Lehrstuhl für Steuerungs- und Regelungstechnik
Technische Universität München
Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss

Safety Assessment for Motion Planning in Uncertain and Dynamic Environments

Daniel Althoff

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. sc. techn. G. Kramer

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing./Univ. Tokio M. Buss
2. Univ.-Prof. G. Cheng, Ph.D.

Die Dissertation wurde am 24.09.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 12.12.2013 angenommen.

Abstract

The progress of robotic systems in the past decades provides robots with capabilities to operate in human populated environments. One of the major challenges on the way to obtain the objective of the robot co-worker is to ensure a safe and reliable operation of robots. Consequently, motion safety is becoming increasingly important in robotic research. This thesis investigates novel safety assessment and motion planning methods for robot navigation in dynamic and uncertain environments with contributions to three problems.

First, the problem of safety assessment of roadmaps in uncertain environments is addressed and a novel approach is presented which computes the safety by the policy with the smallest expected collision probability. This policy determines the optimal route in the roadmap depending on the available information of the environment and enables the robot to replan its route during execution. Compared to the common approach of determining the optimal route, the novel approach is guaranteed to result in a lower collision probability.

Second, novel algorithms for the problem of safety assessment beyond the planning horizon of trajectories are presented. Motion planning approaches for dynamic environments usually generate partial trajectories towards the goal since motion prediction is often not reliable for a long time period. The novel approaches are more efficient than previous methods. Moreover, this problem is also investigated in uncertain environments taking into account the uncertainties in the motion prediction of the surrounding objects of the robot.

Next, the problem of reliable and efficient navigation in uncertain populated environments is addressed that is nowadays still an open problem, especially if the density of moving objects is high. Due to the high density and the uncertain motion prediction of objects the robot may fail to find any admissible trajectory. This problem is addressed by presenting novel safety assessment concepts that consider the avoidance behavior of reactive objects such as humans. It allows a more reliable and less conservative assessment, especially in dynamic environments with a high density of objects.

Finally, the integration of the presented safety assessment approaches into motion planning algorithms is shown. An integration into optimal control approaches is presented that guarantees safety beyond the planning horizon. Based on the idea of the novel roadmap safety assessment approach a generic planner is presented that improves any solution to the motion planning problem by generating additional trajectories resulting in an optimized roadmap. This roadmap is guaranteed to have lower cost than the optimal trajectory. Furthermore, an interactive motion planner is presented considering the avoidance behavior of reactive objects. Thus, the robot and the surrounding objects are reciprocal avoiding each other instead of assuming that only the robot is avoiding the other objects. The effectiveness of all novel methods for safety assessment and motion planning are demonstrated by various simulations from the field of mobile robot navigation and autonomous driving.

Zusammenfassung

Der Fortschritt der letzten Jahrzehnte im Bereich der Robotik ermöglicht es Robotern in der direkten Umgebung von Menschen zu agieren. Um jedoch das Ziel der Mensch-Roboter-Koexistenz zu erreichen, muss ein für Menschen ungefährlicher Betrieb von Robotern gewährleistet werden. Deshalb spielt die Bewegungssicherheit von Robotern eine immer größere Rolle in der Forschung. Diese Dissertation präsentiert neuartige Methoden zur Sicherheitsbewertung und Bewegungsplanung für Roboter in dynamischen und unsicheren Umgebungen.

Zunächst wird die Sicherheit von Roadmaps in unsicheren und dynamischen Umgebungen untersucht und ein neuartiger Ansatz zur Sicherheitsbewertung anhand der Kollisionswahrscheinlichkeit der optimalen Strategie vorgestellt. Diese Strategie wählt die optimale Route der Roadmap abhängig vom aktuellen Zustand der Umgebung aus. Dadurch kann die Route des Roboters neu geplant werden und ermöglicht es dem Roboter auf Veränderungen in seiner Umgebung zu reagieren. Im Vergleich zu vorherigen Ansätzen, welche die Sicherheit anhand der optimalen Route der Roadmap bestimmen, kann so eine weniger konservative Bewertung garantiert werden.

Des Weiteren werden neue Ansätze für die Sicherheitsbewertung von unvollständigen Trajektorien jenseits ihres Planungshorizontes präsentiert. Viele Ansätze der Bewegungsplanung für dynamische Umgebungen generieren nur unvollständige Trajektorien zum Ziel, weil die Bewegungsprädiktion anderer Objekte nur für einen kurzen Zeithorizont verfügbar ist. Neue Methoden werden vorgestellt, die eine effizientere Berechnung als vorherige Methoden ermöglichen. Außerdem wurden neue Methoden für unsichere Umgebungen vorgestellt, die Unsicherheiten in der Prädiktion der umliegenden Objekte berücksichtigen.

Als Nächstes wird das Problem einer zuverlässigen und effizienten Navigation in Umgebungen mit Menschen behandelt. Dieses Problem ist eine besonders große Herausforderung, wenn es sich um eine erhöhte Anzahl von Menschen handelt. Aufgrund der hohen Dichte und der unsicheren Prädiktion der menschlichen Bewegung ist es manchmal unmöglich für den Roboter eine geeignete Trajektorie zu seinem Ziel zu finden. Um dieses Problem zu lösen werden neue Sicherheitskonzepte vorgestellt, die das Ausweichverhalten von reaktiven Objekten, wie zum Beispiel Menschen, berücksichtigen. Diese Methoden erlauben eine zuverlässigere und weniger konservative Sicherheitsbewertung, besonders in unsicheren Umgebungen mit einer hohen Dichte von Objekten.

Abschließend werden basierend auf den neuen Sicherheitskonzepten neue Algorithmen zur Bewegungsplanung vorgestellt. Dazu werden Bewegungsplaner basierend auf dem Konzept der optimalen Steuerung erweitert, um die Sicherheit der resultierenden Trajektorien für einen unendlichen Zeithorizont zu garantieren. Basierend auf dem neuen Ansatz zur Sicherheitsbewertung von Roadmaps wird ein generischer Bewegungsplaner präsentiert, der jede Trajektorie verbessern kann, indem zusätzliche Trajektorien generiert werden, die zu einer optimierten Roadmap führen. Es wurde gezeigt, dass diese Roadmap geringere Kosten besitzt als die optimal Trajektorie. Außerdem wird ein interaktiver Bewegungsplaner vorgestellt, der das Ausweichverhalten von reaktiven Objekten berücksichtigt. Dadurch wird das gegenseitige Ausweichverhalten des Roboters seiner umliegenden Objekte berücksichtigt, anstatt davon auszugehen, dass nur der Roboter den anderen Objekten ausweicht. Die Effektivität aller neu präsentierten Methoden zur Sicherheitsbewertung und Bewegungsplanung wird anhand von zahlreichen Simulationsstudien aus dem Bereich der mobilen Robotik und des autonomen Fahrens veranschaulicht.

Contents

1. Introduction **1**
- 1.1. Motivation . . . 1
- 1.2. Problem Formulation . . . 2
  - 1.2.1. Safety Criteria . . . 3
  - 1.2.2. Environment Description . . . 4
  - 1.2.3. Closed-loop Safety Assessment . . . 6
  - 1.2.4. Safety Assessment Beyond Planning Horizon . . . 7
  - 1.2.5. Interactive Safety Assessment . . . 8
- 1.3. Contributions and Outline of this Thesis . . . 9

2. Closed-loop Assessment **11**
- 2.1. Motivation and Problem Formulation . . . 11
- 2.2. Related Work . . . 12
- 2.3. General Idea . . . 13
- 2.4. Notation and Environment Description . . . 14
  - 2.4.1. Workspace Description . . . 14
  - 2.4.2. Graph Representation . . . 16
  - 2.4.3. Environment Model . . . 16
  - 2.4.4. Safety Assessment . . . 18
- 2.5. Trajectory with Minimum Collision Probability . . . 19
- 2.6. Collision Probability Incorporating Replanning . . . 20
  - 2.6.1. Collision Probability Regarding Multi-edges . . . 20
  - 2.6.2. Collision Probability for Serial Edges . . . 23
- 2.7. Collision Probability for the Entire Graph . . . 23
  - 2.7.1. Reduction of Vertices with Single Output . . . 24
  - 2.7.2. Graph Reduction . . . 25
- 2.8. Implementations . . . 25
  - 2.8.1. Estimation of Collision Probabilities for the Entire Graph . . . 27
  - 2.8.2. Environment Model for Mobile Robot Applications . . . 28
  - 2.8.3. Environment Model for Automotive Applications . . . 31
  - 2.8.4. Roadmap for Mobile Robot . . . 34
  - 2.8.5. Roadmap for Automotive Scenario . . . 35
- 2.9. Simulations . . . 35
  - 2.9.1. Determining the Safest Route . . . 36
  - 2.9.2. Mobile Robot Applications . . . 36
  - 2.9.3. Automotive Applications . . . 39
- 2.10. Discussion . . . 44

3. Assessment beyond Planning Horizon **47**
3.1. Motivation and Problem Formulation . . . 47
3.2. Related Work . . . 49
3.3. Inevitable Collision State and Inevitable Collision Obstacle . . . 50
3.4. Union of Inevitable Collision Obstacles . . . 51
3.5. Motion Safety Regarding Unexpected Objects . . . 54
3.6. Probabilistic Collision State . . . 55
3.7. Overall Collision Probability . . . 56
3.8. Probabilistic Collision Costs . . . 57
3.9. Implementations . . . 58
3.9.1. Inevitable Collision State Checkers . . . 58
3.9.2. Probabilistic Collision State Checker . . . 59
3.10. Simulations . . . 61
3.10.1. Inevitable Collision State Checkers . . . 61
3.10.2. Motion Safety Regarding Unexpected Objects . . . 64
3.10.3. Probabilistic Collision State Checker . . . 65
3.10.4. Overall Collision Probability . . . 68
3.10.5. Probabilistic Collision Cost . . . 68
3.11. Discussion . . . 70

4. Interactive Assessment **73**
4.1. Motivation and Problem Formulation . . . 73
4.2. Related Work . . . 75
4.3. Cooperative Inevitable Collision State . . . 76
4.4. Cooperative Probabilistic Collision State . . . 77
4.4.1. Definition of $\mathrm{cPCS}^{\mathrm{d}}$ . . . 77
4.4.2. Definition of $\mathrm{cPCS}^{\mathrm{u}}$ . . . 78
4.4.3. Discussion . . . 78
4.5. Implementations . . . 80
4.5.1. Automotive Application . . . 80
4.5.2. Mobile Robot Application . . . 84
4.6. Simulations . . . 88
4.6.1. Automotive Applications . . . 89
4.6.2. Mobile Robot Application . . . 92
4.7. Discussion . . . 95

5. Integration into Motion Planning **97**
5.1. Motivation and Problem Formulation . . . 97
5.1.1. Motion Planning in Deterministic Environments . . . 98
5.1.2. Motion Planning in Uncertain Environments . . . 99
5.2. Related Work . . . 101
5.2.1. Autonomous Navigation of Vehicles . . . 101
5.2.2. Mobile Robot Navigation in Populated Environments . . . 103
5.3. Optimal Control Considering Safety Beyond the Planning Horizon . . . 104

5.4. Motion Graph Planning . . . 107
5.4.1. On-line Planning . . . 107
5.4.2. Off-line Planning . . . 110
5.5. Interactive Motion Planning . . . 113
5.6. Implementations . . . 115
5.6.1. On-line Motion Graph Planning . . . 115
5.6.2. Off-line Motion Graph Planning . . . 116
5.6.3. Interactive Motion Planning . . . 116
5.7. Simulations . . . 119
5.7.1. Optimal Control . . . 120
5.7.2. Motion Graph Planning . . . 120
5.7.3. Interactive Navigation . . . 125
5.8. Discussion . . . 128

6. Conclusions **131**
6.1. Summary . . . 131
6.2. Discussion and Future Directions . . . 133

A. Estimation of Collision Probability **135**
A.1. Problem Formulation . . . 135
A.1.1. Notation . . . 135
A.1.2. Collision Probability for a Single Time Point . . . 136
A.2. Workspace Discretization . . . 137
A.3. Monte Carlo Approximation . . . 139
A.4. Discussion . . . 142

Bibliography **145**

Notations

Abbreviations

2D	two-dimensional
3D	three-dimensional
4D	four-dimensional
CDF	cumulative distribution function
cICS	cooperative inevitable collision state
cICSS	cooperative inevitable collision state set
cPCS	cooperative probabilistic collision state
CLRHC	closed-loop receding horizon control
DOF	degrees of freedom
FOV	field of view
FRP	freezing robot problem
ICO	inevitable collision obstacle
ICS	inevitable collision state
IND	index function
LUT	lookup table
MPC	model predictive control
OCP	overall collision probability
PCC	probabilistic collision costs
PCLRHC	partially closed-loop receding horizon control
PCS	probabilistic collision state
PDF	probabilistic distribution function
PMF	probability mass function
PMP	partial motion planning
POMDP	partially observable Markov decision process
PRM	probabilistic road map
RHC	receding horizon control
RRT	rapidly exploring random trees
RVO	reciprocal velocity obstacle
SIS	sequential importance sampling
SMC	sequential Monte Carlo

Symbols

General

$\|\cdot\|, \|\cdot\|_2$	Euclidean norm of a vector
$\|\cdot\|$	absolute value of a scalar
C	collision event
$\mathrm{E}[\cdot]$	expected value
$\mathcal{N}$	Normal distribution
$N_\square$	number of
$P(\cdot)$	probability
$P(\cdot\|\cdot)$	conditional probability
$\tau_\square$	threshold of
$T_\square$	time duration
$\mathrm{Var}(\cdot)$	variance

Subscripts and Superscripts

x^0	initial value of variable x at discrete time point 0
x_0	initial value of variable x for dynamic systems
x_g	goal value of variable x
x^*	optimal value for variable x
$\dot{x}$	time derivative of variable x

Sets

$\emptyset$	empty set
$\mathcal{A}$	subspace of $\mathcal{W}$ occupied by the robot
$\mathcal{A}^b$	subspace of $\mathcal{W}$ occupied by the enlarged robot system
$\mathcal{B}_i$	subspace of $\mathcal{W}$ occupied by the ith object
$\mathcal{B}$	subspace of $\mathcal{W}$ occupied by all objects
$\mathcal{E}$	edges
$\mathcal{G}$	graph containing set of edges $\mathcal{E}$ and vertices $\mathcal{V}$
$\tilde{\mathcal{U}}$	trajectory space
$\tilde{\boldsymbol{\mathcal{U}}}$	trajectory space of all objects in $\mathcal{W}$
$\mathcal{V}$	vertices
$\mathcal{W}$	workspace
$\mathcal{X}$	state space
$\mathcal{X}_\mathcal{W}$	state space of all objects in $\mathcal{W}$

Variables

a_x	acceleration along x
a_y	acceleration along y

d	lateral position
$\mathbf{e}_{ij}$	edge(s) between vertex i and vertex j
μ	mean
$\boldsymbol{\mu}$	mean vector
π	policy
$\mathbf{p}$	position
r	route
s	longitudinal position
σ	standard deviation
$\boldsymbol{\Sigma}$	covariance matrix
t	time
$\tilde{u}$	trajectory
u	control input
$\mathcal{U}$	control input space
$\boldsymbol{v}$	velocity
$\mathbf{v}_i$	vertex i
w	weight factor
x	x-Coordinate
$\mathbf{x}$	system state
$\mathbf{x}_{\mathcal{W}}$	state of all objects in $\mathcal{W}$
y	y-Coordinate

Functions

$\mathrm{cost}(\cdot)$	cost function
$D(\cdot)$	detectability function
$\mathrm{dist}(\cdot)$	distance function
$f(\cdot)$	PDF
$l(\cdot)$	length function
$L(\cdot)$	trajectory cost
$m(\cdot)$	motion model
$q(\cdot)$	PMF
$S(\cdot,\cdot)$	similarity
$\phi(\cdot)$	endpoint cost

Constants

N_{C}	number of collisions
N_{c}	number of control inputs
N_{b}	number of objects excluding the robot
N_{s}	number of samples
N_{k}	number of time steps
N_{m}	number of maneuvers
n	normalizing constant
$N_{\mathcal{V}}$	number of vertices

$N_{\mathcal{E}}$	number of edges
N_{v}	number of vehicles
τ_c	threshold improvement
τ_{coll}	threshold collision probability

1. Introduction

As a result of the overall progress in robotics, robots are entering environments that are populated by humans. These new environments pose particular challenges for the safety concept of robots since robots and humans share the same workspace. Nowadays, the threat of mobile robots constrain its application to specific tasks under certain conditions. Safe autonomous operation of robots is essential for widespread applications in human populated environments. Thus, this thesis focuses on safety assessment for robot motion in dynamic and uncertain environments. The subsequent chapters address substantial challenges of this problem and provide appropriate solutions.

This introduction starts with the motivation and presents various application examples. Following, the problem formulation is given starting with some background information and outlining the challenges of safety assessment. Finally, the outline of this thesis is given, including a short summary of its contributions.

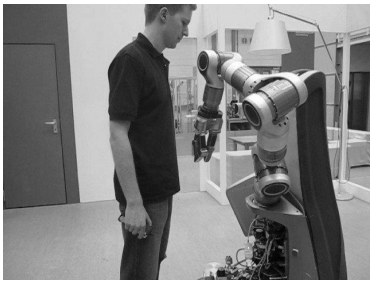
1.1. Motivation

Robots have evolved immensely in the past decades, but there is still a long way to go in making them capable to permanently operate in human populated environments. First achievements include robots which already navigate autonomously to unknown places in urban areas [12, 26] or guided people at exhibitions in the presence of several hundred pedestrians [93]. Apart from mobile platforms, autonomous vehicles navigate in urban environments in the presence of human driven vehicles [118]. Furthermore, robots are able to physical interact with humans, e.g. to hand over objects [107] and to assist elderly people [95]. Since the barrier between robots and humans has been started to fade, one of the major challenges for robotics is to ensure safe and reliable motion in order to achieve the human robot co-existence. This includes all kind of motions such as locomotion of a mobile platform and manipulator movements. In Fig. 1.1 some examples of robots operating in human populated environments are shown. The possible tasks of robot systems vary from navigating to a desired goal location, manipulating objects in the environment or physically interacting with humans. In order that robots can fulfill their task, a motion planning algorithm is used that usually generates a trajectory from the start to the desired goal while minimizing a certain objective. An overview of motion planning algorithms is given in [73].

A perception system which detects all objects in the environment in conjunction with a reliable motion prediction is necessary to provide the motion planning algorithm with a representation of the environment for a certain time horizon. Based on this representation, the aim of the motion planning algorithm is to find a trajectory towards the desired goal that optimizes a certain objective function while avoiding any collision. In human populated environments the most important objective is to prevent any harm to human beings, this is in line with Asimov's

(a) RoboX [106]

(b) Boss [118]



(c) Care-O-bot 3 [61]

Fig. 1.1.: Examples of robotic systems operating in human populated environments.

first law [10]:

> *" A robot may not injure a human being or, through inaction, allow a human being to come to harm. "*

Consequently, *motion safety* is becoming increasingly important in robotic research. The central problem: How to ensure safety in human-robot coexistence? A distinction in this thesis is made between *collision mitigation* and *collision avoidance* systems. A collision avoidance system aims to prevent any collision with objects in the environment, whereas a collision mitigation system aims to reduce the severity of a collision.

In [46] an exhaustive evaluation on collision mitigation systems for robot manipulators is given. Therefore, some soft robot control concepts are presented including the analysis of intrinsic joint compliance which are evaluated through crash tests. The crash tests include blunt impacts as well as soft-tissue injury caused by sharp tools.

The common approach for a collision avoidance system is to integrate a reliable safety assessment approach into a motion planning algorithm in order to avoid possible collisions. Some methods also concentrate on pure collision avoidance meaning that the only objective is to determine a collision-free trajectory. For instance, in [123] a collision avoidance algorithm for vehicles in emergency situations is presented. The basis of every safe motion planning algorithm or collision avoidance approach is a reliable motion safety assessment allowing to evaluate the risk of future motions. In this thesis, the focus is on safety assessment concepts for collision avoidance systems meaning any collision caused by the robot should be prevented.

1.2. Problem Formulation

This thesis addresses various problems associated with safety assessment of trajectories of robots in dynamic environments and as such contributes to the desired increase in safe operation of autonomous systems. A precise mathematical definition of this problem will be given in the following chapters, for now the less formal description is given:

Safety Assessment *Given the current state of the robot, a representation of its environment and an intended trajectory, the goal is to compute the risk that the robot will collide with any object in the environment when executing this trajectory.*

In order to accomplish this, the following subproblems need to be addressed:

Perception The robot is equipped with sensors providing raw sensor data of the environment. They enable robots with the ability to see, to touch, and to hear. Models of the environment are used for identification and interpretation of this sensor data to represent and understand the environment. The common approach for robot perception is to decouple the problem in object detection and object tracking. A brief overview of object detection approaches is given in [14] and for object tracking the reader is referred to [128].

Motion Prediction In order to assess a trajectory of the robot, the robot needs to know how the states of the objects in the environment evolve over time. Hence, the future states of all objects in the environment need to be predicted. Therefore, motion models are used that require information about the current state and goal state of the object which is used to compute the future states of the object. The quality of the safety assessment highly depends on the reliability of the motion prediction. A brief overview of human motion prediction techniques is given in [13] and for prediction of vehicles the reader is referred to [122].

Based on the information available from the environment model, it is possible to assess the motion safety of a robot trajectory. In the following, some criteria are introduced which form the general basis of a reliable safety assessment approach and the challenges associated with them are presented.

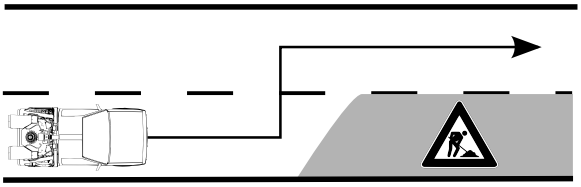
1.2.1. Safety Criteria

Many common safety assessment approaches or collision avoidance algorithms exists preventing collisions in many cases, but most of them cannot ensure safety in all possible situations. In [31] three criteria are introduced for evaluating common navigation approaches regarding their motion safety in dynamic environments: 1. Consider the robot's dynamics 2. Consider the environment objects' future behavior 3. Reason over an infinite time-horizon. The authors postulate that a safety assessment approach has to take into account all of these criteria. In the following, the criteria are explained in more detail.

