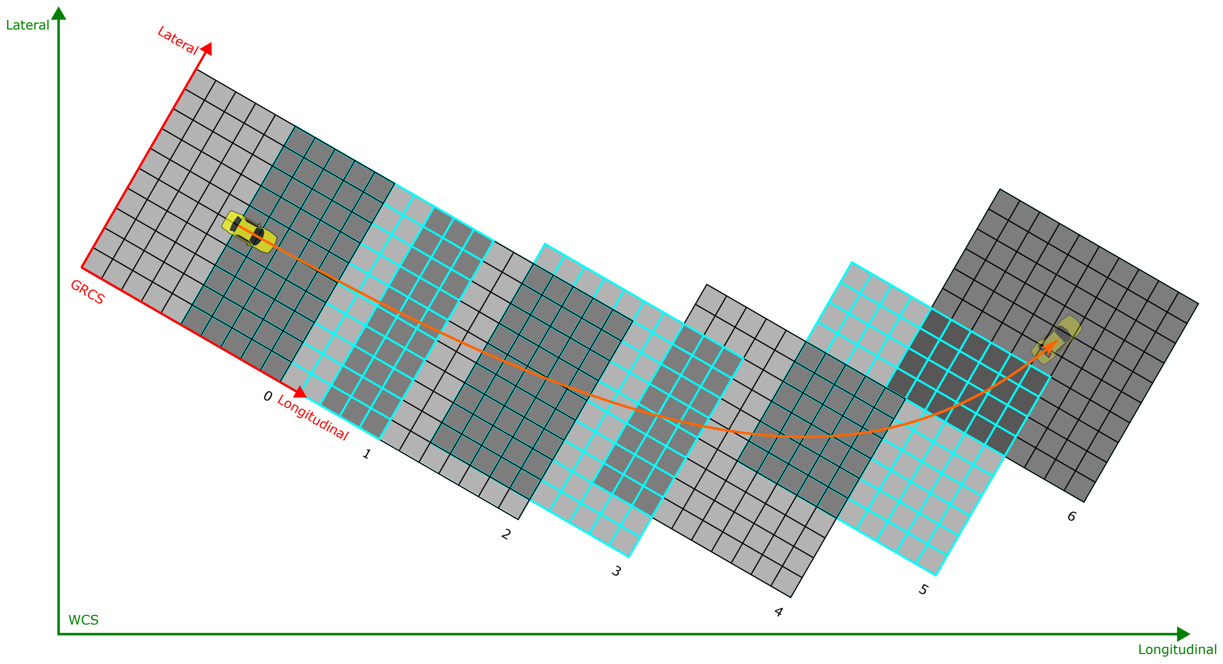


Figure 4.3. Illustration of how the occupancy grid is shifted in 6 consecutive time instants while the host is moving and its orientation varies. The grid at time 0 depicts the initial grid orientation after ignition on.

  - **lateral** [m/s] — *lateral component of host velocity (parallel to the grid lateral axis)*
  - **covariance** $[m^2/s^2]$ — *covariance matrix of host velocity components*
- **position** — position of the host car center
  - **longitudinal** [m] — *longitudinal position of the host center in the occupancy grid*
  - **lateral** [m] — *lateral position of the host center in the occupancy grid*

All of the host car signals are provided in GRCS i.e. in the grid coordinate system. The origin of GRCS is located in the corner of the grid. Figure 4.4 explains some of the input data mentioned. The position of the host car within the occupancy grid is set around the center of the grid. It is not hardcoded due to inconsistency between the host shifting and the grid shifting (host movement compensation) between following cycles.

4.1.2 Output

The output of the proposed algorithm contains the minimum information that the end user needs, sufficient for an unambiguous reconstruction of the free space boundary, i.e. control points position vector. The other necessary part, i.e. the degree of the B-spline is fixed, known to the user, and thus it is not a part of the output.

Properties (in GRCS) of each control point in the output vector (Figure 4.5):

- **position**
  - **longitudinal** [m] — *longitudinal position*
  - **lateral** [m] — *lateral position*
  - **covariance** $[m^2]$ — *covariance matrix of control point position*

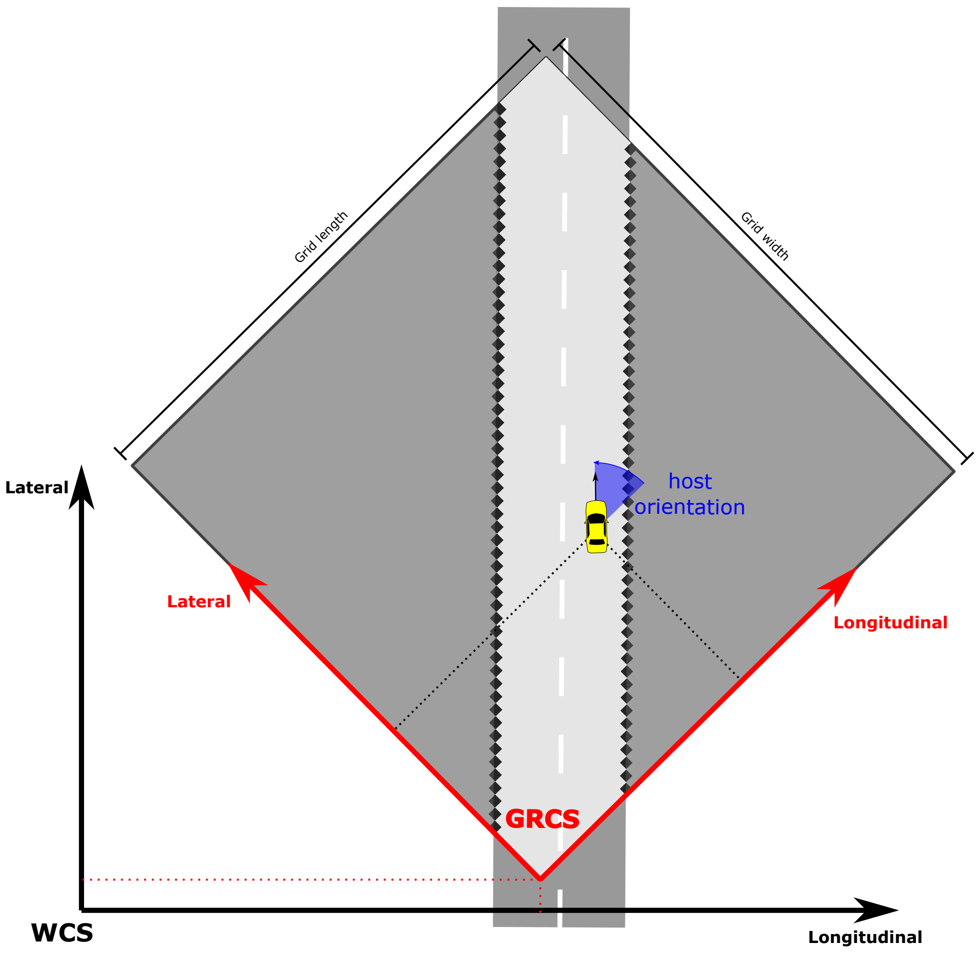


Figure 4.4. Relationship between different coordinate systems.

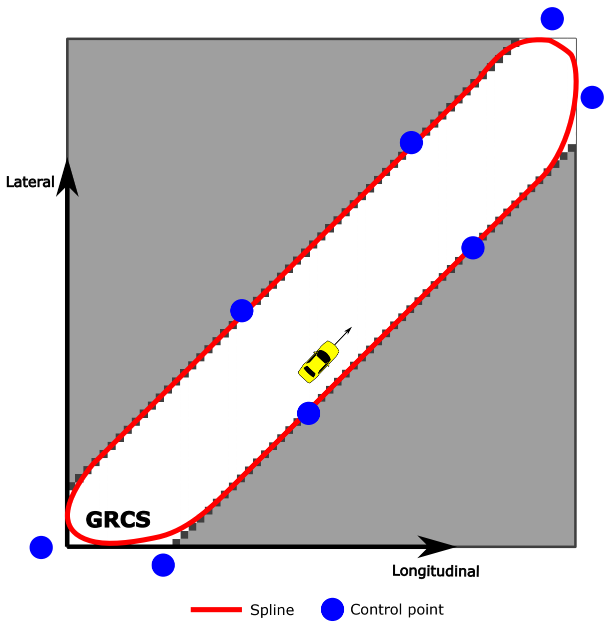


Figure 4.5. The output data within GRCS.

4.2 System fundamentals

The proposed algorithm itself, the input data, the internal data, the output data and all other related elements such as the configuration together form the whole system. This section presents the information about the system starting from the assumptions (4.2.1), through the representation of the internal data structure (4.2.2) and ending with the algorithm initialization (4.2.3).

4.2.1 Main assumptions

The following assumptions were made:

- The free space boundary of the stationary environment is modeled by a 2D closed curve using a B-spline of the second degree.
- The whole system works in discrete time. New input data (sensor information and host signals) is provided at constant rate, that is, each 50 ms.
- The model parameters are refined each time the input data is provided.
- The host car can rotate within the occupancy grid, but the grid itself does not rotate over time with respect to the WCS.
- Generating the occupancy grid is not part of the proposed algorithm. The occupancy grid (occupancy probability matrix) is part of the input data.

4.2.2 Internal representation

The output data is only a subset of the information that is tracked over time. The minimum information allowing the user to reconstruct the spline describing free space boundary is the set of spline control points positions and spline degree. All tracked data is kept in an internal data structure. This section describes the internal data structure as a mathematical model and its extensions.

4.2.2.1 Mathematical model

The proposed algorithm uses a free space model that combines two 1D B-splines into a single 2D parametric spline curve, as follows:

$$\mathbf{r}(s) = \begin{bmatrix} r_x(s) \\ r_y(s) \end{bmatrix} \in \mathbb{R}^2 \tag{4.2.1}$$

where:

s	– free variable, $s \in \langle 0, 1 \rangle$
$r_x(s)$	– longitudinal spline component
$r_y(s)$	– lateral spline component

The set of points on the 2D spline curve in a given tracking cycle can be understood as an observation $\mathbf{r}(s)$ of the system output being a transformation of the state $\mathbf{q}$ through the observation matrix $\mathbf{H}(s)$ as follows:

$$\mathbf{r}(s) = \mathbf{H}(s)\mathbf{q} \tag{4.2.2}$$

where:

$$\mathbf{H}(s) = \begin{bmatrix} \mathring{\mathbf{H}}(\mathbf{s}) & \mathbf{0} \\ \mathbf{0} & \mathring{\mathbf{H}}(\mathbf{s}) \end{bmatrix} \tag{4.2.3}$$

$$\mathring{\mathbf{H}}(\mathbf{s}) = \left[B_1^{[n]}(s)\, B_2^{[n]}(s)\, \cdots\, B_{N_q-1}^{[n]}(s)\, B_{N_q}^{[n]}(s) \right] \tag{4.2.4}$$

$$\mathbf{q} = \begin{bmatrix} q_{x,1} \\ \vdots \\ q_{x,N_q} \\ q_{y,1} \\ \vdots \\ q_{y,N_q} \end{bmatrix} \in \mathbb{R}^{2N_q \times 1} \tag{4.2.5}$$

$\mathbf{H}(s)$	– observation matrix
$\mathbf{q}$	– state vector (complete definition of a B-spline of a given degree)
N_q	– number of control points
$q_{x,i}$	– longitudinal position of the i-th control point
$q_{y,i}$	– lateral position of i-th control point
$B_i^{[n]}$	– i-th basis function of the n-th degree corresponding to the i-th control point.

Each control point $\mathbf{q}_i$ is a point on the Cartesian ground plane, that is, $\mathbf{q}_i = \begin{bmatrix} q_{x,i} & q_{y,i} \end{bmatrix}^T$. The model uses closed uniform periodic B-spline. It is a special case of a B-spline (more details in A.1.1) where the basis functions are wrapped around the entire s-domain and the knots are evenly distributed (Figure 4.6). Thus, the following takes place:

$$\mathbf{r}(s = 0) = \mathbf{r}(s = 1) \tag{4.2.6}$$

And each basis function follows:

$$\begin{gathered} B_i^{[n]}(s) = B^{[n]}(\hat{s} - 1) + B^{[n]}(\hat{s}) + B^{[n]}(\hat{s} + 1) \\ \hat{s} = s - s_i \end{gathered} \tag{4.2.7}$$

$$s_i = \frac{i-1}{N_q} \tag{4.2.8}$$

$$B^{[n]}(s) = \hat{B}_{1,n}^{[n]}(s) \tag{4.2.9}$$

$$\hat{B}_{i,k}^{[n]}(s) = \frac{sN_q + D - i + 1}{k} \hat{B}_{i,k-1}^{[n]}(s) + \frac{i + k - sN_q - D}{k} \hat{B}_{i+1,k-1}^{[n]}(s) \tag{4.2.10}$$

$$D = \frac{n+1}{2} \tag{4.2.11}$$

$$\hat{B}_{i,0}^{[n]}(s) = \begin{cases} 1 & \text{if} \frac{2i-n-3}{2N_q} \le s < \frac{2i-n-1}{2N_q} \\ 0 & \text{otherwise} \end{cases} \tag{4.2.12}$$

where:

i	– basis function index $i \in 1..N_q$
$B_i^{[n]}(s)$	– i-th basis function of degree n.

$B^{[n]}(s)$	– basis even function of degree n.
N_q	– number of control points.

4.2.2.2 Model extension

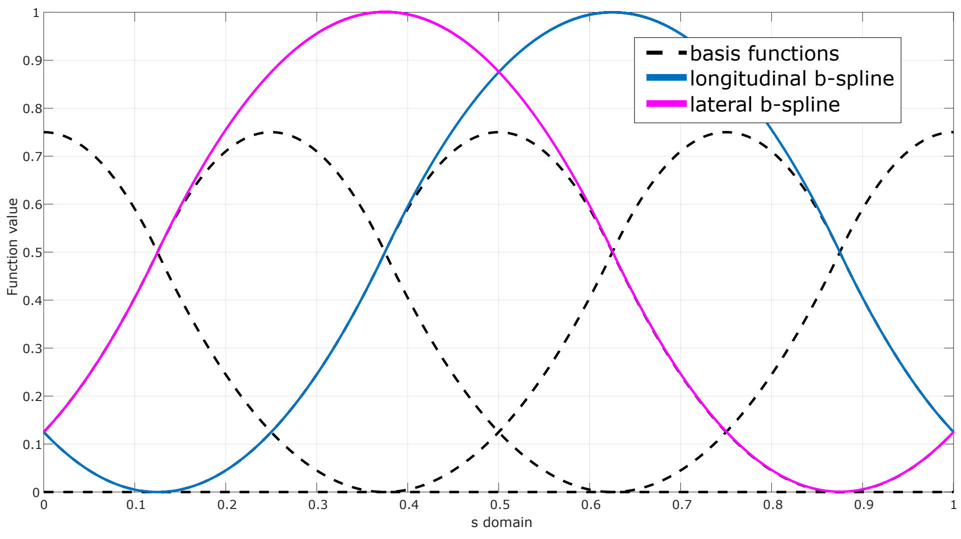
The basic mathematical model (Section 4.2.2.1) is extended by the following additional parameters:

σ_x^2	– longitudinal position variance
σ_y^2	– lateral position variance
$\boldsymbol{\Psi}$	– local shape complexity
$\boldsymbol{\Omega}$	– measurement status
$\boldsymbol{\Phi}$	– approximation error indicator
τ	– number of cycles since the measurement status change.
κ	– number of cycles in which the variance of the position is high. Set to 1 each time the variance drops below the threshold value.

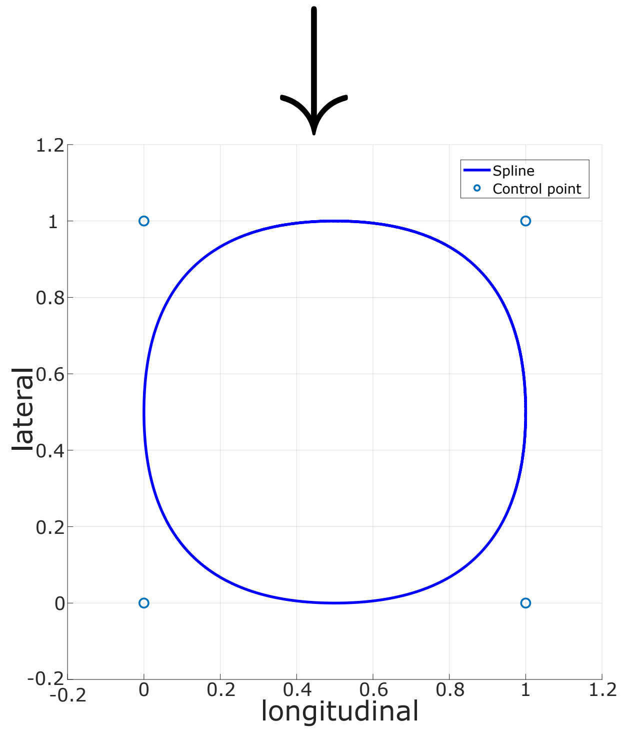
All parameters are described in detail in the following sections.

4.2.3 Before the first call - initialization

Before the proposed algorithm starts to run, it is initialized with a predefined state. The initialization state consists of twenty control points that form a circle. Its center is located in the host car center.



(a) Basis functions with b-spline components in s-domain.



(b) An closed b-spline on 2D plane.

Figure 4.6. An example of a closed curve built on uniform periodic b-splines at four control points, i.e. (0,0), (0,1), (1,1), (1,0).

4.3 Occupancy grid processing

The occupancy grid processing step is responsible for the extraction of measurement points from the occupancy grid. The extracted points represent the discrete boundary of the free space. The term "occupancy grid processing" for this step indicates the use of occupancy grid operations. They are applicable to the occupancy grid, where a cell is treated as a pixel of an image. The occupancy grid processing consists of 7 steps as shown in Figure 4.7. The final result of the occupancy grid processing is shown in Figure 4.8.

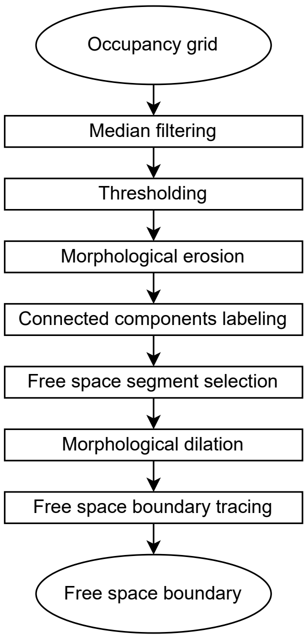


Figure 4.7. The flow of the occupancy grid processing algorithm.

To make it easier to understand, the following symbols are assumed:

E	–	$E = \mathbb{Z}_2 = \{0, 1\}$
$\mathbf{G_o}$	–	occupancy grid
$\mathbf{G_b}$	–	binary grid
c	–	single cell of a grid
$G_o(c)$	–	probability of cell c being occupied
$G_b(c)$	–	binary value (after thresholding) of cell c occupancy probability

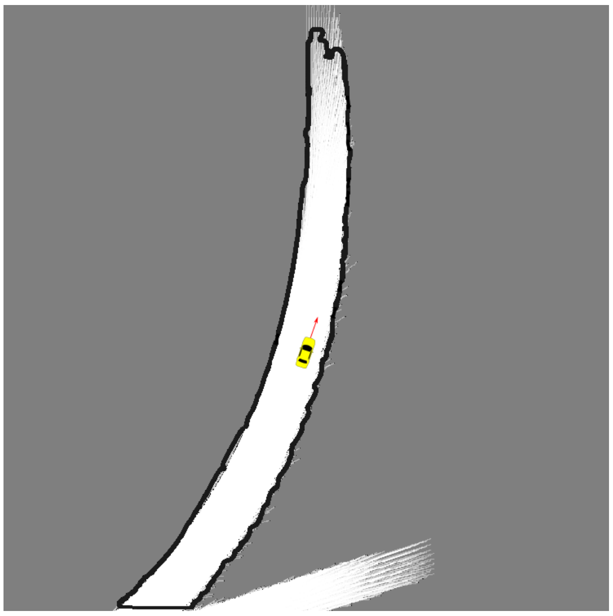


Figure 4.8. The output boundary (bold curve) on an input occupancy grid - result of occupancy grid processing.

4.3.1 Median filtering

The first important step in the processing of the occupancy grid is to remove noise (noisy pixels) while preserving the edges and boundaries of the grid map. It is done by applying in $\mathbf{G_o}$ the 2D median filter with window size I_{median}. The result of the median filtering is shown in Figure 4.9b. Taking as an example $I_{median} = 3$, nine elements are taken on the basis of which the median value is calculated, as shown in the following example.

$$\begin{bmatrix} 0.32 & 0.80 & 0.75 & 0.50 & 0.26 & 0.14 \\ 0.95 & 0.19 & 0.28 & 0.96 & 0.51 & 0.15 \\ 0.03 & 0.49 & 0.68 & 0.34 & 0.70 & 0.26 \\ 0.44 & 0.45 & 0.66 & 0.59 & 0.89 & 0.84 \\ 0.38 & 0.65 & 0.16 & 0.22 & 0.96 & 0.25 \\ 0.77 & 0.71 & 0.12 & 0.75 & 0.55 & 0.81 \end{bmatrix} \rightarrow \begin{bmatrix} 0.32 & 0.75 & 0.75 & 0.50 & 0.26 & 0.15 \\ 0.32 & 0.49 & 0.50 & 0.51 & 0.34 & 0.26 \\ 0.44 & 0.45 & 0.49 & 0.66 & 0.59 & 0.51 \\ 0.44 & 0.45 & 0.49 & 0.66 & 0.59 & 0.70 \\ 0.45 & 0.45 & 0.59 & 0.59 & 0.75 & 0.81 \\ 0.71 & 0.65 & 0.65 & 0.55 & 0.75 & 0.81 \end{bmatrix} \tag{4.3.1}$$

In this example, the boundary of the occupancy grid is padded symmetrically.

4.3.2 Thresholding

Thresholding is a step in transforming the occupancy grid containing continuous probabilities into a binary grid ($\mathbf{G_o} \rightarrow \mathbf{G_b}$). All cells with an occupancy probability greater than $I_{threshold}$ are set to 1 (that is, occupied), and the rest of the cells are set to 0 (that is, free). The result of thresholding is shown in Figure 4.9c.

$$G_b(c) = \begin{cases} 1 & G_o(c) \geq I_{threshold} \\ 0 & G_o(c) < I_{threshold} \end{cases} \tag{4.3.2}$$

4.3.3 Morphological erosion

The morphological erosion operation in the area of mathematical morphology is generally applied to binary images. Erosion of the binary grid $\mathbf{G_b}$ by the structure element $\mathbf{B}$ is defined by the following:

$$\mathbf{G_b} \ominus \mathbf{B} = \{z \in E | B_z \subseteq \mathbf{G_b}\} \tag{4.3.3}$$

$$\mathbf{B_z} = \{b + z | b \in \mathbf{B}\}, \forall z \in E \tag{4.3.4}$$

$\mathbf{B}$ – structuring element
$\mathbf{B_z}$ – translation of $\mathbf{B}$ by the vector $\mathbf{z}$

The consequence of erosion is the reduction of an occupancy grid object, the disappearance of narrow branches and small occupancy grid objects, the elimination of noise, and the expansion of "holes" in an inconsistent area (they take the shape of a structural element). The shape of the structural element $\mathbf{B}$ is a disc shape of a radius $r_{erosion}$ equal to the half width of the host car. In this way, the size of the free space is artificially reduced. Segments through which the host car cannot pass (it just does not fit) are removed. In addition, larger areas of free space, which are connected to each other only by narrow, impassable passages, are separated from each other. The erosion result is shown in Figure 4.9d and in more detail in Figure 4.10.

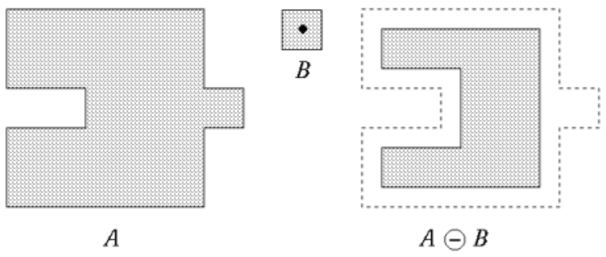


Figure 4.10. Morphological erosion operation. Ref: [Peterlin, 1996].

4.3.4 Connected components labeling

The Connected Component Labeling (CCL) is an algorithmic application of graph theory in which subsets of connected components are uniquely labeled based on a given heuristic. Graphs contain vertices with connecting edges. The vertices contain the information needed for comparison heuristics, while the edges represent the connection between "neighbors". An algorithm traverses a graph and labels the vertices according to the connectivity and relative values of their neighbors. The connectivity is determined by the medium, it can be a 4-connected neighborhood or an 8-connected neighborhood. There are two well-known approaches to do CCL: *multipass* [Rosenfeld and Pfaltz, 1966][Haralick, 1981], *two-pass* [Shapiro and Stockman, 2002][He et al., 2009] and *one-pass* [Johnston and Bailey, 2008][AbuBaker et al., 2007] and *two-pass*. In the case of the occupancy grid processing step, the free grid segments are labeled. The occupied segments are treated as background. The CCL result is shown in Figure 4.9e.

4.3.5 Free space segment selection

Having labeled free space segments, the algorithm chooses the one segment in which the host car is located.

4.3.6 Morphological dilation

The morphological dilation is one of the basic operations in mathematical morphology. The dilation of the binary grid $\mathbf{G_b}$ by the structuring element $\mathbf{B}$ is defined as

$$\mathbf{G_b} \oplus \mathbf{B} = \{z \in E | (\mathbf{B^s})_\mathbf{z} \cap \mathbf{G_b} \neq \varnothing\} \tag{4.3.5}$$

$$(\mathbf{B^s})_\mathbf{z} = \{x \in E | -x \in \mathbf{B}\} \tag{4.3.6}$$

$(\mathbf{B^s})_\mathbf{z}$ – symmetric of $\mathbf{B}$

The dilation operation has the opposite effect with respect to erosion. The consequence of dilation is the enlargement of the occupancy grid object, the disappearance of details, and the filling of "holes" in the inconsistent area. Multiple dilatation joints are often used to achieve the desired effect. In the case of the occupancy grid preprocessing step, morphological dilation is performed on the selected free space segment with the same structuring element as in the erosion operation to bring the reachable free space back to its original size. The result of the dilation is shown in Figure 4.9g and in more detail in Figure 4.11.

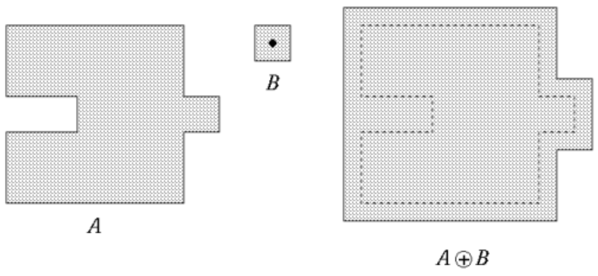
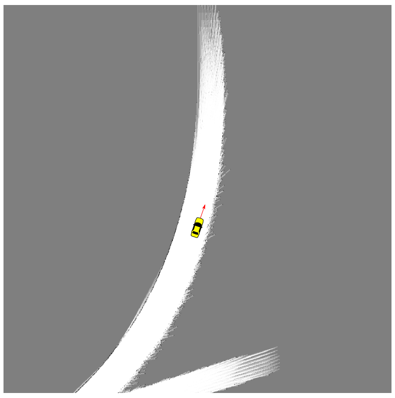
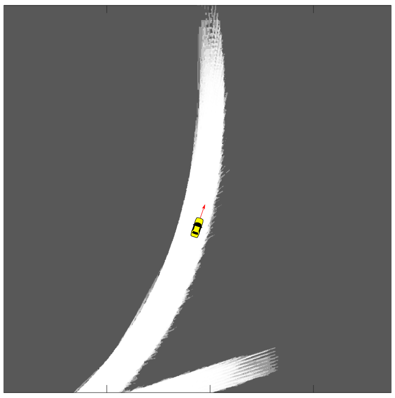


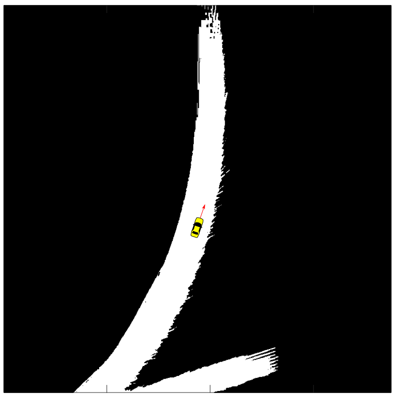
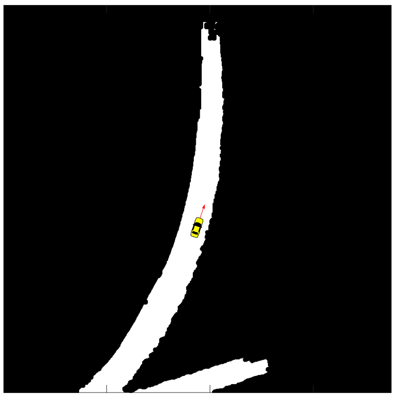
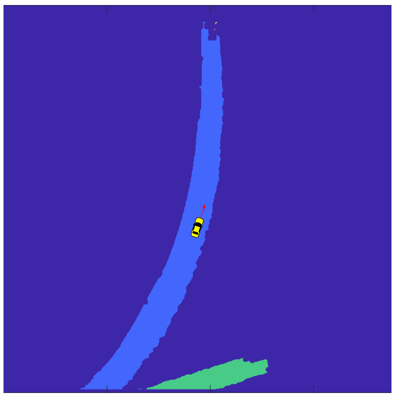
Figure 4.11. Morphological dilation operation. Ref: [Peterlin, 1996].

4.3.7 Free space boundary tracing

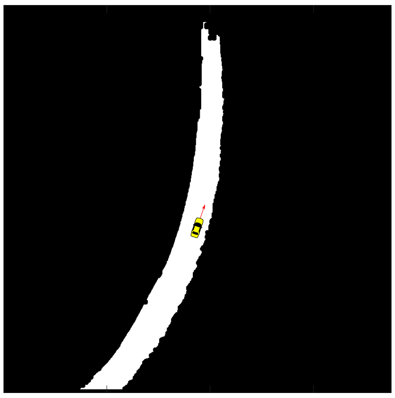
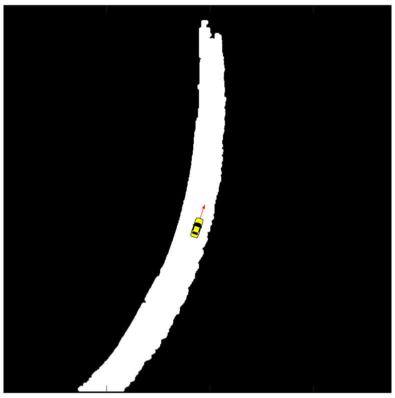
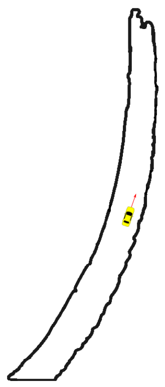
The boundary tracing of the segments of the binary occupancy grid, also known as the contour tracing, can be understood as a segmentation technique to identify the border pixels of the segments of the occupancy grid. Some examples of boundary-tracing algorithms can be found in [Narappanawar et al., 2010] or [Suzuki et al., 1985]. The result of the boundary tracing is shown in Figure 4.9h.

(a) Input occupancy grid. (b) Median filtering.

(c) Thresholding. (d) Morphological erosion. (e) CCL

(f) Selection. (g) Morphological dilation. (h) Boundary tracing.

Figure 4.9. Result of occupancy grid processing steps. Starting from input occupancy grid (a) and finishing with traced boundary as set of pixels (h).

4.4 Closed curve estimation

The main part of the proposed algorithm is the tracking of the boundary of the free space modeled by the closed B-spline curve of the 2^{nd} degree in the ground plane. Tracking (iterative estimation) of a set of control points defining the B-spline uses measurement points provided in each tracking cycle (iteration of the main loop of the algorithm). In the following sections, the single cycle (iteration) of tracking is explained in detail.

The algorithm starts with host movement compensation along with spline prediction (Section 4.4.1). Then measurement processing is performed (Section 4.4.2) where the measurement points are extracted and matched with the predicted spline. It enables the spline update by measurement (Section 4.4.3) followed by the evaluation of the measurement status (Section 4.4.4). Finally, the adjustment of the control points is performed (Sections 4.4.5 and 4.4.6).

4.4.1 State prediction

The B-spline model assumes that the shape being tracked is not movable with respect to the ground. Thus, the state prediction should only impact the state covariance (the state-transition matrix is the identity matrix). However, in each cycle, the host movement over the ground must be compensated. The host changes its position with respect to the ground and to stationary obstacles attached to the ground. When we use the host as the reference frame, then the stationary environment can be seen as moving with respect to the host. As the free space boundary is tracked in time, its B-spline model needs to be shifted with respect to the host by the distance traveled since last tracking cycle, taken with a negative sign. The host pose, that is, its position and velocity, is known with some uncertainty. Thus, the state vector (control points) and the state covariance must be updated due to the host displacement and pose uncertainties. There is no need to include rotation in state prediction because the grid orientation with respect to the ground is fixed and the tracking is performed in the GRCS. Taking into account host rotation and its uncertainty may be needed only by upper software layers to recalculate free space boundary from GRCS to the ISO coordinate system (which origin is set to the center of the host's rear axle while moves and rotates together with the host). The prediction of the state due to host movement is performed as follows.

$$\hat{\mathbf{q}}_{k|k-1} = \mathbf{q}_{k-1|k-1} + \mathbf{B}\mathbf{u}_k \tag{4.4.1}$$

$$\mathbf{B} = \begin{bmatrix} \Delta t & 0 \\ \Delta t & 0 \\ \vdots & \vdots \\ 0 & \Delta t \\ 0 & \Delta t \end{bmatrix}_{2N_q \times 2} \tag{4.4.2}$$

$$\mathbf{u}_k = \begin{bmatrix} u_{x,k} \\ u_{y,k} \end{bmatrix} \tag{4.4.3}$$

$$\hat{\mathbf{P}}_{k|k-1} = \mathbf{P}_{k-1|k-1} + \mathbf{Q}_k \tag{4.4.4}$$

$$\mathbf{Q}_k = \begin{bmatrix} \delta^2_{u_{x,1}} & 0 & & \cdots & & 0 \\ 0 & \ddots & & & & \\ & & \delta^2_{u_{x,N_q}} & & & \vdots \\ \vdots & & & \delta^2_{u_{y,1}} & & \\ & & & & \ddots & 0 \\ 0 & & \cdots & & 0 & \delta^2_{u_{y,N_q}} \end{bmatrix} \Delta t^2 \tag{4.4.5}$$

Information filter matrices are calculated as follows:

$$\boldsymbol{\Upsilon}_{k|k} = \mathbf{P}^{-1}_{k|k} \tag{4.4.6}$$

$$\upsilon_{k|k} = \boldsymbol{\Upsilon}_{k|k}\mathbf{q}_{k|k} \tag{4.4.7}$$

where:

k	–	cycle number
$\mathbf{q}_{k|k}$	–	state vector
$\hat{\mathbf{q}}_{k|k-1}$	–	predicted state vector
$\mathbf{q}_{k-1|k-1}$	–	state vector in (k-1)-th cycle
$\mathbf{B}$	–	control-input matrix
$\mathbf{u}_k$	–	host velocity vector
Δt	–	cycle time (Section 4.2.1)
Q_k	–	process noise matrix
$\delta^2_{u_{x,k}}, \delta^2_{u_{y,k}}$	–	host velocity variances
$\boldsymbol{\Upsilon}_{k|k}$	–	information matrix
$\mathbf{P}_{k|k}$	–	covariance matrix
$\upsilon_{k|k}$	–	information vector

To make a prediction, the following equations are applied.

$$\hat{\boldsymbol{\Upsilon}}_{k|k-1} = \hat{\mathbf{P}}^{-1}_{k|k-1} \tag{4.4.8}$$

$$\hat{\upsilon}_{k|k-1} = \hat{\boldsymbol{\Upsilon}}_{k|k-1}\hat{\mathbf{q}}_{k|k-1} \tag{4.4.9}$$

$$\hat{\mathbf{q}}_{k|k-1} = \mathbf{F_k}\mathbf{q}_{k-1|k-1} = \mathbf{I}\mathbf{q}_{k-1|k-1} = \mathbf{q}_{k-1|k-1} \tag{4.4.10}$$

where:

$\hat{\boldsymbol{\Upsilon}}_{k|k-1}$	–	information matrix in the k-th cycle predicted base on $k-1$-th cycle
$\hat{\mathbf{P}}_{k|k-1}$	–	covariance matrix in the k-th cycle predicted base on $k-1$-th cycle
$\hat{\upsilon}_{k|k-1}$	–	information vector in the k-th cycle predicted base on $k-1$-th cycle
$\hat{\mathbf{q}}_{k|k-1}$	–	state vector in the k-th cycle predicted base on $k-1$-th cycle
$\mathbf{q}_{k-1|k-1}$	–	state vector in $k-1$-th cycle
$\mathbf{F_k}$	–	state transition matrix in the k-th cycle ($\mathbf{F_k}$ = $\mathbf{I}$).

4.4.2 Measurement processing

The measurement processing step focuses on two tasks. They are the extraction of measurement points from the data provided by *occupancy grid processing* and the finding of spline points (points on the spline) corresponding to the measurement points.

4.4.2.1 Measurement extraction

The last step of *occupancy grid processing*, the boundary tracing, provides a set of measurement points that form a closed boundary of the free space. It is not guaranteed that the provided set is sorted (it depends on the algorithm used for tracing). The PFS assumes to get a sorted set of measurement points (pixels). Thus, sorting is necessary. It is done by looking for any boundary point (from which the sorting algorithm starts) and then moving through all points similarly to boundary tracing, but this time assigning an index value to each processed point. Based on the index value, the measurement points are then sorted. The sorted set still forms a closed boundary that does not have a starting or ending point. The "starting" point (first point in the set) is arbitrarily chosen.

4.4.2.2 Measurement downselection

The number of measurements provided by *occupancy grid processing* may be significant. Many of these measurements (pixels) carry redundant information. They do not add value and can be discarded from further processing without affecting the quality of the free-space boundary approximation. It also reduces the computing power demand. This section presents five downselection methods:

1. ray casting,
2. only visible measurments,
3. uniform downselection,
4. line downselection,
5. direction downselection,

Ray casting

The ray casting technique is based on casting a limited number of rays from a specific point (view point) in directions of interest. Each ray provides information about the distance to an obstacle. In the case of the proposed algorithm, the rays are cast in all directions with some increment in the azimuth angle. Once a ray hits a boundary point on the grid, it is recognized as a measurement point (Figure 4.12). In case of multiple obstacles laying on the same ray, the distance to the nearest obstacle is reported. The origin of rays is the center of the host.

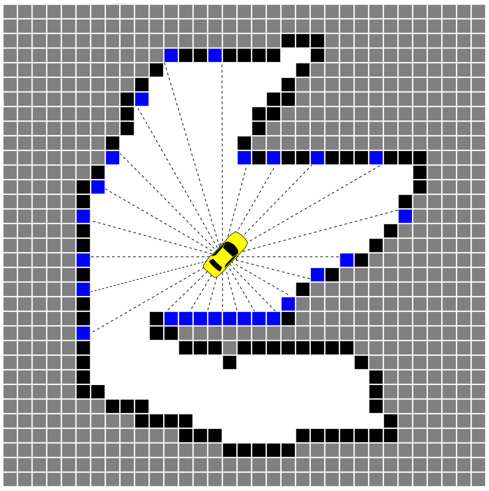


Figure 4.12. Measurement extraction using the ray casting technique. Black cells represent a free-space boundary, and blue cells represent the extracted measurement points.

Only visible measurements

This approach is the technique similar to *ray casting* but using an infinite number of rays. As a result, all visible (from the host point of view) parts of the grid are extracted. Figure 4.13 gives more explanation.

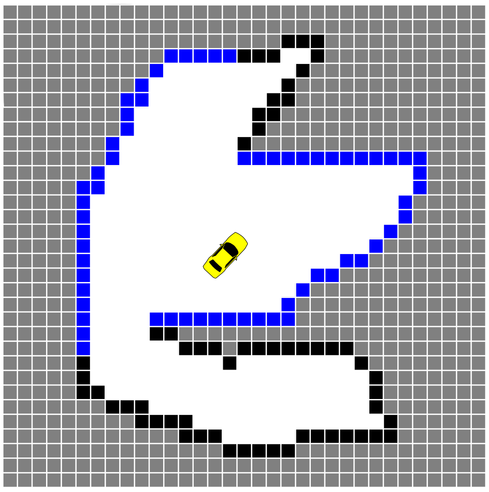


Figure 4.13. Measurement extraction of all visible points. Black cells represent a free-space boundary, and blue cells represent the measurement points extracted.

Uniform downselection

One of the requirements of the reference PFS algorithm is to have all measurement points evenly distributed in the s-domain. Based on this, the naive downselection method (called *uniform*) takes every k-th point $(m_{x,k}, m_{y,k})$ according to the following formula:

$$k = \left\lfloor \frac{N_m}{N_{uniform}} \right\rfloor \cdot (i-1) + 1 \qquad i = 1, 2, 3, ..., N_{uniform} \tag{4.4.11}$$

where:

N_m	– number of measurements.
$N_{uniform}$	– arbitrary divider.

The other downselection methods are based on the idea of keeping only characteristic measurement points, i.e. the ones that provide useful information about geometry.

Line downselection

This method tries to skip (reject) as many collinear measurement points as possible. focuses on points filtration by involving the linear equation $y = ax + b$. The goal is to filter out as many collinear points as possible. The entire boundary length is divided into equal length sections. The ending point of one section is a starting point for the next section. For each measurement point in a section, its orthogonal distance to the section line connecting two endpoints of this section is calculated. If the distance is greater than a threshold, then this measurement point is marked as characteristic (Figure 4.14). The endpoints of each section are also marked as characteristic. All unmarked (non-characteristic) points are rejected from further processing.

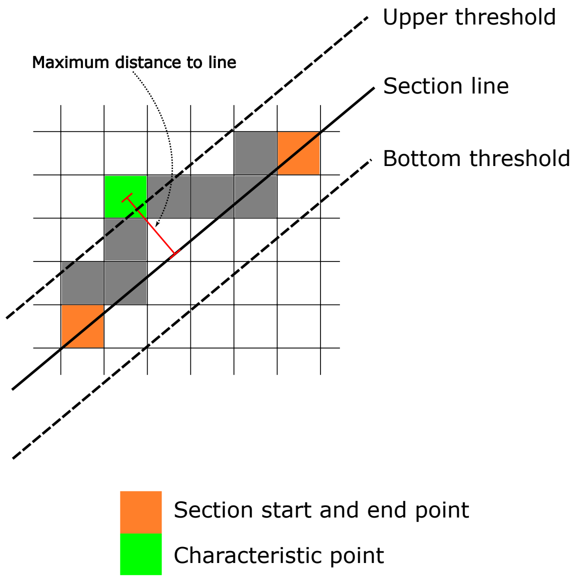


Figure 4.14. The *line* downselection method [Szlachetka et al., 2022].

Direction downselection

This downselection method searches for characteristic points, as the *line* method does. It is limited to basic operations on pixels. The *direction* method focuses on checking if the direction of movement from a previous pixel to a pixel of interest is the same as the direction of movement from a pixel of interest to a next pixel (Figure 4.15). If they differ, the pixel of interest is marked as characteristic.

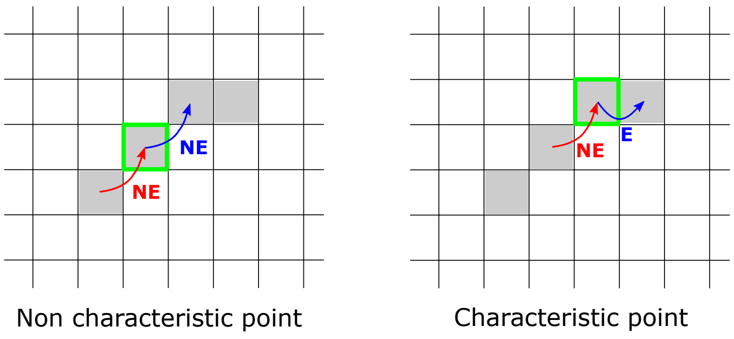


Figure 4.15. The *direction* downselection method [Szlachetka et al., 2022].

4.4.2.3 Measurement association — spline points determination

For each extracted measurement point, we need to know a corresponding spline point, to which the measurement point is associated to perform the measurement update later. It all comes down to the calculation of the s value of the corresponding spline point in the s-domain for each measurement point. This section contains a description of several algorithms that can be used to achieve this goal.

Same distance assumption — original PFS approach

The PFS approach is based on an important assumption, that is, [Schreier et al., 2016]: *„all boundary measurements are treated as equally distributed along s [...] This is a reasonable assumption, because all neighboring boundary pixels have the same distance from each other"*. In other words, since a set of sorted points is constructed on a closed boundary, the distance between two neighboring points is assumed to be exactly the same. It implies following the S-vector:

$$\mathbf{S} = (s_1, s_2, ..., s_{N_m}) \tag{4.4.12}$$

$$s_j = \frac{j-1}{N_m}, \quad j \in [1, \dots, N_m] \tag{4.4.13}$$

where:

s_j	– value of the j-th point in the s-domain corresponding to the j-th measurement point
N_m	– number of measurement points (number of boundary pixels)

This assumption also implies that you have a similar starting measurement point (associated with s_1) position between cycles. It is not guaranteed that the starting point will have a similar position between the two following cycles. Thus, in each cycle, the new starting point is determined on the basis of the starting point from the previous cycle. The new starting point determination is done by making the host movement compensation of the previous starting point and then looking for the closest point (in terms of Euclidean distance) from the provided set of measurement points to the compensated starting point from the previous cycle. Having similar starting points between following tracking cycles prevents from incorrect measurement data association within a single spline tracking cycle. The greater difference between two starting points from the following cycles results in a poorer approximation quality.

It is worth to mention that the PFS's assumption is wrong. The distance between two neighboring pixels can be 1 [unit] (horizontally / perpendicularly) or $\sqrt{2}$ [unit] (slant). Furthermore, the mentioned assumption results in a semiuniform distribution of control points along the spline (in the s-domain). It is one of the main PFS problems resolved in the thesis. The requirement of having similar starting point positions between cycles is another drawback. It can happen that the boundary pixels segment where the starting point is located can disappear between cycles. As a result, the new starting point will be far from the previous one. Thus, incorrect data associations can occur within the spline tracking, resulting in a worse approximation quality.

Scaled distance assumption — PFS extension proposal of the Author

The drawbacks of the main PFS assumption with respect to the measurement points distribution (mentioned above) can be fixed and can still follow the idea of using the measurement points distribution. The PFS approach requires two improvements. The first improvement is to let the measurement points not be equally distributed but to use the actual distances between the neighboring pixels, that is, 1 or $sqrt2$. The second improvement is to allow control points to be distributed depending on the actual needs, not to be semi-equally distributed along the curve (indirect result of the equal measurement point distribution assumption). Such relaxation of assumption implies recalculation of spline points in the s domain (rescaling of the S-vector). In a 1D problem the uniform periodic B-spline has equally distributed basis functions (and control points corresponding to them) in the s-domain. It is no longer true in the 2D case. Figure 4.16 explains the inconsistency of the distribution. The figure consists of a spline built on eight control points and a set of 100 spline points that are evenly distributed in the s-domain $((0, 0.01, 0.02, ..., 0.99, 1))$. The only difference between the subfigures is the position of one control point. In the left subfigure it is the point at position $(0, 2)$, in the right subfigure it is the point at position $(0, 2.5)$. The control point is moved closer to one neighbor and, at the same time, away from another neighbor. A blue circle indicates the control point which position has been changed. The spline itself remains the same. There is almost no difference in shape (unnoticeable difference is possible), but the distribution of the spline points along the curve has changed. The yellow and pink areas indicate parts of the spline that have equal length in the s-domain but not in the 2D plane. It can be noticed that in the left subfigure both areas have the same length and the spline points are equally distributed in the 2D plane. In the right subfigure, the spline points inside the yellow area are more condensed, and in the pink area they are more scattered. Only the spline points around the moved control points are affected. All remaining spline points remain the same. Thus, it is a local problem, not a global one.

To overcome this problem, a recalculation of spline points in the s-domain (rescaling of the $\mathbf{S}$ vector) is needed. First, having a set of measurement points, the distances between each neighbor pair have to be calculated. Compared to the original PFS approach, it takes into account that the distance between two neighboring points can have two possible values. Once the distances are calculated, the normalized accumulative distance vector is determined as follows.

$$d_i = \begin{cases} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} & \text{if } i < N_m \\ \sqrt{(x_{N_m} - x_1)^2 + (y_{N_m} - y_1)^2} & \text{if } i = N_m \end{cases} \quad i = 1, 2, 3, ..., N_m \tag{4.4.14}$$

$$\mathbf{M} = (m_1, m_2, ..., m_{N_m+1}) \tag{4.4.15}$$

$$m_i = \begin{cases} 0 & \text{if } i = 1 \\ \sum_{k=1}^{k<i} d_k & \text{otherwise} \end{cases} \tag{4.4.16}$$

$$\hat{\mathbf{M}} = \mathbf{M} \oslash m_{N_m+1} = \left(0, \frac{m_2}{m_{N_m+1}}, \frac{m_3}{m_{N_m+1}}, , , ..., \frac{m_{N_m}}{m_{N_m+1}}, 1\right) \tag{4.4.17}$$

$$\hat{\mathbf{S}} = (\hat{s}_1 = 0, \hat{s}_2, ..., \hat{s}_{N_s+1} = 1) \tag{4.4.18}$$

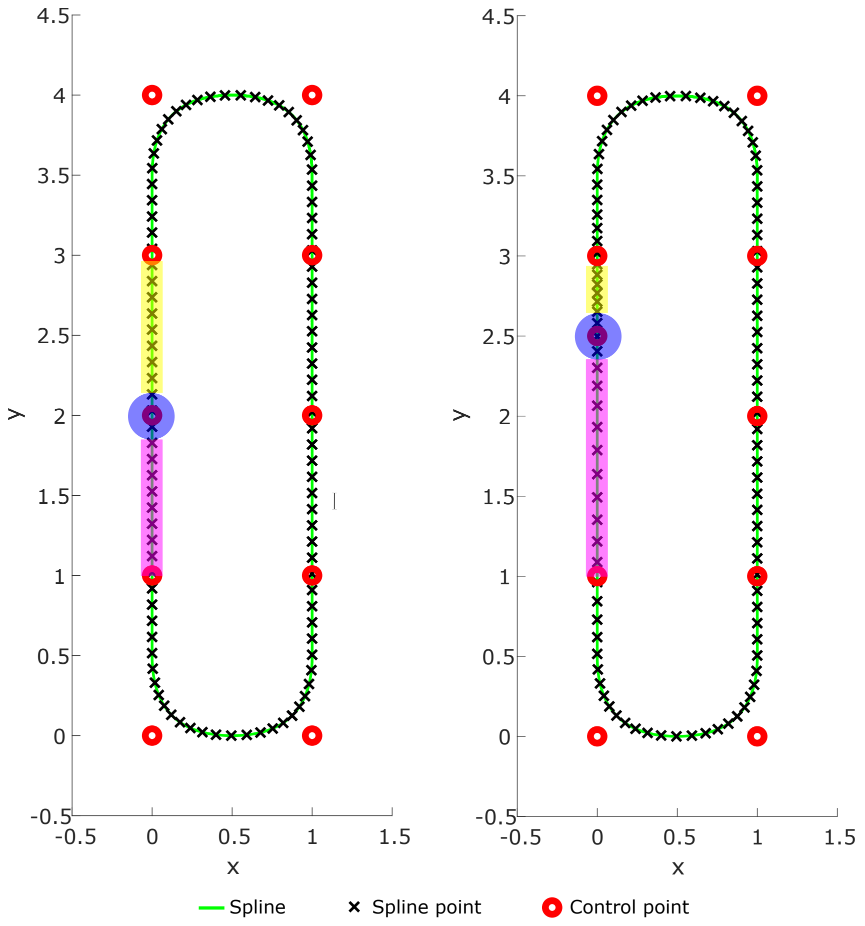


Figure 4.16. An example of spline points distribution inconsistency between the s-domain and along the spline. Both drawings show spline points and control points that are equidistant in the s-domain (equation 4.4.25) but not necessarily along the spline. The original setup is shown on the left side. On the right one can see that after moving one control point on XY plane, the distances between consecutive spline points along the curve are compressed in one section while expanded in another one, even though all control point positions are still equidistant in the s-domain.

$$\mathbf{S} = (s_1 = 0, s_2, ..., s_{N_s+1} = 1) \tag{4.4.19}$$

$$s_i = \frac{i-1}{N_s} \tag{4.4.20}$$

where:

$\mathbf{M}$	–	vector of accumulated Euclidean distances between measurements points;
$\hat{\mathbf{M}}$	–	vector of normalized accumulated Euclidean distances between measurements points;
$\hat{\mathbf{S}}$	–	vector of normalized accumulated Euclidean distances between spline points corresponding to measurement points;
$\mathbf{S}$	–	vector of expected s-domain corresponding to $\hat{\mathbf{S}}$;
d_i	–	Euclidean distance between the i-th and (i + 1) -th points;
N_m	–	number of measurement points;
N_s	–	number of spline samples.

The next step is to sample the spline (equally in the s-domain) and estimate the distance of each spline point along the spline from the beginning of the spline, that is $r(0)$. Then the normalized distance vector $\hat{\mathbf{S}}$ is calculated in the same way as for the determination of the measurement points (equation 4.4.17). The final step is to recalculate $\hat{\mathbf{M}}$ based on $\hat{\mathbf{S}}$ to obtain a new vector $\tilde{\mathbf{M}}$. Then, $\tilde{\mathbf{M}}$ is used to determine the corresponding spline point for each measurement point.

The $\tilde{\mathbf{M}}$ is obtained as follows:

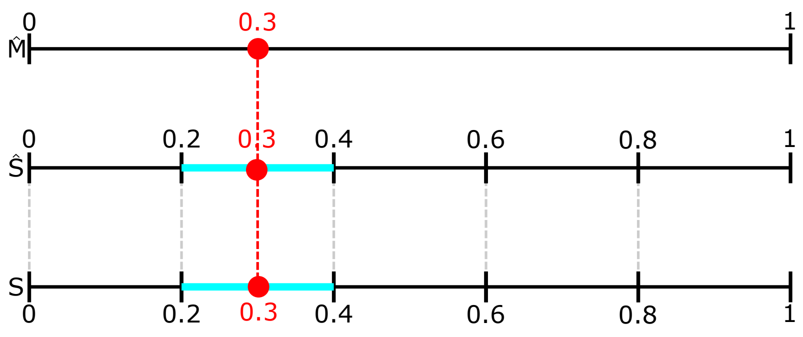
$$\tilde{\mathbf{M}} = (\tilde{m}_1, \tilde{m}_2, ..., \tilde{m}_{N_m}) \tag{4.4.21}$$

$$\tilde{m_k} = (m_k - \hat{s}_k)\frac{s_{k+1} - s_k}{\hat{s}_{k+1} - \hat{s}_k} + s_k \quad \text{if} \quad m_k \in \langle \hat{s}_k, \hat{s}_{k+1} \rangle \tag{4.4.22}$$

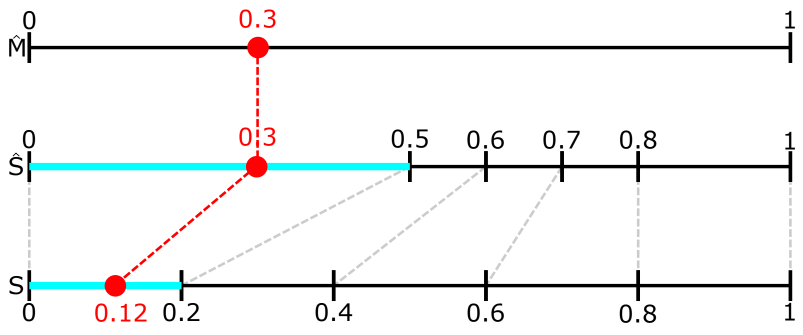
where:

$\tilde{\mathbf{M}}$ – transformed $\hat{\mathbf{M}}$ vector.

Figure 4.17 visualizes the recalculation $\hat{\mathbf{M}}$ to $\tilde{\mathbf{M}}$.



(a) $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ have the same distribution.



(b) $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ have different distribution.

Figure 4.17. An example of rescaling spline points in s-domain.

Taking $m = 0.3$, we obtain $\tilde{m} = 0.12$ as a result of transforming the value from the range $(0.0, 0.5)$ to $(0.0, 0.2)$ based on the following calculations.

$$\tilde{m} = (0.3 - 0)\frac{0.2 - 0}{0.5 - 0} + 0 = 0.12 \tag{4.4.23}$$

The final step is to remove the last element from $\tilde{\mathbf{M}}$. This is necessary because the spline used is a closed spline (Equation 4.2.6).

Distance to measurement — naive

This and the following approaches focus on the idea of finding a spline point for each measurement point that is the closest in terms of the Euclidean distance. The naive approach relies on finding an argument s for which the following function is minimized:

$$f(s) = \sqrt{(r_x(s) - z_x)^2 + (r_y(s) - z_y)^2} \tag{4.4.24}$$

where:

$r_x(s)$	– longitudinal component of the B-spline (4.2.1).
$r_y(s)$	– lateral component of the B-spline (4.2.1).
z_x	– longitudinal component of the measurement point.
z_y	– lateral component of the measurement point.

The value of s determines a spline point in the plane. The starting point for minimization of the function is arbitrarily chosen, that is, $s = 0$.

Distance to measurement — control points position

This approach extends the previous method by better choosing a starting point for the minimization algorithm s_0. It is determined on the basis of the position of the control point, or more precisely, its corresponding spline point ($\mathring{C}_i$).

To calculate the spline point, its s-domain value is needed. Let us define the s-domain position of a control point as follows:

$$s_i = argmax\left(B_i^{[n]}(s)\right) \tag{4.4.25}$$

where:

$B_i^{[n]}(s)$	– i-th basis function of degree n corresponding to $\mathbf{q}_i$ control point.

According to Section 4.2.2.1, the basis functions are equally distributed (fixed offset) in the s-domain. It simplifies equation 4.4.25 to:

$$s_i = \frac{i-1}{N_q} \tag{4.4.26}$$

i	– control point index (1, 2, 3, ..., N_q)
N_q	– number of control points

Thus, a spline point corresponding to a control point is calculated as follows.

$$\mathbf{r}(s_i) = r\left(\frac{i-1}{N_q}\right) \tag{4.4.27}$$

For example, in Figure 4.18, the 9-th control point is considered (marked by a green circle). Its value in the s-domain is $s_9 = 0.88$, so the corresponding spline point is calculated as follows $r(0.88)$ (marked by a teal circle).

In this method, a starting point for the search algorithm is evaluated in two steps:

1. Find the closest (in terms of Euclidean distance) control point to the measurement point of interest.
2. Use the value from the s-domain (s_i) of the control point found as a starting point.

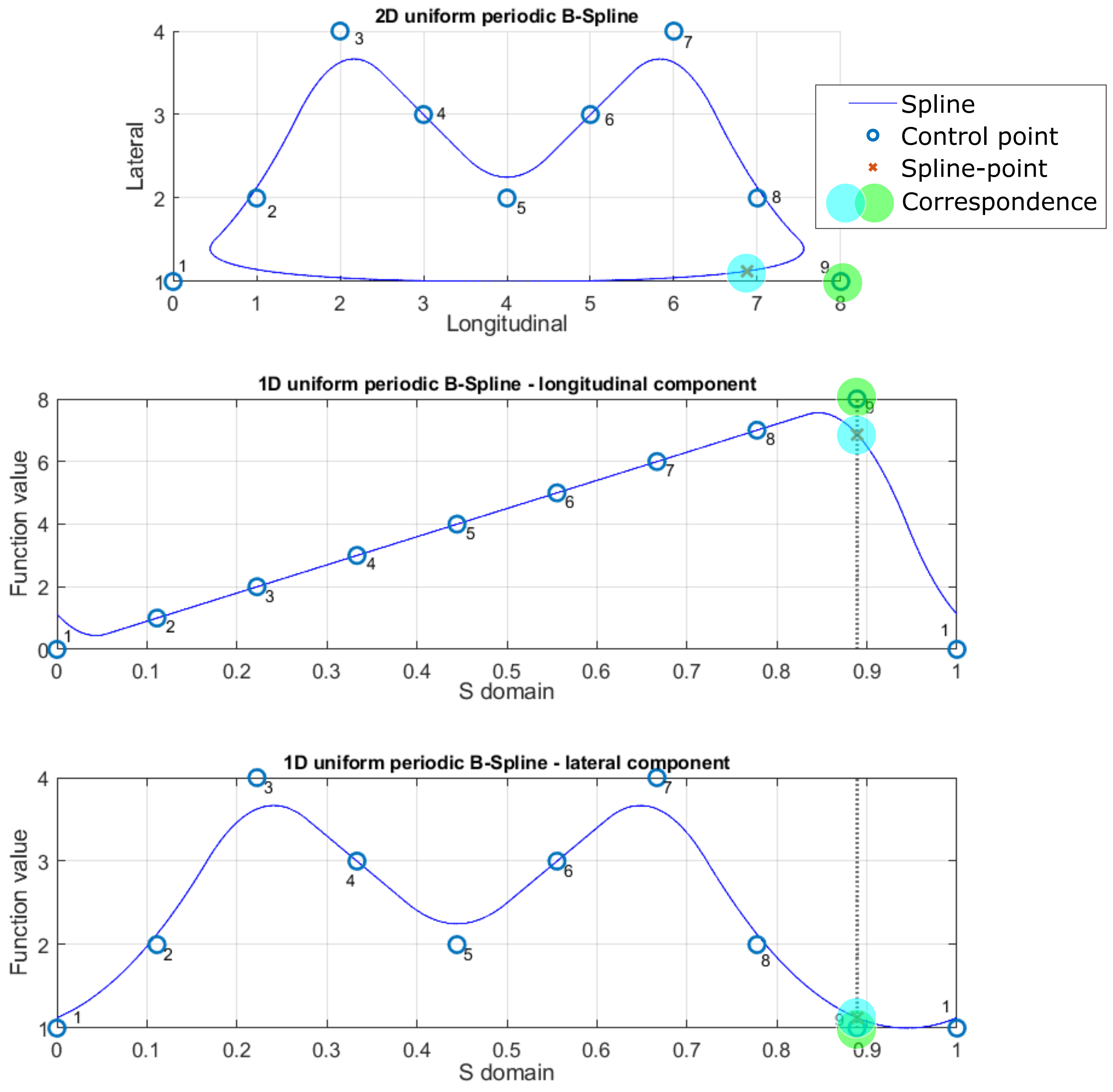


Figure 4.18. The visualization of a control point and its corresponding foot-point.

Distance to measurement — sampled spline

This method differs from the previous one, based on the distance from the measurement, by performing spline sampling beforehand. This way, the calculations are faster. For each measurement point, the corresponding spline point is taken from the sampled set on the basis of the Euclidean distance. The spline feature, the different distribution of spline points along the curve compared to the s-domain, is an advantage here. The spline is sampled equally along the s-domain, but the distribution of spline points along the spline depends on the positions of the control points (Figure 4.16). It is beneficial for more complex local shape areas to have more control points gathered around them. More control points mean more spline points that can be used for a better approximation. The same is applicable for the areas with lower number of control points inside. This time, the local shape is expected to be simple, so there is no need to have more spline points there.

Distance to measurement — normalized sampled spline

Compared to the previous method, this once makes the spline samples equally distributed along the spline. It is done by scaling the s-domain initial vector based on the cumulative distance of the initial spline points.

Perpendicular line from spline points

All methods mentioned earlier focus on finding the corresponding spline point to each measurement point. This approach does the opposite. Focuses on finding a measurement point for each sampled spline point. It starts with the creation of a set of spline samples. Then, for each spline point, the perpendicular line is created. The line is evaluated on the basis of neighboring spline points. The created line is perpendicular to a line defined neighboring spline points and passes the spline point of interest. Figure 4.19 visualizes this method. Once the line is ready, the algorithm looks for the measurement points that are close to the line. From the subset of measurement points, the closest one is chosen.

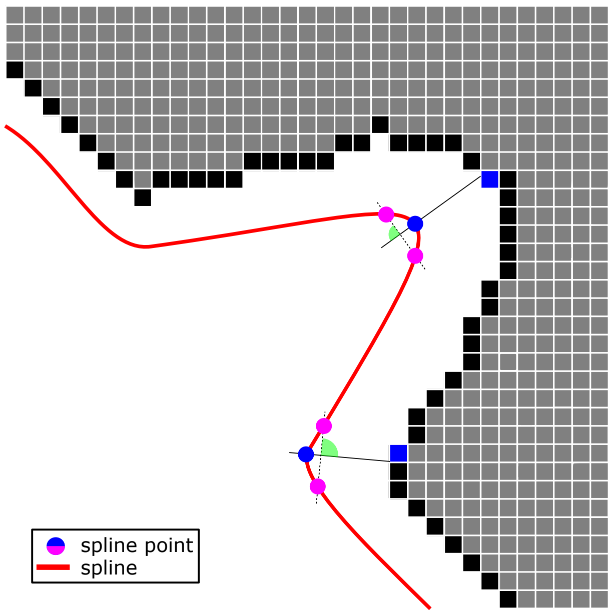


Figure 4.19. The matching between spline points and measurement points is based on the perpendicular line method.

4.4.3 Measurement update

Measurement update of the spline is the final step of the Information Filter used to refine the spline parameters. The measurement update in this case is a trivial sum:

$$\boldsymbol{\Upsilon}_{k|k} = \hat{\boldsymbol{\Upsilon}}_{k|k-1} + \mathbf{L}_k \tag{4.4.28}$$

$$\upsilon_{k|k} = \hat{\upsilon}_{k|k-1} + \mathbf{l}_k \tag{4.4.29}$$

$$\mathbf{L}_k = \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \tag{4.4.30}$$

$$\mathbf{l}_k = \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k \tag{4.4.31}$$

where:

k	– tracking cycle index.
$\mathbf{H}_k$	– observation matrix.
$\mathbf{R}_k$	– measurement noise covariance matrix (observation noise).
$\mathbf{L}_k$	– transformed measurement noise covariance matrix.
$\mathbf{l}_k$	– transformed measurement vector.
$\mathbf{z}_k$	– measurement vector in k-th cycle, containing longitudinal and lateral position of the measurement (selected boundary pixel).

$$\mathbf{z}_k = \begin{bmatrix} z_{x,1} \\ \vdots \\ \\ z_{x,N_m} \\ z_{y,1} \\ \vdots \\ \\ z_{y,N_m} \end{bmatrix} \in \mathbb{R}^{2N_m \times 1} \tag{4.4.32}$$

In the proposed algorithm, the measurement vector $\mathbf{z}_k$ contains positions of selected boundary pixels (which passed measurement downselection). Regarding the measurement uncertainty, expressed by the measurement covariance matrix $\mathbf{R}_k$, the same position variance is assumed for each measurement, with one exception. Each measurement point, i.e. a pixel point at the border of the grid, has assigned higher variance. This is a reasonable assumption, since *occupancy grid processing* always provides pixels that form a closed boundary. It is achieved by marking outer pixels of the grid as an artificial obstacle in each direction in which no obstacle is detected between the center of the host and the border of the grid.