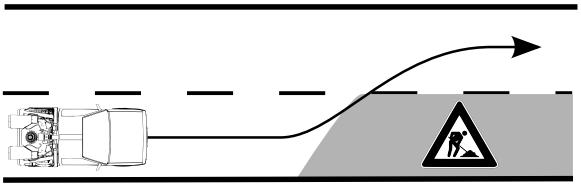
Consider the robot's dynamics The safety of a trajectory can only be evaluated if the robot is able to execute it. Meaning that the trajectory considers the kinematic and dynamic constraints of the robot system. Otherwise, the robot moves along a trajectory that was not evaluated.

Consider the environment objects' future behavior A possible collision can only be foreseen, if the future states for all objects in the environment are predicted. Otherwise, the safety assessment approach is only applicable to static environments.

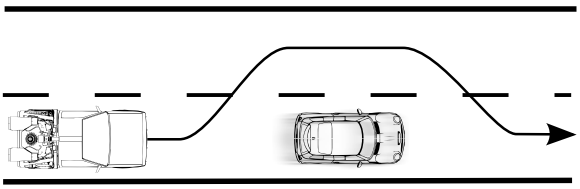
Reason over an infinite time-horizon At first, this criteria is confusing, since a robot trajectory is usually only valid for a finite time horizon. However, trajectories which are collision-free can eventually lead to a collision beyond the valid time horizon of the trajectory. To reason



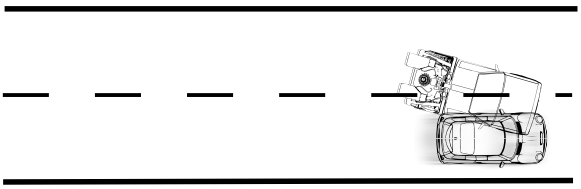
(a) The trajectory of the ego vehicle bypassing the construction area is generated without considering the kinematic or dynamic constraints.



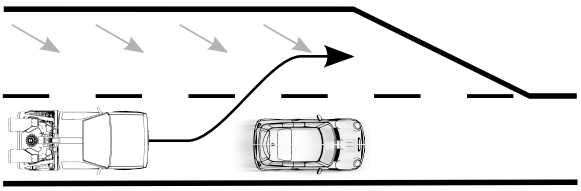
(b) The feasible trajectory of the ego vehicle considering kinematic and dynamic constraints leads to a collision with the construction area.



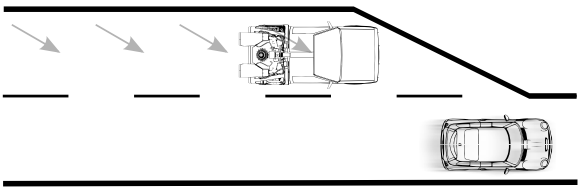
(c) The trajectory of the ego vehicle overtaking the car driving ahead is illustrated.



(d) The trajectory is leading to a collision with the other car, since it is planned without predicting the future behavior of the other car.



(e) The ego vehicle executes a trajectory to overtake the car driving ahead while approaching a road bottleneck.



(f) The trajectory is collision-free during the planning horizon, but will eventually lead to collision immediately after the planning horizon.

Fig. 1.2.: The topmost pictures illustrate the safety criterion *consider its own dynamics*. The criterion *consider the environment objects' future behavior* is illustrated by the middle pictures. The bottom most pictures illustrate the criterion *reason over an infinite time-horizon*.

about an infinite time horizon, the robot system needs to reach a safe state which is proven to be collision-free regardless the time.

These safety criteria are depicted in Fig. 1.2 by some illustrative examples. Depending on the environment, this three safety criteria cause different challenges for the safety assessment problem. In the next section, the different kind of environments are introduced that are considered in this thesis.

1.2.2. Environment Description

The operating space of the robot is called the environment or workspace. Objects present in the environment which change their states over time are called *dynamic* objects. This includes changing their position, their orientation or their configuration. Objects which states are not changing over time are called *static* objects. In this work, four different kind of objects are distinguished:

Static Objects Their states do not change over time, however the information about their states may be uncertain. Walls or furniture are typical examples for static objects in indoor environments. Whereas, crash barriers or buildings are examples for outdoor environments.

Ignoring Dynamic Objects Their states change over time, however, their future states are independent of the future behavior of other dynamic objects. This means, that these objects move in the workspace while ignoring all other objects. An example is a robot which follows a predefined trajectory without collision avoidance (pure trajectory following behavior).

Reactive Dynamic Objects Their states change over time and their future states depend on the future behavior of other objects. This means, that their future behavior depends on the behavior of the other agents which may result in a mutual dependency between the objects. Pedestrians or robots with collision avoidance capabilities are examples for this kind of objects.

Controllable Dynamic Objects Their states change over time, and their future states can be controlled. A robotic system which control inputs can be directly modified is an example for this kind of objects.

In the next section, different kinds of environment models are presented that represent theses kind of objects and their future behavior.

Environment Model

In order to assess robot trajectories, the safety assessment approach needs a model of the environment representing the current and future states of all objects. In this work, three different kinds of environment models are investigated which are illustrated in Fig. 1.3.

Deterministic It is assumed that no uncertainty exists. The complete state of the environment is known including the exact prediction of all objects in the environment. Meaning that there exists no ambiguity in the future behavior of the other objects. This allows performing a binary assessment of motion safety. Every robot trajectory is either safe or unsafe. However, real-world problems are never free of uncertainty since no sensor exists which is noise-free. Since the uncertainty is not taken into account, real-world scenarios may be evaluated wrong, meaning there can occur false safe or false unsafe classifications concerning the motion safety of trajectories.

Bounded Uncertainty The uncertainty is explicitly taken into account and it is assumed that the uncertainty is bounded and that the bounds are known. Thus, the worst case deterministic environment model can be constructed. This model is especially suitable for environments with a low density of objects and a low level of uncertainty. Otherwise, the set of valid future states becomes severely restricted leading to very conservative safety assessment results or to the extent that no valid states exists for the robot. However, the worst case model allows one to guarantee a conservative safety assessment, meaning no false safe classifications are possible.

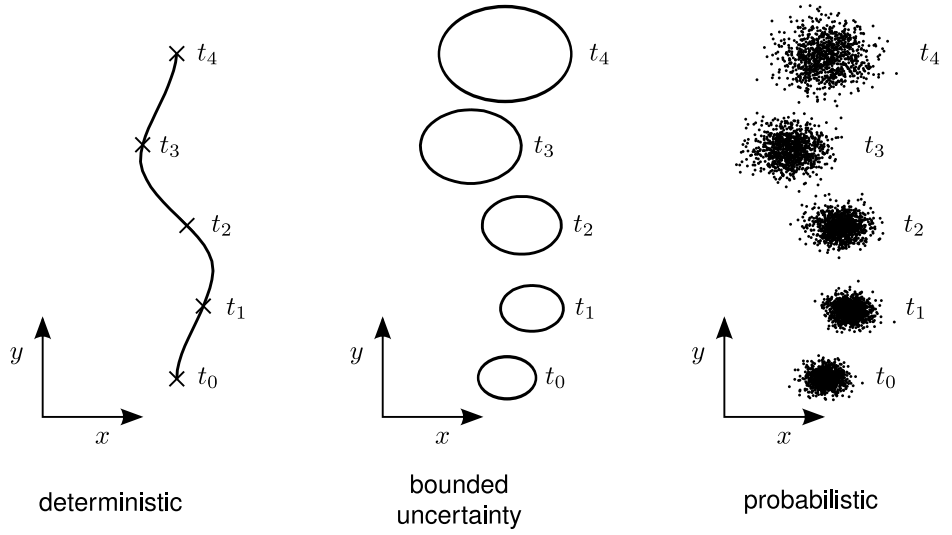


Fig. 1.3.: Different environmental models representing the position of one dynamic object over time (inspired from [34]). The left image shows the deterministic points at certain time points and the reference trajectory of the object. The bounded position uncertainty of the object is depicted in the middle and the right image shows sampled positions of the PDFs at certain time points representing the probabilistic position uncertainty.

Stochastic Instead of using a worst case approximation, the uncertainty is represented by a probabilistic model. Therefore, statistic information of the robot perception system is needed to make some probabilistic estimations of the state of the environment. The uncertainty is usually expressed by probabilistic density functions (PDFs). In contrast to the other environment models, it is possible that motion safety cannot be guaranteed for this kind of environmental model. This is because, the PDFs may have infinite support, such as Gaussian distributions, and thus any trajectory will have a non-zero risk. Instead of a binary assessment, the collision risk of a trajectory is estimated which is known as stochastic safety assessment. This model requires the highest computational effort, however, this kind of model is the only one which is suited for highly dynamic and uncertain environments.

These different environmental models create new challenges for the safety assessment of robotic systems. In the following, the challenges 1) closed-loop assessment 2) assessment beyond planning horizon 3) interactive assessment are briefly explained. These challenges are discussed in more detail in the subsequent chapters of this thesis.

1.2.3. Closed-loop Safety Assessment

In the recent history of motion planning, sampling-based algorithms have been successfully applied to navigation problems for mobile robots and robot arms or a combination of both. The most widely known approaches are rapidly exploring random trees (RRT) [76] and probabilistic roadmaps (PRM) [55]. Both methods generate graphs consisting of many trajectories which contain multiple solutions for the navigation problem. The common safety assessment concept is to separately compute the expected safety of each trajectory in the graph. This implies, that the graph structure is decomposed in all possible trajectories and thus the information about the

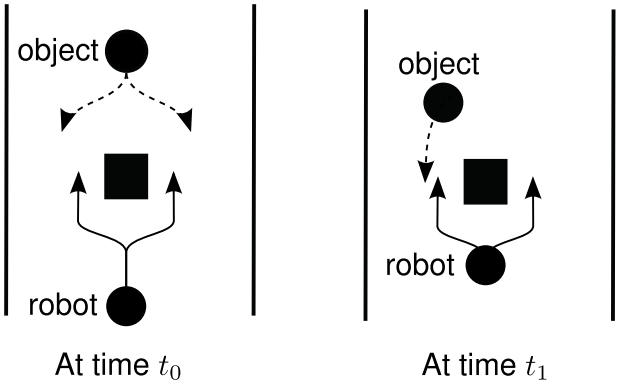


Fig. 1.4.: The scenario contains one robot and one dynamic object (human) in a corridor. Both objects want to move along the corridor in opposite direction and avoid any collision. At time t_0 the robot is unsure about the decision on the moving direction of the object. Thus, both trajectories of the robot have a 50% chance to collide with the object. At time t_1 the robot receives an updated prediction of the object and allows the robot to make a more precise decision on the choice of its trajectory.

connections of the trajectories is lost. In other words, the assessment ignores the fact, that the robot is able to replan its route during the execution of a certain trajectory depending on the future states of the other objects. This can be referred to as an *open-loop* assessment, since it does not consider the future observations of the objects. Especially in stochastic environments populated with dynamic objects this results in an unnecessary conservative assessment. This is mainly caused by the motion prediction of the dynamic objects which uncertainty is continuously increasing over time. One possible way to address this problem is to perform a *closed-loop* assessment instead. The main idea behind the closed-loop assessment is that all possible future trajectories of the robot are considered at once which allows the robot to change its future route. This is achieved by calculating the safety of all trajectories depending on all possible future states of the objects. This can be seen as a feedback rule and is therefore called closed-loop assessment. In contrast to the open-loop assessment that calculates the expected risk of each trajectory individually without considering the replanning possibilities depending on the future states of the objects. In Fig. 1.4 an illustrative scenario shows the idea of closed-loop assessment. If the robot trajectories are assessed separately, the collision probability of the robot is 50%. However, if the replanning possibility at time t_1 is taken into account, it is guaranteed that the robot will avoid the collision (collision probability 0%).

1.2.4. Safety Assessment Beyond Planning Horizon

The goal of motion planning is to find a feasible trajectory from the start to the goal state. Due to the intrinsic complexity of motion planning it is often impossible to compute the complete trajectory during the available time. Furthermore, objects in a real environment can only be reliably predicted for a limited time horizon which is usually much smaller than the time necessary to reach the goal. One possibility to tackle this problem is called the partial motion planning (PMP) concept [92]. The main idea is that the PMP algorithm computes the best partial trajectory towards the goal until it reaches the time constraint. This step is repeated till the robot reaches the desired goal. The basic PMP algorithm contains the following steps:

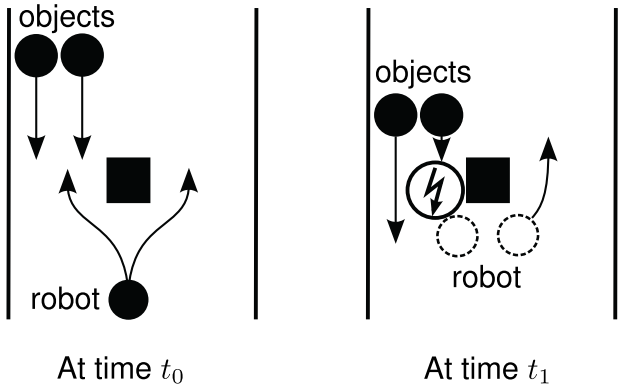


Fig. 1.5.: At time t_0 both possible robot trajectories are safe during the planning horizon, but at time t_1 one trajectory will inevitably lead to a collision.

1. Generate an updated model of the environment including motion prediction of dynamic objects.

2. An incremental motion planning algorithm is used for generating partial trajectories considering the time constraint.

3. When the time constraint is reached, the best available partial trajectory is executed.

The PMP algorithm runs until the partial trajectory reaches the goal. The goal can be a single state or a closed set of states. However, two problems arise by the PMP concept: safety issues after the end of the partial trajectory; convergence problem, the planning algorithm can get stuck in a local minimum and thus will never reach the desired goal. The problem of motion safety beyond the planning horizon is depicted in Fig. 1.5. As illustrated, the robot may choose a trajectory which will inevitably lead to a collision behind the planning horizon. The problem of safety assessment beyond the planning horizon addresses the third safety criteria from Sec. 1.2.1 (*reason about an infinite time-horizon*) for all kind of objects and all kind of environment models.

1.2.5. Interactive Safety Assessment

In environments which are populated by reactive or controllable dynamic objects, interaction between the workspace objects occurs. The interaction originates from the fact that the objects react to each other such that every object has sufficient space for navigation. In this work such kind of environments are named *uncertain and densely packed dynamic environments*. Crowded environments are also referred as environments incorporating interaction between the agents. However, hundreds or even thousands of agents are a necessary requirement for a crowded environment. In contrast, uncertain and densely packed dynamic environments may contain only few dynamic objects. For instance, in structured environments like narrow indoor corridors, already few dynamic objects are sufficient to observe interaction between the objects. In Fig. 1.6 such a scenario is depicted.

Assume a workspace that is populated with several dynamic objects and one robot. All objects including the robot try to reach their goal state while avoiding collisions. In classical motion or path planning the aim is to find a collision-free trajectory for the robot to its goal state by

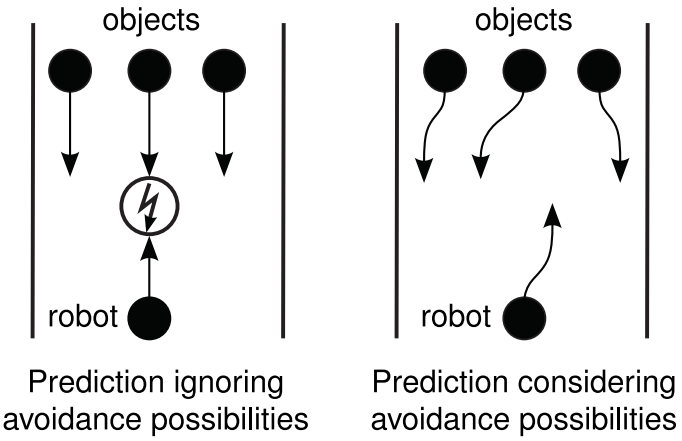


Fig. 1.6.: Without considering the avoidance possibilities of the objects, no trajectory can be found to prevent a collision. By incorporating this ability, it is possible to find a safe trajectory.

minimizing a certain cost function such as required time or energy consumption. Therefore, the problem is decoupled into motion prediction of each object and trajectory planning of the robot. Since in this work, environments are considered that are populated by reactive dynamic objects that mutually interact with each other, this separation is not valid anymore. An unreliable motion prediction will inevitably involve an unreliable safety assessment. This includes results of the safety assessment that are evaluated safer or more dangerous than they actually are.

1.3. Contributions and Outline of this Thesis

This thesis covers three major aspects of safety assessment of robot trajectories: 1. Safety assessment incorporating replanning (closed-loop assessment) 2. Safety assessment beyond the planning horizon (infinite-horizon assessment) 3. Safety assessment considering reactive objects (interactive assessment). It is shown that these aspects improve the performance and reliability of safety assessment for motion planning. Additionally, the proper integration of the improved assessment enhances also the performance of motion planning algorithms. The safety assessment is performed in different environments populated with static and dynamic objects, with or without uncertainty in the environment.

The closed-loop assessment is presented in Chap. 2. Its key contribution is the consideration of the replanning possibilities of the robot while executing a certain trajectory. It is a novel approach for the probabilistic safety assessment of trajectories in uncertain environments which are represented as directed graphs. Therefore, possible future measurements of all objects are considered. These possible future measurements can also be seen as possible future distributions of the objects representing their future position. The main difference compared to other approaches is, that the safety assessment considers the fact, that the chosen route can be replanned during execution if new information about the environment is available. This information decreases the uncertainty in the prediction of objects and allows a more reliable choice of the desired route already before this information is available. In order to take this fact into account earlier, the prediction of each object is not represented by a single distribution but by an infinite set of possible future distributions. It is proven that the collision risk for the whole graph is always smaller than that for a single route, since different routes in the graph have a lower collision probability

compared to others for certain distributions. This results in a less conservative safety assessment approach for uncertain environments.

The infinite horizon assessment is presented in Chap. 3. Its key contribution is the safety assessment of trajectories beyond the time horizon of the motion planning algorithm. In dynamic environments this is essential since it cannot be excluded that the robot will reach a state leading inevitable to a collision with an object beyond the finite time horizon of the motion planner. Such kind of states are called inevitable collision states (ICS). The sequential computation for unions of ICS sets and the concept of robot maneuverability for increasing motion safety are presented. Based on the novel calculation of ICS, two novel ICS-Checker algorithms are introduced allowing a more efficient computation as former implementations. The introduced robot maneuverability showed a significant reduction of robot collisions especially for unexpected objects or for objects with a limited motion prediction. For uncertain environments, the probabilistic collision states (PCS) are presented. They directly consider the uncertainty in the states of the objects and in their motion prediction. In addition, a novel safety assessment cost metric, the probabilistic collision cost (PCC), is introduced which considers the relative speeds and masses of multiple moving objects the robot may collide with. This allows assessing the harm of a collision instead of only considering the probability of the collision.

The interactive assessment is presented in Chap. 4. Its key contribution is the consideration of the avoidance behavior of the dynamic objects in the workspace. Therefore, the extended definitions of the cooperative ICS for deterministic environments and the cooperative PCS for uncertain environments are given. Both approaches take into account the avoidance behavior of the other objects. This interaction originates from the fact that the objects including the robot need to react to each other, such that every object has sufficient space for navigation. This increases the reliability of the safety assessment and results in a less conservative assessment. The later allows the robot to navigate in environments with a high density of objects which is not possible without considering their reactive behavior.

In Chap. 5 possible integrations of the novel safety assessment approaches into motion planning algorithms are presented. An incremental optimization of the closed-loop assessment approach in shown that improves any solution to the navigation problem by generating additional trajectories for the robot. This allows the robot to replan its route depending on the future state of the environment. The idea of interactive safety assessment is extended to demonstrate its impact also for the original motion planning problem. All integrations show substantial improvement compared to common motion planners in different simulation scenarios.

2. Closed-loop Assessment

Summary This chapter discusses the problem of assessing the safety of roadmaps in dynamic and uncertain environments. The roadmaps are a collection of possible trajectories which are represented as a directed graph. This allows the chosen route to be replanned during execution, if new information about the objects is available. Hence, the assessment considers all possible routes depending on the future measurements of the objects, this is called a close-loop assessment. In highly uncertain environments, this method allows a more reliable safety assessment compared to methods ignoring the replanning possibilities.

The chapter is organized as follows: Sec. 2.1 gives the motivation and problem formulation. An overview on related work is presented in Sec. 2.2. In Sec. 2.4 the environment description and notation for this chapter is given. This allows one to discuss the problem of the safest trajectory between two vertices in Sec. 2.5. The problem of safety assessment considering replanning is addressed in Sec. 2.6. This leads to the assessment approach for graphs in Sec. 2.7. Following, an example implementation for this approach is presented in Sec. 2.8 whose simulation results are discussed in Sec. 2.9. Finally, a brief discussion of this chapter is given in Sec. 2.10.

2.1. Motivation and Problem Formulation

Motion planning in uncertain environments is a challenging task, one reason is that the uncertainty of the environment increases over time, resulting in PDFs with high uncertainty. This is especially true for planning problems with long time horizons. Thus, the objects are distributed over a large space of the workspace, meaning the objects are ”everywhere and nowhere”. This can be also expressed with the theory of differential entropy [79]. In the extreme case, the distributions of all objects compose to a uniform distribution in the workspace which is equivalent to the scenario that no information is given about the objects. As a consequence, the collision probability can be considered as constant for all possible trajectories, hence no meaningful assessment can be performed for the trajectories.

One possibility to cope with uncertain environments in motion planning methods is to perform steady replanning. Replanning is performed, if unexpected changes took place in the environment or if due to the lower uncertainty the replanned solution is superior to the old one. The replanning strategy can be seen as generating additional solutions to the motion planning problem, given the robot the possibility to replan its future motion depending on the current state of the environment. This strategy should be also considered in the safety assessment approach, meaning that the possibility of replanning should be directly taken into account. During the assessment the different routes are assessed depending on the possible future object distributions,this is referred to as *closed-loop assessment*.