The measurement covariance matrix is defined as follows:

$$\mathbf{R}_k = \mathbf{R} = \begin{bmatrix} w_{x,1} & 0 & & \cdots & & 0 \\ 0 & \ddots & & & & \\ & & w_{x,N_m} & & & \vdots \\ \vdots & & & w_{y,1} & & \\ & & & & \ddots & 0 \\ 0 & & \cdots & & 0 & w_{y,N_m} \end{bmatrix} \in \mathbb{R}^{2N_m \times 2N_m} \tag{4.4.33}$$

$$w_{x,j} = w_{y,j} = \begin{cases} w_{low} & \text{if measurement point is located at the grid border} \\ w_{high} & \text{otherwise} \end{cases} \tag{4.4.34}$$

where:

$w_{x,j}$	– longitudinal position variance of the j-th measurement.
$w_{y,j}$	– lateral position variance of the j-th measurement.
N_m	– number of measurement points.

The observation matrix, in the k-th cycle, is defined as follows:

$$\mathbf{H}_k = \begin{bmatrix} \mathring{\mathbf{H}}_k & \mathbf{0} \\ \mathbf{0} & \mathring{\mathbf{H}}_k \end{bmatrix} \in \mathbb{R}^{2N_s \times 2N_q} \tag{4.4.35}$$

$$\mathring{\mathbf{H}}_k = \begin{bmatrix} B_1^{[n]}(s_1) & B_2^{[n]}(s_1) & \cdots & B_{N_q-1}^{[n]}(s_1) & B_{N_q}^{[n]}(s_1) \\ B_1^{[n]}(s_2) & & & & \\ \vdots & & \ddots & & \vdots \\ B_1^{[n]}(s_{N_s-1}) & & & & \\ B_1^{[n]}(s_{N_s}) & & \cdots & & B_{N_q}^{[n]}(s_{N_s}) \end{bmatrix} \in \mathbb{R}^{N_s \times N_q} \tag{4.4.36}$$

4.4.4 Local measurement status analysis

One of the control point parameters is its measurement status (Ω). It contains information about the control point's age and measurement association. The measurement status can take one of three possible values: *NEW*, *UPDATED*, and *COASTED*. All control points are initialized with the status *NEW*. It does not matter whether the control point was created during algorithm initialization or was added later. A control point remains in the *NEW* state for a few cycles (e.g. 5). After this time, the control point can take only one of the remaining two status values. The status *UPDATED* of a control point is set when the spline has at least one measurement point associated within the support interval of this control point, otherwise the status is set to *COASTED* (Figure 4.20).

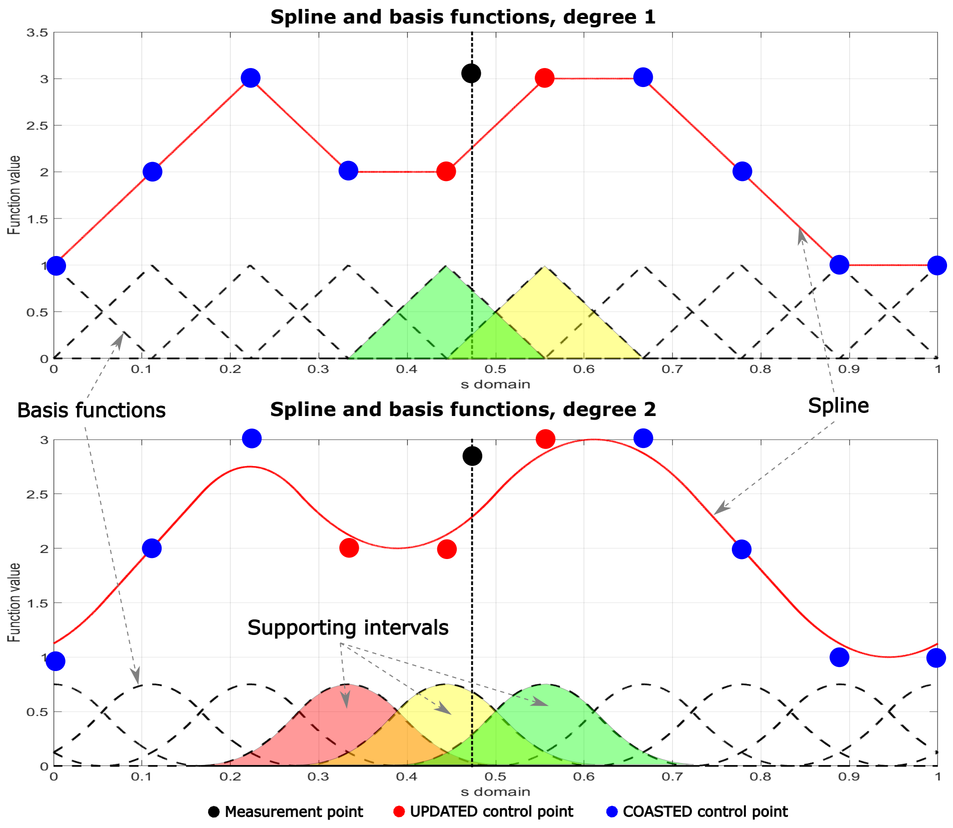


Figure 4.20. The control point status determination based on the presence of a measurement point within control point supporting interval.

The control point support interval is defined as:

$$\Lambda_i = \left\{ s : B_i^{[n]}(s) \neq 0 \right\} \tag{4.4.37}$$

where:

Λ_i	– i-th control point supporting interval.
$B_i^{[n]}$	– i-th basis function of degree n corresponding to the i-th control point.

In case the uniform periodic B-spline is used, equation 4.4.37 can be rewritten as:

$$\Lambda_i = \begin{cases} \left\langle 0, \mathring{\Lambda}_i \right\rangle \cup \left\langle \hat{\Lambda}_i + 1, 1 \right\rangle & \text{if } \hat{\Lambda}_i < 0 \\ \left\langle 0, \mathring{\Lambda}_i - 1 \right\rangle \cup \left\langle \hat{\Lambda}_i, 1 \right\rangle & \text{if } 1 < \mathring{\Lambda}_i \\ \left\langle \hat{\Lambda}_i, \mathring{\Lambda}_i \right\rangle & \text{otherwise} \end{cases} \tag{4.4.38}$$

$$\hat{\Lambda}_i = \frac{i - 0.5n - 1.5}{N_q} \tag{4.4.39}$$

$$\mathring{\Lambda}_i = \frac{i + 0.5n - 0.5}{N_q} \tag{4.4.40}$$

i	– control point index $(1, 2, 3, ..., N_q)$.
n	– degree of spline.
N_q	– number of control points.

Once a control point leaves the *NEW* status, it cannot return to it. Its status can only be changed between *UPDATED* and *COASTED*. Taking as an example Figure 4.20, assume that there is no control point with status *NEW* and the measurement point is identified at $s = 0.48$. In the case of a 1 degree spline, only the 5-th and 6-th control points are affected. Their support intervals are $\Lambda_5 \approx \langle 0.33, 0.56 \rangle$, $\Lambda_6 \approx \langle 0.44, 0.67 \rangle$. In case of 2 degree spline, 4-th, 5-th and 6-th control points are affected. Their support intervals are $\Lambda_4 \approx \langle 0.17, 0.5 \rangle$, $\Lambda_5 \approx \langle 0.28, 0.61 \rangle$ and $\Lambda_6 \approx \langle 0.39, 0.72 \rangle$. All remaining control points are not affected by the measurement point, therefore, their status is *COASTED*.

4.4.5 Local spline shape analysis

By working with the PFS algorithm, after a short time, it starts to be obvious that having a fixed number of control points semi-uniformly distributed along the curve limits the algorithm's ability to adapt a spline to any possible shape of the surrounding environment. The fixed number of control points gives an upper bound above which a complex environment will only be approximated worse. For any assumed number of control points, we can find an environment that is complex enough to not be approximated with a satisfactory quality. Furthermore, increasing this number is also limited by hardware resources such as available memory and computing power. Semi-uniformly distributed control points makes approximation quality even worse. In the same scenario and at the same time, both complex and simple parts can appear in the environment being approximated. It is recommended to keep more control points closer to complex shapes and leave as few control points as possible around simple shapes. Thus, even if the number of control points is fixed, we can still increase approximation quality by moving some control points to more complex shape areas.

Control points adjustment is introduced to make the spline approximation more accurate, that is, to reduce the approximation error of the estimated free-space boundary. It is done by increasing the number of control points in areas of high shape complexity while reducing the number of control points in areas of low shape complexity. Adjustment can be understood as operations such as moving, removing, adding, or merging control points. They are described in the following sections.

To correctly adjust the control points, several novel parameters are introduced and described in the following sections.

4.4.5.1 Local shape complexity

A stationary environment consists of elements that are independent (not correlated). Thus, performing a local analysis is more sufficient than performing a global one. To analyze the spline reflecting the environment, a local shape complicity indicator (LSC) is introduced. The LSC indicator describes the local spline complexity in the vicinity of a single control point. Its raw value is obtained by performing a spline characteristics analysis. Then, the LSC is filtered in time by an infinite impulse response low-pass filter to prevent too aggressive and too frequent adjustments of control points in this area. The LSC can take a value from -1 to 1. The low value, closer to -1, indicates that the local shape is simple and the corresponding control point might be removed. The high value, closer to 1, means a complex local shape and that the number of control points should be increased in this area to improve local approximation quality.

The LSC is defined as follows:

$$\Psi_{k,i} = (1 - c_\Psi)\Psi_{k-1,i} + c_\Psi \tilde{\Psi}_{k,i} \tag{4.4.41}$$

$$k = 1, 2, 3, 4, ...$$

$$i = 1, 2, 3, 4, ..., N_q$$

where:

k	– tracking cycle index (iteration).
i	– control point index.
c_Ψ	– filter factor.
$\tilde{\Psi}_{k,i}$	– raw (instantaneous) value of the shape complexity of i-th control point in k-th cycle.
$\Psi_{k,i}$	– final (filtered) shape complexity value of i-th control point in k-th cycle.

The raw shape complexity $\tilde{\Psi}_{k,i}$ is calculated based on four components as follows (the k index is skipped for clarity):

$$\tilde{\Psi}_i = \min\left(\max\left(f\left(\Delta_i, \Gamma_i, \Theta_i, \Xi_i\right), -1\right), 1\right) \tag{4.4.42}$$

where:

Δ_i	– i-th control point distance complexity indicator.
Γ_i	– i-th control point corresponding spline point complexity indicator.
Θ_i	– i-th control point angle complexity indicator.
Ξ_i	– i-th control point host-dependent indicator.

and

$$f(\Delta_i, \Gamma_i, \Theta_i, \Xi_i) = \begin{cases} \Gamma_i + \Delta_i + \Xi_i & \text{for } \Gamma_i \leq 0 \\ \Theta_i \cdot \Gamma_i + \Delta_i + \Xi_i & \text{otherwise} \end{cases} \tag{4.4.43}$$

The initial value of the LSC for each control point is neutral, that is, set to zero ($\Psi_{0,i} = 0$). The use of the local shape complexity indicator is described in Section 4.4.6.

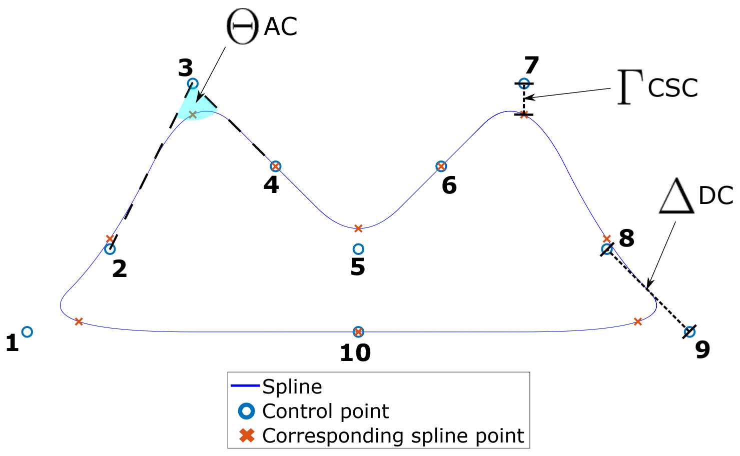


Figure 4.21. The explanation of three shape complexity indicators: AC — angle complexity, DC — distance complexity, CSC — corresponding spline point complexity.

Distance complexity indicator (Δ)

The rule of the distance complexity indicator is to prevent the spline sections being too long. Even if a simple shape is well approximated by a small number of control points, having too long parts may influence the dynamics (may limit the speed of spline adaptation) of the spline adjustment if a new complex shape starts to be visible to the host while the host is moving. Distance complexity can take only positive values and thus can only increase the value of the LSC. An unconstrained distance shape complexity is a Cartesian distance $d()$ between neighboring control points (between the control point of interest and the previous control point) as follows:

$$\check{\Delta}_i = d(\mathbf{q}_i, \mathbf{q}_{i-1}). \tag{4.4.44}$$

The unconstrained distance shape complexity is then normalized from the range $(\Delta_{low}, \Delta_{high})$ to the range $(0, 1)$ with saturation as follows:

$$\Delta_i = \begin{cases} 0, & \check{\Delta}_i \leq \Delta_{low} \\ (\check{\Delta}_i - \Delta_{low}) \cdot \left(\frac{1}{\Delta_{high} - \Delta_{low}}\right) & \check{\Delta}_i \in (\Delta_{low}, \Delta_{high}) \\ 1, & \Delta_{high} \leq \check{\Delta}_i \end{cases} \tag{4.4.45}$$

where Δ_{low} and Δ_{high} are empirically chosen saturation thresholds.

Corresponding spline-point complexity indicator (Γ)

The corresponding spline point ($\mathring{\mathbf{q}}_i$) is described in section 4.4.2.3. An idea of using the corresponding spline point is its mutual relationship with the control point. The distance between a control point and its corresponding spline point describes the local shape complexity. If both points are close to each other, such as points 4, 6 or 10 in Figure 4.21 then the spline sections around are simple. If both points are far from each other, for example, points 1, 3, 7 or 10 in Figure 4.21, then the shape might be complex. In other words, a small distance may indicate a local smoothness of the spline shape and quite often a too high number of control points around the control point of interest, but not the opposite.

An unconstrained corresponding spline point complexity is defined as

$$\check{\Gamma}_i = d(\mathbf{q}_i, \mathring{\mathbf{q}}_i) \tag{4.4.46}$$

The unconstrained corresponding spline point complexity is normalized from the range $(\Gamma_{low}, \Gamma_{high})$ to the range $(-1, 1)$ with saturation afterward as follows:

$$\mathring{\Gamma}_i = \begin{cases} -1, & \check{\Gamma}_i \leq \Gamma_{low} \\ \left(\check{\Gamma}_i - \Gamma_{low}\right) \cdot \left(\frac{2}{\Gamma_{high} - \Gamma_{low}}\right) - 1 & \check{\Gamma}_i \in (\Gamma_{low}, \Gamma_{high}) \\ 1, & \Gamma_{high} \leq \check{\Gamma}_i \end{cases} \tag{4.4.47}$$

Normalization is not the last step in the calculation of the corresponding spline points. Once $\mathring{\Gamma}_i$ is calculated for all control points, an additional pass is made. The final value of the corresponding spline point indicators is the highest value between the corresponding spline point indicator of the control point of interest and the corresponding spline point indicator of the previous control point as follows:

$$\Gamma_i = \max\left(\mathring{\Gamma}_{i-1}, \mathring{\Gamma}_i\right) \tag{4.4.48}$$

This extra pass is done to mark the spline segments to which new control points will eventually need to be added. Since shape complexity is a parameter of the control points, it is not possible to define whether new control points should be added to the left or right of the control points (at high value of shape complexity indicator). Equation (4.4.48) was introduced to solve this problem.

Angle complexity indicator (Θ)

The angle complexity indicator presents the relationship between three consecutive control points, called a triplet. The value of Θ is calculated as the angle between two segments defined by the triplet. Angle values closer to 180 degrees represent smoother shapes, and values closer to 0 degrees represent more complex shapes.

An unconstrained angle complexity indicator is defined as

$$\check{\Theta}_i = h(\mathbf{q}_{i-1}, \mathbf{q}_i, \mathbf{q}_{i+1}) \tag{4.4.49}$$

The unconstrained angle complexity is then normalized from the range $(\Theta_{low}, \Theta_{high})$ to the range $(0, 1)$ with saturation as follows:

$$\Theta_i = \begin{cases} 0, & \check{\Theta}_i \leq \Theta_{low} \\ \left(\check{\Theta}_i - \Theta_{low}\right) \cdot \left(\frac{1}{\Theta_{high} - \Theta_{low}}\right) & \check{\Theta}_i \in (\Theta_{low}, \Theta_{high}) \\ 1, & \Theta_{high} \leq \check{\Theta}_i \end{cases} \tag{4.4.50}$$

Host car dependent indicator (Ξ)

The host car distance dependent indicator is introduced to increase the number of control points near the host car. It is more beneficial to have better approximated the nearest environment than the farthest one. The parking scenario is the target scenario for this indicator. The algorithms responsible for the automatic parking maneuver require a precise description of a free parking slot, next to the host car, to which the host car must be driven. Figure 4.22 shows that the interesting parking space (inside the elliptical area) is better approximated due to the greater number of control points used for its approximation. The free parking slot on the left is outside the area, is estimated with a lower number of control points, and finally has a worse approximation quality. To control the approximation quality in area close to the host car an elliptic area is introduced. The shape of the elliptical area may be dynamically modified based on the movement of the host. Figure 4.23 visualizes the relationship between host movement and the elliptical

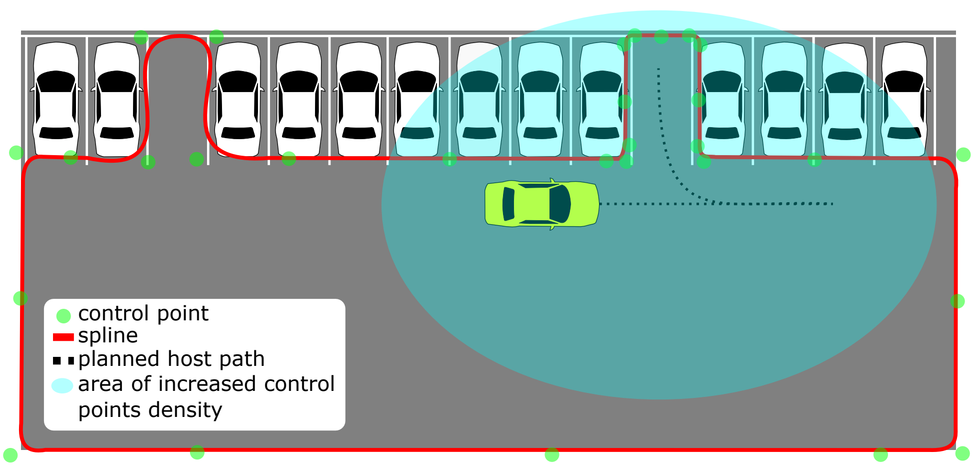


Figure 4.22. The purpose of introducing host car dependent indicator is to have more accurate approximation near the host car.

area. The length of the semi-minor axis b is constant, but the length of the semi-major axis a and the longitudinal offset depend on the host speed. The ellipse orientation is the same as the host orientation. The dependence of the semi-major axis a and the longitudinal offset is defined as follows:

$$b = const \tag{4.4.51}$$

$$a = \begin{cases} 3b & \text{if } v \leq -30 \\ \frac{b}{15}|v| + b & \text{if } v \in (-30, 30) \\ 3b & \text{if } 30 \leq v \end{cases} \tag{4.4.52}$$

$$o = \begin{cases} -\frac{0.8}{30}a & \text{if } v \leq -30 \\ -\frac{0.8}{30}av & \text{if } v \in (-30, 0) \\ \frac{0.8}{30}av & \text{if } v \in \langle 0, 30) \\ \frac{0.8}{30}a & \text{if } 30 \leq v \end{cases} \tag{4.4.53}$$

where:

v	–	the speed of the host car.
a	–	semi-major axis of the ellipse.
b	–	semi-minor axis of the ellipse.
o	–	offset from the center of the host car toward the center of the ellipse.

It can be seen that in equations 4.4.52 and 4.4.53 if the speed of the host car is zero, the ellipse turns into the circle centered in the center of the host car. In all other cases, the ellipse follows the host car towards its movement direction. The higher the host car speed, the longer the ellipse. The value of the host car-dependent indicator for the i-th control point is calculated as follows:

$$\Xi_i = \begin{cases} \Xi_{high} & \text{if } \left(\frac{q_{x,i}-e_x}{a}\right)^2 + \left(\frac{q_{y,i}-e_y}{b}\right)^2 \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.4.54}$$

where:

e_x	–	longitudinal component of the center of the ellipse.
e_y	–	lateral component of the center of the ellipse.

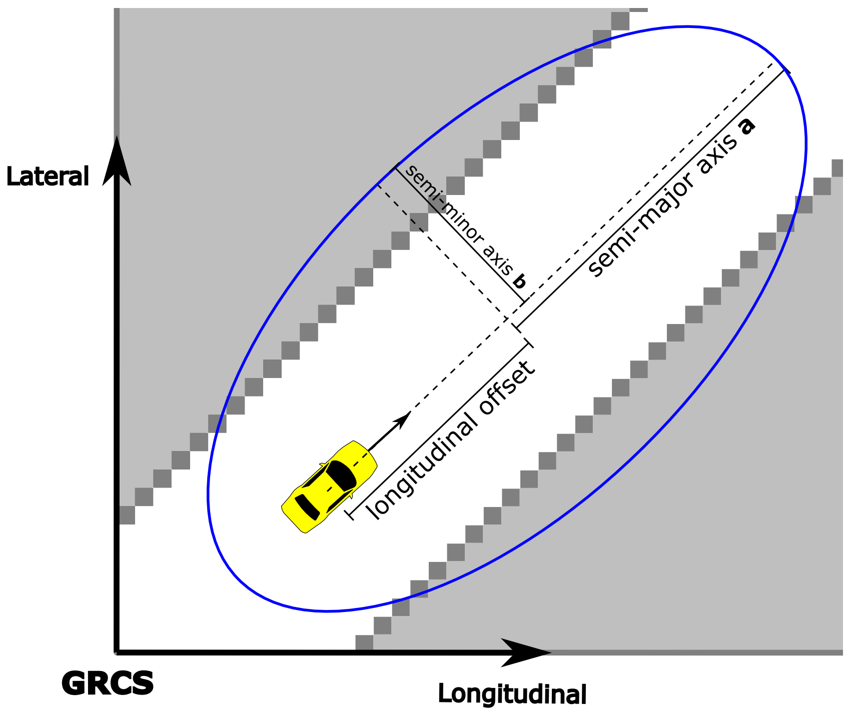


Figure 4.23. The construction of the ellipse is based on the position, orientation, and speed of the host car.

4.4.5.2 Local minima problem

The proposed algorithm focuses mainly on performing the approximation, the best fit, of the provided measurement points. It is an iterative process with a dynamically changing surrounding environment due to host movement. During this process, it is possible that the spline "stucks" somewhere between the measurement points (in a local minimum of approximation error) as shown in Figure 4.24. This result is unacceptable from the point of view of the user. To solve this problem, the control point parameters are extended with an approximation error indicator ($\mathbf{\Phi}$). It is an indicator filtered in time, providing information about too high local approximation error persisting for too long. The local approximation error is estimated on the basis of the distance between the measurement points and the corresponding spline points that lie within the limited supporting interval of this control point.

$$\Phi_{k,i} = (1 - c_\Phi)\Phi_{k-1,i} + c_\Phi \mathring{\Phi}_{k,i} \tag{4.4.55}$$

$$\mathring{\Phi}_{k,i} = \begin{cases} 0, & \check{\Phi}_{k,i} \leq 0 \\ \frac{\check{\Phi}_{k,i}}{\Phi_{high}} & \check{\Phi}_{k,i} \in (0, \Phi_{high}) \\ 1, & \Phi_{high} \leq \check{\Phi}_{k,i} \end{cases} \tag{4.4.56}$$

$$\check{\Phi}_{k,i} = \begin{cases} d_i^{max} & \text{if } d_{th} < d_i^{max} \\ 0 & \text{otherwise} \end{cases} \tag{4.4.57}$$

$$d_i^{max} = \max_j \left(\left\{ d(r(s_j), z_j) : s_j \in \check{\Lambda}_i \right\} \right) \tag{4.4.58}$$

$$\check{\Lambda}_i \subset \Lambda_i \tag{4.4.59}$$

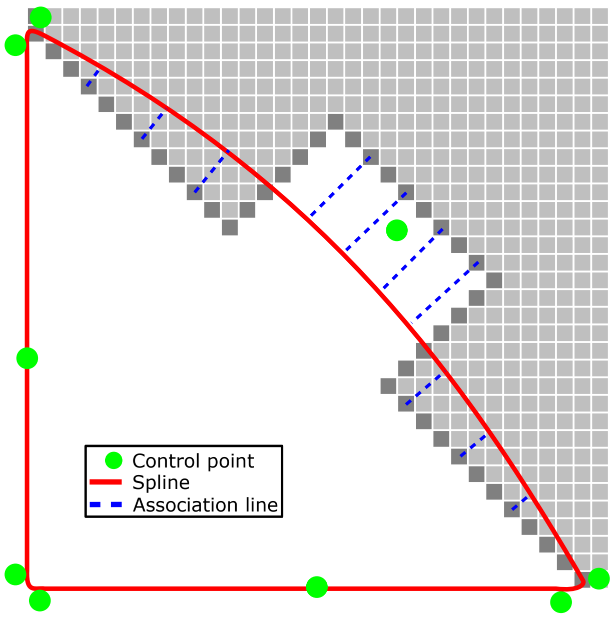


Figure 4.24. An illustration of how the approximation algorithm may stuck in a local minimum.

$\Phi_{k,i}$	– i-th control point approximation error indicator in the k-th cycle
$\check{\Phi}_{k,i}$	– i-th control point raw approximation error indicator in the k-th cycle
$d(a,b)$	– Euclidean distance between the point a and b
d_i^{max}	– maximum $d(a,b)$ of the i-th control point
$\mathbf{r}(s_j)$	– j-th spline point
$\mathbf{z}_j$	– j-th measurement point corresponding to j-th spline point
$\breve{\Lambda}_i$	– limited supporting interval of the i-th control point; it is subinterval of Λ_i limited symmetrically with respect to the interval center
c_Φ	– filter factor

The raw approximation error indicator can take a value of $\langle 0, 1\rangle$. Distances are calculated only for measurement points that the corresponding spline points are located within the limited support interval of the control point of interest (equation 4.4.38) as shown in Figure 4.25. The application of the approximation error indicator is described in Section 4.4.6.

4.4.6 Control points adjustment

All the techniques mentioned so far, such as local measurement status analysis (Section 4.4.4) and local spline shape analysis (Section 4.4.5) provide information about the spline segments that need some modifications. Modification of the spline segments is performed by one of two operations: the addition or removal of a control point. These operations are described in the following sections.

4.4.6.1 Control points addition

The control point addition is an operation that adds a new control point to the existing set of control points. The operation affects only a fragment of the spline surrounding the newly added control point. The algorithm iterates over control points and creates the list of new control points positions. In other words, the addition operation is done once for all positions where a new control point is decided to be added.

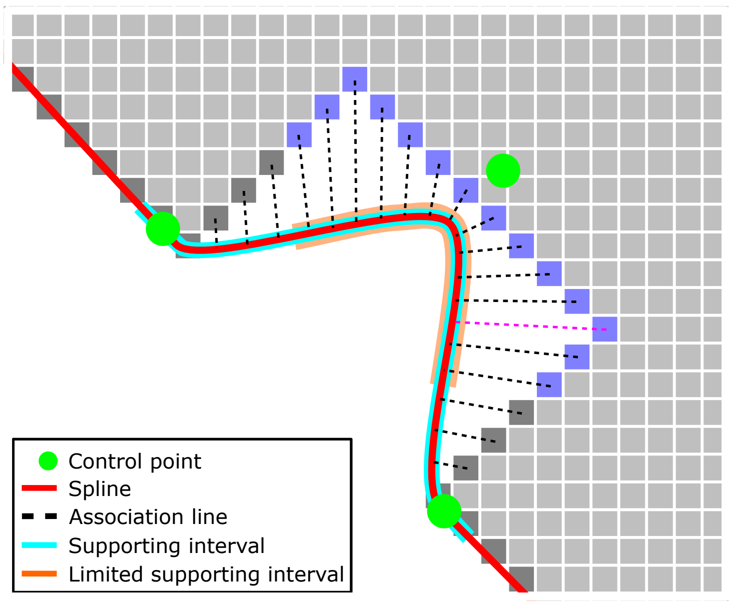


Figure 4.25. Illustration of the maximum distance used in calculation of approximation error indicator. Blue squares indicate measurement points (pixels) that lie within limited supporting interval and pink dashed line the maximum distance.

The decision of adding a new control point is taken when at least one of the following requirements is met.

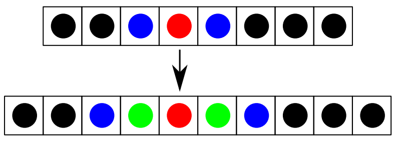
- local shape complexity indicator is higher than the addition threshold.

$$\Psi_i > \Psi_{addition} \tag{4.4.60}$$

- the approximation error indicator is higher than the addition threshold.

$$\Phi_i > \Phi_{addition} \tag{4.4.61}$$

The above requirements are control point specific. The local shape complexity and the approximation error indicators do not describe the specific side of the control point of interest. Thus, if a i-th control point meets the addition requirements, then two new control points are added on its right and left sides (Figure 4.26a). If the i-th and $(i+1)$-th control points satisfy the addition requirements, then new control points are added before the control point i-th, between the control points i-th and $(i+1)$-th, and after the control point $(i+1)$-th (Figure 4.26b). In this way, the algorithm avoids adding a new control point twice between i-th and $(i+1)$-th.


(a) One control point.

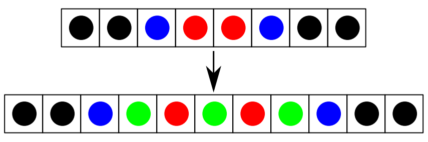

(b) Two neighboring control points.

Figure 4.26. Visualization of control points addition process in the form of the list. Red — control point that meets addition requirements. Green — new control point. Black/Blue — control point that does not meet addition requirements.

The parameters of a new control point are established as follows:

$$
\begin{aligned}
[q_{x,j}, q_{y,j}] &= \left[\frac{q_{x,i} + q_{x,i+1}}{2}, \frac{q_{y,i} + q_{y,i+1}}{2}\right] \\
\sigma^2_{x,j} &= \begin{cases} \sigma^2_{x,i} c_\sigma & \text{if } \sigma^2_{x,i+1} < \sigma^2_{x,i} \\ \sigma^2_{x,i+1} c_\sigma & \text{otherwise} \end{cases} \\
\sigma^2_{y,j} &= \begin{cases} \sigma^2_{y,i} c_\sigma & \text{if } \sigma^2_{y,i+1} < \sigma^2_{y,i} \\ \sigma^2_{y,i+1} c_\sigma & \text{otherwise} \end{cases} \\
\Psi_j &= 0.0 \\
\Phi_j &= 0.0 \\
\Omega_j &= \mathit{NEW} \\
\tau_j &= 1 \\
\kappa_j &= 1
\end{aligned}
\tag{4.4.62}
$$

where:

c_σ – variance extension factor, $c_\sigma > 1.0$.

Additionally, an indicator for an i-th control point, corresponding to a met requirement, is reset to an initial value. In this way, the algorithm forbids adding many control points between the following cycles.

4.4.6.2 Control points removal

The control point removal is an operation to remove a control point from the existing set of control points. It is a local operation. Only the surrounding of the removed control point is affected by this operation. The algorithm goes through all control points and creates a list of control points to remove. The decision to remove a control point is made if at least one of the following requirements is met:

- The local shape complexity is lower than the removal threshold.

$$\Psi_i < \Psi_{removal} \tag{4.4.63}$$

- variances sum is high for too long

$$
\begin{aligned}
\sigma^2_{x,i} + \sigma^2_{y,i} &> \sigma_{removal} \\
\kappa_i &> \kappa_{uncertainty}
\end{aligned}
\tag{4.4.64}
$$

- The status is *COASTED* for a longer period of time.

$$
\begin{aligned}
\Omega_i &= \mathit{COASTED} \\
\tau_i &> \tau_{status}
\end{aligned}
\tag{4.4.65}
$$

- neighboring points are too close to each other (Euclidean distance).

$$d\left(\mathbf{q}_i, \mathbf{q}_{i+1}\right) < d_{close} \tag{4.4.66}$$

Furthermore, the position variances of neighboring control points increase by the factor c_σ. In this way, a slight influence of the removal of control points is reflected.

5 Test and verification

This chapter is dedicated to the testing and verification of the proposed algorithm. This includes a description of the testing methodology (i.e. framework, performance metrics and test scenarios), test results, and results analysis. To demonstrate the performance of the proposed algorithm, it has been compared with the reference PFS algorithm.

5.1 Testing framework

5.1.1 Visualization

During the development of shape tracking algorithms, it is advantageous to have a tool that can visualize the result of the tracking and additional information. Figure 5.1 shows such a tool created as part of the PhD research. The central window can display the actual free space boundary, its spline approximation, and numerous other elements helpful in the development process. The visibility of each element is user selectable. The following items can be displayed:

- host car
  - position
  - orientation
  - size
- spline
  - predicted (before measurement update)
  - updated (after measurement update)
- control points
  - position
  - local shape indicators
- measurement points position
- spline points position
- occupancy grid with free space boundary
- lines connecting measurement points with spline points
- area of increased control points density (Section 4.4.5.1)

Furthermore, the tool generates lateral and longitudinal spline components at the iteration of interest (Figure 5.2) and plots the number of control points throughout the loaded scenario (Figure 5.3) as separate plots.

5.1.2 Artificial scenario generation

A common way to verify the perception algorithm is the simulation study, that is, running the algorithm using artificial input data. Such an approach allows for calculation of performance and quality metrics

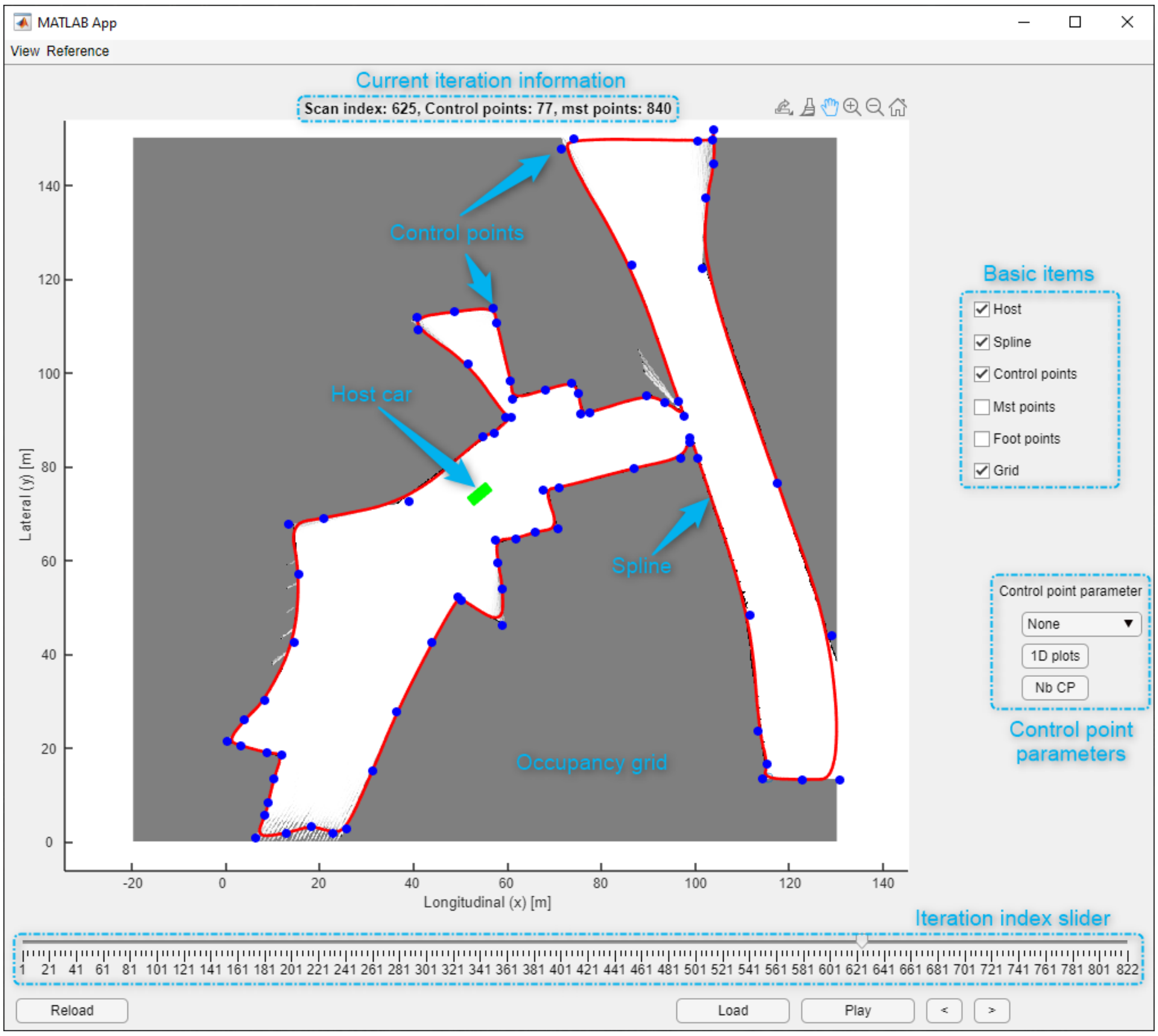


Figure 5.1. Graphical User Interface used for visual analysis of algorithm behavior.

based on a known user-defined reference (true value). The simulation study also saves the cost of testing perception algorithms on real data, which always requires the use of expensive reference measurement systems. The only drawback of the simulation study is that in many cases the models of the environment used to generate artificial data are too simple and do not reflect the real physical phenomena sufficiently. As a result, the evaluation of the algorithms is performed under ideal conditions. In order to overcome the mentioned limitations of too simple hand-made models, the Driving Scenario Designer was used to prepare driving scenarios matching as good the real road cases as possible.

5.1.2.1 Driving Scenario Designer

Driving Scenario Designer (Matlab extension) allows to create synthetic driving scenarios to test automated driving systems [MathWorks, 2023]. It allows fast road scenario prototyping including host car path planning (waypoints, speed, etc.), obstacle creation, and sensors setup configuration. Figure 5.4 shows all the elements mentioned. In this PhD it was used to create test scenarios for the evaluation of the performance of the proposed algorithm.

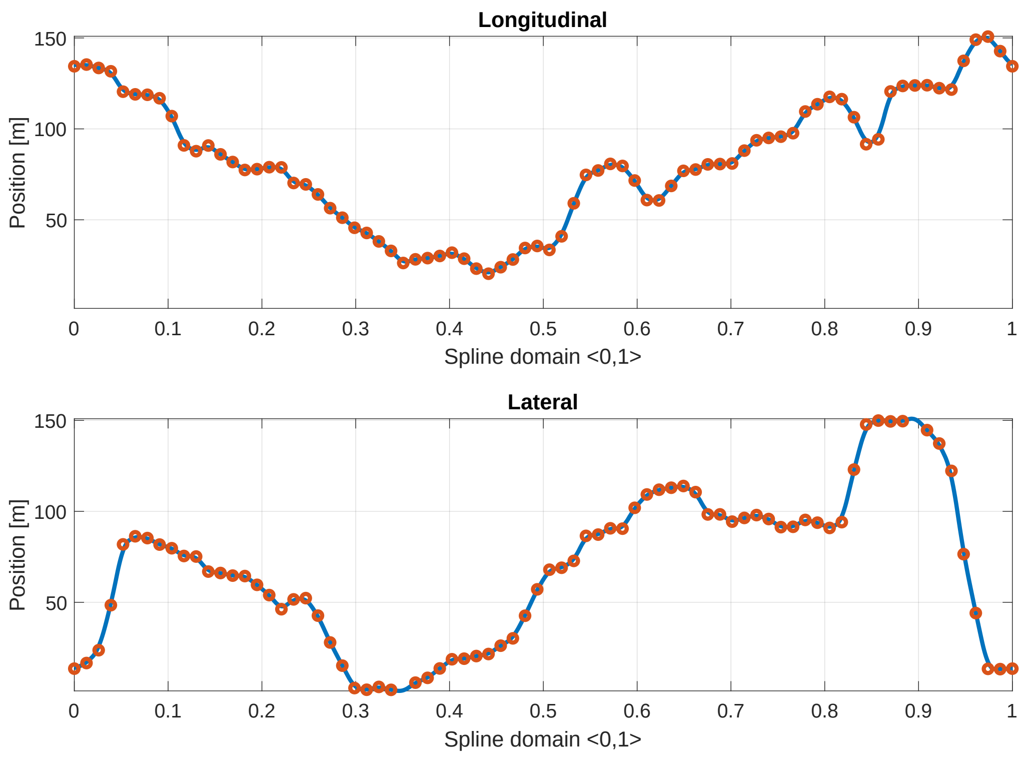


Figure 5.2. Longitudinal and lateral components of the spline and its control points positions in the iteration 625 shown in Figure 5.1.

5.1.2.2 Open street map

OpenStreetMap is a tool (with a web interface [OpenStreetMap, 2023]) that provides map data, including road shapes. OpenStreetMap is developed by a community of mappers who contribute and maintain map content, such as structures and roads, around the world. It can be used together with the Driving Scenario Designer to bring artificial scenarios closer to reality (Figure 5.5).

5.1.2.3 Binary grid generator

Binary grid generator is a basic tool used for the generation of binary (containing only binary occupancy information, i.e., occupied or free) occupancy grids using scenarios created in Driving Scenario Designer as input. The generator treats the area inside the roads as a free space. Other areas (outside roads) are treated as occupied. Additionally, any stationary obstacle located on a road is treated as an occupied area. The snapshot of the generated binary grid is shown in Figure 5.6b.

5.1.2.4 Sensor based occupancy grid generator

It is an advanced generator which is an extension of the binary grid generator (Section 5.1.2.3). It simulates sensors to generate a fully functional occupancy grid. The input to the generator is also a scenario prepared in the Driving Scenario Designer. The generator allows to model occupancy uncertainties via sensor configuration. The snapshot of the generated occupancy grid is shown in Figure 5.6c.

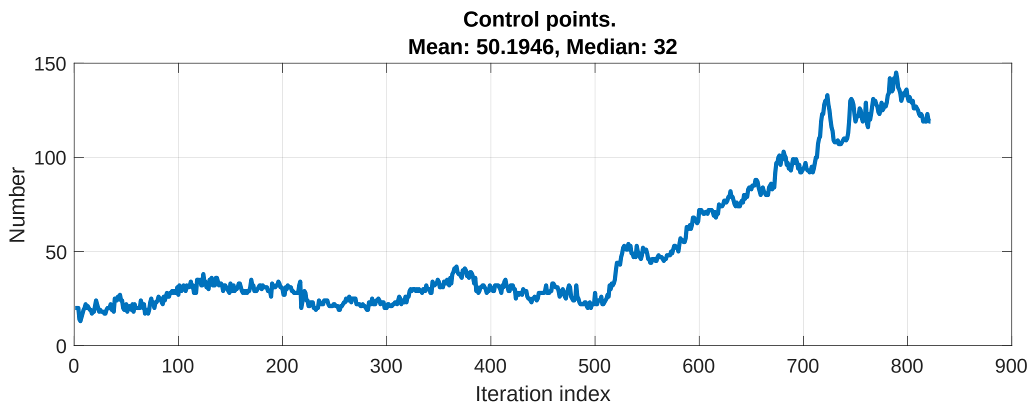


Figure 5.3. Number of control points versus time in scenario shown in Figure 5.1.

5.1.3 Artificial road scenarios

22 scenarios grouped into 5 categories were prepared (reflecting possible circumstances under which the host car can drive). They are as follows:

- highway
- parking
- city
- shapes
- mixed case

Each of the categories with proposed scenarios is described in the following sections.

5.1.3.1 Highway

Highway category is characterized by simple shapes (straight or low curvature road) and the high speed of the host car (around 100 km/h). The highway scenarios used for the tests are as follows:

- Highway *id 1*: smooth part of the A4 road in Cracow.
- Highway *id 2*: smooth part of the A4 road in Cracow with a stopped truck.
- Highway *id 3*: intersection part of the A4 road in Cracow.
- Highway *id 4*: exit lane from the A4 road in Cracow.
- Highway *id 5*: access road to A4 road in Cracow.

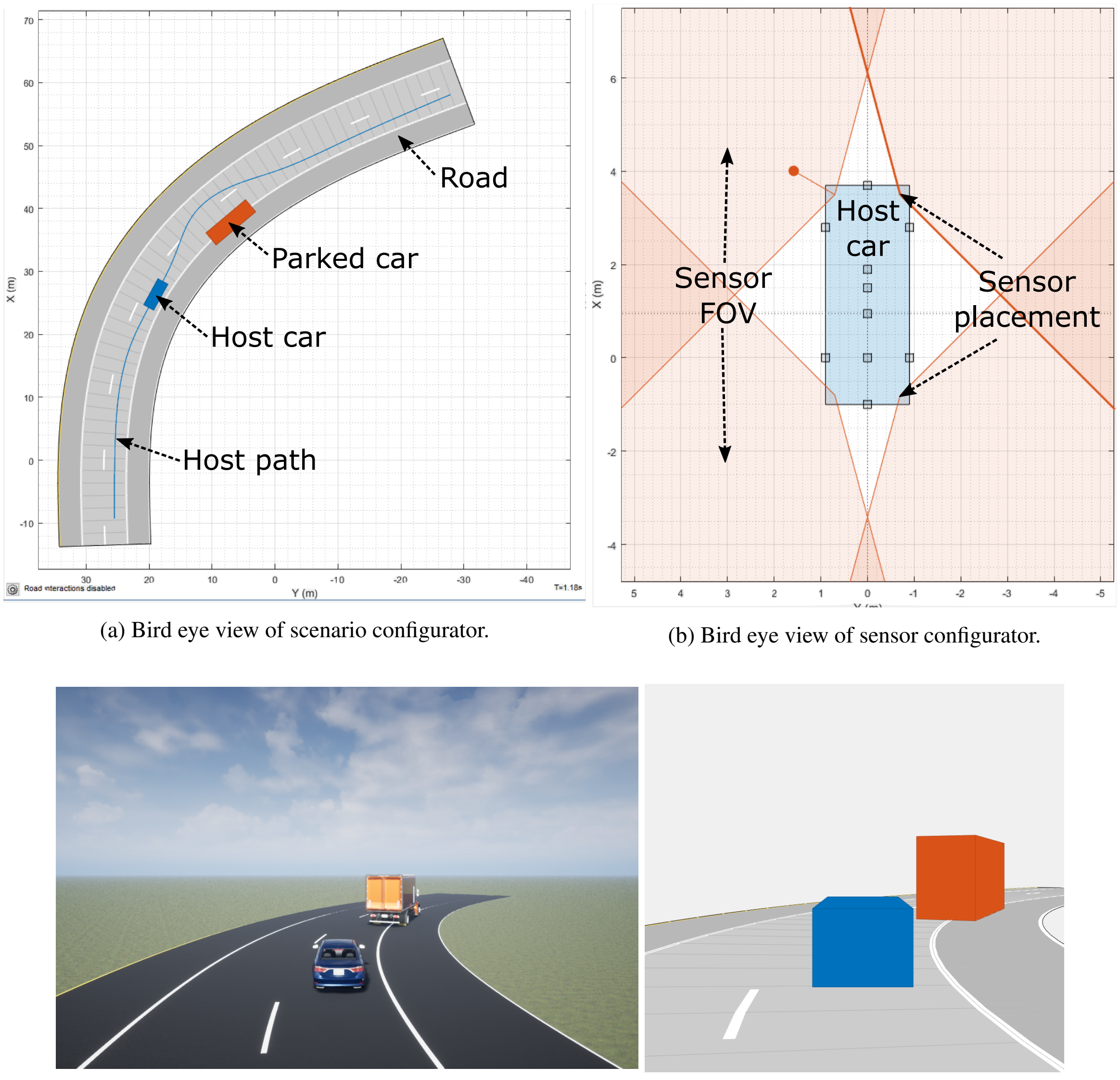
Details of each scenario can be found in Section A.2.1.

5.1.3.2 Parking

Parking category describes the situation in which a user tries to find a free parking slot. The parking scenario is characterized by complex shapes and low host car speed (around 30 km/h). The parking scenarios used for the tests are as follows:

- Parking *id 1*: parking in an open space with a few cars parked.
- Parking *id 2*: driving into parking area with some empty parking slots - variant 1.
- Parking *id 3*: driving into parking area with some empty parking slots - variant 2.
- Parking *id 4*: driving from one parking slot to another.
- Parking *id 5*: driving through the parking lot with few empty parking slots.

Details of each scenario can be found in A.2.3.



(a) Bird eye view of scenario configurator.

(b) Bird eye view of sensor configurator.

(c) Utility visualization feature.

(d) Simple visualization feature.

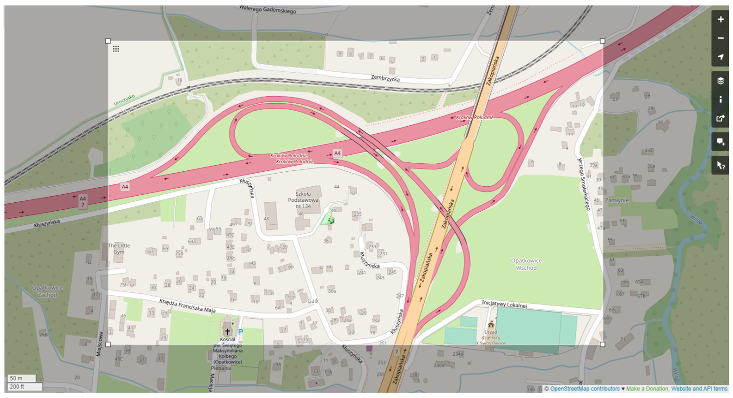
Figure 5.4. An example of a road scenario created in the Driving Scenario Designer tool.

5.1.3.3 City

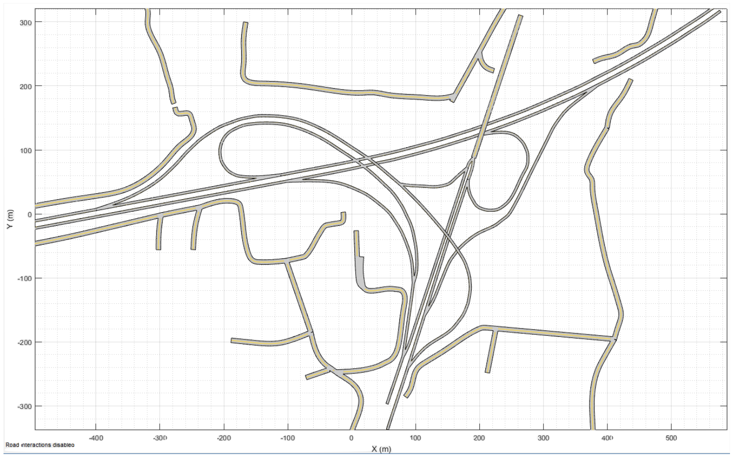
In city category the host car moves at a speed of around 60 km/h within a semi-complex surrounding environment. The field of view of the host car is mostly limited. The city scenarios used for the test are as follows:

- City *id 1*: driving city roads of Cracow.
- City *id 2*: driving along parked cars.
- City *id 3*: smooth crossroad.
- City *id 4*: smooth crossroad with gaps.
- City *id 5*: roundabout.

Details of each scenario can be found in A.2.2.



(a) Screenshot from OpenStreetMap website showing an example of roads intersection in Cracow.



(b) The map from 5.5a imported to Driving Scenario Designer.

Figure 5.5. An example of road export from OpenStreetMap to Driving Scenario Designer.

5.1.3.4 Shapes

This kind of scenario is used to check algorithm performance with unusual shapes of the free space. The host is stationary and the boundary of the free space is fully known from the beginning of each scenario.

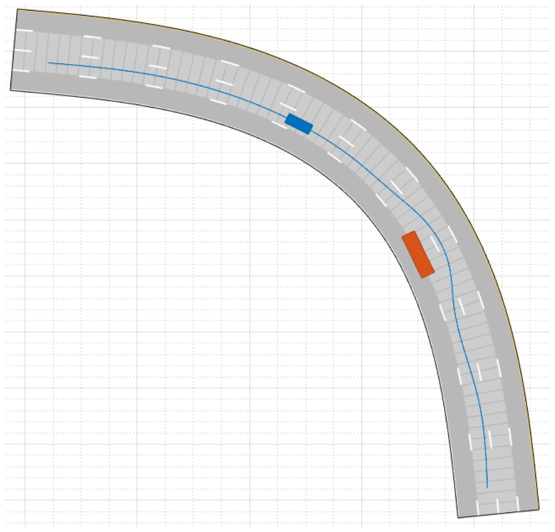
- Shape *id 1*
- Shape *id 2*
- Shape *id 3*
- Shape *id 4*
- Shape *id 5*

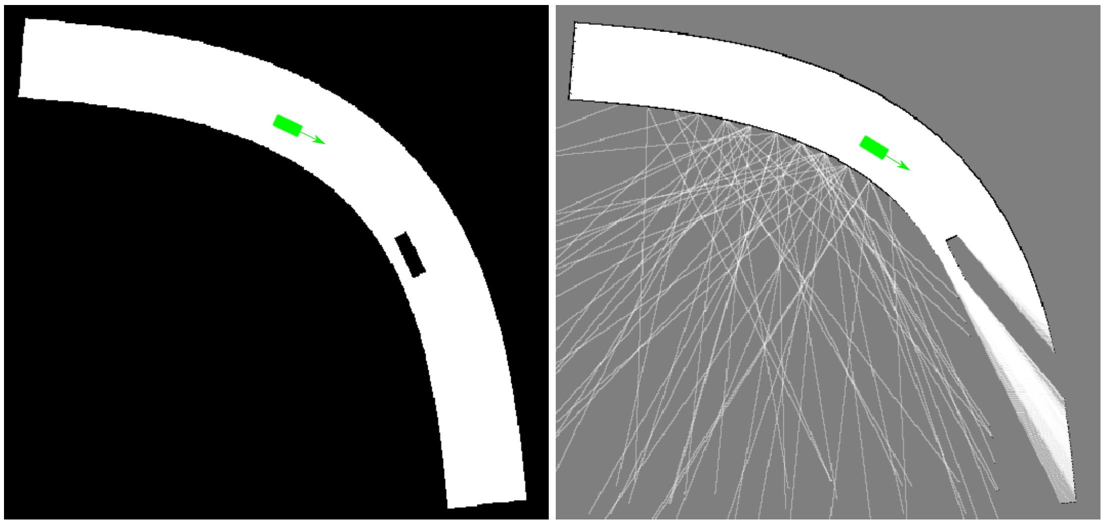
Details of each scenario can be found in A.2.4.

5.1.3.5 Mixed case

Mixed case scenarios are prepared to check the performance of the algorithm during the change of the surrounding environment from smooth to complex and vice versa.

- Mix *id 1*: from highway to city
- Mix *id 2*: from city to highway


(a) Source scenario.


(b) Binary grid generated directly from the source scenario.

(c) Occupancy grid generated by ray casting from the location of moving host.

Figure 5.6. Generation of a grid map.

Details of each scenario may be found in A.2.5.

5.2 Evaluation of free space boundary tracking quality

In this section, all metrics and performance evaluation methodologies are described in detail.

5.2.1 Reference definition

Both the proposed algorithm and the reference algorithm (PFS) use the occupancy grid data as input. In order to make the comparison clear and fair, both algorithms are fed with the same input data generated in the *Occupancy processing step* (described in Section 4.3) in the form of a set of points (pixels). These pixels are known reference data, used later to calculate the errors of the free space boundary approximation by both algorithms.

5.2.2 Approximation error metrics

5.2.2.1 Corresponding spline point approach

To make the evaluation more detailed, a pairing methodology is involved. For each reference point, a corresponding spline point is determined, creating a set of pairs per iteration. For the approximation quality evaluation, the following additional metrics are used.

Mean absolute deviation:

$$\varepsilon_k^{mean} = \frac{1}{N_k^{ref}} \sum_i \varepsilon_i \tag{5.2.1}$$

Maximum absolute deviation:

$$\varepsilon_k^{max} = \max_i \left(\varepsilon_i\right) \tag{5.2.2}$$

Standard absolute deviation:

$$\varepsilon_k^{std} = \sqrt{\frac{1}{N_k^{ref}} \sum_i \left(\varepsilon_i - \varepsilon_k^{mean}\right)^2} \tag{5.2.3}$$

Absolute deviation:

$$\varepsilon_i = d\left(\mathbf{r}(s_i), \mathbf{p}_i\right) \tag{5.2.4}$$

where:

i	–	reference point number $i = 1, 2, 3, ..., N_k^{ref}$
k	–	algorithm iteration number
$\mathbf{p}_i$	–	i-th reference point
$\mathbf{r}(s_i)$	–	i-th sampled spline point (corresponding to $\mathbf{p}_i$), determined by proposed methodologies
N_k^{ref}	–	number of reference points in k-th iteration
$d(a, b)$	–	distance between two points a and b (unsigned)

The corresponding spline point $\mathbf{p}_i$ may be determined either as the *closest spline point* or a *perpendicular spline point*. Both approaches are explained in Figure 5.7.

The *closest spline point* defines a spline point as a point that is the closest, in terms of the Euclidean distance, to the reference point.

The *perpendicular spline point* defines a spline point as a point that is closest, in terms of the Euclidean distance, to a perpendicular line created on the reference point of interest.

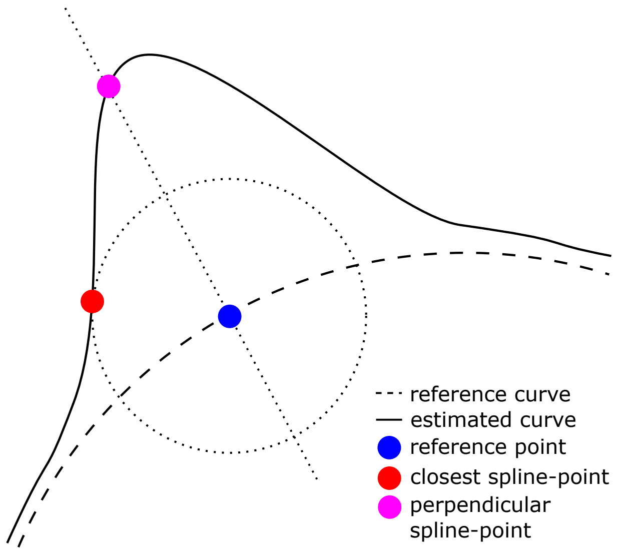


Figure 5.7. Spline-points determination [Szlachetka et al., 2022].

5.2.2.2 Hausdorff distance

Hausdorff distance is a metric that measures the distance between two subsets of a metric space [Kim et al., 2015]. Hausdorff distance may be understood as the longest distance from one set to another. It is defined as:

$$d_{\text{Hausdorff}}\,(X,Y) = max\left\{\sup_{x\in X} d_{\text{inf}}(x,Y), \sup_{y\in Y} d_{\text{inf}}(X,y)\right\} \tag{5.2.5}$$

where:

$$d_{\text{inf}}(x,Y) = \inf_{y\in Y} d(x,y) \tag{5.2.6}$$

is the infimum distance between the point x and a subset Y and:

X, Y	–	nonempty subsets.
$d(a,b)$	–	Euclidean distance between points a and b.
$\inf$	–	the infimum.
$\sup$	–	the supremum.

The Hausdorff distance is calculated as one value per iteration of the algorithm. The sets used in the metric are the reference points and the sampled spline points (equally distributed along the spline, having 10 times more elements than the number of reference points). The Hausdorff distance is a supporting metric that is used to show the worst approximation case per iteration.

5.2.3 Detection error metrics

In free space boundary tracking task it may happen some areas in which deviations of the approximating spline from the actual free space boundary may be unacceptably high. Such areas are referred to as detection errors. In general free space boundary tracking algorithm spline points may be classified as follows [Szlachetka et al., 2022]:

Approximating spline points for which the distance to the reference curve (actual free space boundary) is below a certain safe threshold (e.g. 50 cm) are called true positives (TP), because these estimates quite well match the true boundary (obstacle).

Other spline points not matching well the actual reference curve (actual free space boundary) are called detection errors. There are two possible types of detection error.

Type I error occurs when the approximating spline point distance from the reference boundary (actual boundary) is above the threshold and the point lays inside the free space area. Such a case is called a false positive (FP), because the algorithm falsely reports a non-existing obstacle (free space boundary). This type of error is harmless to the host because it can cause an unnecessary braking or maneuver, but the host will not hit non-existing obstacle reported by the algorithm.

Type II error occurs when the approximating spline point distance from the reference boundary (actual boundary) is above the threshold and the point lays outside the free space area. Such a case is called a false negative (FN), because the algorithm falsely reports that there is no obstacle in place where the actual obstacle is laying. This type of error may be harmful to the host because it can cause the host to hit an obstacle which is not reported close enough.

In order to fully evaluate the performance of the tracking algorithm, the following combined metrics are calculated based on counts, denoted by $\#$ symbol, of spline points defined above:

TPR – True Positive Rate — measures the relation of the count of spline points truly classified as positive to the count of all spline points which were actually positive (i.e. ones either classified truly as positive or wrongly classified as negative).

$$TPR = \frac{\#TP}{\#TP + \#FN} \tag{5.2.7}$$

PPV – Positive Predictive Value — measures the relation of the count of spline points truly classified as positive to the count of all spline points classified as positive (truly or wrongly).

$$PPV = \frac{\#TP}{\#TP + \#FP} \tag{5.2.8}$$

F1 – F1 score — is a frequently used combined measure of detection accuracy.

$$F1 = \frac{2 \cdot \#TP}{2 \cdot \#TP + \#FP + \#FN} \tag{5.2.9}$$

In the case of all of the above metrics the higher the value is the better. Figure 5.8 shows possible types of spline points.

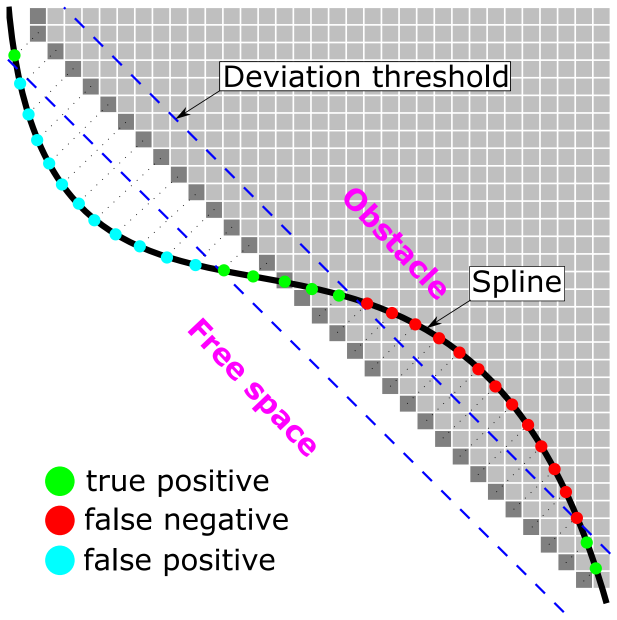


Figure 5.8. Illustration of detection errors.

5.3 Algorithm configuration and parameters tuning

With respect to the proposed algorithm, from the list of methods described in Sections 4.4.2.2 and 4.4.2.3, the following ones were chosen:

- Measurement downselection → *line downselection*
- Measurement association → *distance to measurement: sampled spline*

Furthermore, the values of the parameters of the proposed algorithm were arbitrarily chosen in the beginning and then tuned using the genetic algorithm [Mirjalili and Mirjalili, 2019] to obtain the final values.

5.4 Algorithms comparison

In this section, the cumulative results of the comparison of the proposed algorithm and the reference PFS algorithm are described. The comparison is made based on the metrics mentioned in Section 5.2 and the assumptions described in Section 5.4.1. Additionally, for each category of scenarios, an example scenario is chosen and described in detail.

5.4.1 Comparison assumptions

All the data presented in following sections, together with the plots, is prepared on the basis of the following assumptions:

- The number of control points in the reference algorithm is constant over entire log. However, in the proposed algorithm, the number of control points varies over time. Thus, after running the proposed algorithm, its average and median number of control points are calculated, and then the higher value is used as a fixed number of control points in the reference method.
- Initial iterations (before achieving a convergence) are not used in the error calculation.
- The initial number of control points in the proposed method is set to 20.

- The analysis is done on the set of scenarios described in Section 5.1.3
- Approximation metrics are calculated only from relevant elements (i.e. only samples marked as TP — Section 5.2.3) excluding Hausdorff metric.
- The Hausdorff metric is filtered in time (iterations) using moving window size 5 afterwards in order to filter out pikes affecting the analysis. Such an approach is commonly used in ADAS systems to achieve a higher confidence in an upper layer (data consumer) about the presence or absence of an obstacle.
- Each artificial scenario is generated to grid with 150x150m size and 0.2m resolution.

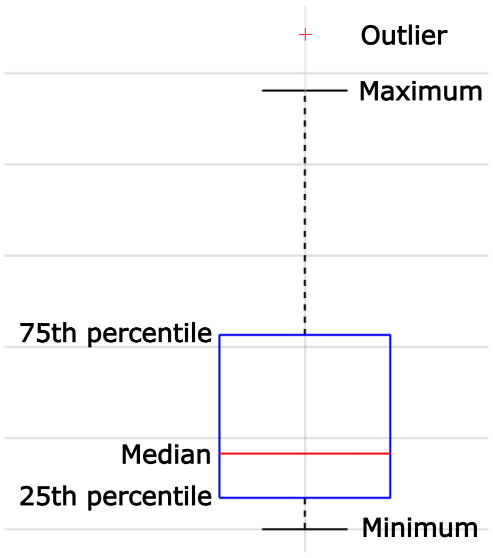


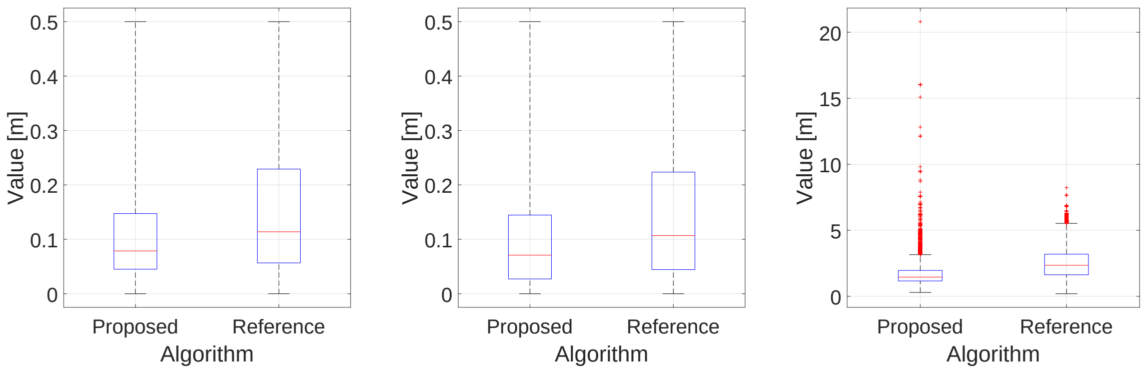
Figure 5.9. Approximation errors box plot description.