In this chapter, a novel approach for the probabilistic safety assessment of trajectories in uncertain environments is presented. The trajectories are represented as directed graphs. Directed

graphs arise by applying rapidly exploring random trees (RRT) [76] or roadmap-based planners [55] to the kinodynamic motion planning problem. The main difference compared to other approaches is, that the safety assessment considers the fact, that the chosen route can be replanned during execution if new information about the objects is available. This information decreases the uncertainty in the prediction of objects and allows a more reliable choice of the desired route already before this information is available. In order to take this fact into account earlier, the prediction of each object is not represented by a single distribution, but by an infinite set of possible future distributions. This set comprises all possible distributions of the object resulting from the possible future measurements. In the following, a brief overview of related work is given.

2.2. Related Work

Following, a brief overview of prior work regarding safety assessment in uncertain and dynamic environments is given. For safety assessment in dynamic and uncertain environments, it is necessary to consider the uncertain information about the future states of other objects. There are two possible approaches [34]: worst case prediction of objects resulting in a deterministic assessment or performing a stochastic assessment. The first approach allows one to perform deterministic (binary) assessment in stochastic environments and can guarantee safety in many situations, but it leads to over-approximations and a very conservative assessment. As for the second possible approach it is common to compute the collision probability for a certain trajectory as a safety criteria. Therefore, stochastic motion prediction is performed for all objects in the environment taking into account the uncertainties. A generic approach for estimating the collision probability for arbitrary shapes and probability density functions representing the future position of the obstacles is presented in [65]. For many applications it is essential to have a quantitative measurement for the intensity of a possible collision. In these cases, the probabilistic collision costs can be used instead of the collision probability as an indication. In [67] the squared speed of the colliding objects is used which was replaced by the internal energy assuming an inelastic impact in [135] taking into account the movement direction of the objects. A stochastic threat assessment algorithm called *collision mitigation by braking system* is presented by [53]. It aims at mitigating the harm of an accident by braking the car once a collision is inevitable. The future trajectories of the other objects are predicted and based on them the probability of collision is estimated to determine if emergency braking should be executed. A more general approach for threat assessment in traffic scenes including a driver model was first published by [18] and extended by [29]. Monte Carlo simulation is used for threat detection of traffic scenes. In stochastic optimal control, the safety is often expressed with chance constraints, meaning that the collision probability or probability of failure must be below a certain threshold. The approach of [15] presents a particle-based approximation technique allowing one to approximate the stochastic optimization problem as a deterministic optimization problem. The estimation error of the collision probability decreases with the number of samples and approaches zero when the number of particles tend to infinity. All these approaches are based on the estimation of the collision probability based on the motion prediction of the objects.

In the field of motion planning, some approaches consider not only the current information of the environment but also possible future measurements. The partially closed-loop receding horizon control (PCLRHC) presented in [114] is one of these approaches. Since future measurement

are taken into account the prediction of the objects becomes more certain. For integration into a control approach, the PCLRHC strategy assumes that the most probable future measurement will occur, instead of considering all possible measurements. Since only one measurement of each object is considered at every state of the robot, the future states of the objects are predictable. However, this leads to a non-conservative safety assessment of the trajectories, since not all possible measurements are considered. The same limitation holds for the approach presented in [54], where the partially observable control problem is transformed to a fully observable underactuated stochastic control problem by assuming maximum likelihood observations. The linear-quadratic Gaussian motion planning (LQP-MP) approach presented in [120] assesses the collision probability of a given path by taking into account the stochastic models representing the motion and sensing uncertainty. This is possible by integrating the sensors and controller, which are applied to execute a given path, into the planning approach. Therefore, the a priori probability distributions of the future states and control inputs along the path are computed. Compared to other approaches of [54] and [114] all possible future measurements are considered instead of only assuming the maximum-likelihood measurements. However, this approach requires the motion and sensing uncertainties represented by Gaussian distributions and has only been applied to static environments.

The motion planning approaches above consider possible future measurements of the environment, but they consider or plan only one motion possibility (trajectory) of the robot system. The concept of bounded uncertainty roadmaps are presented in [45] and addresses the problem of roadmap-based planning in uncertain environments. The roadmap contains all future motion possibilities of the robot. Due to the uncertainty, there are no guarantees that the vertices and edges of the roadmap are collision-free. The goal in this work is to find a route with minimum cost according to a cost function incorporating the path length and collision probability. It is similar to [86] but presents a superior algorithm for approximating the collision probability in terms of scalability to higher dimensions and quantification error.

This chapter presents a safety assessment approach which assesses roadmaps by determining the policy minimizing the collision probability of the robot reaching a goal configuration. This policy considers all motion possibilities and the future measurements of the objects.

2.3. General Idea

In this chapter, a novel approach is presented to assess the safety for a robot system reaching a predetermined goal state in uncertain and dynamic environments. The uncertain future states of the objects are represented by probability distributions. The robot system behaves deterministically but has multiple motion possibilities to reach its goal state which are represented by a directed graph. This graph can also be called a roadmap [55]. The main idea of this safety assessment approach is that a policy is determined which replans the route of the robot if new information about the objects is available. This information decreases the uncertainty in the prediction of objects and allows a more confident choice of the desired route already before this information is available.

This idea is explained by an illustrative example. A robot and a human are walking towards each other in a corridor and are approaching an obstacle. The robot is unsure about the future motion of the human, whether he will choose the upper or the lower trajectory to avoid the

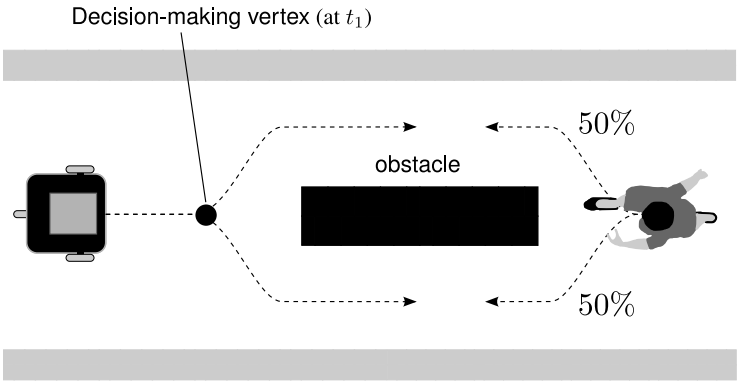
obstacle. The robot has to decide if it passes the obstacle either left or right to reach its goal. In Fig. 2.1 the scenario is depicted. Starting at t_0, the robot moves straight and needs to decide until time t_1 (decision-making vertex) if it takes the upper or the lower trajectory to pass the obstacle. Both trajectories have a collision probability of 50% if they are evaluated separately based on the available information at time t_0. However, if the replanning possibility at the decision-making vertex as well as the possible future information of the human are taken into account, the situation can be assessed more precisely. This is illustrated in Fig. 2.1b and Fig. 2.1c. At time t_1 the robot will have an updated prediction of the human motion which will be less uncertain than at time t_0. On the basis of this information, the robot is able to make a decision which will result in a collision probability of only 5%. In order to incorporate this replanning possibility already in the safety assessment at time t_0, all possible situations (future states of the human) and the associated policy (trajectory) need to be considered. In this case, two possible scenarios are simulated each having a collision probability of 5%. Hence, the resulting collision probability of the roadmap at time t_0 is also 5% instead of the 50% without considering the replanning possibility.

2.4. Notation and Environment Description

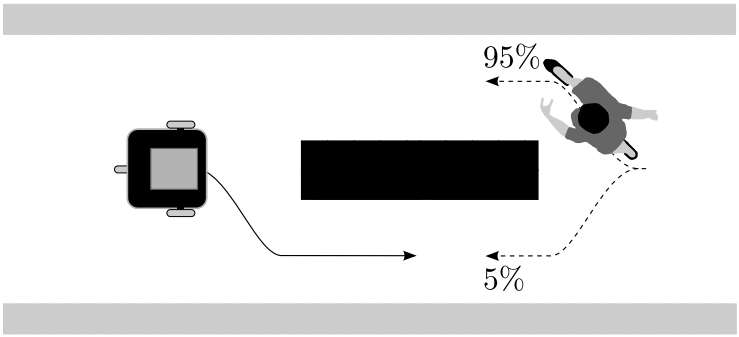
In this chapter, the problem of safety assessment in uncertain and dynamic environments is discussed. The future motion of the robot is deterministic and its future trajectories are represented by directed finite graphs with multi-edges, also called multigraphs. The state of the moving objects are not exactly known and represented by probability distributions. In a nutshell: the safety of trajectories between two vertices is assessed while considering the possibility of replanning based on acquired information about the future states of the objects. The two cases of vertices connected by a single edge and by multi-edges as shown in Fig. 2.2 are discussed in the following. Therefore, some notation is introduced.

2.4.1. Workspace Description

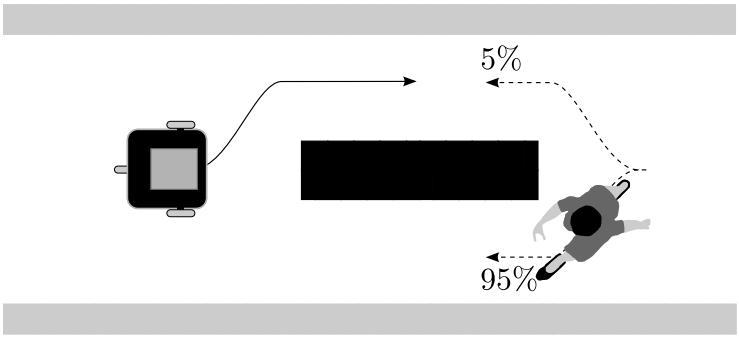
The workspace of the robot system $\mathcal{A}$ is denoted by $\mathcal{W}$ and the subset of the workspace occupied by $\mathcal{A}$ in state $\mathbf{x}(t)$ is expressed as $\mathcal{A}(\mathbf{x}(t)) \subset \mathcal{W}$. The state $\mathbf{x} = [\mathbf{p}, \boldsymbol{v}]$ is represented by its position $\mathbf{p}$ and its velocity $\boldsymbol{v}$. The occupancy of the ith object in the workspace is denoted by $\mathcal{B}_i(t)$. The unified occupancy of all objects is written in short notation as $\mathcal{B} = \bigcup_{i=1,\dots,N_\mathrm{b}} \mathcal{B}_i$ where N_b is the number of workspace objects. An initial state and a sequence of control inputs define a trajectory for $\mathcal{A}$, i.e. a time sequence of states. A trajectory of the robot system is denoted by $\tilde{u}$ and a certain time interval of the trajectory is expressed as $\tilde{u}([t_i, t_j))$, where a round bracket excludes the endpoint and the square bracket includes it. The workspace occupancy generated from the input trajectory is denoted by $\mathcal{A}(\tilde{u}(t))$ and is deterministic. Due to the lack of a perfect model of the environment the states of the objects are represented by probability distributions. The distribution describing the state of the ith object $\mathcal{B}_i$ at time t is denoted by $f_i(\mathbf{x}, t)$ and the distribution representing their position uncertainty is expressed as $f_i(\mathbf{p}, t)$. Since one can only formulate a probability distribution for a random vector and not an occupancy set, f_i represents the probability distribution of the reference point of $\mathcal{B}_i$. The initial information about the objects is expressed as the initial belief $\mathbf{b}_{t_0} = \{f_1^{t_0}(\mathbf{x}, t), \dots, f_{N_\mathrm{b}}^{t_0}(\mathbf{x}, t)\}$ which contains all initial distributions of the objects at time t_0



(a) Situation at time t_0, it is not yet clear which trajectory the human will choose to circumvent the obstacle.



(b) First possible situation at time $t_1 > t_0$. The upper trajectory of the human has a considerably higher probability, thus the robot chooses the lower trajectory.



(c) Second possible situation at time $t_1 > t_0$. The lower trajectory of the human has a considerably higher probability, thus the robot chooses the upper trajectory.

Fig. 2.1.: This example shows the problem of decision making on the basis of an uncertain motion prediction. The robot and the human are approaching an obstacle, whereby both have the option to take the upper or lower trajectory.

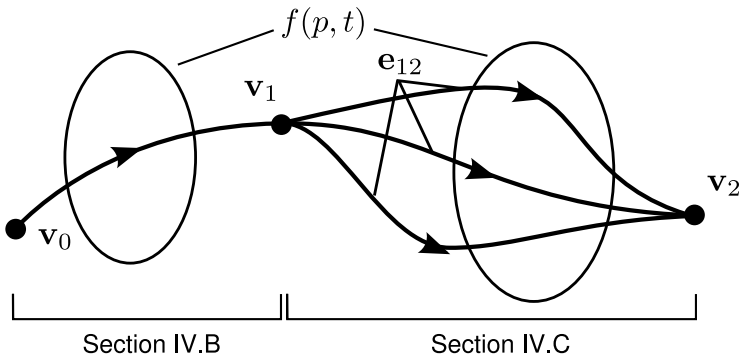


Fig. 2.2.: Example of a graph with multi-edges. Two possible object distributions $f(\mathbf{p}, t)$ are illustrated by its 2-σ ellipsoids for one time point. The safety assessment for single- and multi-edges is treated separately.

2.4.2. Graph Representation

The graph $\mathcal{G} = \{\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}\}$ contains a list of vertices $\mathbf{v}_i \in \mathcal{V}_\mathcal{G}$ and edges $\mathbf{e}_{ij} \in \mathcal{E}_\mathcal{G}$. All vertices in the graph contain information about the state $\mathbf{x}$ of the robot system and at which time t it will reach this vertex (state)

$$\mathbf{v}_i = \{\mathbf{x}_i, t_i\}.$$

Each edge $\mathbf{e}_{ij}$ is a set containing all trajectories $\tilde{u}$ connecting the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$

$$\mathbf{e}_{ij} = \{\tilde{u}_1(\mathbf{v}_i, \mathbf{v}_j), \ldots, \tilde{u}_m(\mathbf{v}_i, \mathbf{v}_j)\},$$

where $\mathbf{e}_{ij}$ is called a *multi-edge* if $m > 1$. If no connection exists, $\mathbf{e}_{ij}$ is empty. A route $r(\mathbf{v}_i, \mathbf{v}_j)$ in the graph consists of multiple partial trajectories (edges) describing a possible trajectory for traversing between vertex $\mathbf{v}_i$ and $\mathbf{v}_j$

$$r(\mathbf{v}_i, \mathbf{v}_j) = \{\tilde{u}(\mathbf{v}_i, \mathbf{v}_{k_1}), \tilde{u}(\mathbf{v}_{k_1}, \mathbf{v}_{k_2}), \ldots, \tilde{u}(\mathbf{v}_{k_n}, \mathbf{v}_j)\}$$

with $i = k_0, j = k_{n+1}$ and

$$\tilde{u}(\mathbf{v}_{k_{l-1}}, \mathbf{v}_{k_l}) \in \mathbf{e}_{k_{l-1}k_l} \qquad \forall l \in \{1, \ldots, n+1\}.$$

In Fig. 2.2 an example for a multigraph is shown. In order to generate such a graph structure, one needs to generate a set of vertices and connect them with trajectories. Such kind of graph can be seen as a *roadmap*, generated from a roadmap-based planner [55]. The vertices are sampled configurations and the edges represent the trajectories to traverse from one configuration to another. Another sampling-based approach which generates a graph structure to solve the motion planning problems are *rapidly exploring random trees* [76], especially in the case of multi-directional rapidly exploring random trees [73].

2.4.3. Environment Model

The representation of robot motions by a graph structure together with the environment model are the core parts leading to the novel safety assessment. For the environment model, represen-

tation and motion prediction of the workspace objects, two different situations are distinguished: motion prediction during a time interval $(t_i, t_j]$ between two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ and the prediction at the time point of the vertices t_i and t_j. This distinction is necessary to model the information gain about the states of the objects which is available for the robot system at the vertices.

For predicting the future states of an object for a certain time interval, any state-of-the-art probabilistic motion prediction algorithm can be used representing the future state $\mathbf{x}$ as a distribution $f^{t'}(\mathbf{x}, t)$ for a certain time point t. The superscript t' indicates the time point of the information which is used for the prediction. This time point is called the observation time, since it represents the time of the information at which the object was last detected or observed.

During the execution of a trajectory, it is assumed that the robot system receives new measurements of the workspace objects which are available at each vertex. These new measurements or information of the objects result in updated distributions for the positions and states of the objects. The updated distributions will have a lower variance Var

$$\mathrm{Var}(f^{t_j}(\mathbf{p}, t)) \leq \mathrm{Var}(f^{t_i}(\mathbf{p}, t)), \quad \text{with } t_i < t_j \leq t,$$

where t_j and t_i indicate the observation times which are used for the prediction at time t. In this case, the distribution $f^{t_i}(\mathbf{p}, t)$ is more uncertain than $f^{t_j}(\mathbf{p}, t)$, since it is based on the information available at time t_i which is older and thus the predicted position for time t is more uncertain. This assumption also holds if no new information is available (e.g. occluded objects), in this case the distribution is not updated.

Since the future information or measurements at time t_j are not known at time t_i, the possible distributions at time t_j are represented as a compound distribution. A compound probability distribution [44] is described by a parameterized distribution with a parameter vector $\boldsymbol{\theta}$ that is distributed according to another distribution $f(\boldsymbol{\theta})$. The compound distribution results from marginalizing over the distribution of the parameter vector. It is assumed that the future measurements at time t_j $(t_j > t_i)$ are consistent with the prediction f^{t_i}. Thus, the distribution at time t_i can be seen as the compound distribution comprising from all new distributions at time t_j

$$f^{t_i}(\mathbf{x}, t) = \mathrm{E}_{\boldsymbol{\theta}(t_i)}[f^{t_j}(\mathbf{x}, t|\boldsymbol{\theta})] = \int f^{t_j}(\mathbf{x}, t|\boldsymbol{\theta}) f\big(\boldsymbol{\theta}(f^{t_i}(\mathbf{x}, t_j))\big)\, \mathrm{d}\boldsymbol{\theta}. \tag{2.1}$$

The parameters $\boldsymbol{\theta}$ of $f^{t_j}(\mathbf{x}, t|\boldsymbol{\theta})$ are distributed according to $f\big(\boldsymbol{\theta}(f^{t_i}(\mathbf{x}, t_j))\big)$ which depends on the distribution $f^{t_i}(\mathbf{x}, t_j)$ at time t_j with the observation time t_i. The idea of compound distributions is also known from Bayesian Interference [17], there this is called the *prior predictive distribution* [39, 99]. It addresses the following problem: Before new information of the environment is available which future measurements (distributions) of the objects can be expected?

For better understanding, one example implementation of this environment model illustrates the interplay of the separated prediction for one moving obstacle in Fig. 2.3. At time t_0 the position uncertainty is represented by one Gaussian distribution $f^{t_0}(\mathbf{p}, t_0)$. For the time interval $(t_0, t_1]$ the object is predicted with the constant velocity model (Sec.2.8.2) resulting in the distribution $f^{t_0}(\mathbf{x}, t_1)$, the corresponding position distribution is depicted in the figure. As expected, the uncertainty of the prediction based on the information at time t_0 has increased over time. At time t_1, it is expected that the robot system perceives the object with its sensors. It is assumed that the updated future distributions have equal variance (measurement noise constant) but the mean

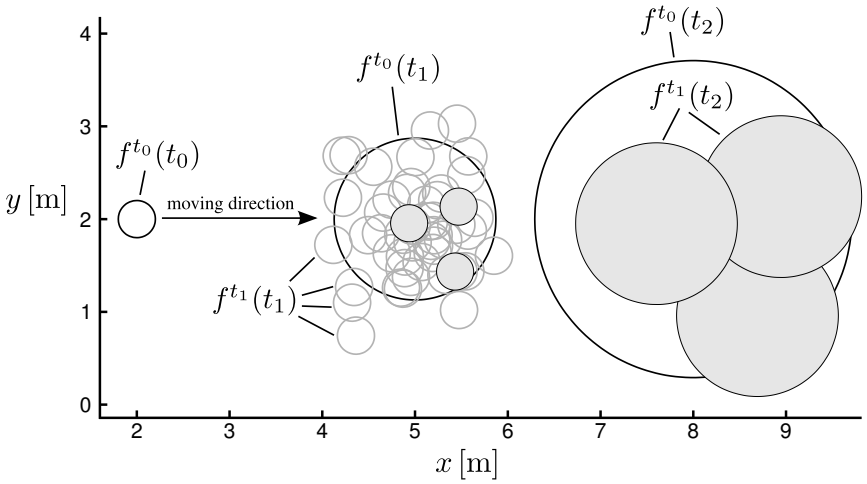


Fig. 2.3.: The proposed environment model is sketched for one moving object using Gaussian distributions to represent its position uncertainty. The constant velocity model is used for the prediction during the time intervals. Two different time bases of information exist, t_0 and t_1 which model new measurements of the object. The compound distribution of f^{t_1} is depicted by 50 possible samples. Three of them (small gray solid circles) are again predicted with the constant velocity model till time t_2 (big gray solid circles).

value of the Gaussians is unknown. 50 position distributions according to possible future distributions $f^{t_1}(\mathbf{x}, t_1)$ based on the information at time t_1 are shown. For the time interval $(t_1, t_2]$ the possible future distributions are again predicted resulting in the $f^{t_1}(\mathbf{x}, t_2)$ distributions, whereby three example position distributions are depicted in Fig. 2.3. At time t_2, these distributions are replaced by the updated distributions $f^{t_2}(\mathbf{x}, t_2)$ which are used for predicting the object in the next time interval. Based on this environment model, the problem of assessing the motion safety for roadmaps is formulated.

2.4.4. Safety Assessment

This chapter addresses the problem to assess the safety of a given graph of motion possibilities in an uncertain environment. Instead of determining and assessing the safety of the route $r^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})$ with the minimum collision probability or collision costs in an uncertain roadmap [45, 86], this work aims to determine the collision probability $P(C|\mathcal{G}, \mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})$ for traversing from a specified start vertex $\mathbf{v}_s$ to a goal vertex $\mathbf{v}_g$ by considering the entire graph $\mathcal{G}$ and the initial belief $\mathbf{b}_{t_s}$. Thus, the optimal policy to the goal vertex is determined by considering the current and possible future distributions of all objects and the possibility to replan the route during execution. This can be seen as a feedback-based or closed-loop assessment (according to feedback-based planner or closed-loop optimal control) which determines the safest route depending on the predicted distribution of the objects at the vertices of the roadmap. This problem is also closely related to the partially observable Markov decision process (POMDP) [113] problem. A POMDP models the behavior of the robot which tries to maximize its reward by a sequence of actions in an uncertain environment. The POMDP formulation considers uncertainties in the future motion of the robot and the future observations. The solution to the POMDP

problem is to determine the optimal policy π^* of the robot which maximizes the expected reward. A policy $\pi : B \mapsto A$ is a mapping from the belief space B to the available action space A. In this work, the motion of the robot system is deterministic but the future states of the objects are uncertain. Thus the robot system has only a belief of the future states of the objects. The goal is to determine the optimal policy which minimizes the expected collision probability for a given graph $\mathcal{G}$ and a given initial belief $\mathbf{b}_{t_s}$

$$\pi^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s}) := \arg\min_{\pi} P(C|\mathcal{G}, \pi(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})). \tag{2.2}$$

For a given belief $\mathbf{b}$ the policy returns a certain action $\tilde{u} \in \mathcal{E}$. The collision probability of the graph is defined as the probability that the robot system collides with any object while executing the optimal policy

$$P(C|\mathcal{G}, \mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s}) := P(C|\mathcal{G}, \pi^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})).$$

The belief of the state of an object is represented by the compound distribution described in Sec. 2.4.3. The presented approach for solving this POMDP like safety assessment problem is closely related to point-based POMDP approaches [108]. In this work a hierarchical tree of future beliefs (object distributions) is generated, too. For each sample the optimal policy π^* is determined allowing one to approximate the expected collision probability for the graph.

2.5. Trajectory with Minimum Collision Probability

Before the safety assessment for vertices connected by multi-edges or single-edges incorporating replanning are discussed in Sec. 2.6, the problem of determining the collision probability of a single trajectory is discussed. Since the focus of this work is on the effect and incorporation of replanning into safety assessment and not on a sophisticated approach for estimating collision probabilities, some simplifications are used. It is assumed, that the collision probabilities of the edges of the graph are independent, meaning that truncated distributions are not taken into account. This allows a separated estimation of the collision probabilities of all edges. In [43] and [91] the problem of truncated Gaussian distributions is addressed. Furthermore, it is assumed that the objects move independently thus the collision probability of a trajectory $\tilde{u}$ considering all N_b objects $\mathcal{B}_i$ is derived as

$$P(C|\tilde{u}, \mathbf{b}_{t'}) = 1 - \prod_{i=1}^{N_\mathrm{b}} \big(1 - P_i(C|\tilde{u}, f_i^{t'}(\mathbf{x}, t))\big),$$

where C is the collision event and $P_i(C|\tilde{u}, f_i^{t'}(\mathbf{x}, t))$ is the probability that the trajectory $\tilde{u}$ will lead to a collision with the ith object with the information available at observation time t'. Using the compound distribution (2.1) the collision probability is computed as

$$P(C|\tilde{u}(\mathbf{v}_i, \mathbf{v}_j), f^{t'}(\mathbf{x}, t)) = \int P\big(C|\tilde{u}(\mathbf{v}_i, \mathbf{v}_j), f^{t_i}(\mathbf{x}, t|\boldsymbol{\theta})\big) f\big(\boldsymbol{\theta}(f^{t'}(\mathbf{x}, t_i))\big)\, \mathrm{d}\boldsymbol{\theta},$$

where $P\big(C|\tilde{u}(\mathbf{v}_i, \mathbf{v}_j), f^{t'}(\mathbf{x}, t|\boldsymbol{\theta})\big)$ is the collision probability during the time interval $[t_i, t_j]$ based on the prediction $f^{t'}(\mathbf{x}, t|\boldsymbol{\theta})$. Two possible implementations for estimating the collision

probabilities $P\big(C|\tilde{u}, f(\mathbf{x}, t)\big)$ are described in Sec. 2.8. The collision probability for one trajectory considering all workspace objects allows us to define:

Definition 1 (Minimum collision trajectory $\tilde{u}^*$ between two vertices $\mathbf{v}_i$, $\mathbf{v}_j$)**.**

$$\tilde{u}^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t'}) := \underset{\tilde{u} \in \mathbf{e}_{ij}}{\arg\min}\, P(C|\tilde{u}, \mathbf{b}_{t'}),$$

where t' is the observation time.

This definition only considers the edge with the lowest risk to assess the safety for traversing between the vertices $\mathbf{v}_i, \mathbf{v}_j$ and ignores all other edges.

For calculating the collision probability of vertices connected by multi-edges, the minimum collision trajectory concerning one parameterized distribution of one object is defined.

Definition 2 (Minimum collision trajectory $\tilde{u}^*$ between two vertices $\mathbf{v}_i$, $\mathbf{v}_j$ regarding one distribution)**.**

$$\tilde{u}^*(\mathbf{v}_i, \mathbf{v}_j, f^{t'}(\mathbf{x}, t|\boldsymbol{\theta}_k)) := \underset{\tilde{u} \in \mathbf{e}_{ij}}{\arg\min}\, P(C|\tilde{u}, f^{t'}(\mathbf{x}, t|\boldsymbol{\theta}_k)).$$

Following, it is shown that new information about the location of the objects lead to a more precise prediction allowing a novel safety assessment for multi-edges.

2.6. Collision Probability Incorporating Replanning

During the execution of a certain trajectory $\tilde{u}$, the robot system has the possibility to decide at each vertex how to continue. Additionally, new information about the objects will be available. These two assumptions are used for assessing the safety for traversing between two configurations.

This is shown only for one object, $\mathbf{b}_{t'} = \{f^{t'}(\mathbf{x}, t)\}$ but this can be easily extended to multiple objects by determining the optimal trajectory minimizing the collision probability regarding all objects instead of only one.

2.6.1. Collision Probability Regarding Multi-edges

If two nodes are connected by multi-edges, all possible trajectories $\tilde{u} \in \mathbf{e}_{ij}$ between $\mathbf{v}_i$ and $\mathbf{v}_j$ have to be considered.

Definition 3 (Collision probability between two adjacent vertices $\mathbf{v}_i$, $\mathbf{v}_j$ with multi-edges)**.** *The collision probability between two adjacent vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ connected by multi-edges based on the information from time t_k is defined as*

$$\begin{aligned} P(C|\mathbf{e}_{ij}, \mathbf{b}_{t_k}) :&= \mathrm{E}_{\boldsymbol{\theta}(t_k)}\left[P\big(C|\tilde{u}^*(\mathbf{v}_i, \mathbf{v}_j, f^{t_i}(\mathbf{x}, t|\boldsymbol{\theta}))\big)\right] \\ &= \int P\big(C|\tilde{u}^*(\mathbf{v}_i, \mathbf{v}_j, f^{t_i}(\mathbf{x}, t|\boldsymbol{\theta}))\big)\, f\big(\boldsymbol{\theta}(f^{t_k}(\mathbf{x}, t_i))\big)\, \mathrm{d}\boldsymbol{\theta}, \end{aligned}$$

with $t_i > t_k$.

The time t_k is the observation time of the information of the state of the objects and t_i is the prediction time based on this information. The according optimal policy for traversing between two vertices is defined as

$$\pi^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}) : f^{t_i}(\mathbf{x}, t|\boldsymbol{\theta}) \mapsto \tilde{u}^*(\mathbf{v}_i, \mathbf{v}_j, f^{t_i}(\mathbf{x}, t|\boldsymbol{\theta})).$$

For a given distribution f the policy returns the optimal trajectory $\tilde{u}^*$ to get from $\mathbf{v}_i$ to $\mathbf{v}_j$. The resulting expected collision probability is smaller than or equal to the minimum collision probability of all edges

$$P(C|\mathbf{e}_{ij}, \mathbf{b}_{t_k}) \leq \min_{\tilde{u} \in \mathbf{e}_{ij}} P(C|\tilde{u}, \mathbf{b}_{t_k}).$$

This follows from the fact that each edge represents a possible trajectory between the two vertices and, therefore, the safest trajectory can be chosen that depends on the predicted distributions $f^{t_i}(\mathbf{x}, t|\boldsymbol{\theta})$. This is shown by the following Propositions 1 and 2. Therefore, the collision probability of all trajectories $\mathbf{e}_{ij} = \{\tilde{u}_1, \ldots, \tilde{u}_m\}$ is interpreted as a random variable $X_k = P(C|\tilde{u}_k, f(\mathbf{x}, t|\boldsymbol{\theta}))$, depending on the distribution $f(\mathbf{x}, t|\boldsymbol{\theta})$ which is distributed according to $f(\boldsymbol{\theta})$. The collision probability regarding all edges is also a random variable $Y = P(C|\mathbf{e}_{ij}, f(\mathbf{x}, t|\boldsymbol{\theta}))$ which is defined as $Y = \min\{X_1, \ldots, X_m\}$.

Proposition 1. *Let X_1 and X_2 be independent random variables on the support interval $[x_l, x_u]$ with distribution f_i and cumulative distribution function (cdf) F_i, with $i \in \{1, 2\}$. Let $Y = \min\{X_1, X_2\}$, with the minimum distribution $F_Y(x) = 1 - (1 - F_1(x))(1 - F_2(x))$. Then*

$$\mathrm{E}[Y] \leq \mathrm{E}[X_i], \quad i \in \{1, 2\}$$

where E *is the expectation value.*

Proof. The expectation value is computed as

$$\mathrm{E}[Y] = \int_{x_l}^{x_u} \big(f_1(x)(1 - F_2(x)) + f_2(x)(1 - F_1(x))\big) x \, \mathrm{d}x.$$

Thus, with $F_i(x) = \int_{x_l}^{x} f_i(\xi) \, \mathrm{d}\xi$

$$\mathrm{E}[Y] = \underbrace{\int_{x_l}^{x_u} f_1(x) x \, \mathrm{d}x}_{\mathrm{E}[X_1]} - \int_{x_l}^{x_u} \int_{x_l}^{x} f_2(\xi) \, \mathrm{d}\xi \; f_1(x) x \, \mathrm{d}x + \int_{x_l}^{x_u} \int_{x}^{x_u} f_1(\xi) \, \mathrm{d}\xi \; f_2(x) x \, \mathrm{d}x,$$

where the third term can be reformulated by Fubini's Proposition to

$$\int_{x_l}^{x_u} \int_{x_l}^{\xi} f_1(\xi) f_2(x) x \, \mathrm{d}x \, \mathrm{d}\xi = \int_{x_l}^{x_u} \int_{x_l}^{x} f_2(\xi) \xi \, \mathrm{d}\xi \; f_1(x) \, \mathrm{d}x$$

with substituting x by ξ and vice versa. Finally, this leads to

$$\mathrm{E}[Y] = \mathrm{E}[X_1] - \int_{x_l}^{x_u} \int_{x_l}^{x} (x - \xi) f_2(\xi) \,\mathrm{d}\xi \; f_1(x) \,\mathrm{d}x.$$

The statement follows, as the integral over positive functions is again positive. The same holds for $\mathrm{E}[X_2]$ as the indices can be interchanged. □

Proposition 2. *Let X_i be a set of independent random variables with distribution $f_i(x)$ and cdf $F_i(x)$ for $i \in \{1, \ldots, n\}$ and $Y = \min\{X_1, \ldots, X_n\}$. Then*

$$\mathrm{E}[Y] \leq \mathrm{E}[X_i], \qquad \forall i \in \{1, \ldots, n\}.$$

Proof. By complete induction over n.
Base case $n = 2$: Proven by Proposition 1.
Step $n - 1 \to n$: Let $Z = \min\{X_1, \ldots, X_{n-1}\}$ then $Y = \min\{Z, X_n\}$. The statement follows with Proposition 1. □

If edges contain only one trajectory $\mathbf{e}_{ij} = \{\tilde{u}\}$, they can be considered as a special case of the above. Therefore, the collision probability results in

$$P(C|\mathbf{e}_{ij}, \mathbf{b}_{t_k}) = P(C|\tilde{u}(\mathbf{v}_i, \mathbf{v}_j), \mathbf{b}_{t_k}).$$

In contrast to the multi-edge case, the collision probability based on observation time t_k for single edges is the same as the expected collision probability with observation time t_i

$$P(C|\tilde{u}(\mathbf{v}_i, \mathbf{v}_j), \mathbf{b}_{t_k}) = \mathrm{E}_{\boldsymbol{\theta}(t_k)} \left[P(C|\tilde{u}(\mathbf{v}_i, \mathbf{v}_j), \mathbf{b}_{t_i}) \right], \tag{2.3}$$

with $t_i > t_k$. This is shown by Proposition 3.

Proposition 3. *Let the collision probability $P(C|\tilde{u}, \mathbf{b}_{t_i})$ be defined like in Sec. A.2 for a given trajectory $\tilde{u}$ and object distribution $f^{t_i}(\mathbf{p}, t) = \mathbf{b}_{t_i}$ which is defined as a compound distribution like in* (2.1)*. Then*

$$P(C|\tilde{u}, \mathbf{b}_{t_i}) = \mathrm{E}_{\theta_{t_i}}[P(C|\tilde{u}, \mathbf{b}_{t_j})], \quad t_i < t_j.$$

Proof. The expectation value is computed as

$$\begin{aligned}
\mathrm{E}_{t_i}[P(C|\tilde{u}, t_j)] &= \int P(C|\tilde{u}, t_j) f(\boldsymbol{\theta}) \,\mathrm{d}\boldsymbol{\theta} \\
&= \int_{\mathcal{A}^{\mathrm{b}}(\tilde{u}(t))} \int f^{t_j}(\mathbf{p}, t|\boldsymbol{\theta}) \,\mathrm{d}\mathbf{p} \; f(\boldsymbol{\theta}) \,\mathrm{d}\boldsymbol{\theta} \\
&= \int_{\mathcal{A}^{\mathrm{b}}(\tilde{u}(t))} \underbrace{\int f^{t_j}(\mathbf{p}, t|\boldsymbol{\theta}) f(\boldsymbol{\theta}) \,\mathrm{d}\boldsymbol{\theta}}_{f^{t_i}(\mathbf{p}, t)} \,\mathrm{d}\mathbf{p} \\
&= P(C|\tilde{u}, t_i). \qquad \square
\end{aligned}$$

This means, that the observation time of the belief is not changing the collision probability for serial edges. Thus, the initial belief can be used to calculate the collision probability for serial edges. This result seems obvious, since the robot has no other choice than to take this single edge.

2.6.2. Collision Probability for Serial Edges

Consider a vertex $\mathbf{v}_k$ with indegree $\deg^-(\mathbf{v}_k) \geq 1$ and outdegree $\deg^+(\mathbf{v}_k) = 1$. The collision probability for the combination of each ingoing and the outgoing edge can be computed separately, as no decision has to be made at $\mathbf{v}_k$. Therefore, a virtual edge $\mathbf{e}'_{ij}$ is defined which can be parallel to an already existing edge $\mathbf{e}_{ij}$. For further collision probability considerations the new edge $\mathbf{e}_{ij}$ is assumed to be a multi-edge consisting of the old edge $\mathbf{e}_{ij}$ and the virtual edge $\mathbf{e}'_{ij}$. The collision probability for serial edges $\mathbf{e}_{ik}$,$\mathbf{e}_{kj}$ is calculated as

$$P(C|\mathbf{e}'_{ij}, \mathbf{b}_{t'})) = P(C|\mathbf{e}_{ik}, \mathbf{b}_{t'}) + (1 - P(C|\mathbf{e}_{ik}, \mathbf{b}_{t'}))P(C|\mathbf{e}_{kj}, \mathbf{b}_{t'}) \tag{2.4}$$

for all ingoing edges $\mathbf{e}_{ik}$. The resulting collision probability is computed by the combination of the two possible cases: 1) a collision occurs on edge $\mathbf{e}_{ik}$ 2) a collision occurs not on edge $\mathbf{e}_{ik}$ but on $\mathbf{e}_{kj}$.

In the next section, the algorithm for assessing the safety for an entire graph is presented.

2.7. Collision Probability for the Entire Graph

The previous definitions allow one to compute the expected collision probability between serial and multi-edges. In order to assess the safety of a complete graph, an iterative algorithm is used which combines multi-edges and removes vertices with $\deg^- \geq 1$ and $\deg^+ = 1$.

As a preprocessing, the Graph $\mathcal{G}$ is pruned so that it only contains all possible routes from $\mathbf{v}_s$ which can reach $\mathbf{v}_g$. The pruned graph $\mathcal{H}$ is defined as:

Definition 4 (Reachable Subgraph $\mathcal{H}$)**.**

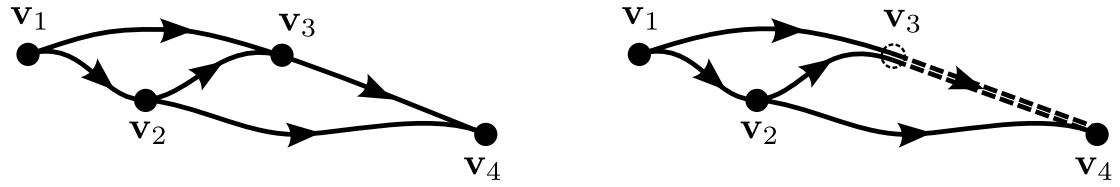
$$\mathcal{H} = \{\mathcal{V}, \mathcal{E}\} \subseteq \mathcal{G} = \{\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}\}$$

where,

$$\mathcal{V} = \{\mathbf{v}_s, \mathbf{v}_g\} \cup \{\mathbf{v}_i \in \mathcal{V}_\mathcal{G} \mid \exists r(\mathbf{v}_s, \mathbf{v}_i), \exists r(\mathbf{v}_i, \mathbf{v}_g)\}$$
$$\mathcal{E} = \{\mathbf{e}_{ij} \in \mathcal{E}_\mathcal{G} \mid \mathbf{v}_i, \mathbf{v}_j \in \mathcal{V}\}$$

In other words, every vertex $\mathbf{v}_i$ can be reached by at least one valid route from $\mathbf{v}_s$ and there exists at least one valid route r to $\mathbf{v}_g$. This pruning can be achieved by using the Dijkstra algorithm and initialize it at the goal vertex.

In the following subsection, a rule is presented that allows removing certain vertices of the graph, allowing one to assess the entire graph with Def. 3 and (2.4).



(a) Vertex $\mathbf{v}_3$ with $\deg^-(\mathbf{v}_3) = 2$, $\deg^+(\mathbf{v}_3) = 1$ is removed.

(b) The dashed lines represent the old edge $\mathbf{e}_{34}$ which was appended to $\mathbf{e}_{13}$ and $\mathbf{e}_{23}$.

Fig. 2.4.: Graph reduction by rule (2.5): vertex $\mathbf{v}_3$ is removed by appending $\mathbf{e}_{34}$ to edge $\mathbf{e}_{13}$ and $\mathbf{e}_{23}$.

2.7.1. Reduction of Vertices with Single Output

In order to reduce the graph $\mathcal{H}$ to a graph only consisting of two vertices and one edge, Def. 3 is used to replace multi-edges by a single edge. Similarly, (2.4) can be applied to remove vertices with $\deg^-(\mathbf{v}_k) \geq 1$ and $\deg^+(\mathbf{v}_k) = 1$ from the graph by attaching the outgoing edge to all input edges. Given a set of vertices $\mathcal{V}^k$ which are all connected to the vertex $\mathbf{v}_k$ which in turn is connected only to the vertex $\mathbf{v}_j$, then the vertex $\mathbf{v}_k$ can be removed by modifying all the edges of the vertices in $\mathcal{V}^k$ going to $\mathbf{v}_k$

$$\begin{aligned} \mathcal{V}^k &= \{\mathbf{v}_i \in \mathcal{V} \mid \mathbf{v}_i \neq \mathbf{v}_k, \mathbf{e}_{ik} \neq \emptyset\} \\ \mathbf{e}'_{ij} &= \{\mathbf{e}_{ik}, \tilde{u}_{kj}\} \cup \mathbf{e}_{ij}, \end{aligned} \tag{2.5}$$

where the original edge $\mathbf{e}_{ij}$ is replaced by the new edge $\mathbf{e}'_{ij}$. Loosely speaking, every vertex with $\deg^+ = 1$ has no influence on the safety assessment, since the future trajectory is predetermined. In Fig. 2.4 a graph is shown where this definition is applied.

Algorithm 1: Graph Reduction

Input : $\{\mathcal{G}, \mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s}\}$
Output : $P(C|\mathcal{G}, \mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s}), \pi^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})$

Initialize: Get $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\} \subseteq \mathcal{G}$ (Def. 4)
while $|\mathcal{V}| > 2$ **do**
  Merge multi-edges $\{\mathbf{e} \in \mathcal{E} \mid |\mathbf{e}| > 1\}$ (Def. 3) and determine $g(\mathbf{e}, \mathbf{b})$ (2.6)
  Find all vertices $\mathcal{V}^g$ which are connected to $\mathbf{v}_g$
  foreach $\mathbf{v}_k \in \mathcal{V}^g$ **do**
    Remove vertices with $\deg^+(\mathbf{v}_k) = 1$ (2.5) and
    determine $g(\mathbf{e}_{ig}, \mathbf{b}_{t_l})$ (2.7)
$P(C|\mathcal{G}, \mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s}) = P(C|\mathbf{e}_{sg}, \mathbf{b}_{t_s})$, with $|\mathbf{e}_{sg}| = 1$
$\pi^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s}) \leftarrow g(\mathbf{e}_{sg}, \mathbf{b}_{t_s})$
return $P(C|\mathcal{G}, \mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s}), \pi^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})$

2.7.2. Graph Reduction

After obtaining the reduced Graph $\mathcal{H}$, the iterative Alg. 1 is applied to it. For every pair of vertices connected by multi-edges in $\mathcal{H}$, Def. 3 is applied to determine the function

$$g(\mathbf{e}_{ij}, \mathbf{b}_{t_k}) \rightarrow [\pi^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}), P(C|\mathbf{e}_{ij}, \mathbf{b}_{t_k})] \tag{2.6}$$

which identifies the optimal policy π^* for a given observation time t_i and the collision probability between the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ for a given t_k. Note, that π^* and the collision probability can depend on different observation times.