5.4.2 Artificial scenarios

The performance of both algorithms has been evaluated on 22 scenarios. Figure 5.10 shows the results accumulated from all scenarios in the form of box plots (explained in Figure 5.9). All subfigures point out that a better approximation quality is achieved by the proposed algorithm. In general its approximation error is lower (median absolute deviation is about 31% and mean 28% lower). The box plots shown in the subfigure 5.10c point to the advantage of the proposed algorithm compared to the reference algorithm.

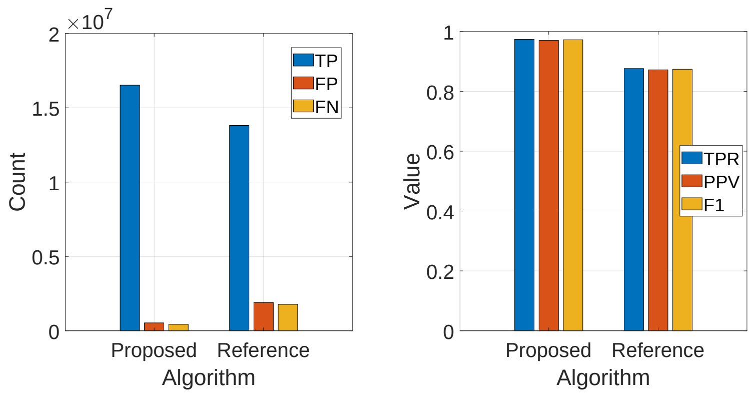
The results analysis with respect to the detection errors also indicates better performance of the proposed algorithm (Figure 5.11). The TP count is 20% higher, the FP count is 72% lower, and the FN count is 76% lower.

In terms of execution time, the proposed algorithm is almost two times faster than the reference algorithm, as shown in Figure 5.12. The OCG processing time does not affect the result because both algorithms use exactly the same OCG processing stage.



(a) Closest spline point. (b) Perpendicular spline point. (c) Hausdorff distance.

Figure 5.10. Approximation error box plots aggregated from all scenarios.



(a) Detection error. (b) Detection error rates.

Figure 5.11. Detection error counts and metrics aggregated from all scenarios.

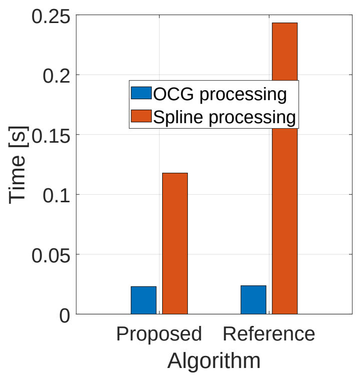


Figure 5.12. Algorithms' execution time aggregated from all scenarios.

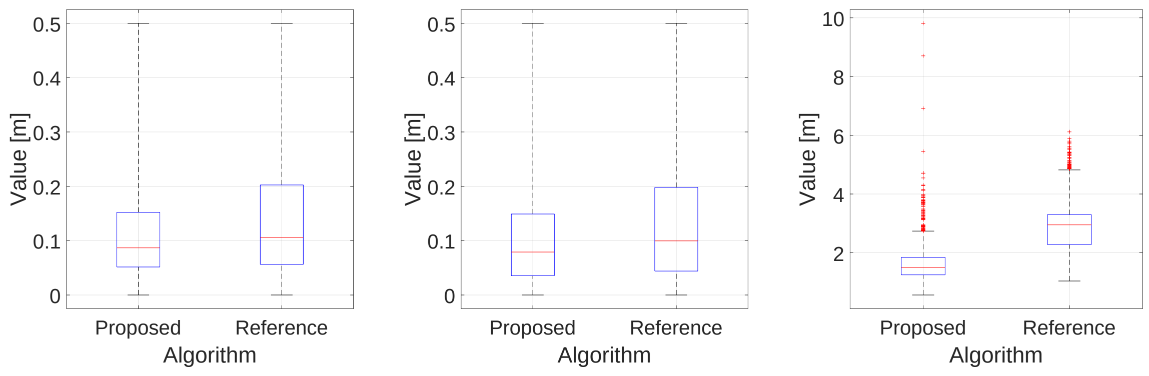
In the following sections, a similar analysis is performed for each category of scenarios. Also, an example scenario is analyzed in detail. Table 5.1 shows the number of control points used in the simulations for each of the scenarios taken (the complete list, which contains 22 scenarios, can be found in Table 5.5).

Table 5.1. Number of control points used in the tests scenarios.

Category	ID	Proposed		Reference
		Median	Mean	Used
Highway	2	18	18,68	19
City	2	20,85	21	21
Parking	4	26,58	28	28
Shapes	2	49,3	50	50
Mix	1	50,19	32	50

5.4.2.1 Highway

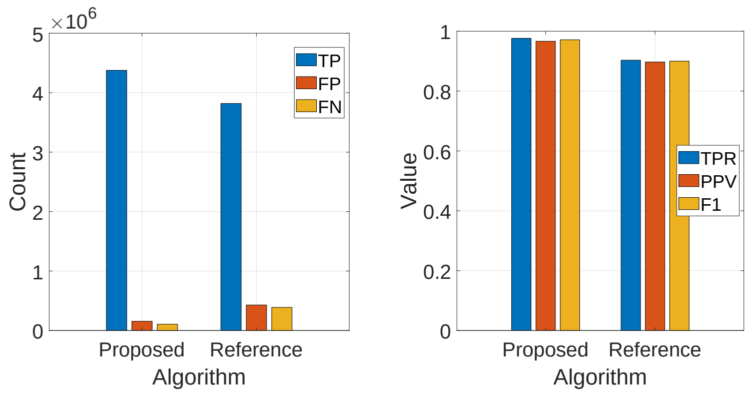
The analysis of the results of the highway scenarios does not differ much from the overall results. The analysis of the approximation error indicates better performance of the proposed algorithm (Figure 5.13). The mean absolute deviation is 18% and its median is 18% lower (better) than the reference algorithm. The results of Hausdorff distance analysis point to the advantage of the proposed algorithm compared to the reference algorithm.



(a) The closest spline point. (b) The perpendicular spline point. (c) The Hausdorff distance.

Figure 5.13. Approximation error box plots aggregated from highway scenarios.

The results analysis with respect to the detection errors shows a 14% higher TP count, a 64% lower FP count and a 73% lower FN count for the proposed algorithm (Figure 5.14).



(a) Detection error. (b) Detection error rates.

Figure 5.14. Detection error counts and rates aggregated from highway scenarios.

The execution time is similar to the overall results. The proposed algorithm is two times faster than the reference one (Figure 5.15).

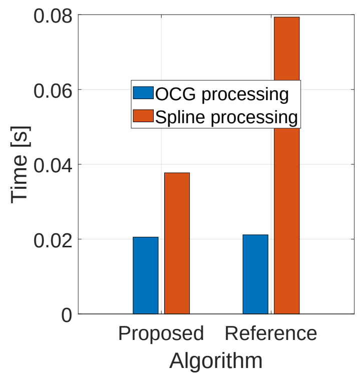


Figure 5.15. Algorithms' execution time aggregated from highway scenarios.

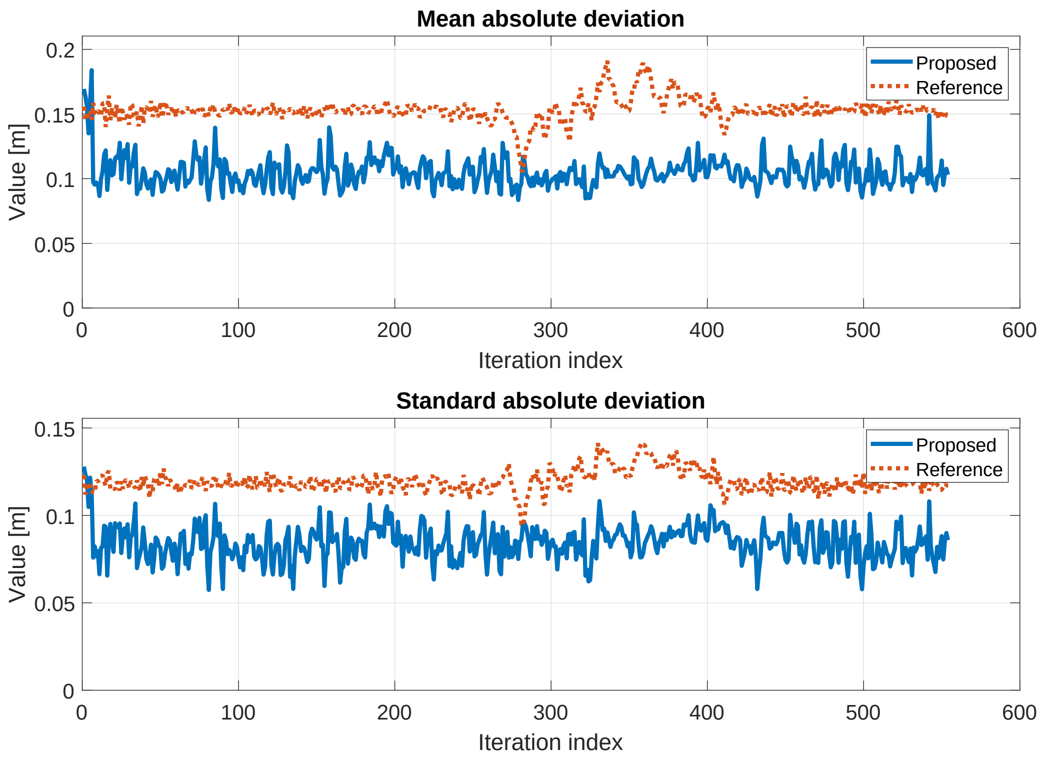


Figure 5.16. An example of approximation error per iteration for highway scenario *id 2*.

Scenario *id 2* is taken as an example scenario to be described in detail. Figure 5.16 shows the means of the approximation error metrics for each iteration. Most of the time, when a road has a simple shape, both algorithms have stable approximation errors (still better for the proposed algorithm). Starting from the 280th iteration to the 420th iteration, an obstacle is present inside the occupancy grid. It can be seen that the proposed algorithm maintains its performance, which cannot be said about the reference algorithm (Figure 5.17).

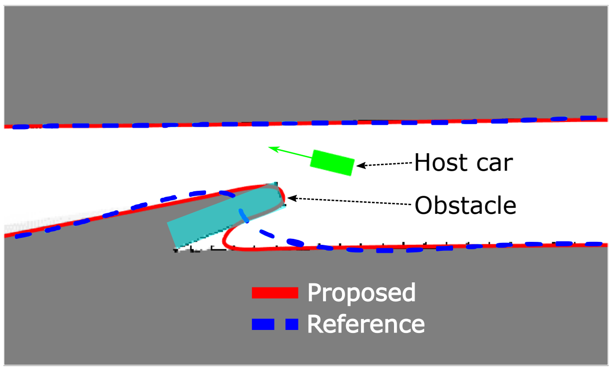


Figure 5.17. The approximation quality of the proposed algorithm and the reference one (PFS) in case of an obstacle appearance on a road.

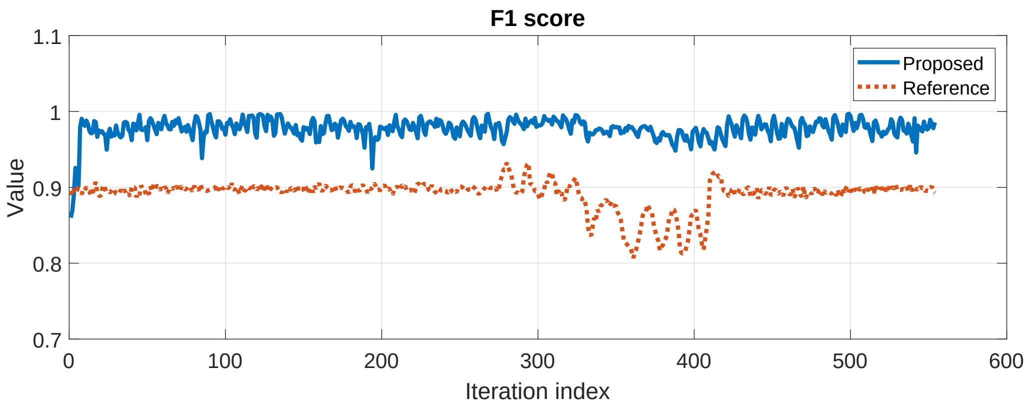


Figure 5.18. An example of F1 score value per iteration for highway scenario *id 2*.

The advantage of the proposed algorithm is also reflected in F1 score metric (Figure 5.18). The reason why the proposed algorithm can maintain a high approximation quality is the dynamic adjustment of the control points. Once the obstacle appears in the occupancy grid, the proposed algorithm adds some control points around it, making the approximation much better. The increase in the number of control points can be seen in Figure 5.19. The pick around the 320th iteration is caused by increased shape complexity. After a few iterations, the proposed algorithm made the decision to decrease this value slightly, but still maintaining high approximation quality. It is a typical behavior of the proposed algorithm, i.e., adding more control points around new complex shapes and then after a short time removing some control points. In this way, the proposed algorithm can reflect a complex shape very quickly.

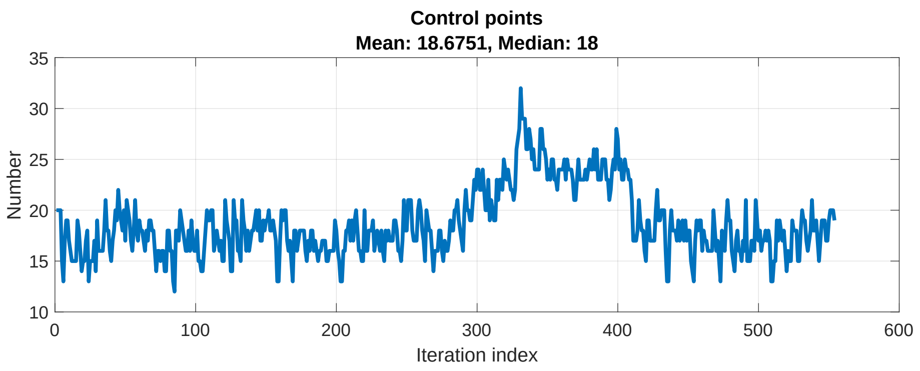
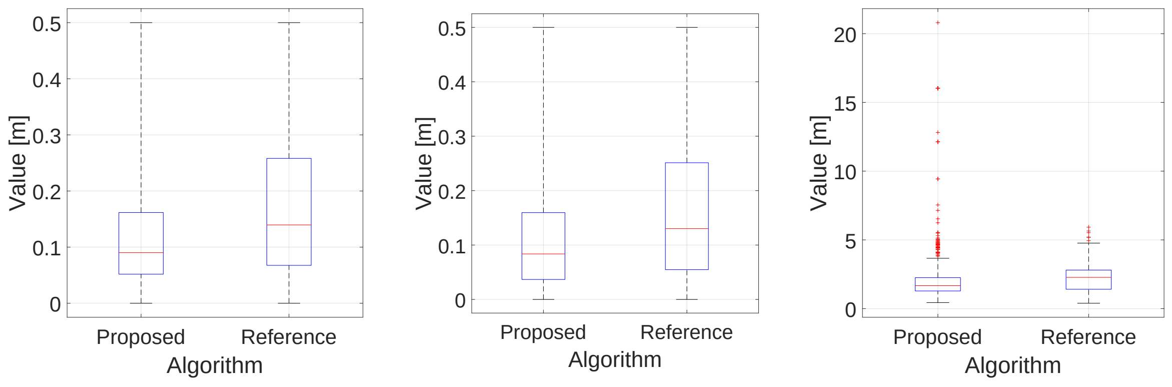


Figure 5.19. Number of control points per iteration for highway scenario *id 2*.

5.4.2.2 City

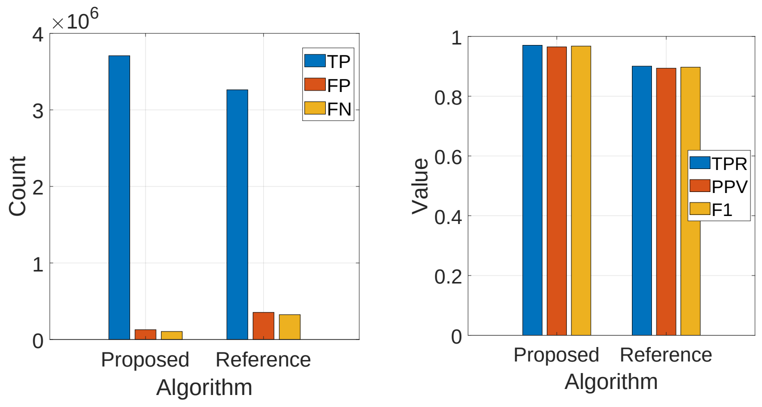
The analysis of the results of the city scenarios does not differ much from the overall results. The analysis of the approximation error indicates a better performance of the proposed algorithm (Figure 5.20). The mean absolute deviation is 30% and its median is 35% lower (better) than the reference algorithm. Hausdorff distance values are similar for both compared algorithms (but slightly better for the proposed algorithm).



(a) The closest spline point. (b) The perpendicular spline point. (c) The Hausdorff distance.

Figure 5.20. Approximation error box plots aggregated from city scenarios.

The results analysis with respect to detection errors shows a 14% higher TP count, 64% lower FP count, and a 68% lower FN count for the proposed algorithm (Figure 5.21).



(a) Detection error. (b) Detection error rates.

Figure 5.21. Detection error counts and rates aggregated from city scenarios.

The execution time is similar to the overall results. The proposed algorithm is two times faster than the reference algorithm (Figure 5.22).

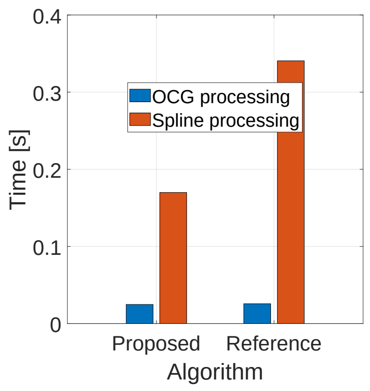


Figure 5.22. Algorithms' execution time aggregated from city scenarios.

Scenario *id 2* is taken as an example scenario to be described in detail. Figure 5.23 illustrates the means of the approximation error metrics for each iteration. All the time, both algorithms maintain stable approximation quality. It can be said that the proposed algorithm performs twice better here. This is due to the fact that the scenario does not have as many complex shapes, and the number of control points in the proposed algorithm does not change significantly over time (Figure 5.25). The value of detection metric, expressed by F1 score, is better for the proposed algorithm most of the time (Figure 5.24).

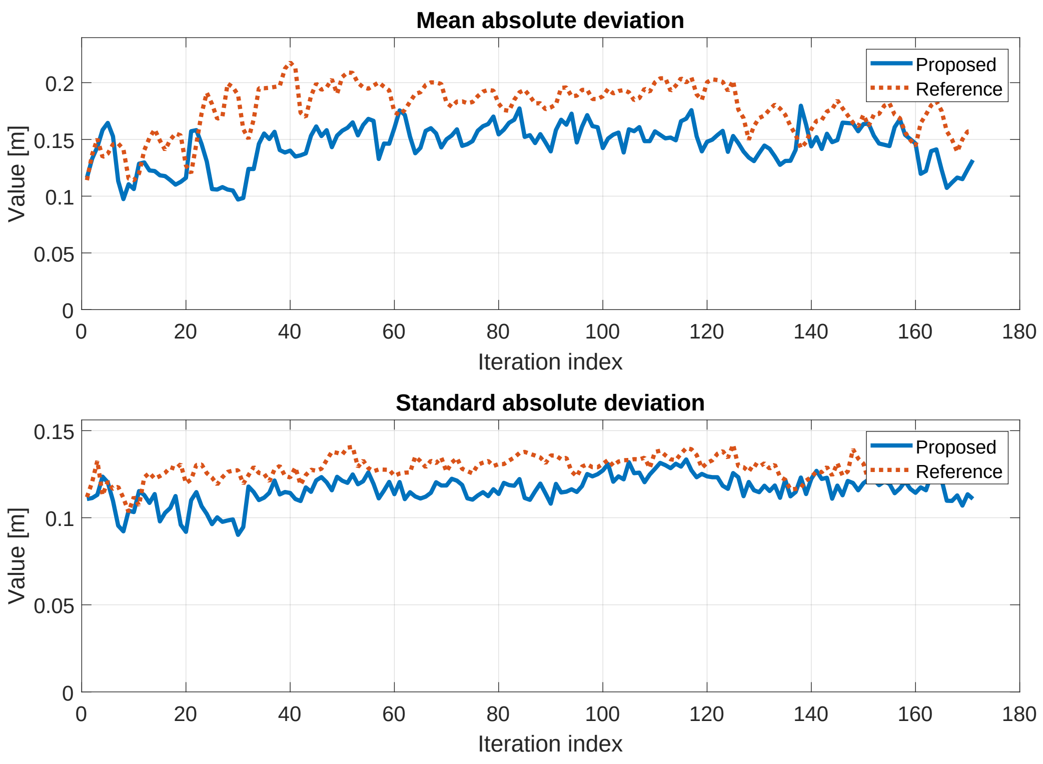


Figure 5.23. An example of approximation error per iteration for city scenario *id 2*.

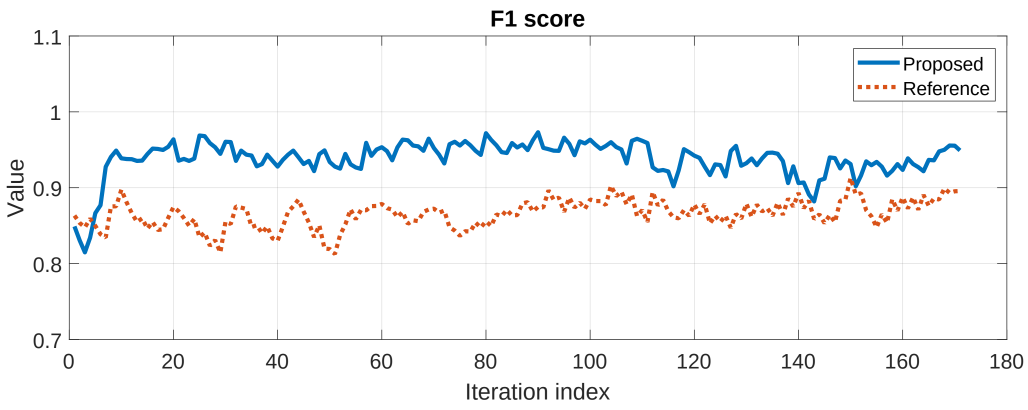


Figure 5.24. An example of F1 score value per iteration for city scenario *id 2*.

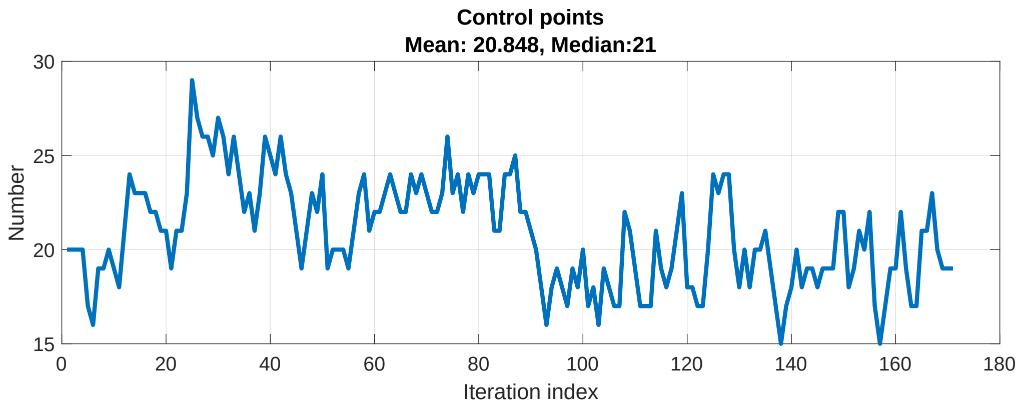
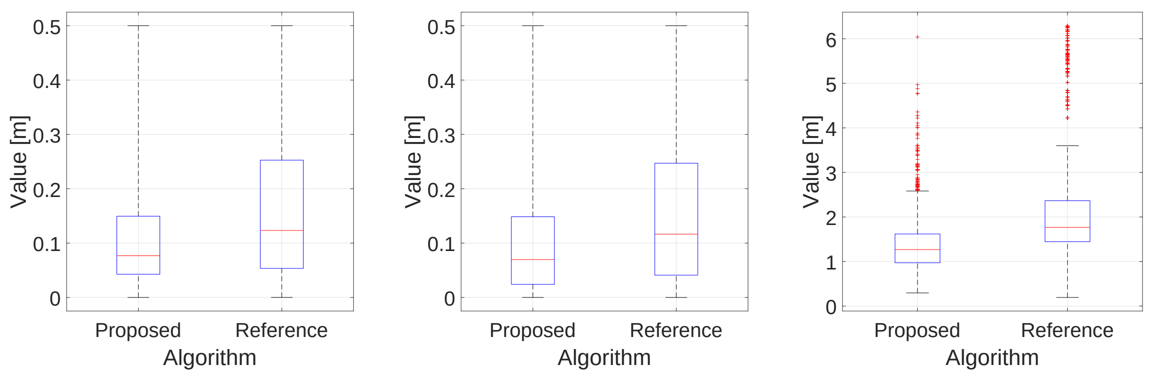


Figure 5.25. Number of control points per iteration for city scenario *id 2*.

5.4.2.3 Parking

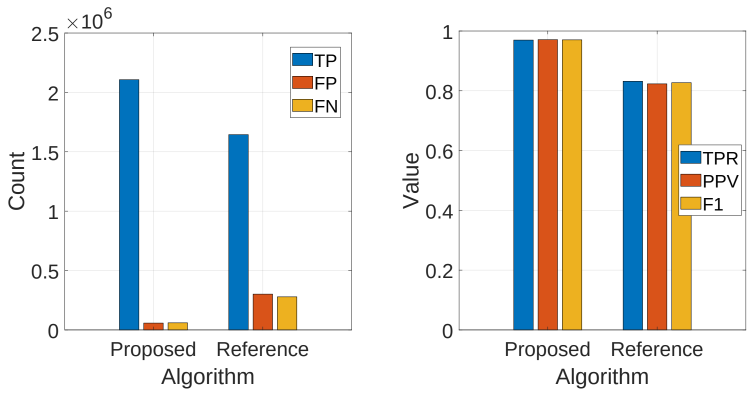
The analysis of the parking scenarios results is similar to the overall results. The approximation error is lower for the proposed algorithm (Figure 5.26). The mean absolute deviation is 33% and its median is 38% lower compared to the reference algorithm. The outcomes of the Hausdorff distance analysis suggest that the proposed algorithm has an advantage over the reference algorithm.



(a) The closest spline point. (b) The perpendicular spline point. (c) The Hausdorff distance.

Figure 5.26. Approximation error box plots aggregated from parking scenarios.

The analysis of the results with respect to the detection errors shows a 28% higher TP count, 81% lower FP count and a 79% lower FN count for the proposed algorithm (Figure 5.27).



(a) Detection error. (b) Detection error rates.

Figure 5.27. Detection error counts and rates aggregated from parking scenarios.

The execution time is similar to the overall results. The proposed algorithm is two times faster than the reference algorithm (Figure 5.28).

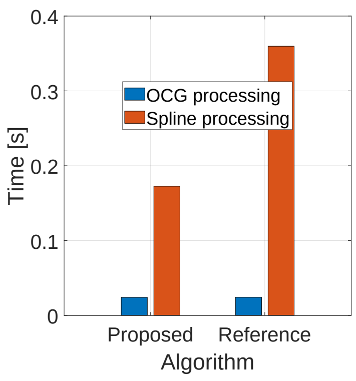
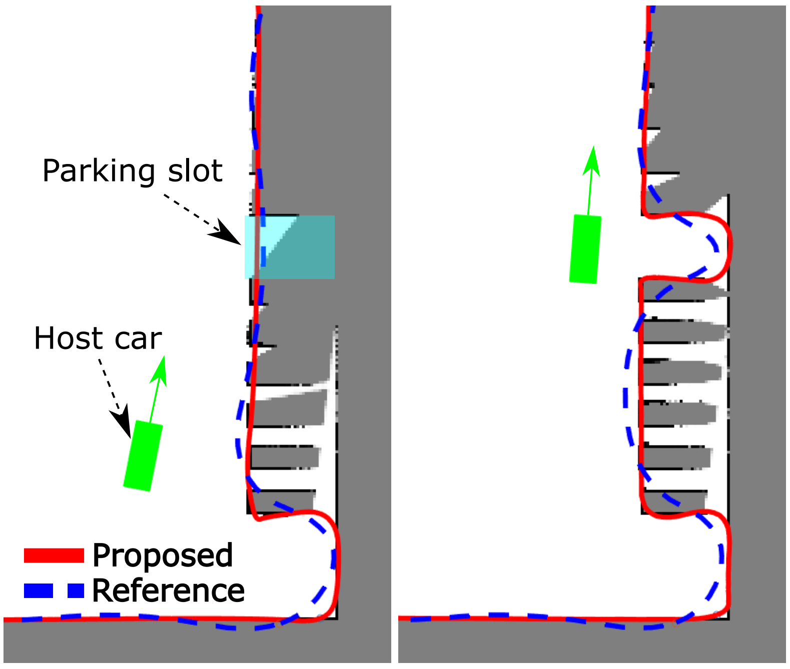


Figure 5.28. Algorithms' execution time aggregated from parking scenarios.

Scenario *id 4* is taken as an example, which is described in detail. Figure 5.30 shows the means of the approximation error metrics for each iteration. For most of the time, the proposed algorithm has better approximation quality. As may be noticed, the approximation quality of the proposed algorithm starts to increase around the 30th iteration. It is caused by the delay in adding control points. After some iterations, the algorithm starts to add some control points (around the 60th iteration). After that, the approximation quality increases significantly. Changes in the number of control points are reflected in Figure 5.32. The increase in the number of control points is caused by the detection of an empty parking space. Figure 5.29 illustrates this slot before and after making control points adjustment. Most of the time, the proposed algorithm tends to have a higher F1 score, indicating a better value for the detection metric (Figure 5.31).



(a) The spline approximation at 40th iteration.

(b) The spline approximation at 75th iteration.

Figure 5.29. The empty parking slot shape approximation. Initially the slot is not visible entirely for the host (self occlusion). After host movement the host can map and approximate entire slot.

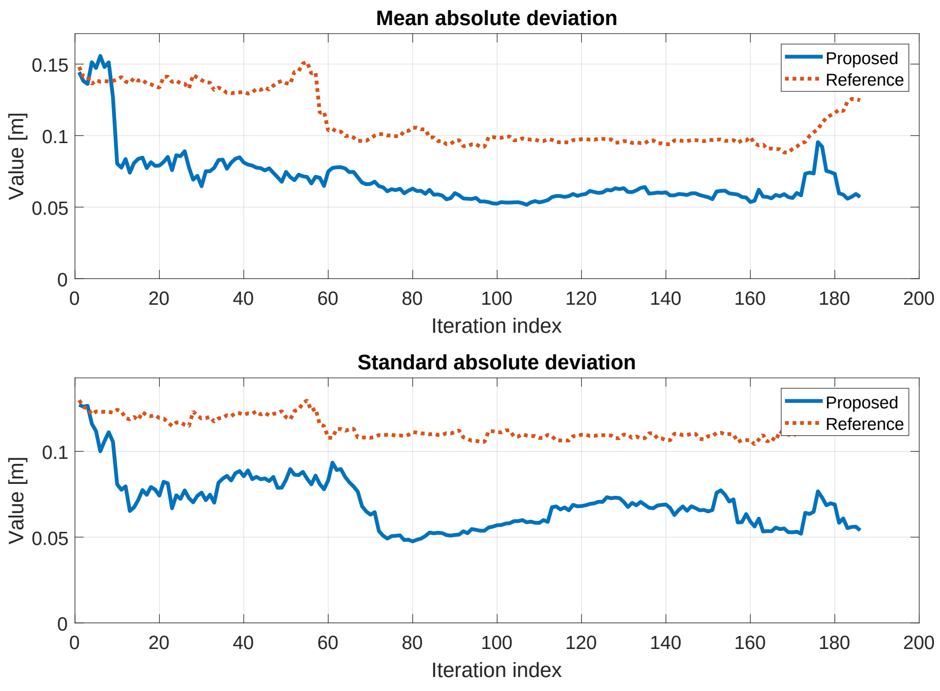


Figure 5.30. An example of approximation error per iteration for parking scenario *id 4*.

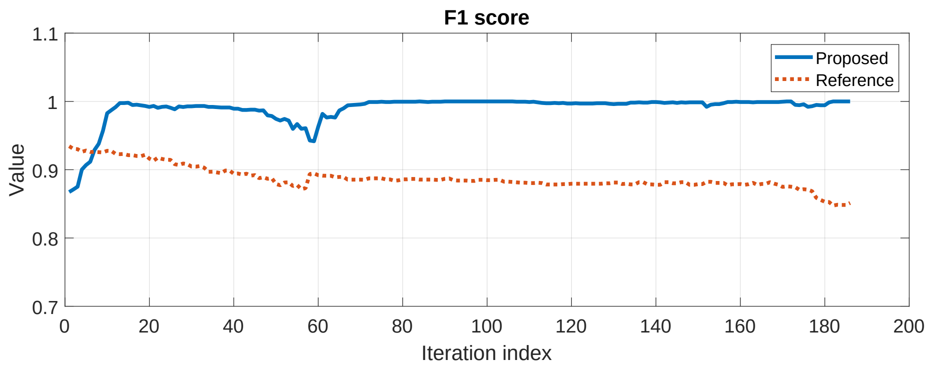


Figure 5.31. An example of F1 score value per iteration for parking scenario *id 4*.