After merging all multi-edges in $\mathcal{H}$, rule (2.5) is applied to all vertices $\mathbf{v}_k$ with $\deg^-(\mathbf{v}_k) \geq 1$ and $\deg^+(\mathbf{v}_k) = 1$ which are connected to the goal vertex $\mathbf{v}_g$. Therefore, the two functions $g(\mathbf{e}_{ik}, \mathbf{b}_{t_l})$ and $g(\mathbf{e}_{kj}, \mathbf{b}_{t_l})$ need to be merged by

$$\begin{aligned} g(\mathbf{e}_{ik}, \mathbf{b}_{t_l}) \times g(\mathbf{e}_{kj}, \mathbf{b}_{t_l}) &\rightarrow g(\mathbf{e}_{ij}, \mathbf{b}_{t_l}) \\ \pi^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}) &= \{\pi^*(\mathbf{v}_i, \mathbf{v}_k, \mathbf{b}_{t_i}), \pi^*(\mathbf{v}_k, \mathbf{v}_j, \mathbf{b}_{t_k})\} \\ P(C|\mathbf{e}_{ij}, \mathbf{b}_{t_l}) &= P(C|\mathbf{e}_{ik}, \mathbf{b}_{t_l}) + (1 - P(C|\mathbf{e}_{ik}, \mathbf{b}_{t_l}))P(C|\mathbf{e}_{kj}, \mathbf{b}_{t_l}) \end{aligned} \tag{2.7}$$

where $\{\pi, \pi'\}$ is the concatenation of two policies.

The operating principle of the single steps are sketched in Fig. 2.5. The result of the algorithm is a graph which contains only the start vertex, the goal vertex and its edge. The edge represents the function $g(\mathbf{e}_{sg}, \mathbf{b}_{t_s})$ allowing one to calculate the expected collision probability of the entire graph $P(C|\mathbf{e}_{sg}, \mathbf{b}_{t_s})$ and to determine the associated optimal policy $\pi^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})$.

The following proposition shows that the graph reduction algorithm always converges in finite time.

Proposition 4. *Let $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$ be a graph according to Def. 4 with $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$. Then Alg. 1 converges after at most $n + m$ steps.*

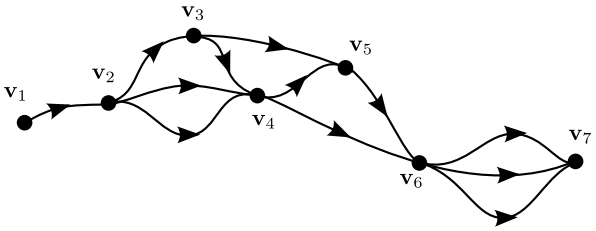
Proof of Proposition 4. In every step of our algorithm the number of vertices or edges is reduced by at least one. So there can be $n + m$ steps at most. In the following the convergence is proven by contradiction. Assume that the algorithm has not converged, so there has to be nodes

$$\mathcal{V}^+ = \{\mathbf{v} \in \mathcal{V}/\{\mathbf{v}_s, \mathbf{v}_g\} \mid \deg^+(\mathbf{v}) > 1\},$$

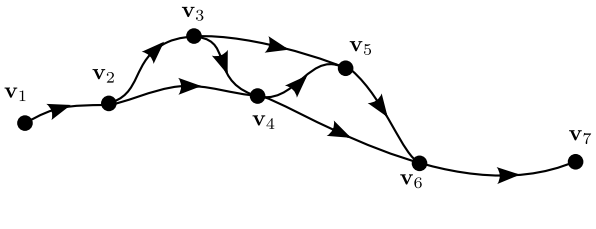
because if $\deg^+(\mathbf{v}) = 1$ the node would have been removed from the graph in any step by rule (2.5). Choose $\mathbf{v}_i$ as the latest such node, i.e. $t_i > t, \forall t \in \mathbf{v} \in \mathcal{V}^+$. The outgoing edges cannot be multi-edges because they would have been removed in any step. Thus, there has to be an edge $\mathbf{e}_{ij}$ to $\mathbf{v}_j \neq \mathbf{v}_g$, with $t_j > t_i$. As $\mathbf{v}_j$ was not removed in a previous step, $\deg^+(\mathbf{v}_j) > 1$ must hold. Therefore, $\mathbf{v}_j$ was not the latest node with $\deg^+ > 1$. □

2.8. Implementations

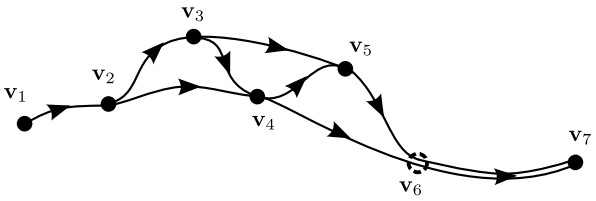
In this section two example implementations are presented for the safety assessment approach presented in Sec. 2.7. The implementation for mobile robot applications uses Gaussian distributions for representing the uncertain states of the objects and constructs the roadmaps based on



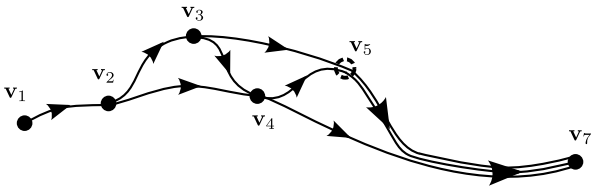
(a) Example graph with multi-edges.



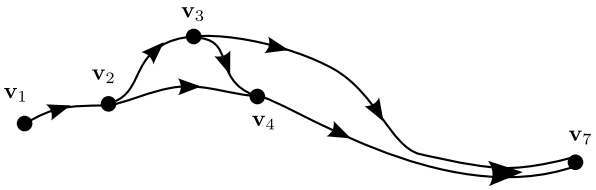
(b) Merged edges $\mathbf{e}_{24}$ and $\mathbf{e}_{67}$ (Def. 3).



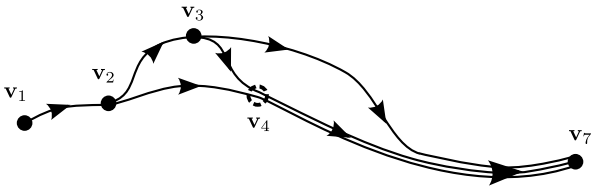
(c) Removed vertex $\mathbf{v}_6$ (rule (2.5)).



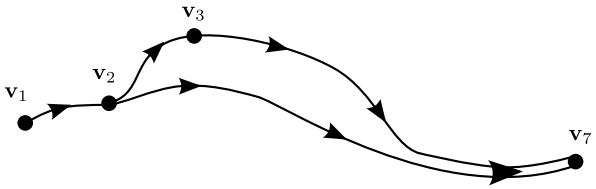
(d) Removed vertex $\mathbf{v}_5$ (rule (2.5)).



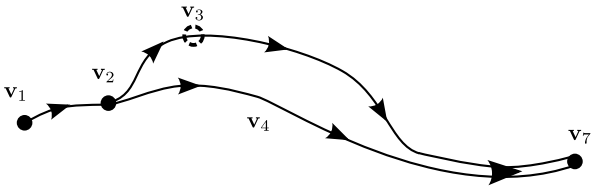
(e) Merged edge $\mathbf{e}_{47}$ (Def. 3).



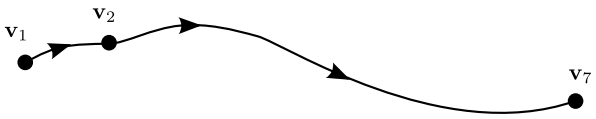
(f) Removed vertex $\mathbf{v}_4$ (rule (2.5)).



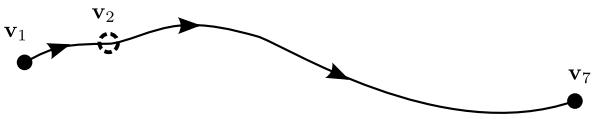
(g) Merged edge $\mathbf{e}_{37}$ (Def. 3).



(h) Removed vertex $\mathbf{v}_3$ (rule (2.5)).



(i) Merged edge $\mathbf{e}_{27}$ (Def. 3).



(j) Removed vertex $\mathbf{v}_2$ (rule (2.5))

Fig. 2.5.: Figures depicting single steps of Alg. 1 for an example graph, until the graph contains only the start and goal vertex connected with one edge. In order to show the result of rule (2.5), the appended edges are drawn in parallel to the existing edges.

Bézier curves. Gaussian distributions are widely used in mobile robots e.g. representing localization error or representing the uncertain position of objects [113]. For generating roadmaps, many different approaches are available in literature, all having their own advantages for solving the motion planning problem (e.g. [5, 40]). However, our focus is different, since it should be possible to directly modify the shape of the trajectories for generating multi-edges. Additionally, the trajectories should be smooth, i.e. continuous in position, direction, and velocity. Hence, an implementation based on Bézier curves in combination with a triangular velocity profile is used.

The other implementation for automotive applications uses motion patterns for representing the uncertain future motion of other vehicles and an optimal control approach for constructing the roadmap. Motion patterns, also called motion hypotheses, are a common representation for motion prediction in structured environments such as indoor environments or roadways. In [13] motion pattern are used to represent the future motions of humans and in [51] the problem of long term vehicle prediction is addressed with prerecorded trajectories of vehicles. It is also possible to use e.g. Gaussian distributions [13] or Gaussian processes [71] to represent a motion pattern. In literature many approaches are available for generating trajectories for autonomous navigation of vehicles on roadways. Approaches using numeric optimization are not directly applicable for constructing roadmaps since they do not succeed in generating trajectories ending in a certain state after a predefined time. For this implementation an optimal control approach [125] is used which generates minimum jerk trajectories in the lane coordinates of the road by using quintic polynomials.

In the following, an approximation technique for the future belief of the workspace objects is presented which is used in both implementations.

2.8.1. Estimation of Collision Probabilities for the Entire Graph

The initial belief contains one distribution for each object representing its uncertain state, but due to its prediction based on compound distributions (2.1) the future belief of each object contains an infinite set of parameterized distributions. For implementing Alg. 1, it is necessary to estimate the collision probability for multi-edges according to Def. 3. The definition is not directly implementable, since for each parameterized distribution of the belief the trajectory with the lowest collision probability $\tilde{u}^*$ needs to be determined. To overcome this problem, the belief of the object is represented by a finite set of sampled distributions from the compound distribution. This idea is similar to the belief tree known from POMDP planning [62]. The root of the tree is the initial belief $\mathbf{b}_{t_0}$ and for every new time point the belief is propagated according to its compound distribution, whereby the compound distribution is represented by a finite set of sampled distributions. The propagation rule for the belief tree is presented for one object $\mathbf{b}_{t_0} = \{f^{t_0}\}$. Importance sampling is used to draw samples from the parameter distribution $f(\boldsymbol{\theta}(f^{t_{k-1}})))$ depending on the distribution from the previous time step t_{k-1}. Each sampled parameter vector results in one sampled distribution f_s, where s denotes the number of the sample. The samples are generated according to

$$\boldsymbol{\theta} \sim f\big(\boldsymbol{\theta}(f^{t_{k-1}}(\mathbf{x}, t_k))\big) \rightarrow f^{t_k}(\mathbf{x}, t_k|\boldsymbol{\theta})$$
$$\mathbf{b}^{t_k}_{t_{k-1}}(f^{t_{k-1}}) = \{f^{t_k}_s(\mathbf{x}, t_k|\boldsymbol{\theta}_s)|s = (n-1)N_\mathrm{s}(k)+1, \ldots, nN_\mathrm{s}(k))\}$$

Tab. 2.1.: Sampling tree for four time steps.

t_0	t_1	t_2	t_3
f^{t_0}	$\mathbf{b}_{t_0}^{t_1}(f^{t_0}) = \{f_s^{t_1} \vert s = 1, \dots, N_\mathrm{s}\}$	$\mathbf{b}_{t_1}^{t_2}(f_1^{t_1}) = \{f_s^{t_2} \vert s = 1, \dots, N_\mathrm{s}\}$	$\mathbf{b}_{t_2}^{t_3}(f_1^{t_2}) = \{f_s^{t_3} \vert s = 1, \dots, N_\mathrm{s}\}$
		$\mathbf{b}_{t_1}^{t_2}(f_2^{t_1}) = \{f_s^{t_2} \vert s = N_\mathrm{s}+1, \dots, 2N_\mathrm{s}\}$	$\mathbf{b}_{t_2}^{t_3}(f_2^{t_2}) = \{f_s^{t_3} \vert s = N_\mathrm{s}+1, \dots, 2N_\mathrm{s}\}$
		$\vdots$	$\vdots$
		$\mathbf{b}_{t_1}^{t_2}(f_{N_\mathrm{s}}^{t_1}) = \{f_s^{t_2} \vert s = {N_\mathrm{s}}^2 - N_\mathrm{s}+1, \dots, {N_\mathrm{s}}^2\}$	$\mathbf{b}_{t_2}^{t_3}(f_{N_\mathrm{s}}^{t_2}) = \{f_s^{t_3} \vert s = {N_\mathrm{s}}^2 - N_\mathrm{s}+1, \dots, {N_\mathrm{s}}^2\}$
			$\vdots$
			$\mathbf{b}_{t_2}^{t_3}(f_{{N_\mathrm{s}}^2}^{t_2}) = \{f_s^{t_3} \vert s = {N_\mathrm{s}}^3 - N_\mathrm{s}+1, \dots, {N_\mathrm{s}}^3\}$

where $N_\mathrm{s}(k)$ is the number of samples of $\mathbf{b}_{t_{k-1}}^{t_k}$ and n denotes the level of the belief tree. The subscript of $\mathbf{b}_{t_{k-1}}^{t_k}$ denotes the observation time of the sampled distribution $f^{t_{k-1}}$ and the superscript denotes the observation time of the samples. The resulting belief tree for four time steps is shown in Tab. 2.1. One can see, that the information of the object is represented by multiple beliefs, all depending on a sampled distribution from a belief of an earlier time step. It is crucial, that the sampling is not performed independently for each belief, since for the rules described in (2.6) and (2.7) it is necessary that each sample can be traced back till the initial belief. It is noted, that the number of samples depends on the kth time step, in order to adapt the number of samples to the uncertainty of the sampled distribution. The number of all samples needed for a complete graph is

$$\sum_{k=1}^{N_\mathrm{t}-1} N_\mathrm{b}(N_\mathrm{s}(k))^k,$$

where N_b is the number of objects and N_t is the number of time steps.

The belief tree allows one to approximate the collision probability according to Def. 3 by

$$P(C|\mathbf{e}_{ij}, \mathbf{b}_{t_k}) \approx \hat{P}(C|\mathbf{e}_{ij}, \mathbf{b}_{t_k}) := \frac{1}{N_\mathrm{s}} \sum_{s=1}^{N_\mathrm{s}} P\big(C|\tilde{u}_s^*(\mathbf{v}_i, \mathbf{v}_j, f_s^{t_i}), t_k\big), \quad f_s^{t_i} \in \mathbf{b}_{t_k}^{t_i},$$

where N_s is the total number of samples and $\tilde{u}_s^*$ is the trajectory with the lowest collision probability regarding f_s. For estimating the collision probability of $\tilde{u}_s^*$, the collision probability for all possible trajectories is evaluated

$$P\big(C|\tilde{u}_s^*(\mathbf{v}_i, \mathbf{v}_j, f_s^{t_i}), t_k\big) = \min_{\tilde{u}_s \in \mathbf{e}_{ij}} P\big(C|\tilde{u}_s(\mathbf{v}_i, \mathbf{v}_j, f_s^{t_i}), t_k\big).$$

The estimations of $\hat{P}\big(C|\tilde{u}_s^*, \mathbf{b}_{t_k}\big)$ and $\hat{P}(C|\mathbf{e}_{ij}, \mathbf{b}_{t_k})$ allows one to implement Alg. 1. The environment model based on Gaussian distributions is presented in the next subsection.

2.8.2. Environment Model for Mobile Robot Applications

First an example implementation for the environment model is shown which assumes a Gaussian distribution $\mathcal{N}$ for the state of each object at a certain time point

$$\mathbf{x}(t) \sim \mathcal{N}(\boldsymbol{\mu}(t), \boldsymbol{\Sigma}(t)) = f(\mathbf{x}, t),$$

where $\boldsymbol{\mu}(t)$ is the predicted state of the object and $\boldsymbol{\Sigma}(t)$ the covariance matrix representing the associated uncertainty. For implementing the environment model described in Sec. 2.4.3, the compound distribution according to (2.1) and the prediction for the time intervals need to be specified. For predicting the future states of the objects, the constant velocity model is used. The position is denoted by $\mathbf{p} = [x, y]^T$ and the velocity as $\boldsymbol{v} = [v_x, v_y]^T$. After a time discretization, where $t_k = kT$, $k \in \mathbb{N}^+$ is a time step and $T \in \mathbb{R}^+$ is the step size, the dynamic model is transformed into the discrete time form

$$\underbrace{\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}(t_{k+1})}_{\mathbf{x}(t_{k+1})} = \underbrace{\begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}(t_k)}_{\mathbf{x}(t_k)}, \quad t_k = kT.$$

The initial state has the multivariate Gaussian distribution $\mathbf{x}(t_0) \sim \mathcal{N}(\boldsymbol{\mu}(t_0), \boldsymbol{\Sigma}(t_0))$. With the multiplication rule and the addition rule for independent random variables with Gaussian distribution the mean value and covariance of the state $\mathbf{x}$ can be updated as

$$\begin{aligned} \boldsymbol{\mu}(t_{k+1}) &= \mathbf{A}\boldsymbol{\mu}(t_k) \\ \boldsymbol{\Sigma}(t_{k+1}) &= \mathbf{A}\boldsymbol{\Sigma}(t_k)\mathbf{A}^T \end{aligned}$$

for every time point. This update rule allows one to predict the workspace objects for a certain time interval.

Additionally, the compound distribution according to (2.1) needs to be specified. It is assumed that at each vertex all objects are detected by the robot perception system and that the noise in the estimation of their state is constant. The distribution of the object is updated at every vertex, the updated mean is denoted by $\boldsymbol{\mu}^+$ and the covariance as $\boldsymbol{\Sigma}^+$. The parameter vector is the mean value $\boldsymbol{\theta} = \boldsymbol{\mu}^+(t_i)$ which is distributed according to $f(\boldsymbol{\mu}^+) = \mathcal{N}\big(\boldsymbol{\mu}(t_i), \boldsymbol{\Sigma}(t_i) - \boldsymbol{\Sigma}(t_0)\big)$ and the parameterized distribution is $f(\mathbf{x}, t|\boldsymbol{\mu}^+) = \mathcal{N}(\boldsymbol{\mu}^+, \boldsymbol{\Sigma}^+)$. The updated Gaussian distribution representing the state of the object is specified by

$$\begin{aligned} \boldsymbol{\mu}^+(t_i) &\sim \mathcal{N}\big(\boldsymbol{\mu}(t_i), \boldsymbol{\Sigma}(t_i) - \boldsymbol{\Sigma}(t_0)\big) \\ \boldsymbol{\Sigma}^+(t_i) &= \boldsymbol{\Sigma}(t_0). \end{aligned}$$

This means the new prediction depends on the estimated prediction at the current time and the covariance $\boldsymbol{\Sigma}^+(t_i)$ will be set to the initial covariance $\boldsymbol{\Sigma}(t_0)$ assuming that the uncertainty arises only from measurement noise which is assumed to be constant. The proof that this parameterized distribution satisfies (2.1) is given with the following Propositions.

Proposition 5. *Let X be a random variable of a one dimensional Gaussian compound distribution*

$$f(X) = \mathrm{E}_\theta[f(X|\theta)] = \int f(X|\theta) f(\theta)\, \mathrm{d}\theta.$$

with the mean value μ_c and variance σ_c^2

$$X \sim \mathcal{N}(\mu_c, \sigma_c^2).$$

Then a valid parameterized distribution $f(X|\theta)$ is a Gaussian distribution $\mathcal{N}(\mu_p, \sigma_p)$ with the variance σ_p and the mean value as the parameter $\mu_p = \theta$ distributed according to $f(\mu_p)$ with

$$\mu_p \sim \mathcal{N}(\mu_c, \sigma_c^2 - \sigma_p^2).$$

Proof. It will be shown, that the Gaussian compound distribution has the expected mean and variance. The variance of the compound distribution can be determined as

$$\mathrm{Var}[X] = \int ((\mu_p - \mu_c)^2 + \sigma_p^2) f(\mu_p)\,\mathrm{d}\mu_p$$

by the law of total variance. Since the covariance of the parameterized distribution is constant $\sigma_p^2 = \mathrm{const}$ this can be rewritten as

$$\mathrm{Var}[X] = \underbrace{\int f(\mu_p)\sigma_p^2\,\mathrm{d}\theta}_{\sigma_p^2} + \underbrace{\int f(\mu_p)(\mu_p - \mu_c)^2\,\mathrm{d}\theta}_{\mathrm{Var}(f(\mu_p))}$$

with the variance of the compound distributions being $\sigma_p^2 + (\sigma_c^2 - \sigma_p^2)$.

The expectation of the compound distribution is defined as

$$\mathrm{E}[X] = \int \mathrm{E}[f(X|\theta)] f(\theta)\,\mathrm{d}\theta$$

according to the law of total expectation. Since $\mathrm{E}[f(X|\theta)] = \mu_p \sim \mathcal{N}(\mu_c, \sigma_c^2 - \sigma_p^2)$ one can see that $\mathrm{E}[X] = \mu_c$. □

Proposition 6. *Let the Gaussian compound distribution $f(\mathbf{X})$ be defined like in* (2.1) *with the mean value $\boldsymbol{\mu}_c$ and variance $\boldsymbol{\Sigma}_c$, $\mathbf{X} = [X_1, X_2] \sim \mathcal{N}_2(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$. Then a valid parameterized distribution $f(\mathbf{X}|\boldsymbol{\theta})$ is a Gaussian distribution $\mathcal{N}_2(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ with the mean value $\boldsymbol{\mu}_p \sim f(\boldsymbol{\theta}) = \mathcal{N}_2(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c - \boldsymbol{\Sigma}_p)$ and the variance $\boldsymbol{\Sigma}_p$.*

Proof. The Covariance matrix of the compound distribution is defined as

$$\boldsymbol{\Sigma}_c = \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] \end{bmatrix}.$$

Applying Proposition 5 to each element will show

$$\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}_p + (\boldsymbol{\Sigma}_c - \boldsymbol{\Sigma}_p).$$

□

This environment model is illustrated with one object in Fig. 2.3. The presented environment model is a very simplified one. However, it can be easily exchanged by a more sophisticated model considering e.g. distance to obstacles, sensor model and occlusion of obstacles.

In the following the estimation of the collision probability for one edge is presented. The method as presented in Sec. A.2 is used for estimating the collision probability for a single trajectory. Therefore the occupancy $\mathcal{A}$ of the robot system is enlarged by the Minkowski addition

of the area $(-\mathcal{B}_i + \mathbf{c}_i)$ to $\mathcal{A}$, so the new occupancy of the robot is $\mathcal{A}^{b_i} = \mathcal{A} \oplus (-\mathcal{B}_i + \mathbf{c}_i)$. The point $\mathbf{c}_i$ is the reference point of the object. The probability of the robot system $\mathcal{A}$, applying the input trajectory $\tilde{u}$ having a collision C with another object $\mathcal{B}_i$ at time t is defined as

$$P_i(C|\tilde{u}(t), \mathbf{b}_{t'}) = P(\mathcal{A}(\tilde{u}(t)) \cap \mathcal{B}_i \neq \emptyset | \mathbf{b}_{t'}) = \int\limits_{\mathcal{A}^{b_i}(\tilde{u}(t))} f_i^{t'}(\mathbf{p}, t)\, \mathrm{d}\mathbf{p},$$

where the index i of the collision probability refers to the ith workspace object and t' is the observation time. This integral is approximated by discretizing the workspace and the time dimension. This is done by generating an occupancy grid with equidistant segmentation.

2.8.3. Environment Model for Automotive Applications

For this implementation, the future motion of the objects is represented by motion patterns. The set of motion patterns for the ith object is denoted by

$$\mathcal{M}_i = \{\tilde{m}_{i,1}, \tilde{m}_{i,2}, \ldots, \tilde{m}_{i,N_\mathrm{m}}\} \quad |M_i| = N_\mathrm{m},$$

where each trajectory $\tilde{m}$ is a motion pattern. These trajectories have to fulfill the kinematic and dynamic constraints of the object. In order to represent the uncertainty of the state of the object a discrete probability function also known as the probability mass function (PMF) is used. The PMF relates the value of a discrete random variable to its probability. In the case of motion patterns, the discrete random variables are the states of the motion patterns and the value of the PMF is the probability that this pattern will be executed by the object. For each motion pattern a weight w exists representing the probability of the associated motion pattern. The PMF is denoted by

$$q(\mathbf{x}, t) = \begin{cases} w_1, & \mathbf{x} = \tilde{m}_1(t), \\ w_2, & \mathbf{x} = \tilde{m}_2(t), \\ & \vdots \\ w_{N_\mathrm{m}}, & \mathbf{x} = \tilde{m}_{N_\mathrm{m}}(t), \end{cases}$$

with $\sum w_i = 1$. For this environmental model there is no need for a motion prediction algorithm, since the prediction of the future states of the objects are already included in the motion patterns. For modeling the possible future distributions of the objects at the vertices, the compound distribution according to (2.1) is defined as

$$q^{t_i}(\mathbf{x}, t) = \int q^{t_j}(\mathbf{x}, t|\mathbf{w}) f(\mathbf{w}(t_i))\, \mathrm{d}\mathbf{w}, \quad t_i < t_j \leq t, \tag{2.8}$$

where $\boldsymbol{\theta} = \mathbf{w} = \{w_1, \ldots, w_{N_\mathrm{m}}\}$ is the parameter vector containing all weights and the parameterized distribution $q^{t_j}(\mathbf{x}, t|\mathbf{w})$ is the same PMF as $q^{t_i}(\mathbf{x}, t)$ only with another weight vector. The motion patterns cannot be changes, since this would lead to another compound distribution. For this implementation, the distribution $f(\mathbf{w}(t_i))$ is modeled by simulated future measurements at the time of the vertices. Since measurements are never noise free, a Gaussian measurement model is used. The measurement is generated according to a Gaussian which mean is the actual position of the vehicle according to the motion pattern and the covariance is assumed to be

constant. The future measurements are distributed according to

$$\mathbf{z}(t_j) \sim \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_z), \quad \boldsymbol{\mu}_s \sim q^{t_i}, \quad t_i < t_j$$

where $\boldsymbol{\Sigma}_z$ is the covariance of the measurement model and $\mathbf{z}(t_j)$ is one possible measurement at time t_j based on the pmf q^{t_i} with the observation time t_i. The simulated measurements are used to generate the novel weight vectors $\hat{\mathbf{w}}(t_j)$. The update process is very similar to a particle filter approach [28] also called sequential Monte Carlo method for tracking objects. Since the focus of this work is not on a realistic simulation of sensor or tracking models, a very simplified update step is used. The weights of the weight vector are updated according to the measurement $\mathbf{z}$ by the Euclidean distance of the expected position at the motion pattern and the measured position $\hat{w}_i = \|\tilde{m}_i(t) - \mathbf{z}(t)\|$ which requires the normalization of all weights

$$w_i = \frac{\hat{w}_i}{\sum \hat{w}_i}, \quad w_i \in \hat{\mathbf{w}}.$$

It is noted that this update step can be easily exchanged by a more sophisticated one e.g. by considering measurement noise and using an appropriate distance metric.

For generating the belief tree according to Sec. 2.8.1 a finite number of weight vectors $\hat{\mathbf{w}}$ are generated by simulating a finite number of measurements. However, the generated weight vectors do not fulfill (2.8), since the expectation of the updated weight vectors $\hat{\mathbf{w}}(t_j)$ is not equal to the weight vector $\mathbf{w}(t_i)$. This is done by performing a second normalization step for all generated weight vectors. The ith weight vector is normalized by

$$\mathbf{w}_i(t_j) = \hat{\mathbf{w}}_i(t_j) \frac{\mathbf{w}(t_i)}{\mathrm{E}[\hat{\mathbf{w}}(t_j)]},$$

where $\mathrm{E}[\hat{\mathbf{w}}(t_j)]$ is the expectation of all generated weight vectors. The environmental model including the update steps is illustrated in Fig. 2.6 for one propagated sample.

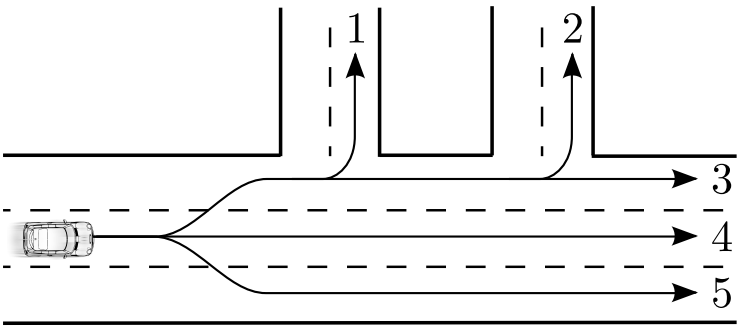
In the following a Monte Carlo based approach as described in Sec. A.3 is used to estimate the collision probabilities based on the motion patterns of the objects. The principle of motion patterns can be seen as a specific sampling strategy for the Monte Carlo approach. Thus the probability of the robot to collide with the ith object while navigating between two vertices can be estimated by the weighted sample estimator ([100])

$$\hat{P}_i(C|\tilde{u}(\mathbf{v}_j, \mathbf{v}_k), \mathbf{b}_{t'}) = \sum_{n=1}^{N_\mathrm{m}} \mathrm{Ind}(C|\tilde{u}(\mathbf{v}_j, \mathbf{v}_k), \tilde{m}_{i,n}) w_n(t'),$$

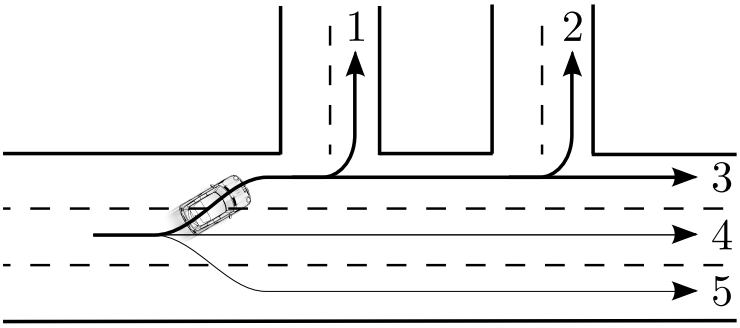
with $\sum_{n=1}^{N_\mathrm{m}} w_n(t') = 1$ and the indicator function

$$\mathrm{Ind}(C|\tilde{u}(\mathbf{v}_j, \mathbf{v}_k), \tilde{m}_{i,n}) = \begin{cases} 1, & \exists t \in [t_j, t_k]\ \mathcal{A}(\tilde{u}(t)) \cap \mathcal{B}_i(\tilde{m}_i(t)) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases}$$

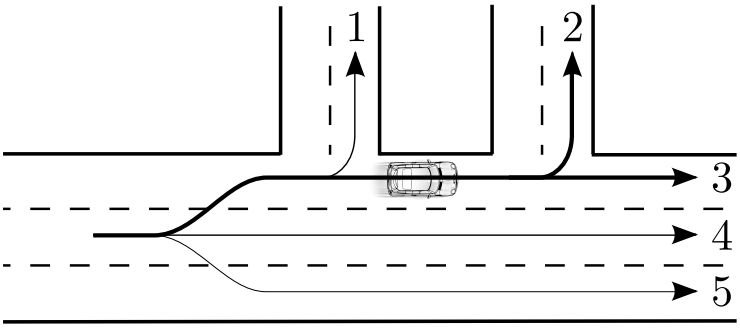
The indicator function is one, if a collision between the robot trajectory $\tilde{u}$ and the nth trajectory $\tilde{m}_{n,i}$ of the ith object occurs and is zero otherwise. In other words, the probability that the



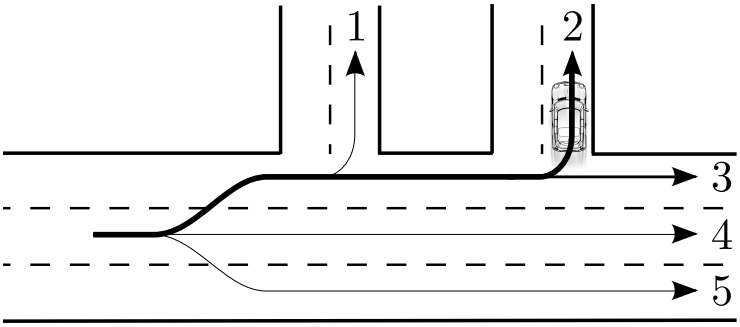
(a) The weights of the motion patterns are equally initialized.



(b) It is likely that the car follows motion pattern $1, 2$ or 3.



(c) It is likely that the car follows motion pattern 2 or 3.



(d) It is likely that the car follows motion pattern 2.

Fig. 2.6.: The principle of the environment model for motion patterns is sketched for some time steps. There are 5 possible motion pattern and the weights of the patterns are illustrated by the line thickness.

robot trajectory $\tilde{u}$ will collide with the object, is the sum of the weights w_n of all trajectories of the object leading to a collision. For detecting collisions, the states of the ego vehicle and the other object are sampled along their trajectory according to a fixed time discretization. If their geometric shapes overlap in at least one time point their trajectories are classified as colliding trajectories. In this work, rectangles are used to approximate the shape of the vehicles and used the OBB-tree algorithm [42] for collision detection. It is noted, that compared to the occupancy grid approach, this method does not discretize the workspace.

In the next section, the implementation for generating roadmaps for mobile robot applications is presented.

2.8.4. Roadmap for Mobile Robot

Fifth-order Bézier curves $B(\lambda), \lambda \in [0, 1]$ and a triangular velocity profile are used for generating roadmaps for mobile robot applications. The path $B(\lambda)$ between two vertices $\mathbf{v}_i, \mathbf{v}_j$ is defined by control points $P_k, k = 0, \ldots, 4$, with $B(0) = P_0$ and $B(1) = P_4$. The first two control points are used to define the start position and its initial direction $[x_i, y_i, \theta_i]$, the last two control points are used to define the goal position and direction $[x_j, y_j, \theta_j]$. The third control point allows one to vary the shape of the Bézier curve $B(\lambda)$.

$$P_0 = [x_i, y_i], \quad P_1 = P_0 + \frac{\|\overrightarrow{P_0P_4}\|}{4}[\cos(\theta_i), \sin(\theta_i)]$$

$$P_4 = [x_j, y_j], \quad P_3 = P_4 - \frac{\|\overrightarrow{P_0P_4}\|}{4}[\cos(\theta_j), \sin(\theta_j)]$$

The control point P_2 is set

$$P_2 = P_0 + \frac{\overrightarrow{P_0P_4}}{2} + \alpha\|\overrightarrow{P_0P_4}\|\mathbf{n},$$

where $\mathbf{n} \perp \overrightarrow{P_0P_4}$ and $\alpha \in [-1, 1]$ is the parameter determining the shape of the path. In Fig. 2.7 an example path is illustrated.

The velocity profile $v(t)$ is defined by two linear velocity sections and the absolute velocity of the robot at the ith vertex is denoted as $v_i = \|\boldsymbol{v}_i\|$.

$$v(t, v') = \begin{cases} v_i + \dfrac{v' - v_i}{T_{\tilde{u}}}(t - t_i), & t_i \le t < t_i + T_{\tilde{u}} \\ v' + \dfrac{v_j - v'}{T_{\tilde{u}}}(t - t_i - T_{\tilde{u}}), & t_i + T_{\tilde{u}} \le t \le t_j \end{cases}$$

where $T_{\tilde{u}} = \frac{t_j - t_i}{2}$ is the desired time duration of the trajectory. The via velocity $v' = v(t_i + T_{\tilde{u}})$ is defined such that

$$\int_{t_i}^{t_j} v(t, v')\,\mathrm{d}t = \int_0^1 \|\nabla B(\lambda)\|\,\mathrm{d}\lambda =: l(B) = l(\tilde{u}),$$

where $l(B)$ is the length of the Bézier curve and $\| \bullet \|$ is the Euclidean norm.

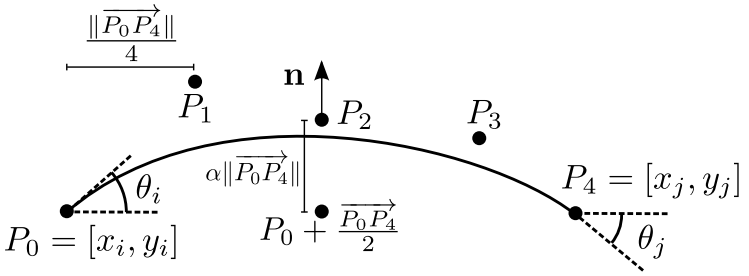


Fig. 2.7.: Generation of one possible Bézier curve between the vertices v_i and v_j for one α.

2.8.5. Roadmap for Automotive Scenario

An optimal control approach based on a combined unconstrained optimization of the lateral and longitudinal movements in street-relative coordinates, so-called Frenet frame [21], with terminal state sets is used to generate the roadmaps. The shape of the trajectories for lateral and longitudinal trajectories can be changed by varying the desired time to reach the specified end constraints. The end constraints are position, velocity and acceleration of the associate longitudinal or lateral coordinate. The trajectories are checked for kinematic and dynamic constraints in a second step. Typical constraints of a vehicle are e.g. limited acceleration, maximum velocity, maximum curvature. This approach has been successfully evaluated in experiments with the autonomous vehicle platform Junior [80]. In [137] an extension of this approach is presented addressing former limitations concerning the safety assessment of the trajectories. For more information to the trajectory generation approach, the reader is referred to Sec. 5.3 and [124, 125]

In order to generate roadmaps, the vertices needs to be specified and the edges are generated with the optimal control approach mentioned above. One possibility is to learn or optimize the states of the vertices in order to achieve a good coverage of the possible trajectories with respect to the present road course. The roadmap should include trajectories for e.g. overtaking, lane changes and velocity keeping. In this work the focus is on the safety assessment of roadmaps and not on its creation process. Thus, in this work, the vertices are manually set on the center of the different lanes with equal spaces to allow lane changes to all available lanes. The desired time to reach the vertices is also constant for all trajectories, so no multi-edges are generated in the roadmap. They appear during the graph reduction algorithm (Alg. 1). An example roadmap is illustrated in Fig. 2.11b.

2.9. Simulations

In this section the safety assessment concept from Sec. 2.7 is evaluated based on the implementations presented in Sec. 2.8. Therefore, the safety assessment concept is applied to mobile robot scenarios as well as to automotive scenarios. It is evaluated by comparison with the common approach determining the route with the smallest collision probability in the graph. In contrast to the proposed graph assessment approach, this method does not consider replanning possibilities. In the following, the common assessment is explained in more detail.

2.9.1. Determining the Safest Route

The simulation results of the proposed graph assessment approach are compared to the safety assessment approach identifying the route r^* with the lowest collision probability starting at $\mathbf{v}_s$ and ending at $\mathbf{v}_g$. This corresponds to the minimum collision trajectory $\tilde{u}^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})$ according to Def. 1. It is determined by estimating the collision probability for all possible routes from the start to the goal vertex. Since the considered graphs do not allow loops (robot cannot reach a vertex more than once at the same time if standing still is excluded), the set of all possible routes is finite. All objects are predicted from t_s till t_g by using the constant velocity model. Based on this prediction, the collision probabilities of all edges are estimated allowing one to calculate the collision probabilities of all routes.

Approaches such as the minimum collision cost planner [86] are similar, but not identical. This approach applies the A* algorithm to the graph, where the weights of the edges represent the collision probability. Since the A* algorithm expects a additive cost function, the cost of a route is the sum of the weighted edges, thus the approach determines the route with the minimum sum of collision probabilities. This solution may be also the route with the minimum collision probability, but there can be also a clear difference. One possibility to determine the route with the minimum collision probability is to apply the more general best-first search algorithm [27] which does not suffer from the additive restriction on the evaluation function. Since the graphs in this work contain not more than 9 vertices, a simple brute-force algorithm is used to find all possible routes which collision probability is calculated in a second step allowing one to determine the route with the minimum collision probability.

2.9.2. Mobile Robot Applications

In this section the evaluation of the proposed graph assessment approach is shown. Therefore, some scenarios related to mobile robot navigation in uncertain and dynamic environments are simulated. The example implementations presented in Sec. 2.8 are used. The robot and the objects have circular shape with a radius of $0.25\,\mathrm{m}$. For the following simulation scenarios, an occupancy grid with a cell size of $0.1\,\mathrm{m} \times 0.1\,\mathrm{m}$ and a time resolution of $0.1\,\mathrm{s}$ was used for estimating the collision probabilities of the single edges.

Assessment of Multi-edges

The simulation environment consists of one robot system and one dynamic object represented by a Gaussian PDF. The generated graph (Fig. 2.8) contains the three vertices

$$\mathbf{v}_0 = [0\,\mathrm{m}, 2\,\mathrm{m}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,\mathrm{s}] \quad \mathbf{v}_1 = [3\,\mathrm{m}, 2\,\mathrm{m}, 3\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 3\,\mathrm{s}]$$
$$\mathbf{v}_2 = [6\,\mathrm{m}, 2\,\mathrm{m}, 3\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 4\,\mathrm{s}]$$

with $\mathbf{v}_i = [x_i, y_i, v_{x,i}, v_{y,i}, t_i]$. The first edge $\mathbf{e}_{01}$ and the middle edge of $\mathbf{e}_{12}$ are generated with $\alpha = 0$. For the upper edge of $\mathbf{e}_{12}$ $\alpha = 1$ and for the lower one $\alpha = -1$ is used. The initial state

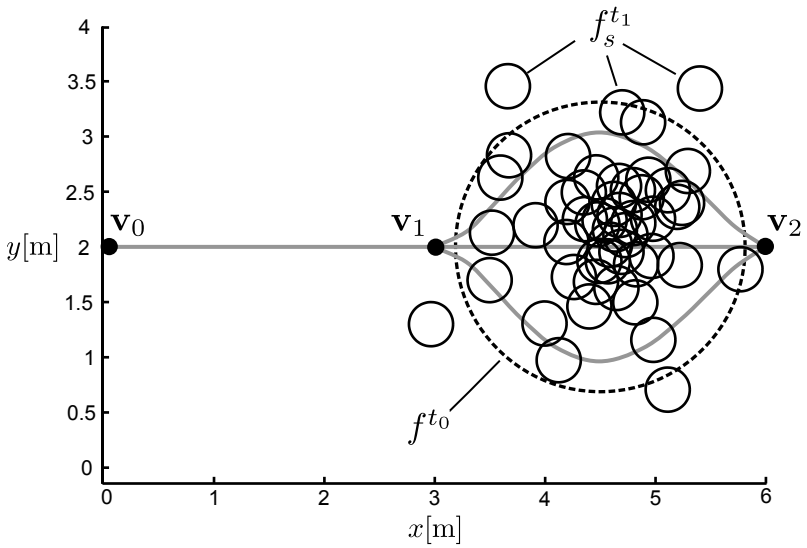


Fig. 2.8.: The vertices of the graph are shown by black solid circles and the gray lines are the associated edges. the dashed circle depicts the 2-σ ellipsoid of f^{t_0} and the solid circles represent the 2-σ ellipsoids of 50 sampled $f_s^{t_1}$.

and the covariance matrix of the object are

$$\mathbf{x}(t_0) = [4.5\,\mathrm{m}, 2.0\,\mathrm{m}, 0.0\,\mathrm{m/s}, 0.0\,\mathrm{m/s}]$$
$$\Sigma(t_0) = \mathrm{diag}[0.01\,\mathrm{m}^2, 0.01\,\mathrm{m}^2, 0.05\,\mathrm{m^2/s^2}, 0.05\,\mathrm{m^2/s^2}].$$

The estimated collision probability of $\mathbf{e}_{01}$ is 0 and $[0.204, 0.307, 0.204]$ for $\mathbf{e}_{12}$ starting with the upper one. The safest route $r^*(\mathbf{v}_0, \mathbf{v}_2, t_0)$ to $\mathbf{v}_2$ has a collision probability of 0.204, based on the information available at t_0 and without considering the possibility to replan the route at $\mathbf{v}_1$.