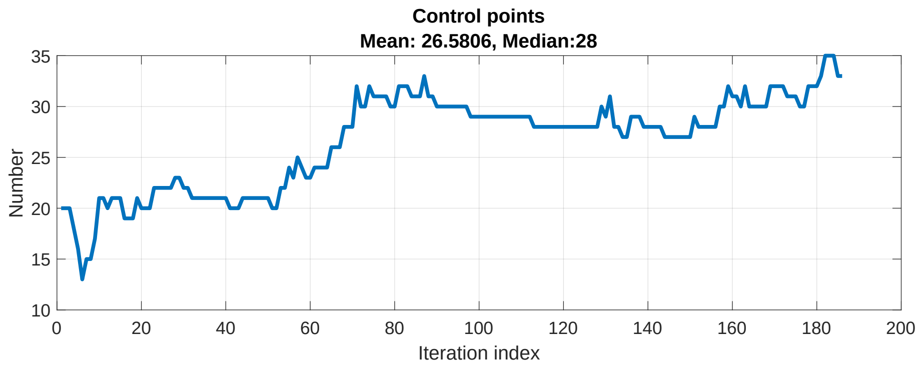
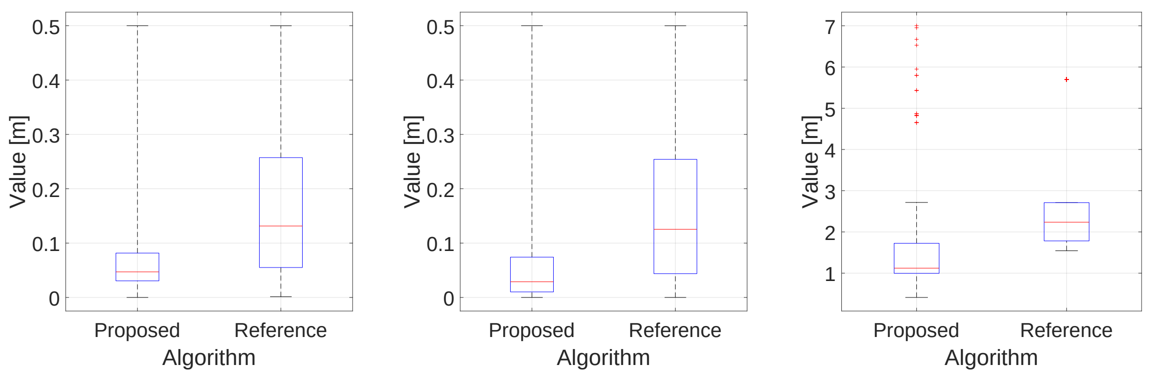


Figure 5.32. Number of control points per iteration for parking scenario *id 4*.

5.4.2.4 Shapes

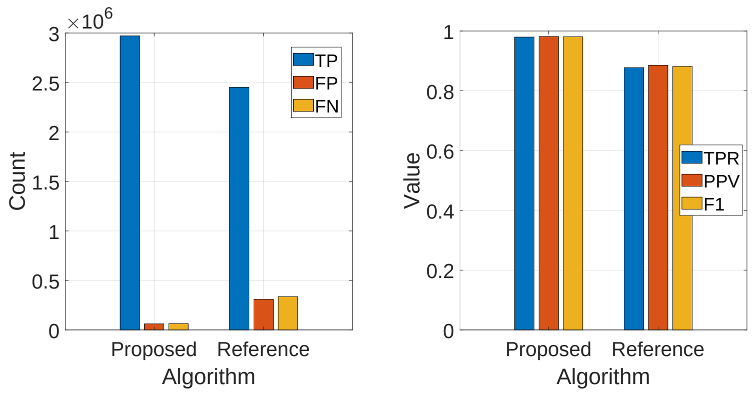
The results analysis of shapes scenarios are much better compared to the overall results. The approximation error is lower for the proposed algorithm (Figure 5.33). The mean absolute deviation is 57% and its median is 64% lower with respect to the reference algorithm (PFS). The findings from the Hausdorff distance analysis indicate that the proposed algorithm performs better than the reference algorithm.



(a) The closest spline point. (b) The perpendicular spline point. (c) The Hausdorff distance.

Figure 5.33. Approximation error box plots aggregated from shape scenarios.

The analysis of the results with respect to detection errors shows a 21% higher (better) TP count, 80% lower FP count, and 81% lower FN count for the proposed algorithm (Figure 5.34).



(a) Detection error. (b) Detection error rates.

Figure 5.34. Detection error counts and rates aggregated from shape scenarios.

The execution time is similar to the overall results. The proposed algorithm is more than two times faster than the reference algorithm (Figure 5.35).

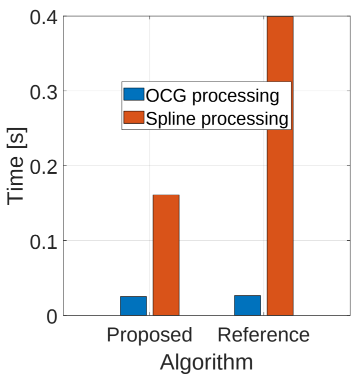


Figure 5.35. Algorithms' execution time aggregated from shape scenarios.

As an example scenario to be described in detail, scenario *id 2* is taken. Figure 5.36 shows the means of the approximation error metrics for each iteration. All shapes scenarios are used to check the algorithms performance from pure approximation point of view. Thus, the host is stopped and the generated occupancy grid is a constant binary grid (it does not change over time). This is why the approximation error plots are so stable. In addition, in this case, it can be said that the performance of the proposed algorithm is twice better. Its worse approximation error at the beginning is caused by an initialization time the proposed algorithm requires to calculate and refine shape complexity indicators etc. The stability of the approximation error plots is also reflected in the number of control points used in the plot shown in Figure 5.38. In general, the proposed algorithm outperforms other methods in terms of the detection metric, with a higher F1 score being the indicator of this superior performance (Figure 5.37).

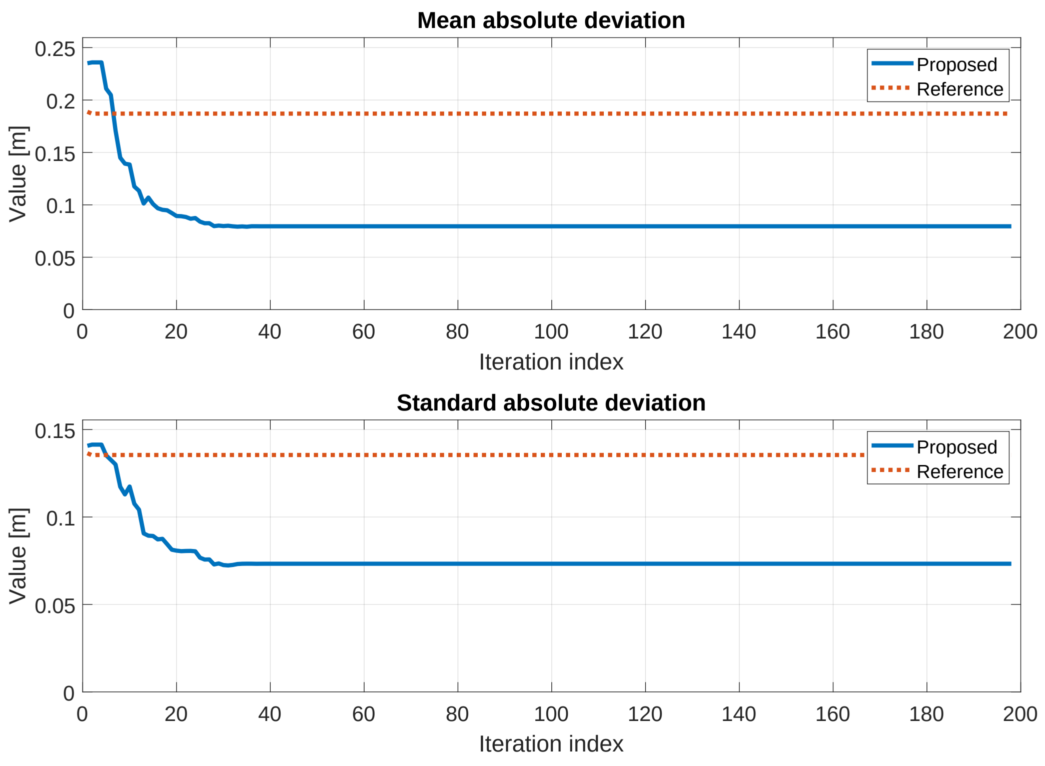


Figure 5.36. An example of approximation error per iteration for shape scenario *id 2*.

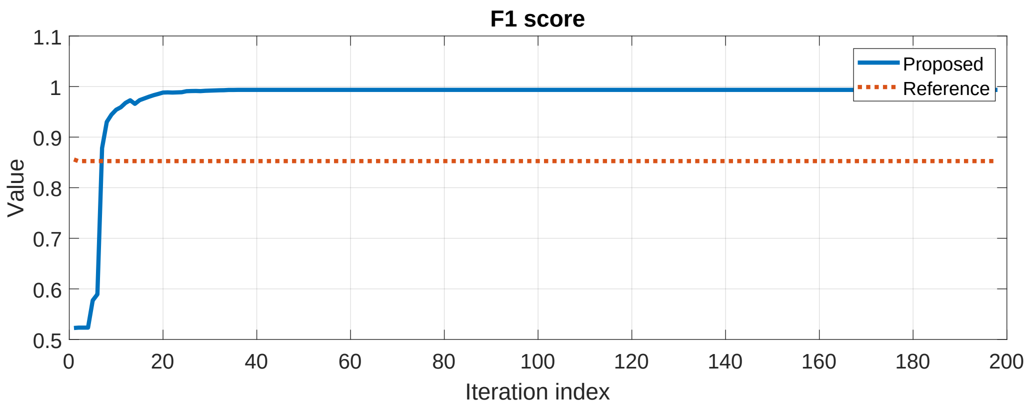


Figure 5.37. An example of F1 score value per iteration for shape scenario *id 2*.

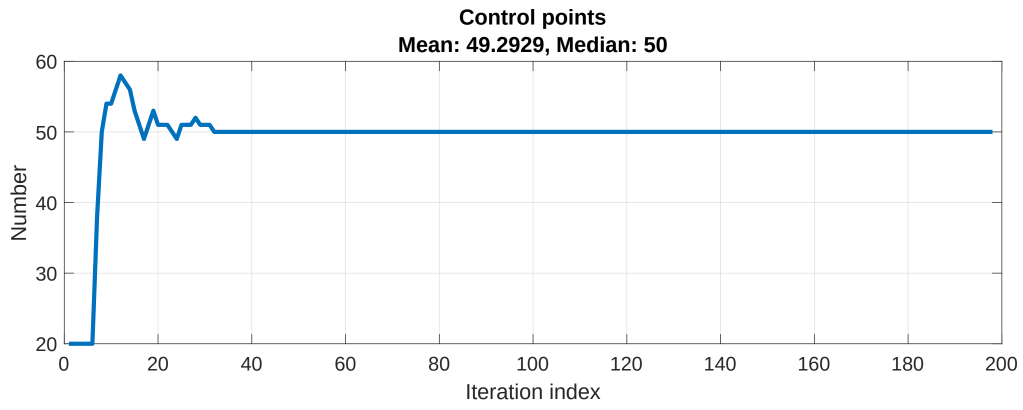
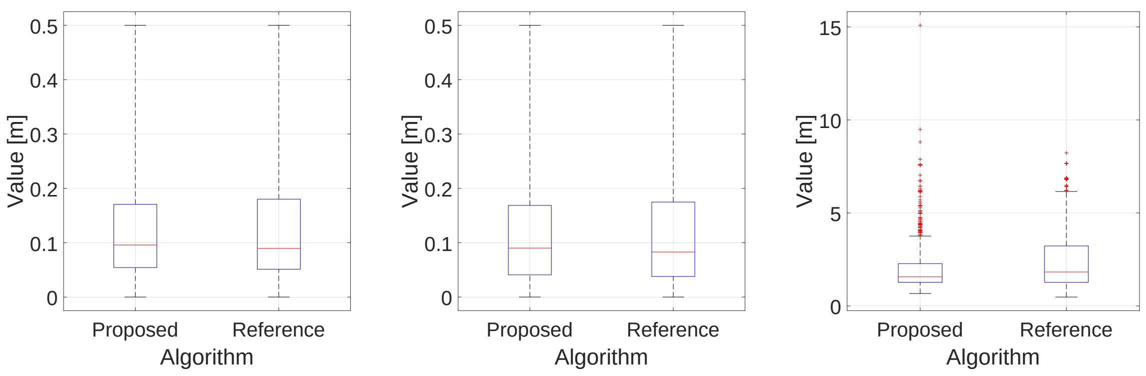


Figure 5.38. Number of control points per iteration for shape scenario *id 2*.

5.4.2.5 Mix

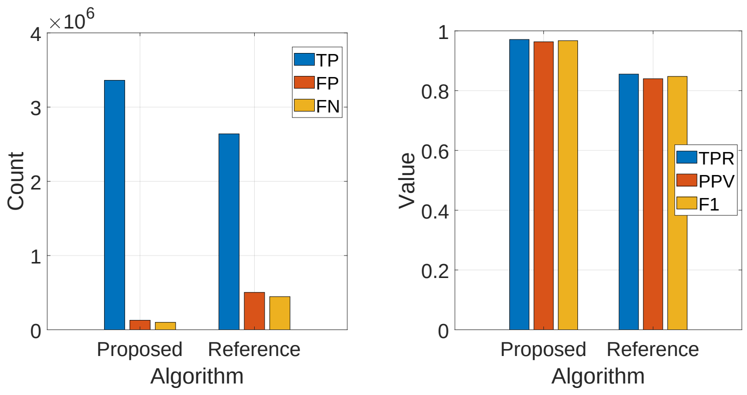
The results analysis of mix scenarios are better compared to the overall results. The approximation errors are comparable for the proposed algorithm and the reference one (Figure 5.39). The mean absolute deviation is 5% lower (better) while its median is 7% higher (worse) with respect to the reference algorithm. Based on the analysis of the Hausdorff distance, it can be concluded that the proposed algorithm has an edge over the reference algorithm.



(a) The closest spline point. (b) The perpendicular spline point. (c) The Hausdorff distance.

Figure 5.39. Approximation error box plots aggregated from mix scenarios.

The results analysis with respect to the detection errors shows 27% higher TP count, 74% lower FP count and 78% lower FN count for the proposed algorithm (Figure 5.40).



(a) Detection error. (b) Detection error rates.

Figure 5.40. Detection error counts and rates aggregated from mix scenarios.

The execution time is similar to the overall results. The proposed algorithm is about two times faster than the reference algorithm (Figure 5.41).

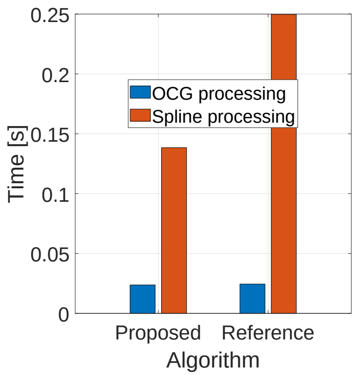
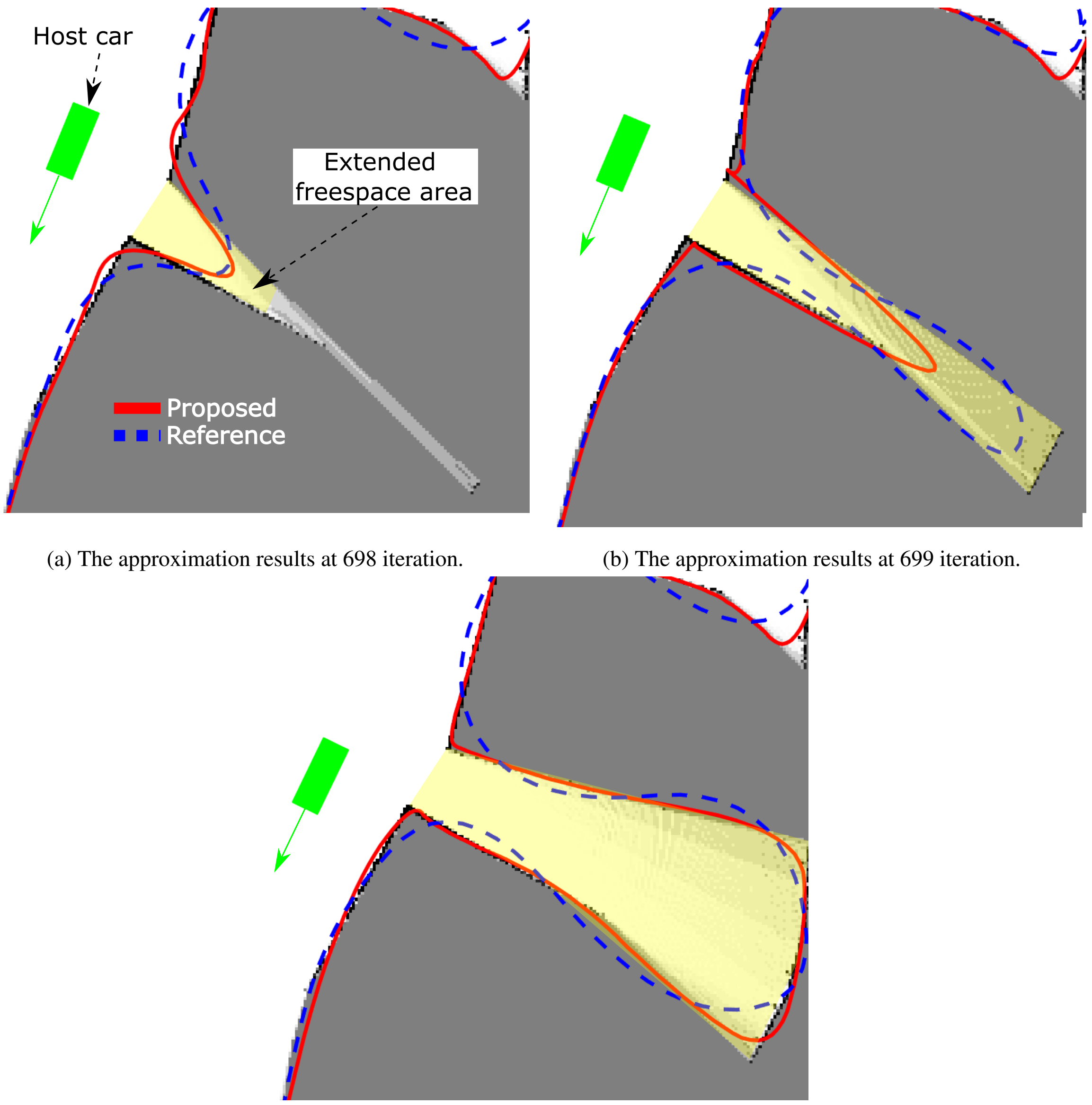


Figure 5.41. Algorithms' execution time aggregated from mix scenarios.



(a) The approximation results at 698 iteration.

(b) The approximation results at 699 iteration.

(c) The approximation results at 702 iteration.

Figure 5.42. The approximation of shapes that are changed significantly between two following iterations.

As an example scenario to be described in detail, scenario *id 1* is taken. Figure 5.43 shows the means of the approximation error metrics for each iteration.

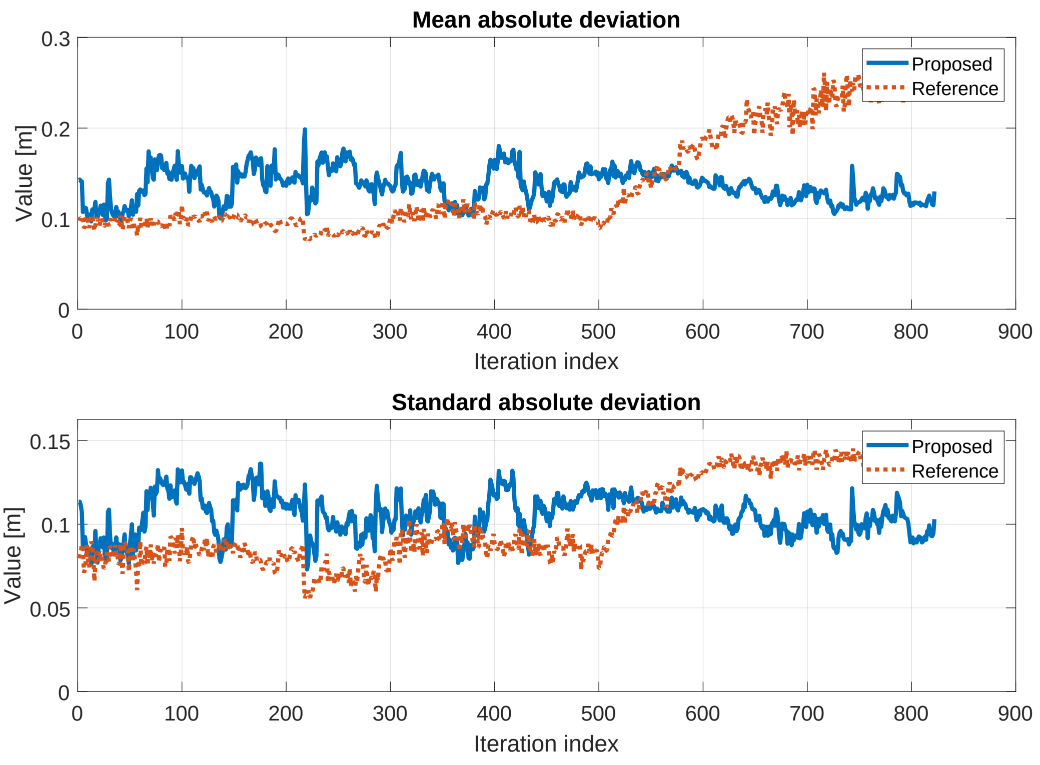


Figure 5.43. An example of approximation error per iteration for mix scenario *id 1*.

The first part of the example scenario is the highway type. Around 500th iteration, it is changed to the city type. It can be seen that, starting from the 500th iteration, the approximation quality of the reference algorithm starts to decrease and is worse and worse until the end of the scenario. The same behavior can be seen in Figure 5.44 showing F1 score value per iteration. In the highway part, the F1 score is similar for both algorithms (a little better for the reference one). In the city part, the F1 score for the reference algorithm drops drastically. The F1 score of the proposed algorithm remains the same level.

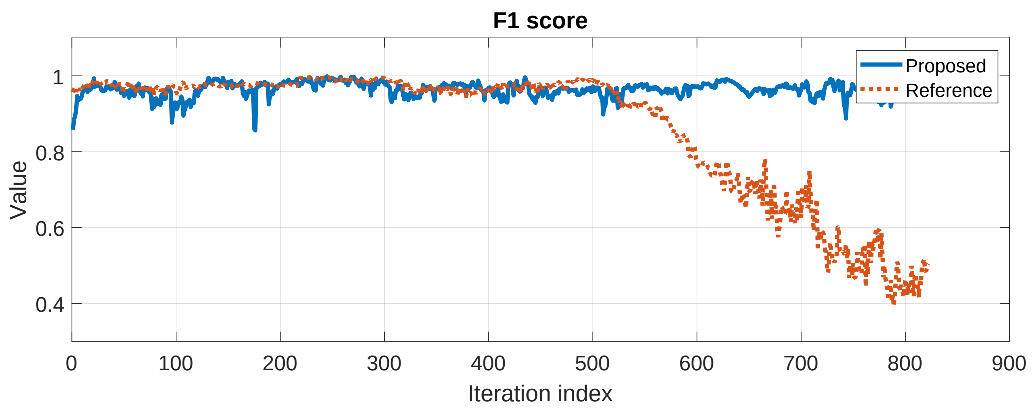
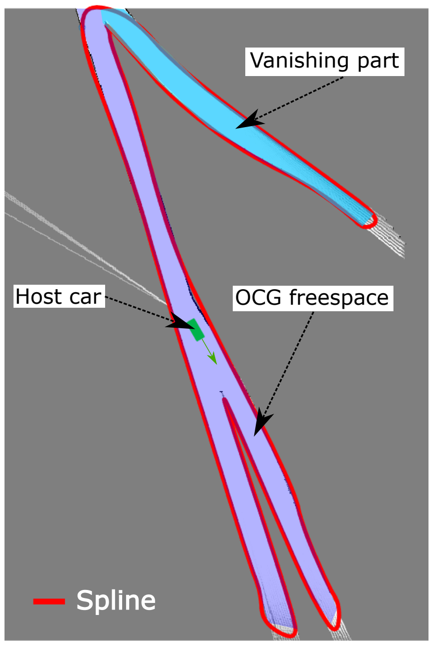
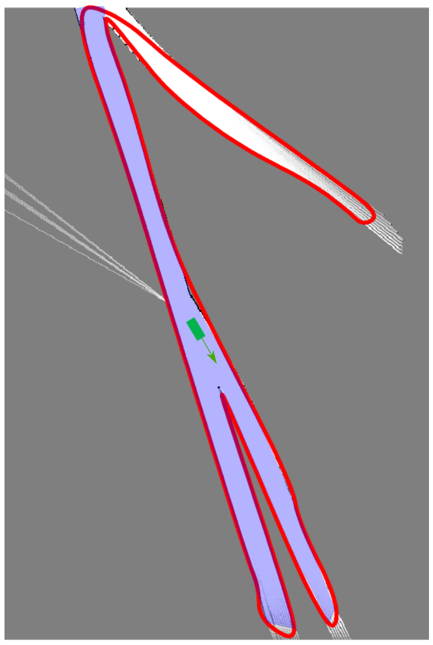


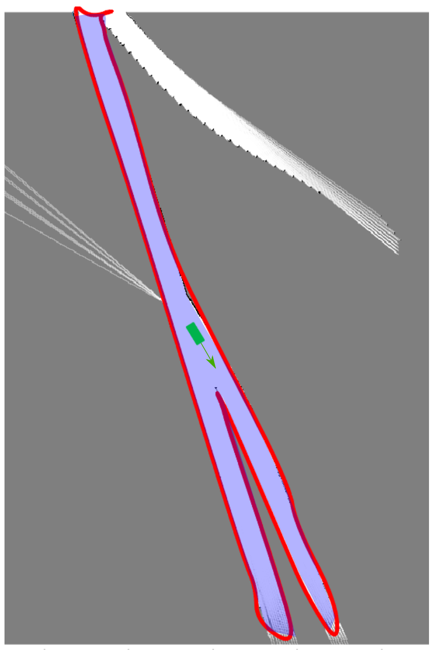
Figure 5.44. An example of F1 score value per iteration for mix scenario *id 1*.



(a) The approximation results at 130-th iteration.



(b) The approximation results at 131-th iteration.



(c) The approximation results at 132-th iteration.

Figure 5.45. Vanishing shape influencing approxmiation errors.

The number of control points in the reference algorithm is fixed, and the proposed algorithm iteratively adds control points (Figure 5.46). The fixed number of control points in the reference algorithm was enough for the highway type scenario, but not for the city type.

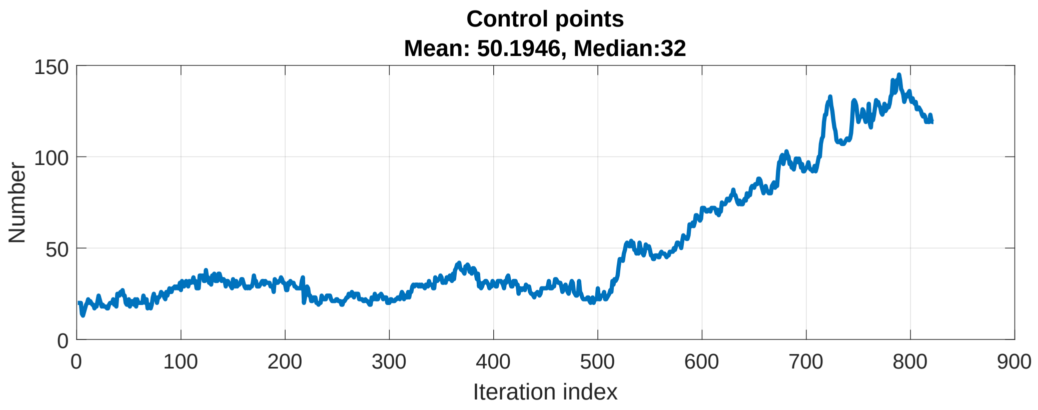


Figure 5.46. Number of control points per iteration for mix scenario *id 1*.

Some peaks visible in the proposed algorithm approximation error plots, like those around 520th, 590th and 700th iterations are caused by the time the algorithm requires to approximate a new long shape that appears between two following iterations (Figure 5.42). Another reason is the vanishing of the already approximated shape. Figure 5.45 shows this situation that occurs in the following iterations. The free space is marked by the blue area. There is a big change in the actual free space between two iterations, which is reflected by the approximating spline with some delay. The reason for this is the same as mentioned before, i.e. indicators and some other parameters are filtered in time and thus require some time to converge.

Compared to the reference algorithm, the goal of the proposed algorithm is not to make a one-time-instance approximation (low measurement noise), but to balance the approximation quality over time versus the computing power demand (proportional to the number of control points).

5.4.2.6 Summary

All presented results prove that the proposed algorithm has a better overall performance (approximation quality and execution time). It is worth to mention that even though the execution time of the proposed algorithm is about two times shorter than that of the reference algorithm for a comparable number of control points, its approximation quality is still much better than that of the reference algorithm. It may be stated that measurement points downselection combined with the dynamic spline points adjustment make estimated spline much closer to the reference free space boundary. Table 5.2, Table 5.3 and Table 5.4 contain summary results for the 22 scenarios tested.

Table 5.2. Approximation errors in all scenarios. P — proposed algorithm, R — reference algorithm (the PFS). "The closest spline point" metric is used.

Category	**ID**	**Mean**		**Median**		**Deviation**	
		P	R	P	R	P	R
City	1	0.13	0.18	0.09	0.14	0.10	0.12
City	2	0.15	0.18	0.11	0.14	0.12	0.13
City	3	0.11	0.18	0.08	0.14	0.09	0.13
City	4	0.10	0.17	0.07	0.13	0.08	0.13
City	5	0.13	0.17	0.10	0.13	0.10	0.13
Highway	1	0.10	0.15	0.07	0.11	0.09	0.12
Highway	2	0.10	0.15	0.08	0.12	0.09	0.12
Highway	3	0.12	0.15	0.09	0.11	0.10	0.12
Highway	4	0.13	0.13	0.10	0.10	0.10	0.11
Highway	5	0.13	0.14	0.10	0.10	0.10	0.11
Mix	1	0.13	0.13	0.10	0.09	0.11	0.12
Mix	2	0.12	0.14	0.09	0.09	0.10	0.12
Parking	1	0.14	0.20	0.10	0.16	0.12	0.14
Parking	2	0.11	0.16	0.08	0.12	0.10	0.13
Parking	3	0.11	0.18	0.08	0.15	0.10	0.13
Parking	4	0.07	0.11	0.04	0.06	0.07	0.11
Parking	5	0.12	0.18	0.10	0.14	0.09	0.13
Shape	1	0.07	0.16	0.04	0.12	0.07	0.13
Shape	2	0.08	0.19	0.06	0.15	0.08	0.14
Shape	3	0.07	0.16	0.04	0.09	0.07	0.14
Shape	4	0.09	0.17	0.05	0.14	0.09	0.13
Shape	5	0.06	0.17	0.04	0.12	0.06	0.13
Average		0.11	0.16	0.08	0.12	0.09	0.13

Table 5.3. Detection error counts in all scenarios. P — proposed algorithm, R — reference algorithm (the PFS), P/R — ratio (value above 1.0 indicates better performance of the proposed algorithm in case of TP, while less than 1.0 are better in case of FP and FN). "The closest spline point" metric is used.

Category	ID	TP			FP			FN		
		P	**R**	**P/R**	**P**	**R**	**P/R**	**P**	**R**	**P/R**
City	1	1349485	1269513	1.06	57317	98649	0.58	51709	90349	0.57
	2	231104	199523	1.16	15334	31753	0.48	14919	30081	0.5
	3	755750	652669	1.16	13956	60582	0.23	10753	67208	0.16
	4	529403	457718	1.16	9057	45255	0.2	6709	42196	0.16
	5	842015	682867	1.23	32930	117951	0.28	20829	94956	0.22
Highway	1	739797	588828	1.26	21992	95951	0.23	17085	94095	0.18
	2	795306	668414	1.19	19401	73089	0.27	17410	90614	0.19
	3	829617	730096	1.14	32518	78716	0.41	26745	80068	0.33
	4	880350	803506	1.1	35302	74606	0.47	20420	57960	0.35
	5	1130711	1027279	1.1	45552	106591	0.43	23787	66180	0.36
Mix	1	1728450	1356719	1.27	72347	262360	0.28	52415	234133	0.22
	2	1632426	1282467	1.27	55886	240981	0.23	47529	212393	0.22
Parking	1	91598	57661	1.59	4828	19929	0.24	5502	24368	0.23
	2	864194	712497	1.21	21908	98051	0.22	23667	99221	0.24
	3	449632	307658	1.46	17031	89678	0.19	16859	86186	0.2
	4	205031	166819	1.23	2043	26368	0.08	1940	15827	0.12
	5	496667	399838	1.24	11777	67382	0.17	11617	52841	0.22
Shape	1	730624	601320	1.22	14930	72272	0.21	16350	88312	0.19
	2	585257	451256	1.3	9987	72270	0.14	12022	83740	0.14
	3	302417	261764	1.16	3673	26737	0.14	5958	23547	0.25
	4	844428	756750	1.12	24921	70691	0.35	22443	64351	0.35
	5	508313	380355	1.34	7588	66540	0.11	7215	76221	0.09
Average		751026	627978	1.23	24104	86200	0.27	19722	80675	0.25

Table 5.4. Detection error rates in all scenarios. P — proposed algorithm, R — reference algorithm (the PFS). The "closest spline point" metric is used.