For calculating the novel expected minimum collision probability according to Sec. 2.8.1 for $\mathbf{e}_{12}$, 500 distributions $f_s^{t_1}$ are sampled from $f^{t_0}(\mathbf{x}, t_1)$ to generate the belief $\mathbf{b}_{t_0}^{t_1}$. In Fig. 2.8 the distribution f^{t_0} and 50 samples $f_s^{t_1}$ are depicted. The resulting expected minimum collision probability according to Def. 3 is 0.023. This is a reduction of 88.73% compared to the optimal trajectory without considering the opportunity of replanning. This is caused by the fact, that the robot is able to choose the safest trajectory at time t_1 depending on the occurring distribution f^{t_1}. Additionally, the variance of the object distributions is not large enough to have a significant effect on all three trajectories of $\mathbf{e}_{12}$ regarding only one possible PDF f^{t_1}. This is illustrated in Fig. 2.9 for all possible edges at time $t = 3.3\,\mathrm{s}$. One can see, that there exists no configuration of the object, such that its 2-σ ellipsoid of its PDF overlaps with all three possible enlarged robot systems $\mathcal{A}^b$ at this time step.

Graph Assessment

For evaluating the graph assessment approach presented in Sec. 2.8.1, an example graph is generated and is shown in Fig. 2.10. This example shows, that multi-edges can also appear during the reduction of the graph effecting the collision probability of the whole graph. The states of

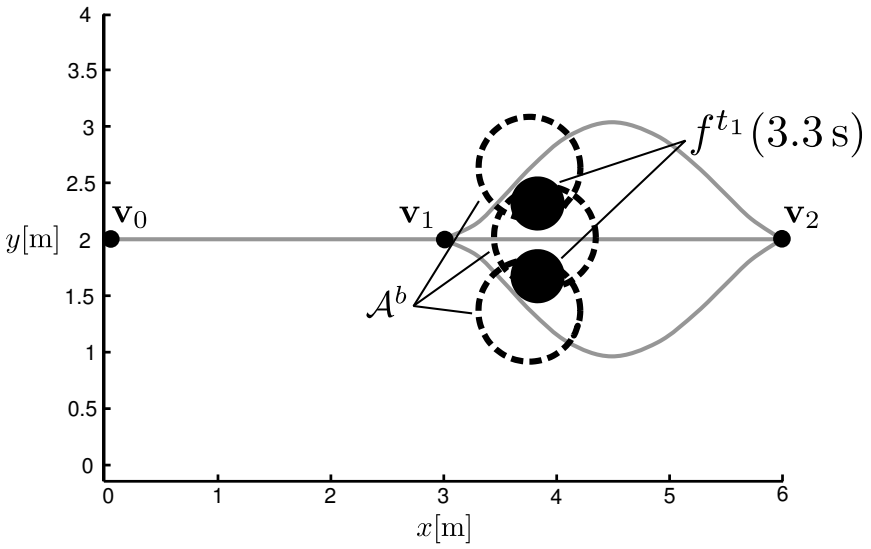


Fig. 2.9.: The dashed circles represent the area of the enlarged robot system $\mathcal{A}^b$ at time $t = 3.3\,\mathrm{s}$ for the three possible edges. The solid black circles show possible 2-σ ellipsoids of the obstacle at the same time.

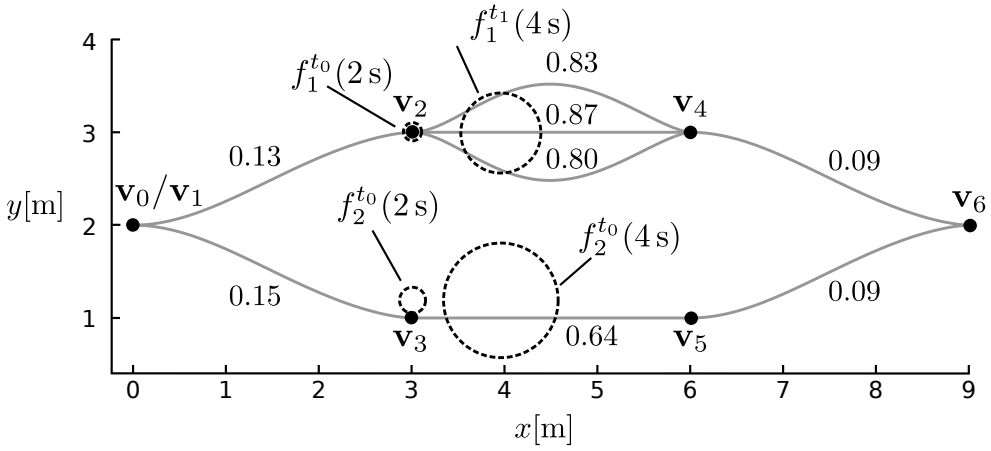


Fig. 2.10.: Example graph for safety assessment. The numbers next to the edges are the collision probabilities.

the vertices are

$$\begin{aligned}
\mathbf{v}_0 &= [0\,\mathrm{m}, 2\,\mathrm{m}, 0\,\mathrm{m/s}, 0\,\mathrm{m/s}, 0\,\mathrm{s}] \quad \mathbf{v}_1 = [0\,\mathrm{m}, 2\,\mathrm{m}, 0\,\mathrm{m/s}, 0\,\mathrm{m/s}, 2\,\mathrm{s}] \\
\mathbf{v}_2 &= [3\,\mathrm{m}, 3\,\mathrm{m}, 2\,\mathrm{m/s}, 0\,\mathrm{m/s}, 4\,\mathrm{s}] \quad \mathbf{v}_3 = [3\,\mathrm{m}, 1\,\mathrm{m}, 2\,\mathrm{m/s}, 0\,\mathrm{m/s}, 4\,\mathrm{s}] \\
\mathbf{v}_4 &= [6\,\mathrm{m}, 3\,\mathrm{m}, 2\,\mathrm{m/s}, 0\,\mathrm{m/s}, 6\,\mathrm{s}] \quad \mathbf{v}_5 = [6\,\mathrm{m}, 1\,\mathrm{m}, 2\,\mathrm{m/s}, 0\,\mathrm{m/s}, 6\,\mathrm{s}] \\
\mathbf{v}_6 &= [9\,\mathrm{m}, 2\,\mathrm{m}, 0\,\mathrm{m/s}, 0\,\mathrm{m/s}, 8\,\mathrm{s}].
\end{aligned}$$

The vertex $\mathbf{v}_0$ is equal to $\mathbf{v}_1$ except for the time information. This allows the robot to gain more accurate information about the objects before it decides to take the edge $\mathbf{e}_{12}$ or $\mathbf{e}_{13}$ at time t_1. Two objects are placed in the workspace and their initial states are modeled as Gaussians

$$\begin{aligned}
\boldsymbol{\mu}_1(t_1) &= [2.5\,\mathrm{m}, 3.0\,\mathrm{m}, 0.5\,\mathrm{m/s}, 0\,\mathrm{m/s}] \quad \boldsymbol{\mu}_2(t_1) = [2.5\,\mathrm{m}, 1.1\,\mathrm{m}, 0.5\,\mathrm{m/s}, 0\,\mathrm{m/s}] \\
\boldsymbol{\Sigma}_1(t_1) &= \mathrm{diag}[0.01\,\mathrm{m}^2, 0.01\,\mathrm{m}^2, 0.05\,\mathrm{m^2/s^2}, 0.05\,\mathrm{m^2/s^2}] \\
\boldsymbol{\Sigma}_2(t_1) &= \mathrm{diag}[0.02\,\mathrm{m}^2, 0.02\,\mathrm{m}^2, 0.10\,\mathrm{m^2/s^2}, 0.10\,\mathrm{m^2/s^2}].
\end{aligned}$$

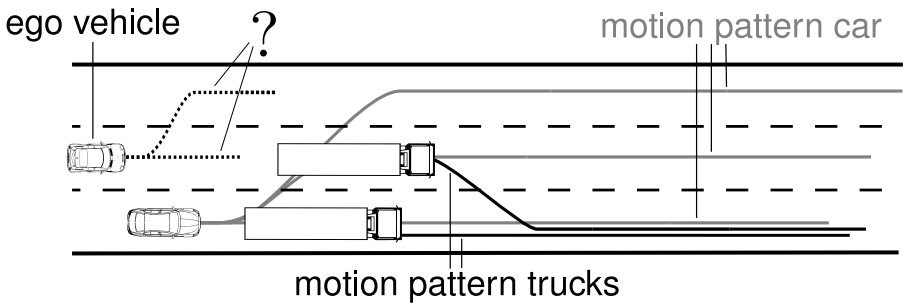
The objects have the same size and the radius of the enlarged robot system is $0.5\,\mathrm{m}$. Using the common assessment algorithm, the collision probability is 0.73 and the optimal route is $r^* = \{\mathbf{e}_{13}, \mathbf{e}_{35}, \mathbf{e}_{56}\}$. Using the novel graph assessment algorithm with $N_\mathrm{s}(k) \equiv 10$ results in the collision probability of 0.32 which is a relative reduction of 56%. This follows from the expected collision probability of the multi-edge $\mathbf{e}_{24}$ being smaller than the minimum collision probability of the single trajectories. The collision probability of the best trajectory is 0.80, however the expected collision probability of $\mathbf{e}_{24}$ is 0.38. During the graph reduction Alg. 1 another multi-edge occurs between $\{\mathbf{e}_{12}, \mathbf{e}_{24}, \mathbf{e}_{46}\}$ and $\{\mathbf{e}_{13}, \mathbf{e}_{35}, \mathbf{e}_{56}\}$ leading to the final collision probability.

2.9.3. Automotive Applications

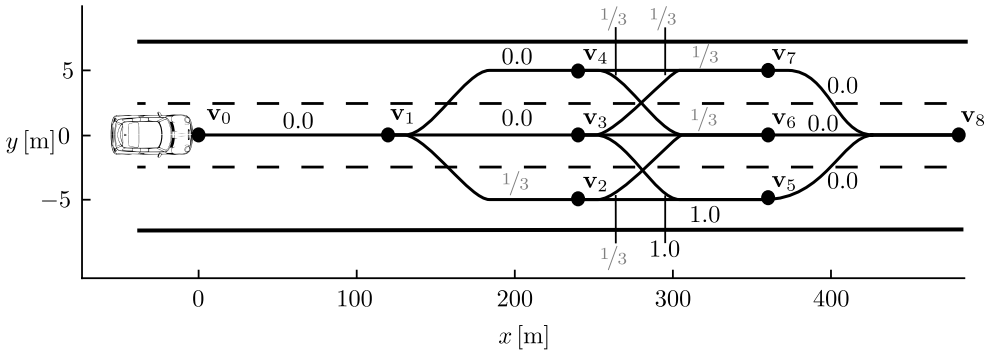
In this section the implementation from Sec. 2.4.1 is applied to assess the safety for autonomous vehicle driving. Therefore, an overtaking scenario on a highway is used. In this scenario, the ego vehicle approaches a group of two trucks and one car. One truck is at the end of its overtaking maneuver and the future motion of the car is uncertain. Therefore, three different motion patterns of the other car are considered: slowing down and staying behind the truck; maintaining speed and overtaking one truck at the middle lane; speeding up and overtaking both trucks at the left lane. The goal of the ego vehicle is to pass this group without collision. The whole scenario is sketched in Fig. 2.11a. The initial states of the other vehicles, the trucks $\mathbf{x}_{\mathrm{t}_1}$, $\mathbf{x}_{\mathrm{t}_2}$ and the car $\mathbf{x}_\mathrm{c}$ are

$$\begin{aligned}
\mathbf{x}_\mathrm{c} &= [\ 50\,\mathrm{m}, -5\,\mathrm{m}, 30\,\mathrm{m/s}, 0\,\mathrm{m/s}], \\
\mathbf{x}_{\mathrm{t}_1} &= [110\,\mathrm{m}, -5\,\mathrm{m}, 23\,\mathrm{m/s}, 0\,\mathrm{m/s}], \\
\mathbf{x}_{\mathrm{t}_2} &= [120\,\mathrm{m}, \quad 0\,\mathrm{m}, 25\,\mathrm{m/s}, 0\,\mathrm{m/s}].
\end{aligned}$$

The motion patterns of the objects are generated with the same approach which is used for the generation of the roadmap of the ego vehicle. For a clear presentation, the roadmap of the ego vehicle is generated with a constant $v_x = 40\,\mathrm{m/s}$ and it has the option to change the lanes at two



(a) Sketched scenario



(b) Roadmap of ego vehicle with resulting collision probabilities. The black numbers refer to collision probabilities caused by the trucks and the gray ones to collision probabilities caused by the other car.

Fig. 2.11.: The top figure shows the sketched scenario with the ego vehicle, two trucks and the other vehicle. The bottom figure shows the roadmap of the ego vehicle with the corresponding collision probabilities of the edges.

different time points. The vertices of the roadmap are

$$\begin{aligned}
\mathbf{v}_0 &= [0\,\mathrm{m}, \quad 0\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, \quad 0\,\mathrm{s}] \quad \mathbf{v}_1 = [0\,\mathrm{m}, \quad 0\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, \quad 3\,\mathrm{s}] \\
\mathbf{v}_2 &= [3\,\mathrm{m}, -5\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, \quad 6\,\mathrm{s}] \quad \mathbf{v}_3 = [3\,\mathrm{m}, \quad 0\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, \quad 6\,\mathrm{s}] \\
\mathbf{v}_4 &= [6\,\mathrm{m}, \quad 5\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, \quad 6\,\mathrm{s}] \quad \mathbf{v}_5 = [6\,\mathrm{m}, -5\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, \quad 9\,\mathrm{s}] \\
\mathbf{v}_6 &= [9\,\mathrm{m}, \quad 0\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, \quad 9\,\mathrm{s}] \quad \mathbf{v}_7 = [9\,\mathrm{m}, \quad 5\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, \quad 9\,\mathrm{s}] \\
\mathbf{v}_8 &= [9\,\mathrm{m}, \quad 0\,\mathrm{m}, 40\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 0\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}, 12\,\mathrm{s}],
\end{aligned}$$

where $\mathbf{v}_0$ is the initial state of the ego vehicle.

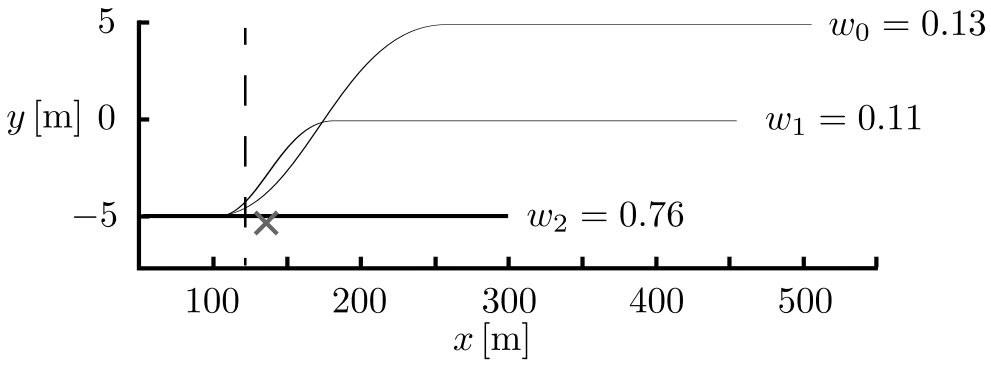
The resulting collision probabilities for the edges are show in Fig. 2.11b. The collision probabilities for the edges $\mathbf{e}_{25}$ and $\mathbf{e}_{35}$ are caused by the trucks and the other ones by the overtaking car. The result of the common graph assessment is a collision probability of $\frac{1}{3}$ and the safest route (there are multiple ones with the same collision probability) is $r^* = \{\mathbf{e}_{01}, \mathbf{e}_{12}, \mathbf{e}_{26}, \mathbf{e}_{68}\}$.

However, the ego vehicle has the option to react on the future PMFs of the overtaking car at the vertices $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ and $\mathbf{v}_4$. The result of the novel graph assessment Alg. 1 incorporating replanning capabilities is 0.057. Therefore, the mean of 10 trials were calculated with $N_{\mathrm{s}}(k) \equiv 10$ with a variance of 1.14×10^{-4}. For the generation of the measurements the same covariance matrix $\mathbf{\Sigma}_z = \mathbf{\Sigma}(t_0)$ as in Sec. 2.9.2 is used. Compared to the common assessment, this means a relative decrease of 82.9%.

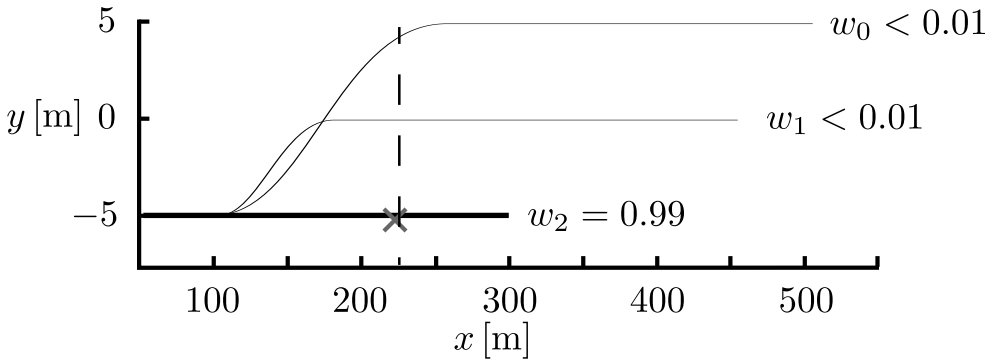
The calculation of the graph assessment incorporating replanning is illustrated by two samples. Therefore, only the car is considered since the motion prediction of the trucks is deterministic and results always in a collision at the edges $\mathbf{e}_{25}$ and $\mathbf{e}_{35}$. The motion prediction for the car at time steps t_1 and time $t_{2/3/4}$ is shown in Fig. 2.12 and Fig. 2.13 for two different sampled distributions.

In Fig. 2.12 the probability that the car will slow down and will stay behind the trucks is already quite high with 76% at time t_1. Thus the ego vehicle can choose to take the left lane or stay at the middle lane to prevent a collision already at vertex $\mathbf{v}_1$. At time $t_{2/3/4}$ the motion prediction of the car is even sharper and it makes practically no difference if the ego vehicle takes the left or middle lane.

In Fig. 2.13 the sampled PMF at time t_1 is different and the most likely motion pattern of the car is to take the left lane with a probability of 52%. From the situation at this time the right lane seems to be the best choice for the ego vehicle. However, the situation at time $t_{2/3/4}$ shows that the car is changing to the middle lane with a probability of 93%. The high probability for the left lane at time t_1 was caused by a noisy measurement. As can be seen in Fig. 2.13b, the ego vehicle can still change its lane at time $t_{2/3/4}$ before colliding with the car. It would be the best choice at time t_1 for the ego vehicle to stay at the middle lane. Since taking the right lane would force the ego vehicle at time $t_{2/3/4}$ to change to the middle lane due to the trucks on the right lane. This would cause most probable an accident with the other car. This behavior of the ego vehicle, to stay on the middle lane to allow the vehicle to stay on the middle lane or to change to the left lane, is the result of the optimal policy calculation. is taken into account in the novel graph assessment algorithm by the optimal policy π^*.

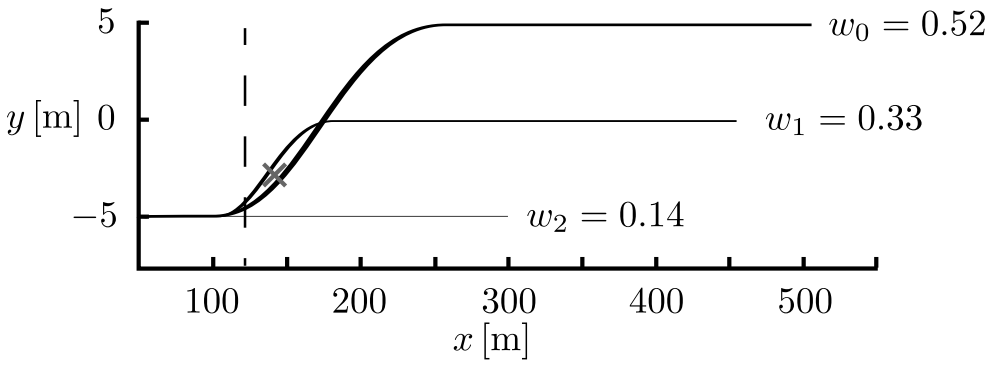


(a) Motion prediction and measurement at time t_1 (3 s).

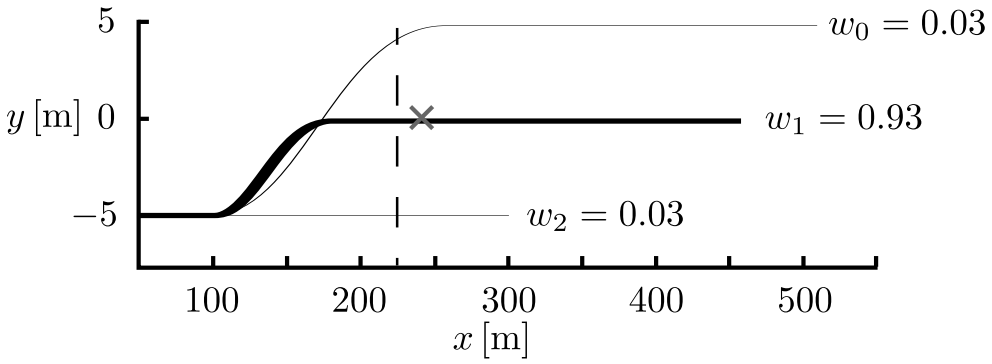


(b) Motion prediction and measurement at time $t_{2/3/4}$ (6 s).

Fig. 2.12.: The motion prediction of the car is shown at two different time steps. The solid lines are the motion patterns and the thickness is proportional to their weights. The cross depicts the measurement regarding to the sampled distribution. The dashed line represents the x-position of the ego vehicle at the time steps.



(a) Motion prediction and measurement at time t_1 (3 s).



(b) Motion prediction and measurement at time $t_{2/3/4}$ (6 s).