Category	**ID**	**TPR**		**PPV**		F1	
		P	**R**	P	**R**	P	**R**
City	1	0.96	0.93	0.96	0.93	0.96	0.93
	2	0.94	0.87	0.94	0.86	0.94	0.87
	3	0.99	0.91	0.98	0.92	0.98	0.91
	4	0.99	0.92	0.98	0.91	0.99	0.91
	5	0.98	0.88	0.96	0.85	0.97	0.87
Highway	1	0.98	0.86	0.97	0.86	0.97	0.86
	2	0.98	0.88	0.98	0.9	0.98	0.89
	3	0.97	0.9	0.96	0.9	0.97	0.9
	4	0.98	0.93	0.96	0.92	0.97	0.92
	5	0.98	0.94	0.96	0.91	0.97	0.92
Mix	1	0.97	0.85	0.96	0.84	0.97	0.85
	2	0.97	0.86	0.97	0.84	0.97	0.85
Parking	1	0.94	0.7	0.95	0.74	0.95	0.72
	2	0.97	0.88	0.98	0.88	0.97	0.88
	3	0.96	0.78	0.96	0.77	0.96	0.78
	4	0.99	0.91	0.99	0.86	0.99	0.89
	5	0.98	0.88	0.98	0.86	0.98	0.87
Shape	1	0.98	0.87	0.98	0.89	0.98	0.88
	2	0.98	0.84	0.98	0.86	0.98	0.85
	3	0.98	0.92	0.99	0.91	0.98	0.91
	4	0.97	0.92	0.97	0.91	0.97	0.92
	5	0.99	0.83	0.99	0.85	0.99	0.84
Average		0.97	0.88	0.97	0.87	0.97	0.87

5.4.3 Real world scenario

In the real test case, the vehicle has been equipped with four radar sensors mounted on each corner. These sensors have the field of view of 150° and can detect objects up to a maximum distance of 100 meters (Figure 5.47). They use frequency-modulated continuous-wave technology operating in a millimeter-wave bandwidth. The occupancy grid data has been created using the assistance of [Porebski, 2022] and has a resolution of 0.2 meters within a 100 x 100 meter grid. Raw data for the scenario were gathered on the parking with some obstacles present within sensors field of view (Figure 5.48).

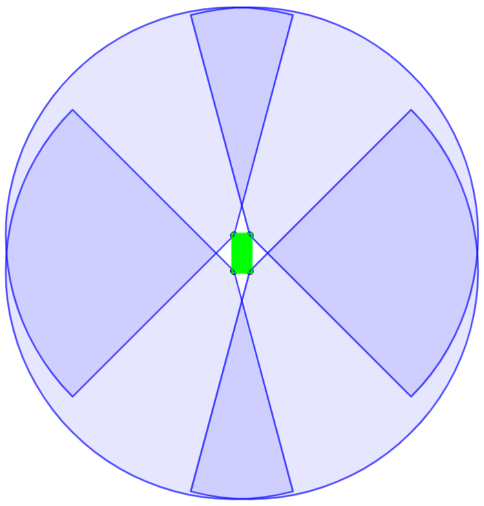


Figure 5.47. Radars coverage of the vehicle (depicted as the green rectangle). The relation between vehicle size and radar range is not kept to improve the clarity.

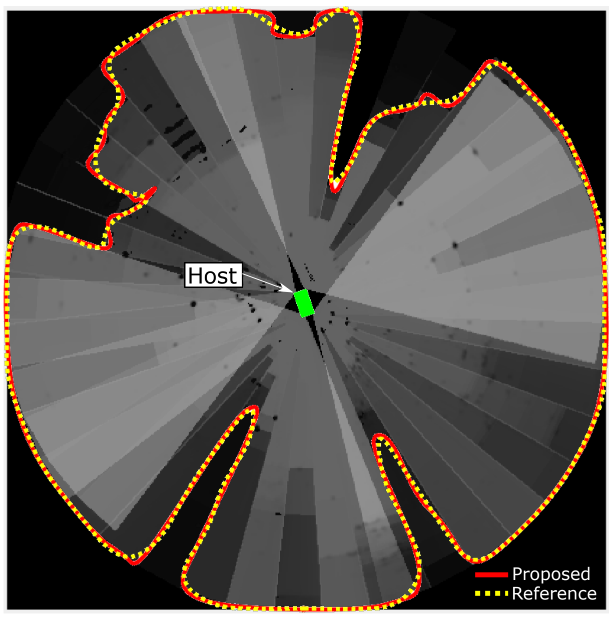
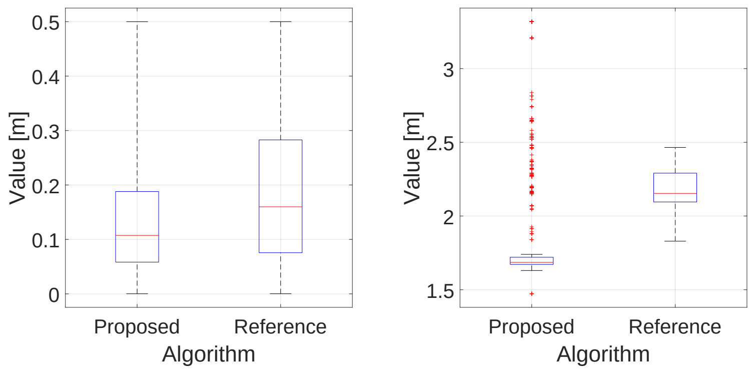


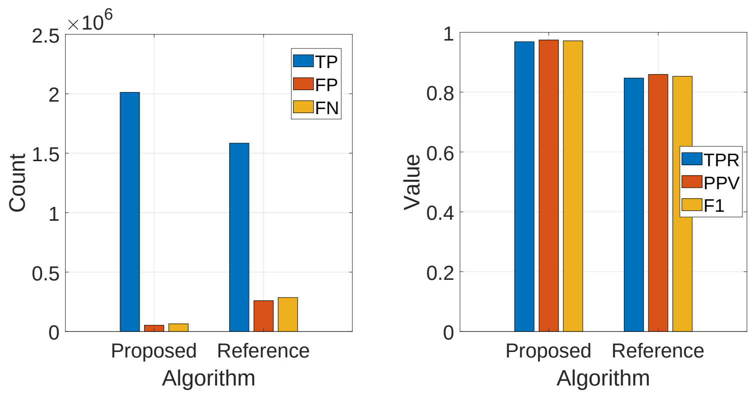
Figure 5.48. An visualization of one iteration from the real scenario.

The overall results show the advantage of the proposed algorithm over the reference one. The approximation errors are lower for the proposed algorithm (Figure 5.49). The same happens with the detection errors (Figure 5.50). The proposed algorithm is about two times faster than the reference (Figure 5.51).



(a) The closest spline point. (b) The Hausdorff distance.

Figure 5.49. Approximation errors aggregated from real scenario.



(a) Detection error. (b) Detection error rates.

Figure 5.50. Detection errors and rates aggregated from real scenario.

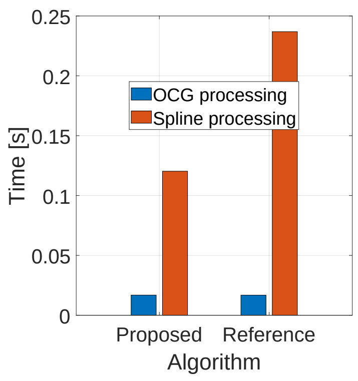
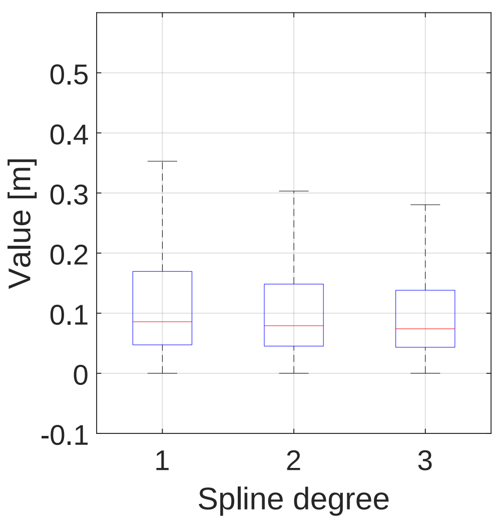


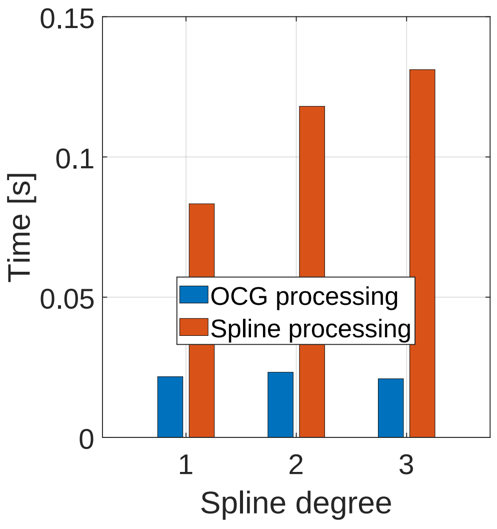
Figure 5.51. Algorithms' execution time aggregated from real scenario.

5.4.4 Spline degree influence

The one of the basic spline parameters is its degree. In this test, the influence of the degree of the spline on the approximation quality is verified. As expected, the higher degree of spline provides a better approximation quality, while at the same time increasing the execution time of an algorithm (Figure 5.52). The median values in the box plots are 0.086 (degree 1), 0.079 (degree 2), and 0.074 (degree 3). The execution time means are, respectively, 0.0833, 0.118 and 0.131 seconds.



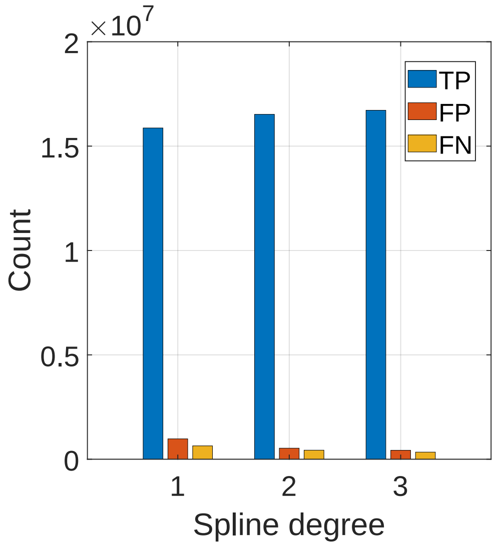
(a) Approximation error



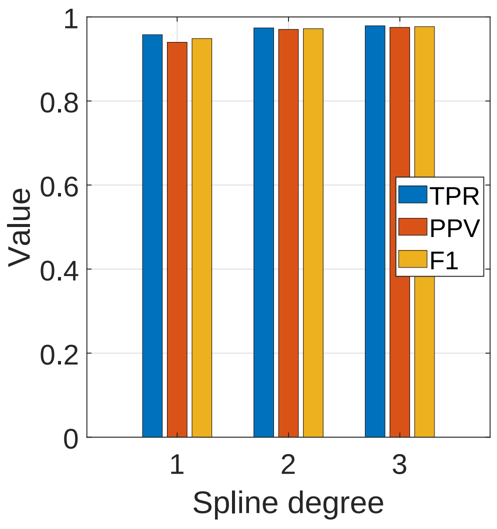
(b) Time

Figure 5.52. The results of the spline degree influence verification — basic metrics.

The same situation may be observed for the results of detection error rates. The TP count is higher for higher degree, whereas the FP and FN counts are lower (Figure 5.53).



(a) Detection error



(b) Detection error rates

Figure 5.53. The results of the spline degree influence verification — detection metrics.

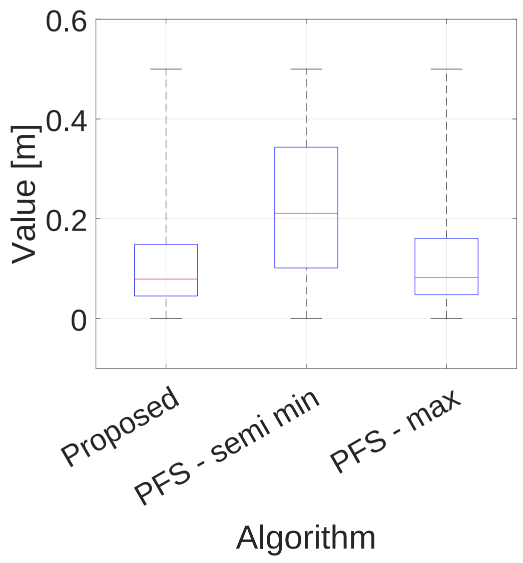
5.4.5 Impact of number of control points.

The number of control points has a direct influence on the quality of the approximation. In general, it can be expected that the higher number of control points improves the quality of the approximation. For testing purposes, two different simulations are performed to check how the reference algorithm behaves with a different fixed number of control points. The first test assumes that the number of control points is set to the maximum number that occurs in the proposed algorithm throughout the scenario. The other approach takes the so-called "semi-minimum" number of control points. Its value is equal to the rounded 35% of the value taken in the main comparison (Section 5.4.1). Table 5.5 shows the fixed number of control points used in the reference algorithm.

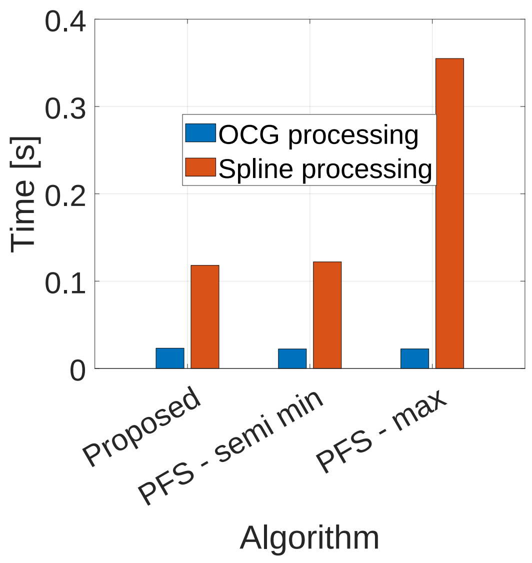
Table 5.5. The number of control points used in the PFS simulations.

Category	ID	Basic	Maximum	Semi-minimum
City	1	117	157	41
	2	21	29	7
	3	64	75	22
	4	36	49	13
	5	38	76	13
Highway	1	17	22	6
	2	19	32	7
	3	23	42	8
	4	31	48	11
	5	22	44	8
Mix	1	50	145	18
	2	55	127	19
Parking	1	33	48	12
	2	125	144	44
	3	70	116	25
	4	28	35	10
	5	54	98	19
Shape	1	60	70	21
	2	50	58	18
	3	38	41	13
	4	110	126	39
	5	34	49	12
Average		49.77	74.14	17.55

Compared to the proposed algorithm, the semi-minimum version of the reference algorithm is approximately 2.6 times worse (the median value is 63% greater) in approximation quality with similar execution time. The maximum version of the reference algorithm has only 5% worse approximation quality with about 3 times longer execution time. It should be noted that the reference algorithm in its best (the maximum number of control points) is worse than the proposed algorithm (Figure 5.54). In terms of detection error rates (Figure 5.55), the proposed algorithm achieved the best results.

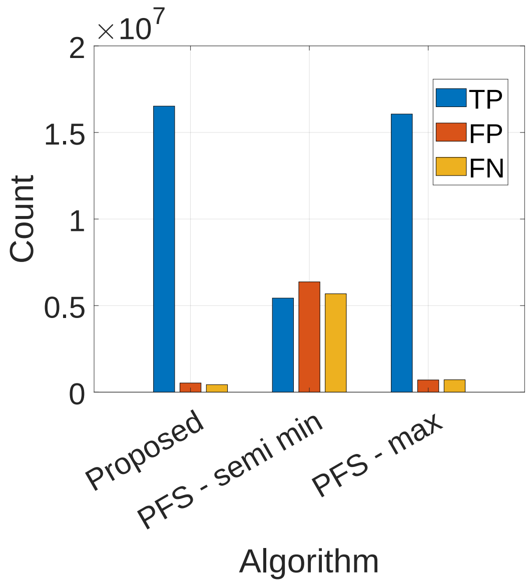


(a) Approximation error

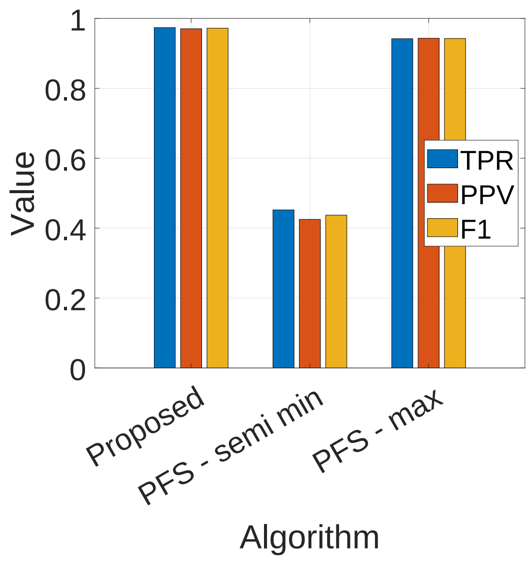


(b) Time

Figure 5.54. Impact of the number of control points on basic metrics.



(a) Detection error



(b) Detection error rates

Figure 5.55. Impact of the number of control points on detection error metrics.

5.4.6 Measurement shuffling test

This simple test has been performed to prove that the proposed algorithm can handle unsorted measurement points compared to the reference algorithm, which cannot do that. Once the measurement points are extracted, the shuffling is performed. As a result, the unsorted set of measurement points is provided to the spline estimation part. Figure 5.56 shows the behavior of both algorithms. The proposed algorithms make the shape approximation as expected, while the result from the reference algorithm (PFS) is a total mess.

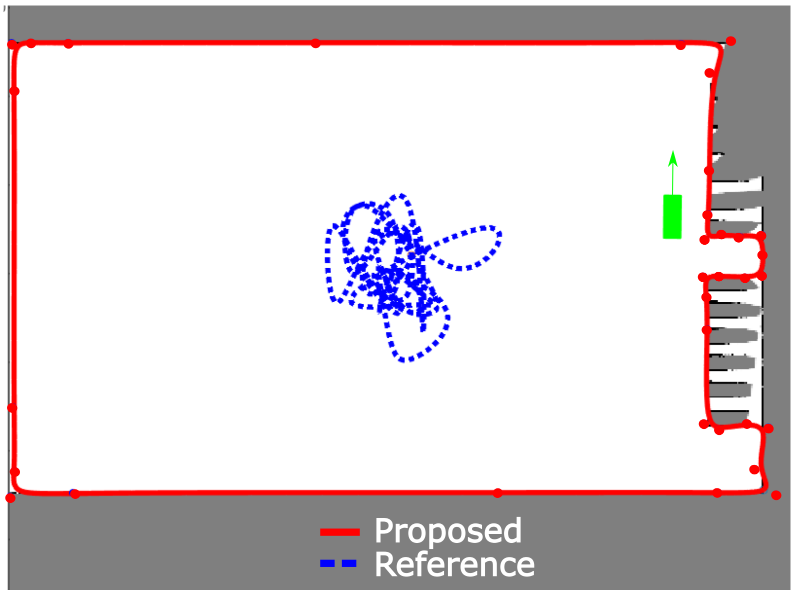


Figure 5.56. Shape estimation results with unsorted measurement points.

6 Conclusion and future work

6.1 Summary

In this thesis, a new free space boundary tracking algorithm is introduced. From a variety of mathematical models, the parametric B-spline curve is chosen as a model of tracked boundary. The decision was made based on a comparative analysis of the described models (Section 3.3). The B-spline was chosen mainly because of its property (ability) to refine local approximation when needed which does not affect other parts of the curve.

The proposed algorithm consists of two main stages: the occupancy grid processing and the spline processing. All innovations are introduced in the spline processing stage only. Several approaches for the measurement processing are proposed, including measurement extraction and downselection, measurement association, including searching for a spline point corresponding to the measurement point. The proposed algorithm is also enhanced by the control point adjustment methodology. It focuses on dynamic control point adjustment, i.e., adding or removing a control point. The method is based on the local shape complexity analysis, which employs local shape complexity indicators introduced by the Author. Additionally, to solve the local minima problem, an approximation error indicator that is introduced.

In the final part of this doctoral dissertation, tests and verification of the proposed algorithm are performed. It is done by comparison of the proposed algorithm with the chosen reference algorithm (here, the PFS). The comparison is made mostly on artificial data by involving metrics such as the Hausdorff distance, and two approximation error metrics proposed by the Author i.e. (*closest spline point* and *perpendicular spline point*) and safety looking metric called *detection gaps classifier*.

The comparison analysis proves that the proposed algorithm is better in terms of approximation quality (in all metrics used) and execution time. Moreover, some additional tests were performed that evaluate impact of spline degree, impact of number control point, and the impact of measurement shuffling on both algorithms. In all those analyzes the proposed algorithm outperforms the reference one.

6.2 Further improvements

Perfect solutions do not exist in the real word. The same applies to the proposed algorithm. There are still several areas where the proposed algorithm can be improved.

Control point merging

Mentioned in Section 4.4.6 operations on control points consist of only two operations: addition and removal. It may be beneficial to extend operations by merging two neighboring control points.

Raw sensor data input

The current implementation uses the occupancy grid as input. The way in which the proposed algorithm was written was as general as possible to handle an input from various sensor types and to eliminate the occupancy grid processing stage in the future. There is still work to be done to achieve 100% separation from the occupancy grid.

Traversability determination

Nowadays, the autonomous vehicle needs to know not only the boundary of the free space, but also the traversability classification (i.e. whether the boundary is under, over, or non-drivable). Examples are road elements like speed bumpers (overdriveble) or bridges (underdravible). The idea is to handle 3-dimensional input and add a third dimension information to the mathematical model.

Maximum number of control points

The autonomous vehicle has limits set on the software due to hardware resource limits (memory and computing power). Thus, an unlimited number of control points is not an acceptable idea. The maximum number of control points should be set. As a result, additional logic is required that makes the decision on what to do in case the maximum number of control points is reached.

Host distance dependent measurement noise

It is known that the algorithm should reflect the closest surrounding environment better than farther in terms of approximation quality. By relating the measurement position noise of the point to the distance from the host, one can expect to have better approximation of the nearest environment.

Optimal control point placement when adding

The way how the proposed algorithm place a new control point when adding is not an optimal one. Based on [Yang et al., 2004], it should be possible to find better way of control point placement that minimize spline deformation.

A Appendix

A.1 Parametric curves

A parametric curve is a set of points which position is defined by functions of the same parameter. In case of the planar (2D) curve we can say that the coordinates (x, y) of the points creating the curve **C** are defined by two functions of a parameter s:

$$\mathbf{C}(s) = (x, y) \in \mathbb{R}^2 \tag{A.1.1}$$

$$\begin{aligned} x &= f(s) \\ y &= g(s) \end{aligned} \tag{A.1.2}$$

s – free parameter

A.1.1 B-Spline

The B-spline is a particular representation of a spline. The B-spline can be described as follows: for a given knots vector **t**, a B-spline curve $b(s)$ of the n-th degree is defined as a linear combination of basis functions $B_{i,n}(s)$. i-th basis function is weighted by the corresponding control point p_i:

$$b(s) = \sum_{i=1} p_i B_{i,n}(s) \tag{A.1.3}$$

p_i – i-th control point

A basis function of the n-th degree is defined using the recursive Cox de Boor formula [de Boor, 1971]:

$$B_{i,n}(s) = \frac{s - t_i}{t_{i+n} - t_i} B_{i,n-1}(s) + \frac{t_{i+n+1} - s}{t_{i+n+1} - t_{i+1}} B_{i+1,n-1}(s) \tag{A.1.4}$$

$$B_{i,0}(s) = \begin{cases} 1, & if\ t_i \le s < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{A.1.5}$$

A knot vector **t** is defined as set of non-decreasing values in s-domain where the pieces of polynomial meet each other.

Figure A.1 illustrates an example of the b-spline with its basis functions and Figure A.2 visualizes a 2 dimensional B-spline.

The most important properties of the basis functions are the following:

- **Local support**
  Basis function is non-zero only within its natural domain called its support interval.

$$B_{i,n}(s) = 0 \qquad s \notin (t_i, t_{i+n+1}) \tag{A.1.6}$$

- **Boundary values**

$$B_{i,n}(t_i) = B_{i,n}(t_{i+n+1}) = 0 \tag{A.1.7}$$

- **Continuity**
  For a given degree n basis function belongs to the highest possible continuity class for a piecewise polynomial function

$$B_{i,n}(t_i) \in \mathcal{C}^{n-1}\left((t_i, t_{i+n+1}]\right) \tag{A.1.8}$$

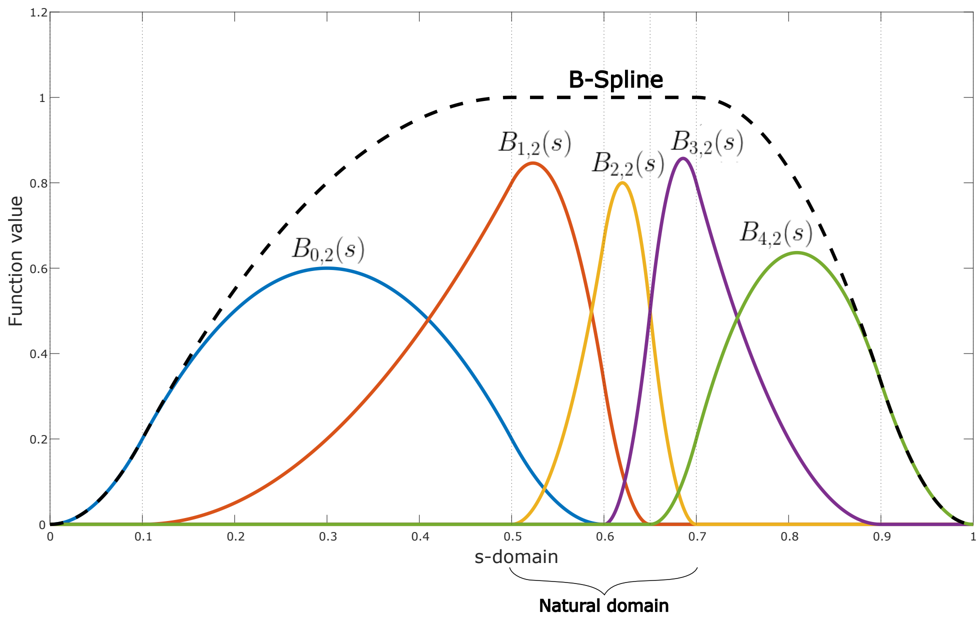


Figure A.1. An example of 1-dimensional B-spline of degree 2 with its basis functions. Control points vector: $[1, 1, 1, 1, 1]$, knots vector: $[0, 0.1, 0.5, 0.6, 0.65, 0.7, 0.9, 1]$

- **Derivatives**
  The derivative of a basis function of degree n is a linear combination of basis functions of degree $n-1$:
  $$\frac{dB_{i,n}(s)}{ds} = n\left(\frac{B_{i,n-1}(s)}{t_{i+n}-t_i} - \frac{B_{i+1,n-1}(s)}{t_{i+n+1}-t_{i+1}}\right) \tag{A.1.9}$$

Similar properties, which are presented above, may also be defined for a B-spline:

- **Domain definition**
  For a B-spline we can define two domains.
  The *full domain* is the interval
  $$[t_1, t_k] \tag{A.1.10}$$
  The *natural domain* is the interval
  $$[t_{n+1}, t_{k-n}] \tag{A.1.11}$$
  Within which, the B-spline is a combination of $n+1$ non-zero basis functions.

- **Limited influence region of control points**
  Each control point affects only a limited region of the B-spline. In other words, given control point p_i affects only the following interval of the B-spline:
  $$[t_i, t_{i+n+1}] \tag{A.1.12}$$

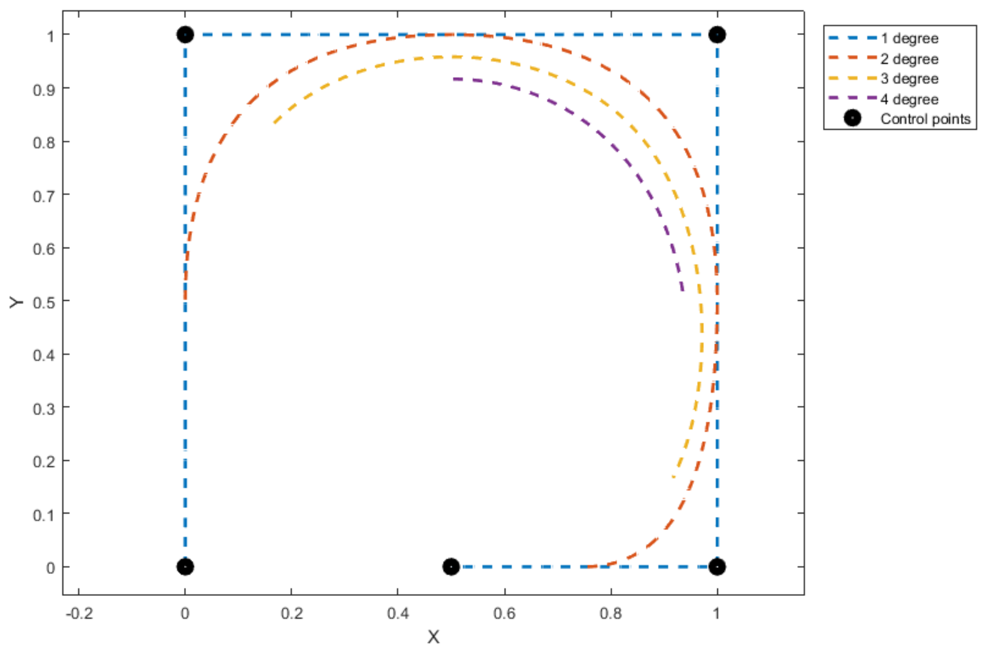


Figure A.2. An example of 2-dimensional B-spline within its natural domain.

A.2 Artificial scenario details

This section contains extended information on the artificial scenarios used for the performance evaluation of the proposed algorithm. Details of the scenarios are given in Table A.1. The following figures illustrate those scenarios:

- Highway — Figure A.4
- City — Figure A.5
- Parking — Figure A.6
- Shapes — Figure A.7
- Mix — Figure A.8

Table A.1. Each of the tested scenarios in detail.

Category	ID	Total iterations	Total reference points
City	1	480	1458511
	2	171	261357
	3	252	780459
	4	252	545169
	5	416	895774
Highway	1	515	778874
	2	554	832117
	3	521	888880
	4	502	936072
	5	735	1200050
Mix	1	822	1853212
	2	736	1735841
Parking	1	160	101920
	2	340	909769
	3	267	483522
	4	186	209014
	5	400	520061
Shape	1	198	761904
	2	198	607266
	3	198	312048
	4	198	891792
	5	198	523116

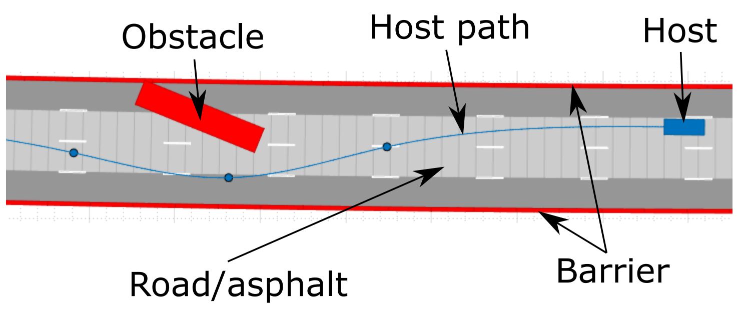
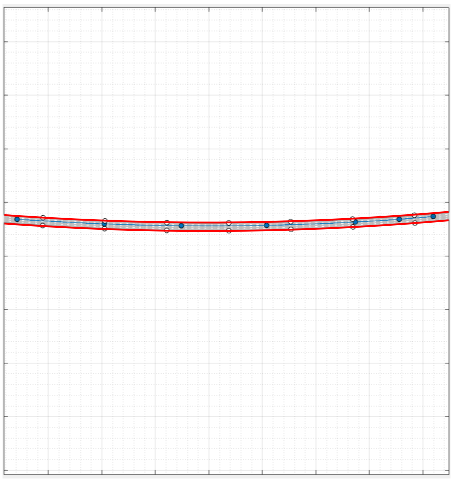
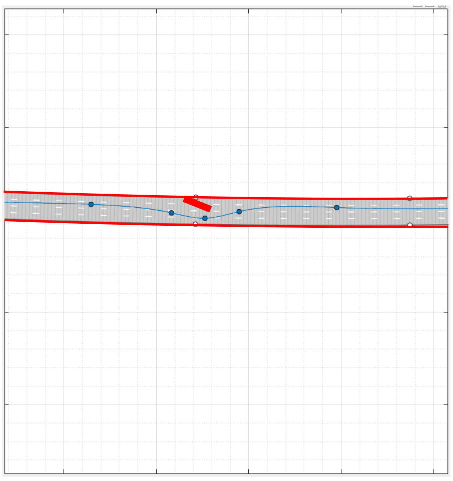
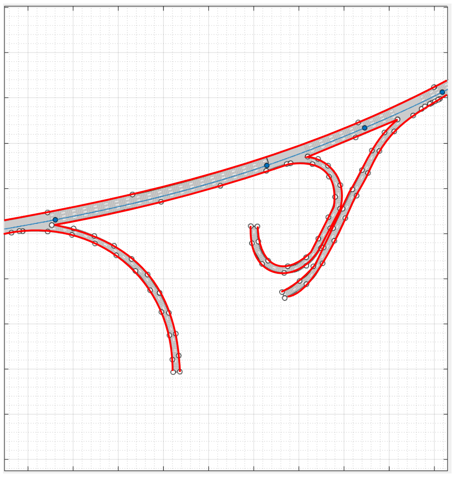


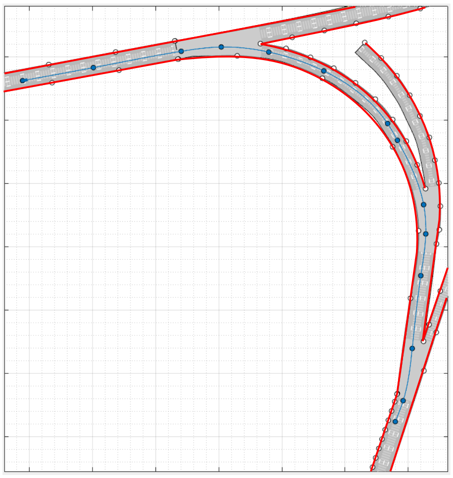
Figure A.3. Legend of elements used in visualization of each scenario.

A.2.1 Highway


(a) Scenario 1


(b) Scenario 2


(c) Scenario 3


(d) Scenario 4

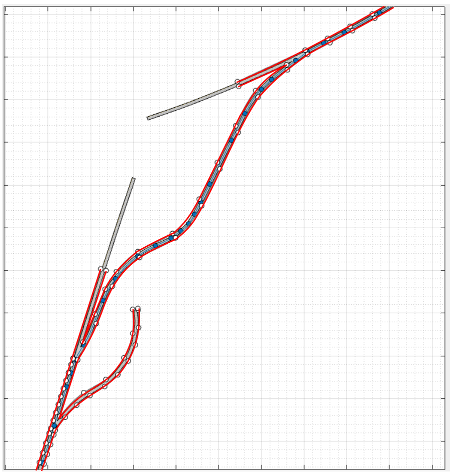
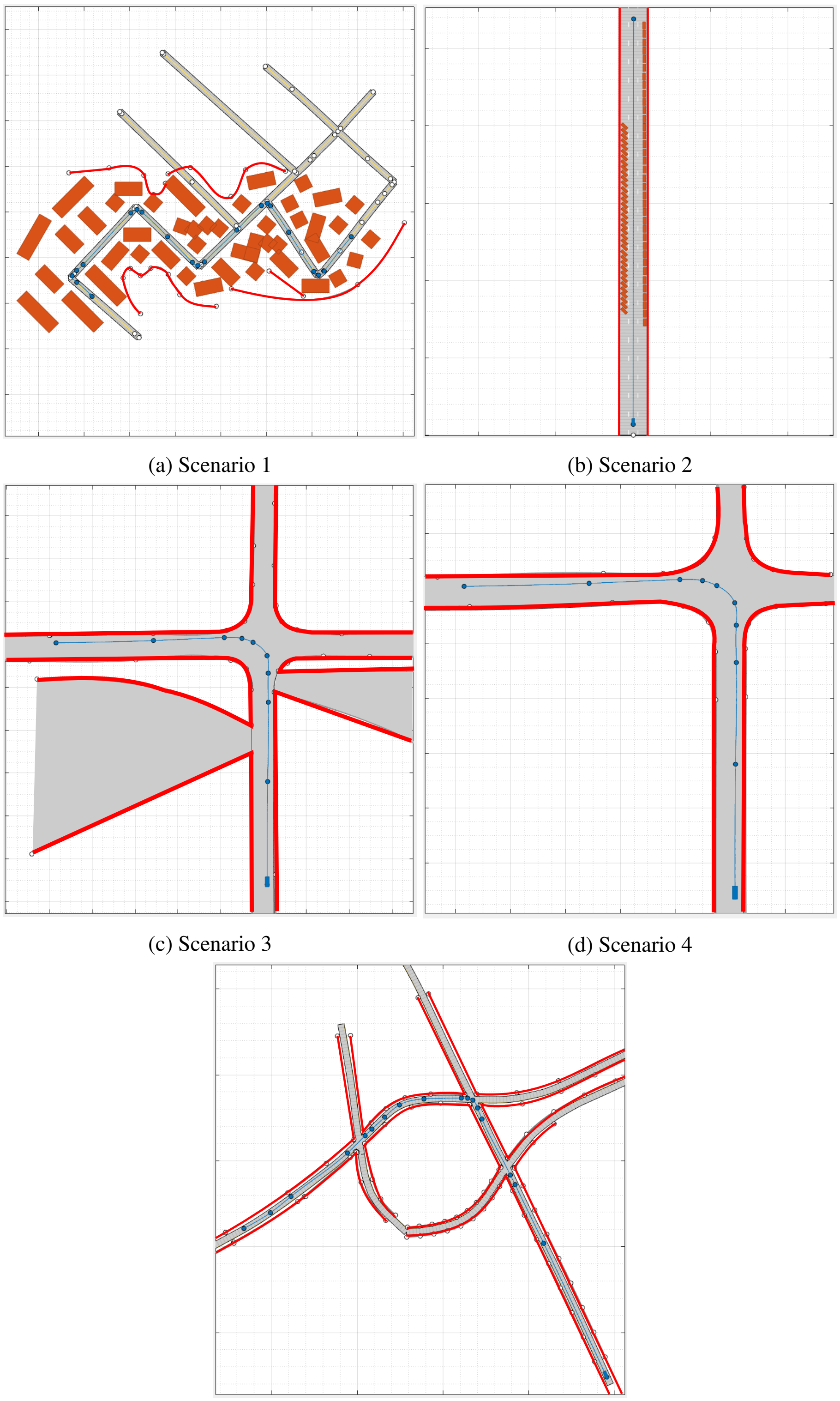

(e) Scenario 5

Figure A.4. The visualization of highway scenarios.

A.2.2 City



(a) Scenario 1

(b) Scenario 2

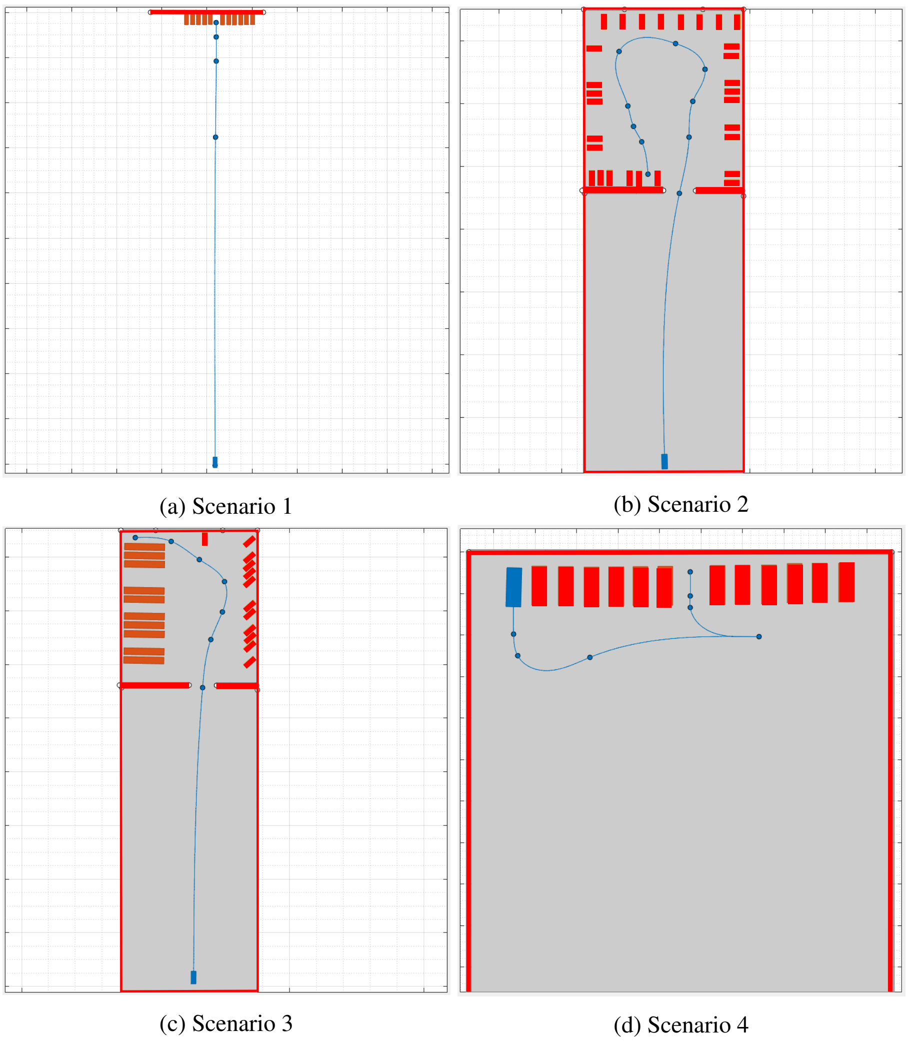
(c) Scenario 3

(d) Scenario 4

(e) Scenario 5

Figure A.5. The visualization of city scenarios.

A.2.3 Parking


(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

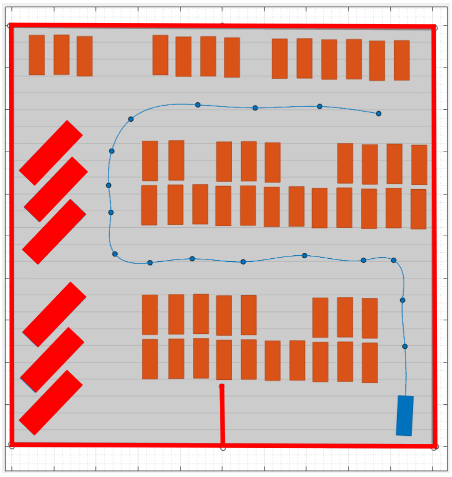
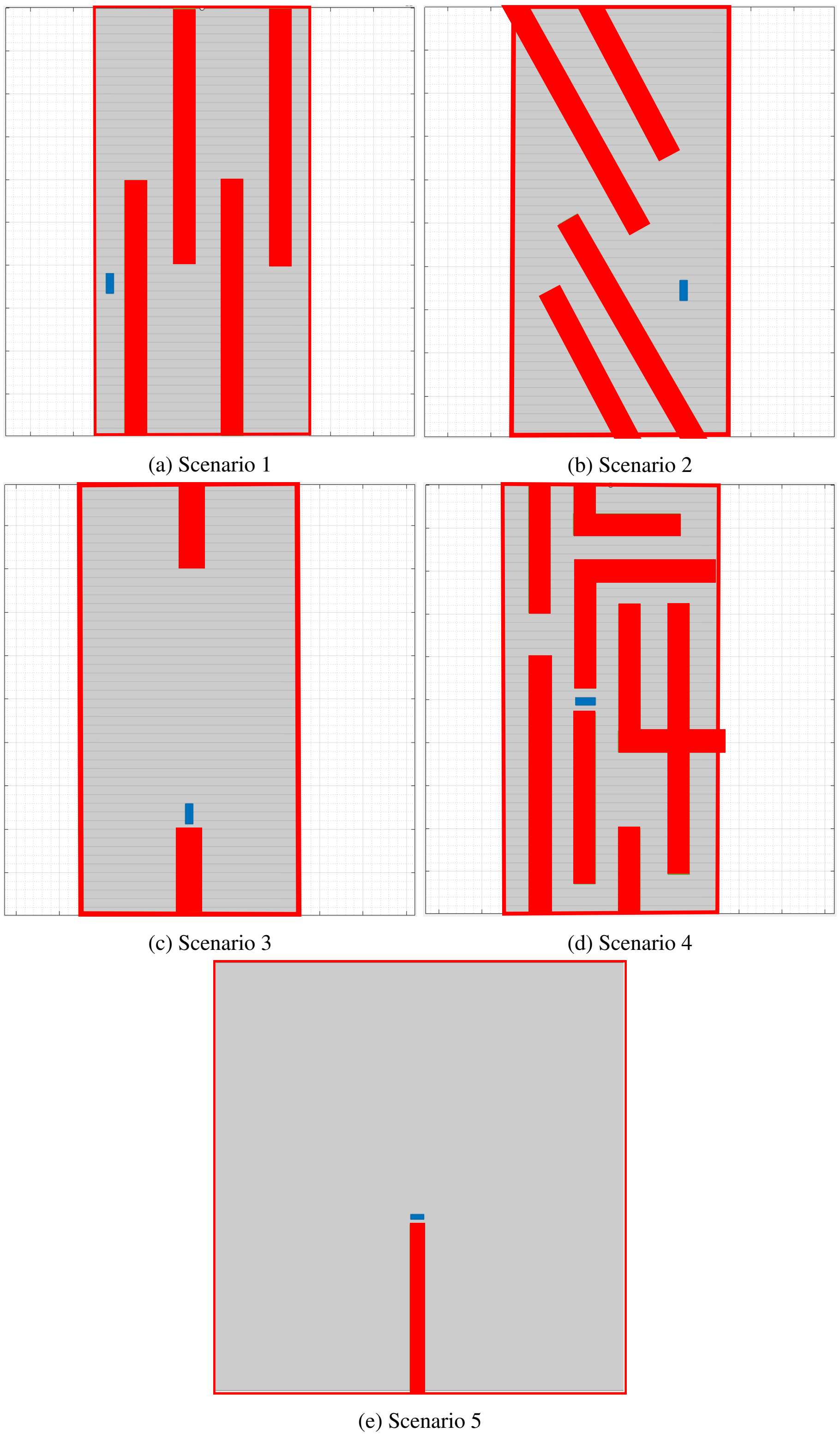

(e) Scenario 5

Figure A.6. The visualization of parking scenarios.

A.2.4 Shapes



(a) Scenario 1

(b) Scenario 2

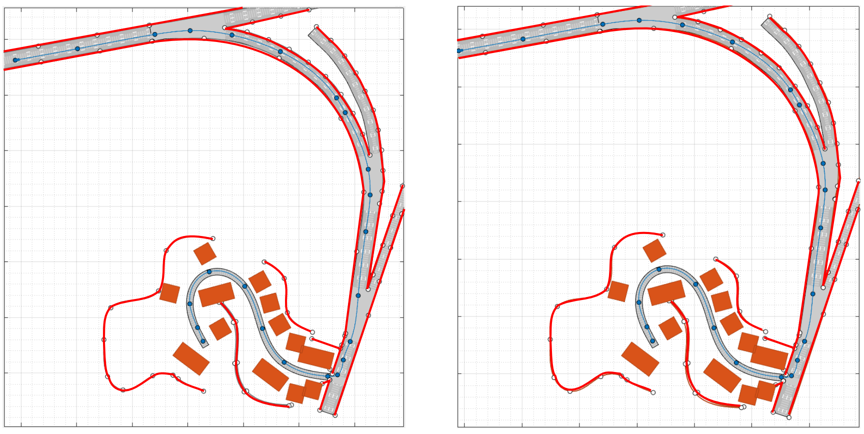
(c) Scenario 3

(d) Scenario 4

(e) Scenario 5

Figure A.7. The visualization of shapes scenarios.

A.2.5 Mix



(a) Scenario 1

(b) Scenario 2

Figure A.8. The visualization of mix scenarios.

A.3 Hardware and software

During the PhD work, all scripts, algorithms, and tools were written in MATLAB R2021b, including the following.

- the proposed algorithm
- the reference algorithm
- scripts used for performance evaluation
- occupancy grid generators
- GUI

The simulations were performed on a computer equipped with:

- 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz
- 64 GB

Bibliography

A. AbuBaker, R. Qahwaji, S. Ipson, and M. Saleh. One scan connected component labeling technique. In *2007 IEEE International Conference on Signal Processing and Communications*, pages 1283–1286. IEEE, 2007.

T. R. Andriamahefa. *Integer Occupancy Grids: a probabilistic multi-sensor fusion framework for embedded perception*. PhD thesis, Université Grenoble Alpes, 2017.

N. Assimakis, M. Adam, and A. Douladiris. Information filter and kalman filter comparison: Selection of the faster filter. In *Information Engineering*, volume 2, pages 1–5, 2012.

AUTOSAR. Autosar, 2022. URL *www.autosar.org*. Accessed 2022-12-10.

M. Bersani, M. Vignati, S. Mentasti, S. Arrigoni, and F. Cheli. Vehicle state estimation based on kalman filters. In *2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, pages 1–6, 2019. doi: 10.23919/EETA.2019.8804527.

M. Bertozzi, A. Broggi, E. Cardarelli, R. I. Fedriga, L. Mazzei, and P. P. Porta. Viac expedition toward autonomous mobility [from the field]. *IEEE Robotics & Automation Magazine*, 18(3):120–124, 2011.

J. W. Betz. *Beidou System*, pages 252–265. 2016a. doi: 10.1002/9781119141167.ch11.

J. W. Betz. *Satellite-Based Augmentation Systems*, pages 201–211. 2016b. doi: 10.1002/9781119141167.ch8.

M. Buehler. The 2005 darpa grand challenge, 2007.

M. Buehler. The darpa urban challenge: Autonomous vehicles in city traffic, 2009.

C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3-4): 507–525, 2001.

J. Carvalho and R. Ventura. Comparative evaluation of occupancy grid mapping methods using sonar sensors. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 889–896. Springer, 2013.

S. Challa and D. Koks. Bayesian and dempster-shafer fusion. *Sadhana*, 29(2):145–174, 2004.

W. Chen, W. Wang, K. Wang, Z. Li, H. Li, and S. Liu. Lane departure warning systems and lane line detection methods based on image processing and semantic segmentation: A review. *Journal of traffic and transportation engineering (English edition)*, 7(6):748–774, 2020.

X. Chen, X. Wang, and J. Xuan. Tracking multiple moving objects using unscented kalman filtering techniques. *arXiv preprint arXiv:1802.01235*, 2018.

W. Chu, Y. Luo, Y. Dai, and K. Li. In–wheel motor electric vehicle state estimation by using unscented particle filter. *International Journal of Vehicle Design*, 67(2):115–136, 2015.

I. E. Commission. Iec 61508, 2022. URL *www.iec.ch*. Accessed 2022-12-10.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2022.

C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessière. Bayesian occupancy filtering for multitarget tracking: an automotive application. *The International Journal of Robotics Research*, 25(1): 19–30, 2006.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

I. J. Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

R. Danescu, S. Sobol, S. Nedevschi, and T. Graf. Stereovision-based side lane and guardrail detection. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1156–1161, 2006. doi: 10.1109/ ITSC.2006.1707378.

C. de Boor. Subroutine package for calculating with b-splines. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 1971.

A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. In *Classic works of the Dempster-Shafer theory of belief functions*, pages 57–72. Springer, 2008.

E. W. Dijkstra. A note on two problems in connexion with graphs:(numerische mathematik, 1 (1959), p 269-271). 1959.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

L. Earnest. Stanford cart, December 2012. URL *https://web.stanford.edu/~learnest/sail/oldcart.html*.

A. Elfes. Occupancy grids: a probabilistic framework for robot perception and navigation. 1989.

C. Engelking. The 'driverless' car era began more than 90 years ago, December 2017. URL *https://www.discovermagazine.com/technology/the-driverless-car-era-began-more-than-90-years-ago*.

T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184, 1983.

W. Gex and N. Campbell. Local free space mapping and path guidance. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 424–431. IEEE, 1987a.

W. Gex and N. Campbell. Local free space mapping and path guidance. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 424–431. IEEE, 1987b.

M. S. Grewal, A. P. Andrews, and C. G. Bartone. *Differential GNSS*, pages 293–330. 2020. doi: 10.1002/9781119547860.ch8.

A. Guerra, F. Guidi, J. Dall'Ara, and D. Dardari. Occupancy grid mapping for personal radar applications. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pages 766–770, 2018. doi: 10.1109/SSP.2018.8450813.

J. Guo, Y. Wang, X. Yin, P. Liu, Z. Hou, and D. Zhao. Study on the control algorithm of automatic emergency braking system (aebs) for commercial vehicle based on identification of driving condition. *Machines*, 10(10):895, 2022.

J.-S. Gutmann, M. Fukuchi, and M. Fujita. A floor and obstacle height map for 3d navigation of a humanoid robot. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1066–1071, 2005. doi: 10.1109/ROBOT.2005.1570257.

D. Harabor and A. Grastien. Online graph pruning for pathfinding on grid maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1114–1119, 2011.

R. Haralick. Some neighborhood operators. In *Real-Time Parallel Computing*, pages 11–35. Springer, 1981.

P. E. Hart. An asymptotic analysis of the nearest-neighbor decision rule. Technical report, STANFORD UNIV CA STANFORD ELECTRONICS LABS, 1966.

P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

L. He, Y. Chao, K. Suzuki, and K. Wu. Fast connected-component labeling. *Pattern recognition*, 42(9): 1977–1987, 2009.

J. Hooper. From darpa grand challenge 2004darpa's debacle in the desert, June 2004a. URL *https://www.popsci.com/scitech/article/2004-06/darpa-grand-challenge-2004darpas-debacle-desert/*.

J. Hooper. Leonardo's car brought to life, April 2004b. URL *https://www.theguardian.com/world/2004/apr/24/italy.arts*.

C. Ilas. Electronic sensing technologies for autonomous ground vehicles: A review. In *2013 8TH INTERNATIONAL SYMPOSIUM ON ADVANCED TOPICS IN ELECTRICAL ENGINEERING (ATEE)*, pages 1–6, 2013. doi: 10.1109/ATEE.2013.6563528.

S. International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, April 2021a.

S. International. Sae j3016tm levels of driving automation, April 2021b. URL *https://www.sae.org/binaries/content/assets/cm/content/blog/sae-j3016-visual-chart_5.3.21.pdf*.

International Organization for Standardization. Iso 26262-1:2018 road vehicles — functional safety, 2018. URL *https://www.iso.org/standard/68383.html*. Accessed: 2023.04.07.

International Organization for Standardization. Iso 21448:2022 road vehicles — safety of the intended functionality, 2022. URL *https://www.iso.org/standard/77490.html*. Accessed: 2022.10.23.

T. J, T. E, N. R, and A. M. *Ultrasonic Fluid Quantity Measurement in Dynamic Vehicular Applications*, chapter 2. Springer, 2013. Chapter 2: Ultrasonic Sensing Technology.

C. T. Johnston and D. G. Bailey. Fpga implementation of a single pass connected components algorithm. In *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*, pages 228–231. IEEE, 2008.

R. E. Kalman. A new approach to linear filtering and prediction problems. 1960.

S. Karutin, N. Testoedov, A. Tyulin, and A. Bolkunov. *GLONASS*, pages 87–103. 2021. doi: 10.1002/9781119458449.ch4.

J. Kim, H.-R. Choi, J. Kwon, and T. Kim. Recognition of face orientation by divided hausdorff distance. In *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, pages 564–567, 2015. doi: 10.1109/TSP.2015.7296326.

D. Kok and J. Fu. Signal processing for automotive radar. In *IEEE International Radar Conference, 2005.*, pages 842–846, 2005. doi: 10.1109/RADAR.2005.1435944.

P. D. Konstantinova, A. Udvarev, and T. Semerdjiev. A study of a target tracking algorithm using global nearest neighbor approach. In *Compsystech*, volume 3, pages 290–295, 2003.

F. Kraus, N. Scheiner, W. Ritter, and K. Dietmayer. Using machine learning to detect ghost images in automotive radar. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, 2020. doi: 10.1109/ITSC45102.2020.9294631.

F. Kröger. Automated driving in its social, historical and cultural contexts, 2016.

R. B. Langley. Rtk gps. *Gps World*, 9(9):70–76, 1998.

M. Li, Z. Feng, M. Stolz, M. Kunert, R. Henze, and F. Küçükay. High resolution radar-based occupancy grid mapping and free space detection. pages 70–81, 03 2018. doi: 10.5220/0006667300700081.

S. Lim, S. Lee, and S.-C. Kim. Clustering of detected targets using dbscan in automotive radar systems. In *2018 19th International Radar Symposium (IRS)*, pages 1–7, 2018. doi: 10.23919/IRS.2018.8448228.

G. Liu, L. Wang, and S. Zou. A radar-based blind spot detection and warning system for driver assistance. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 2204–2208, 2017. doi: 10.1109/IAEAC.2017.8054409.

J. Liu and W. Li. Aggressive heuristic search for sub-optimal solution on path planning. In *2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 16–20. IEEE, 2018.

Á. Llamazares, E. J. Molinos, and M. Ocaña. Detection and tracking of moving obstacles (datmo): a review. *Robotica*, 38(5):761–774, 2020.

K. Małecki, M. Kamiński, and J. Wąs. A multi-cell cellular automata model of traffic flow with emergency vehicles: Effect of a corridor of life and drivers' behaviour. *Journal of Computational Science*, 61:101628, 2022.

T. MathWorks. Driving scenario designer, 2023. URL *https://www.mathworks.com/help/driving/ref/drivingscenariodesigner-app.html*. Accessed: 2023.2.22.

G. Micula and S. Micula. *Handbook of splines*, volume 462. Springer Science & Business Media, 2012.

S. Mirjalili and S. Mirjalili. Genetic algorithm. *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pages 43–55, 2019.

MISRA. Misra, 2022. URL *www.misra.org.uk*. Accessed 2022-12-10.

H. Moravec. Robot spatial perceptionby stereoscopic vision and 3d evidence grids. *Perception*, 483:484, 1996.

H. P. Moravec. Sensor fusion in certainty grids for mobile robots. In A. Casals, editor, *Sensor Devices and Systems for Robotics*, pages 253–276, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg. ISBN 978-3-642-74567-6.

K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12):2221–2229, 1992.

T. Nam, J. Shim, and Y. Cho. A 2.5d map-based mobile robot localization via cooperation of aerial and ground robots. *Sensors*, 17(12):2730, Nov 2017. ISSN 1424-8220. doi: 10.3390/s17122730. URL *http://dx.doi.org/10.3390/s17122730*.

N. L. Narappanawar, B. M. Rao, T. Srikanth, and M. Joshi. Vector algebra based tracing of external and internal boundary of an object in binary images. *Journal of Advances in Engineering Science*, 3(1): 57–70, 2010.

Navlab. Navlab: The carnegie mellon university navigation laboratory, 2022. URL *https://www.cs.cmu.edu/afs/cs/project/alv/www/index.html*. Accessed: 2022.12.10.

S. Nedevschi, R. Danescu, D. Frentiu, T. Marita, F. Oniga, C. Pocol, R. Schmidt, and T. Graf. High accuracy stereo vision system for far distance obstacle detection. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 292–297. IEEE, 2004a.

S. Nedevschi, R. Schmidt, T. Graf, R. Danescu, D. Frentiu, T. Marita, F. Oniga, and C. Pocol. 3d lane detection system based on stereovision. In *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*, pages 161–166. IEEE, 2004b.

J. v. Neumann. Theory of self-reproducing automata. *Mathematics of Computation*, 21:745, 1966.

R. Oliveira, M. Cirillo, J. M. artensson, and B. Wahlberg. Combining lattice-based planning and path optimization in autonomous heavy duty vehicle applications. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 2090–2097, 2018. doi: 10.1109/IVS.2018.8500616.

OpenStreetMap. Open street map, 2023. URL *https://www.openstreetmap.org/about*. Accessed: 2023.2.22.

R. K. Ozguner U, Acarman T. *Autonomous Ground Vehicles*. Artech House, 2011.

S. Parsons. Probabilistic robotics by sebastian thrun, wolfram burgard and dieter fox, mit press, 647 pp., isbn 0-262-20162-3. *The Knowledge Engineering Review*, 21(3):287–289, 2006.

P. Pavitha, K. B. Rekha, and S. Safinaz. Perception system in autonomous vehicle: A study on contemporary and forthcoming technologies for object detection in autonomous vehicles. In *2021 International Conference on Forensics, Analytics, Big Data, Security (FABS)*, volume 1, pages 1–6, 2021. doi: 10.1109/FABS52071.2021.9702569.

Y. Peng, D. Qu, Y. Zhong, S. Xie, J. Luo, and J. Gu. The obstacle detection and obstacle avoidance algorithm based on 2-d lidar. In *2015 IEEE International Conference on Information and Automation*, pages 1648–1653, 2015. doi: 10.1109/ICInfA.2015.7279550.

pete. Self-drive cars and you: A history longer than you think, August 2004. URL *https://velocetoday.com/self-drive-cars-and-you-a-history-longer-than-you-think/*.

P. Peterlin. Morphological operations: An overview, 1996. URL *https://www.inf.u-szeged.hu/~ssip/1996/morpho/morphology.html*. Accessed 2021-8-10.

J. Porebski. *Occupancy grid environmental modeling for automotive applications*. PhD thesis, AGH, 2022.

C. Rablau. Lidar–a new (self-driving) vehicle for introducing optics to broader engineering and non-engineering audiences. In *Education and Training in Optics and Photonics*, page 11143_138. Optica Publishing Group, 2019.

H. Rahman. *Fundamental Principles of Radar*. CRC Press, 2019.

R. Rajamani. Adaptive cruise control. *Encyclopedia of Systems and Control*, pages 20–26, 2021.

P. Raju. Fundamentals of gps. In *Satellite Remote Sensing and GIS Applications in Agricultural Meteorology*, 2003.

W. P. Rathnayake. Google maps based travel planning & analyzing system (tpas). In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–5, 2018. doi: 10.1109/ICCTCT.2018.8550996.

D. Reid. An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*, 24(6): 843–854, 1979.

M. I. Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, 43(46):3736–3741, 2004.

J. Rodríguez, J. Hahn, M. M. Bautista, and E. Chatre. *GALILEO*, pages 105–142. 2021. doi: 10.1002/9781119458449.ch5.

M. Roelofsen, J. Bie, L. Jin, and B. van Arem. Assessment of safety levels and an innovative design for the lane change assistant. In *2010 IEEE Intelligent Vehicles Symposium*, pages 83–88, 2010. doi: 10.1109/IVS.2010.5548095.

A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM (JACM)*, 13(4):471–494, 1966.

Y. Roth-Tabak and R. Jain. Building an environment model using depth information. *Computer*, 22(6): 85–90, 1989.

A. Sadat, M. Ren, A. Pokrovsky, Y.-C. Lin, E. Yumer, and R. Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3949–3956, 2019. doi: 10.1109/IROS40897.2019.8967615.

M. Saval-Calvo, L. Medina-Valdés, J. M. Castillo-Secilla, S. Cuenca-Asensi, A. Martínez-Álvarez, and J. Villagrá. A review of the bayesian occupancy filter. *Sensors*, 17(2):344, 2017.

H. H. Schmid. *Three-dimensional triangulation with satellites*, volume 7. National Oceanic and Atmospheric Administration, 1974.

M. Schreier, V. Willert, and J. Adamy. Compact representation of dynamic driving environments for adas by parametric free space and dynamic object maps. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):367–384, 2016. doi: 10.1109/TITS.2015.2472965.

C. Sentouh, A.-T. Nguyen, M. A. Benloucif, and J.-C. Popieul. Driver-automation cooperation oriented approach for shared control of lane keeping assist systems. *IEEE Transactions on Control Systems Technology*, 27(5):1962–1978, 2018.

L. Shapiro and G. Stockman. Connected components labeling. In *Prentice Hall, editor, Computer Vision*, pages 69–73. Prentice Hall, 2002.

F. Smarandache and J. Dezert. Information fusion based on new proportional conflict redistribution rules. In *2005 7th international conference on information fusion*, volume 2, pages 8–pp. IEEE, 2005.

G. Spampinato, S. Curti, I. Guarneri, and A. Bruna. Optical flow based system for cross traffic alert. *International Journal of Mechanical and Mechatronics Engineering*, 12(7):705–710, 2018.

A. Stentz. Optimal and efficient path planning for partially known environments. In *Intelligent unmanned ground vehicles*, pages 203–220. Springer, 1997.

S. Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

M. Szlachetka, D. Borkowski, and J. Was. Stationary environment models for advanced driver assistance systems. In *2020 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 116–121, 2020. doi: 10.23919/SPA50552.2020.9241306.

M. Szlachetka, D. Borkowski, and J. Wąs. The downselection of measurements used for free space determination in adas. *Journal of Computational Science*, 63:101762, 2022.

A. N. Talbot. *The railway transition spiral*. Engineering News Publishing Company, 1901.

TISAX. Tisax, 2022. URL *https://portal.enx.com/en-us/tisax*. Accessed 2022-12-10.

J. Todd. No hands across america, 2022. URL *https://www.cs.cmu.edu/~tjochem/nhaa/nhaa_home_page.html*. Accessed: 2022.12.10.

M. T.R. and A. M. Obstacle detection and obstacle avoidance algorithm based on 2-d rplidar. In *2019 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–4, 2019. doi: 10.1109/ICCCI.2019.8821803.

R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2276–2282, 2006. doi: 10.1109/IROS.2006.282632.

C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics*, 25(8):425–466, 2008.

VDA QMC Working Group 13 / Automotive SIG. *Automotive SPICE Process Assessment*, 2017.

VisLab S.r.l., 2022. URL *https://vislab.it/viac/*. Accesssed: 2022.12.12.

J. Wąs and R. Lubaś. Towards realistic and effective agent-based models of crowd dynamics. *Neurocomputing*, 146:199–209, 2014.

M. Weber. Where to? a history of autonomous vehicles, May 2014. URL *https://computerhistory.org/blog/where-to-a-history-of-autonomous-vehicles/l*.

T. Weiherer, E. Bouzouraa, and U. Hofmann. A generic map based environment representation for driver assistance systems applied to detect convoy tracks. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 691–696. IEEE, 2012.

T. Weiherer, S. Bouzouraa, and U. Hofmann. An interval based representation of occupancy information for driver assistance systems. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 21–27, 2013. doi: 10.1109/ITSC.2013.6728205.

L. Wenzheng, L. Junjun, and Y. Shunli. An improved dijkstra's algorithm for shortest path planning on 2d grid maps. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 438–441, 2019. doi: 10.1109/ICEIEC.2019.8784487.

S. Wolfram et al. *A new kind of science*, volume 5. Wolfram media Champaign, 2002.

K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. volume 2, 01 2010.

H. Yang, W. Wang, and J. Sun. Control point adjustment for b-spline curve approximation. *Computer-Aided Design*, 36(7):639–652, 2004.

N. Z, A. G, W. B, and P. S. *TI Gives Sight to Vision-Enabled Automotive Technologies*. Texas Instruments, October 2013.

J. Zhu, Z. Wang, L. Zhang, and W. Zhang. State and parameter estimation based on a modified particle filter for an in-wheel-motor-drive electric vehicle. *Mechanism and Machine Theory*, 133:606–624, 2019.

List of Figures

List of Tables