Fig. 2.13.: The motion prediction of the car is shown at two different time steps. The solid lines are the motion patterns and the thickness is proportional to their weights. The cross depicts the measurement regarding to the sampled distribution. The dashed line represents the x-position of the ego vehicle at the time steps.

2.10. Discussion

In this chapter a novel safety assessment concept for trajectories represented as directed graphs is discussed. This is the main contribution of this chapter and includes the consideration of two facts during the execution of a trajectory: the possibility of replanning; the collection of new information of the environment. To the best of the author's knowledge, this is the first time that these facts are taken into account in the field of safety assessment. It was shown, that the collision probability of an entire graph is always smaller than that for a single route, since all possible trajectories of the graph and future positions of the objects are considered. The presented approach is independent of the uncertainty model representing the future states of the objects and is directly applicable to any roadmap-based planner.

Graph Reduction Algorithm The graph reduction algorithm differs from other graph-based algorithms like solving the dynamic shortest path problem [22] or determining the route with the minimum collision costs [45]. These approaches use weighted graphs and apply graph search algorithms such as the A* or Dijkstra algorithm to identify the optimal route. These routes have the minimum summed costs. But the route with the minimum sum of collision probabilities is not necessarily the route with the minimum collision probabilities since the collision probability is not an additive function. Additionally, they do not consider the possibility of replanning by determining a policy depending on the future state of the environment.

Implementation Two example implementations, one using Gaussian distributions and one using motion patterns, for representing the future uncertain states of the objects are presented. Both implementations use a belief tree to approximate the infinitely many possible future distributions of the objects.

The novel closed-loop assessment approach considers the possibility of replanning the future route at any vertex in the graph based on the information acquired during the execution. This entails a considerably higher computational effort. For instance, a graph with 4 consecutive vertices at which the robot can replan its future route results in a belief tree with 1111 distributions for one object if the compound distribution is approximated by e.g. 10 sampled distributions. For determining the optimal route with the minimum collision probability, the collision probability is estimated only once for each edge in the graph based on the initial belief. Applying the novel algorithm, the collision probability must be estimated 1110 times more often for determining the optimal policy than for determining the optimal route in the graph. The computational load of the graph reduction algorithm (without the estimation of collision probabilities) can be neglected compared to the computational effort necessary for estimating the collision probabilities. However, there are two possibilities to clearly reduce the computational complexity. First of all, the subgraphs with a high collision risk could be determined and the other subgraphs with a low collision risk should be assessed with the common approach. Second, the samples of the sampling tree can be merged into one sample at vertices with only one output edge. Thus, the number of samples can be clearly reduced. This is possible due to Proposition 3. The situation is different regarding the implementation for automotive applications. Since only the weights of the PMFs change in the belief tree but not the motion primitives, the number of collision checks is equal to the common approach. For the implementation for automotive applications, the computational

effort for the novel algorithm is comparable to the common approach.

Simulation Results All simulation scenarios showed a considerable reduction of the estimated collision probability for mobile robot and automotive applications. This is caused by considering all possible future trajectories of the robot in connection with the possible future distributions of the objects. It could be argued, that the same is possible by continuously reassessing the safety of the robot during its navigation. But this could lead the robot to make a wrong decision regarding its future route due to a wrong assessment at the beginning. In summary, this work clearly states that considering the possibility of replanning is essential for a more reliable safety assessment approach.

3. Assessment beyond Planning Horizon

Summary This chapter discusses the problem of assessing the safety of trajectories beyond the planning horizon. These trajectories are partial solutions to the motion planning problem, since they do not reach the goal state of the robot, but they bring the robot closer to it. Although these trajectories are guaranteed to be safe during their time horizon, they may end in a state leading eventually to a collision. Thus, the final state of the partial trajectories require a specific safety assessment that evaluates the safety beyond the planning horizon. Different algorithms are presented to identify these states in deterministic as well as in uncertain environments populated with dynamic objects. A special focus is on the efficient computation.

The outline of this chapter is as follows: Sec. 3.1 is motivating this chapter and gives the problem formulation. Sec. 3.2 provides an overview of related work. Then the definition and properties of the inevitable collision state (ICS) and inevitable collision obstacle (ICO) are presented in Sec. 3.3. This leads to the discussion of unions of ICS sets in Sec. 3.4 which allows one to define the robot maneuverability in Sec.3.5. The next section discusses the problem in a stochastic modeled environment starting with the probabilistic collision state (PCS) in Sec. 3.6. Based on PCS, the overall collision probability (OCP) is presented in Sec. 3.7 allowing a complete probabilistic assessment of partial trajectories. The probabilistic collision costs (PCC) are introduced in Sec. 3.8 which are an alternative safety indicator to the collision probability. Implementations of the proposed concepts are shown in Sec. 3.9 and the associate simulation results are discussed in Sec. 3.10. Finally, a discussion of this chapter is given in Sec. 3.11.

3.1. Motivation and Problem Formulation

For navigation in uncertain or dynamic environments it is often not reasonable to plan the robot motion until its goal state, since the prediction of such environments is not reliable for a long time horizon due to uncertainties or the future prediction is not given for the whole time interval. Thus, the robot plan needs many adaptations that are often unnecessary, since they take place in the distant future. For this reason, model predictive control (MPC) [38] also called receding horizon control (RHC) is applied to these kinds of problems. The main idea of RHC is to apply iterative optimization inside a finite time horizon. Thereby, the problem is partitioned into subproblems which allow an on-line optimization. For the motion planning problem this means that instead of optimizing a trajectory from the initial state to the goal state, only a partial trajectory is optimized for a smaller time horizon leading the robot to its goal state. However, two problems arise from the partial trajectories: problem of stability (in the sense of reaching the goal) and problem of feasibility. In this chapter the focus is on the problem of feasibility, since it ensures that no constraints will be violated (i.e. no collision until reaching the goal). Fig. 3.1

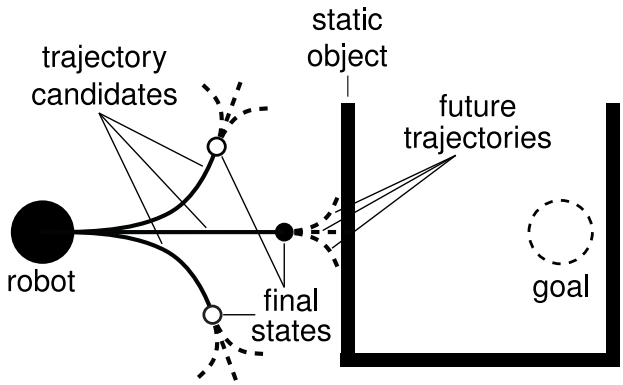


Fig. 3.1.: The environment contains one robot (black solid circle) which task is to reach a goal state (dashed circle) lying inside a U-shaped object. Trajectory candidates are represented by solid lines and the dashed lines depict future possible trajectories. The final states are depicted as circles. The best candidate is the solid circle, but the robot will eventually collide with the static object beyond the planning horizon.

illustrates these two problems. From the three trajectory candidates the middle one ends closest to the goal. But as illustrated, all succeeding trajectories will end in a collision with the object.

Another very similar approach for this problem is to apply partial motion planning (PMP) as proposed by [92]. The PMP approach also calculates the best partial trajectory to the goal for a certain time horizon. The problem of feasibility is addressed with the concept of inevitable collision states. If the robot is not colliding during the time interval of a certain trajectory and if the final state of the trajectory is not an ICS, this trajectory is declared safe.

In this chapter, the problem of feasibility of partial trajectories is discussed which can also be called the problem of safety assessment beyond the planning horizon. The problem of safety assessment beyond the planning horizon for deterministic environments which are populated by static objects, ignoring dynamic objects, and a single robot according to Sec. 1.2.2, is formulated as:

Safety Beyond the Planning Horizon in Deterministic Environments *Given the future state of the robot system after the planning horizon and the future state of the environment, the state of the robot is safe if there exists at least one trajectory ending in a state that is guaranteed to be safe for an infinite time horizon.*

The aspect of the infinite time horizon is a necessary criterion as otherwise the safety is again only guaranteed for a finite time horizon. This would extend the time horizon for which safety is guaranteed, however, this arises again the same problem of safety assessment beyond this extended time horizon. This can be illustrated by means of the example from [23] which is depicted in Fig. 3.2:

> *" For example, a car-like vehicle with momentum moving at high speed towards a wide brick wall obstacle might not have enough braking distance or maneuverability to avoid impact. Thus, even though the car's current state is free of collision, the car will inevitably collide regardless of the future control actions applied. "*

In this chapter, this problem is also investigated for uncertain environments that are populated by static objects, ignoring dynamic objects, reactive objects, and one robot system according to

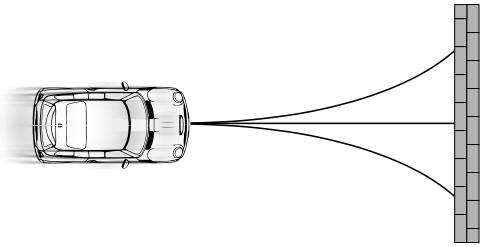


Fig. 3.2.: Vehicle is driving with high speed towards a brick wall and no control input exists which can prevent a collision. Three example trajectories are illustrated, representing full braking and full steering to the left and to the right.

Sec. 1.2.2. The initial state of the objects and their future motion is uncertain. The robot system behaves deterministically meaning that its future trajectory is completely known. The problem of safety assessment beyond the planning horizon for uncertain environments is formulated as:

Safety Beyond the Planning Horizon in Uncertain Environments *Given the future state of the robot system after the planning horizon and the uncertain future state of the environment, the state of the robot is assessed by the trajectory with the minimum collision probability ending in a state which allows assessing the safety for an infinite time horizon.*

Compared to the formulation for deterministic environments, a binary assessment may not be possible anymore due to the uncertainties in the future states of the objects. Hence, the collision probability is used to rate the safety of the last state of the trajectory. As mentioned above, the problem is to find a trajectory ending in a certain state that allows reasoning about the safety of the robot for an infinite time horizon and at the same time having the lowest possible collision probability. Thereby, the collision probability of this trajectory as well as the collision probability of the final state of this trajectory is taken into account.

Next, an overview about related work addressing these problems is given.

3.2. Related Work

The problem of assessing the safety of a given state for an infinite time horizon is also known as the feasibility problem in *receding horizon control* (RHC). In order to guarantee feasibility, no violation of constraints including the collision constraint, *invariant set theory* [85] is used in RHC applications. In [56] a *non-linear model predictive control* framework is presented which guarantees nominal feasibility using invariant sets. The concept of terminal feasible invariant set is introduced in which safety is guaranteed for an infinite time horizon in [103]. The main difference to former approaches is, that this method is applied on-line and needs no complex offline computation which is not flexible to changes in the environment. A collection of affine transformations is used to represent these sets. The sets can be seen as an a priori known fallback solution that is executed if no feasible trajectory can be found in the next iterative optimization.

One of the first works, addressing the problem of identifying states leading inevitably to a collision in the field of motion planning is presented in [75, 76]. Therefore, the *region of inevitable collision* (RIC) is defined as the set of states which are already in collision or which are inevitable leading to a collision. In [130] the analytic computation of RIC for convex poly-

gons and the approximation for unions of RIC sets is discussed with the aim to make motion planning more efficient and safer. The effectiveness of the RIC approach for motion planning of vehicles with underactuated dynamics is shown in [23]. Additionally, the RIC extensions *region of potential collision* (RPC) and *the region of near-collision* (RNC) are presented. Instead of distinguishing between RIC and non-RIC, RPC rates states depending on the number of control inputs leading to collision and RNC rates the states depending on the time horizon after the state is inside the RIC. This allows one to perform not only a binary assessment. A state belonging to RIC is also called an *inevitable collision state* (ICS) [33]. This concept has already been applied in dynamic environments [83, 84] and also for car-like vehicles [90]. The ICS approach is also used in the *partial motion planning* (PMP) [92] framework. It directly considers the real-time constraint of the motion planning problem. Therefore, the motion planning algorithms has a finite time horizon for the computation of the next partial trajectory. This trajectory is guaranteed to be safe during and beyond the planning horizon. This is done by applying an ICS check to the final state of the trajectory.

In the following, the concept of ICS is presented identifying states of the robot which will eventually lead to a collision.

3.3. Inevitable Collision State and Inevitable Collision Obstacle

The concept of ICS was first introduced in [32] and discussed in more detail in [33] together with the inevitable collision obstacle (ICO). The concept of ICS is developed for deterministic environments in which the motion of all objects is known for an infinite time horizon. That implies, that all information about the environment is noise-free and that the future motion of the robot system is also deterministic. In the following, the definition and properties of ICS are recalled from literature based on [33].

Definition 5 (**Inevitable Collision State).** *Given the state* $\mathbf{x}$*, the inevitable collision state is defined as*

$$\mathrm{ICS}(\mathbf{x}) = \begin{cases} 1, & \forall \tilde{u}(\mathbf{x}) \in \tilde{\mathcal{U}}, \exists t, \exists i : \mathcal{A}(\tilde{u}(\mathbf{x}, t)) \cap \mathcal{B}_i(t) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases}$$

Loosely speaking, the robot is in an inevitable collision state if there exists no trajectory $\tilde{u}(\mathbf{x})$ which begins at state $\mathbf{x}$ and can avoid a collision regarding all workspace objects. If only one object $\mathcal{B}_i$ is considered, this is written as $\mathrm{ICS}(\mathbf{x}, \mathcal{B}_i)$. If a reduced set of trajectories $\mathcal{I} \subset \tilde{\mathcal{U}}$ is considered, this is denoted as $\mathrm{ICS}(\mathbf{x}, \mathcal{B}_i, \mathcal{I})$. The definition of ICS leads to the definition of ICO:

Definition 6 (**Inevitable Collision Obstacle).** *Given an object* $\mathcal{B}_i$*, the inevitable collision obstacle is defined as*

$$\mathrm{ICO}(\mathcal{B}_i) = \{\mathbf{x} \in \mathcal{X} | \, \mathrm{ICS}(\mathbf{x}, \mathcal{B}_i) = 1\}.$$

Loosely speaking, the ICO of one object is defined as the set of all states which are an ICS regarding this object. If only a subset of trajectories $\mathcal{I} \subset \tilde{\mathcal{U}}$ is used, this is denoted as $\mathrm{ICO}(\mathcal{B}_i, \mathcal{I})$. In the following two ICO properties presented in [33] are summarized. A conservative approximation of ICO can be calculated by using only a subset of all possible future trajectories.

Property 1 (ICO Approximation [33]).

$$\mathrm{ICO}(\mathcal{B},\tilde{\mathcal{U}}) \subseteq \mathrm{ICO}(\mathcal{B},\mathcal{I}),\ \textit{with}\ \mathcal{I} \subset \tilde{\mathcal{U}} \tag{3.1}$$

The following property shows, that $\mathrm{ICO}(\mathcal{B})$ can be derived from $\mathrm{ICO}(\mathcal{B}_i,\tilde{u})$ for every possible future trajectory $\tilde{u}$.

Property 2 (ICO Characterization [33]).

$$\mathrm{ICO}(\mathcal{B}) \;=\; \bigcap_{\tilde{u}\in\tilde{\mathcal{U}}} \bigcup_{i=1}^{N_b} \mathrm{ICO}(\mathcal{B}_i,\tilde{u}) \tag{3.2}$$

It is pointed out, that the union of ICO computations is *not* equal to the ICO computation of the union of objects.

Property 3 (ICO Union).

$$\mathrm{ICO}(\mathcal{B}) \;\neq\; \bigcup_{i=1}^{N_b} \mathrm{ICO}(\mathcal{B}_i) \tag{3.3}$$

This is a major drawback of ICO computation, since the computation must start from scratch, if a new object appears. This property is illustrated in Fig. 3.3 with an example using two rectangular objects. In the following section a modified computation of ICS is introduced which allows one to compute the union of ICS sets. Furthermore, this method allows one to compute the ICS property of a certain robot state more efficiently than other state of the art approaches.

3.4. Union of Inevitable Collision Obstacles

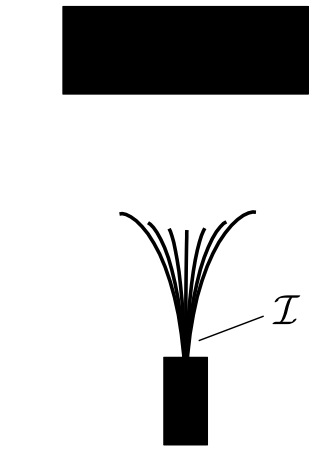
As shown by (3.3), the union of ICS sets (ICOs) is not equal to the ICS set of the union of objects. That implies, that the computation of $\mathrm{ICO}(\mathcal{B})$ cannot be used anymore if a new object appears in the environment or if the motion prediction of any object is not valid anymore (e.g. unexpected change of direction) and the computation needs to be done from scratch. In order to address this problem, the computation is modified, thus that the union of ICS sets can be efficiently computed in a sequential manner. It is shown, that the union of ICO can be computed by using only the reduced set of trajectories which are still collision-free regarding the already investigated objects. Therefore, the case of two objects is considered $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$.

Proposition 7. *One can show, that the union can be calculated as*

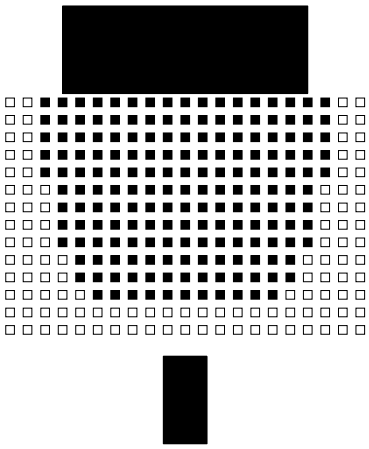
$$\mathrm{ICO}(\mathcal{B}_1 \cup \mathcal{B}_2,\tilde{\mathcal{U}}) \;=\; \mathrm{ICO}(\mathcal{B}_1,\tilde{\mathcal{U}}) \cup \mathrm{ICO}(\mathcal{B}_2,\tilde{\mathcal{U}}^a(\mathcal{B}_1)), \tag{3.4}$$

where $\tilde{\mathcal{U}}^a(\mathcal{B}_1)) \subseteq \tilde{\mathcal{U}}$ are all admissible trajectories which are not leading to a collision with $\mathcal{B}_1$

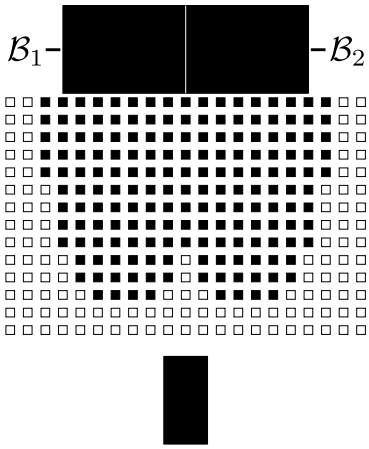
$$\tilde{\mathcal{U}}^a(\mathbf{x},\mathcal{B}_1) = \{\tilde{u} \in \tilde{\mathcal{U}} | \forall t, \mathcal{A}(\tilde{u}(\mathbf{x},t)) \cap \mathcal{B}_1 = \emptyset\}.$$



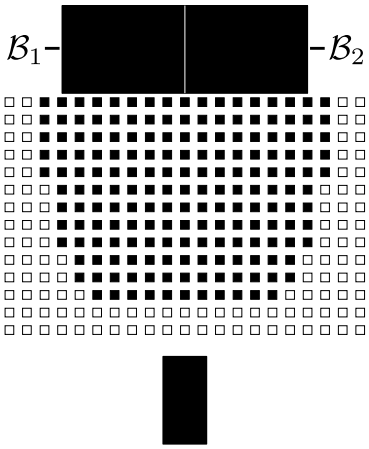
(a) Used configuration.



(b) $\mathrm{ICO}(\{\mathcal{B}_1, \mathcal{B}_2\}, \mathcal{I})$.



(c) $\mathrm{ICO}(\mathcal{B}_1, \mathcal{I}) \cup \mathrm{ICO}(\mathcal{B}_2, \mathcal{I})$.



(d) $\mathrm{ICO}(\mathcal{B}_1, \mathcal{I}) \cup \mathrm{ICO}(\mathcal{B}_2, \mathcal{I}^a(\mathcal{B}_1))$

Fig. 3.3.: The pictures illustrate different ICS calculations for the same setup. A rectangular robot is moving towards a static object. The squares depict the states that have been checked for ICS, whereas solid black squares are ICS states. The example illustrates, that the union of ICO is not equal to the ICO of the union of objects. This is visible by the gap of states in the union of the two ICOs which are all ICS states regarding the ICO of the union of objects.

For the ICS calculation of $\mathcal{B}_2$, only $\mathcal{U}^a(\mathcal{B}_1)$ needs to be considered. The proof is done for a single state $\mathbf{x}$.

Proof. According to Def. 5, each trajectory $\tilde{u}$ needs to be checked for collision with all objects $\mathcal{B}$. If one trajectory $\tilde{u}$ exists which is not colliding with any object of $\mathcal{B}$, the state is not an ICS. Hence, it is sufficient to check if all trajectories collide with at least one object. It is shown that

$$\mathrm{ICS}(\mathbf{x}, \mathcal{B}, \tilde{\mathcal{U}}) = \mathrm{ICS}(\mathbf{x}, \mathcal{B}_1, \tilde{\mathcal{U}}) \vee \mathrm{ICS}(\mathbf{x}, \mathcal{B}_2, \tilde{\mathcal{U}}^a(\mathbf{x}, \mathcal{B}_1))$$

is true which is the equivalent of (3.4) for just one state. It is shown that all possible trajectories $\tilde{\mathcal{U}}$ are checked for collision with all objects $\mathcal{B}$. The first term $\mathrm{ICS}(\mathbf{x}, \mathcal{B}_1, \tilde{\mathcal{U}})$ determines which trajectories collide with the object $\mathcal{B}_1$ and which are collision-free, denoted as $\tilde{\mathcal{U}}^a(\mathbf{x}, \mathcal{B}_1)$. Since it is sufficient that a trajectory collides with $\mathcal{B}_1$ or $\mathcal{B}_2$ only the set $\tilde{\mathcal{U}}^a(\mathbf{x}, \mathcal{B}_1)$ needs to be considered for object $\mathcal{B}_2$. If there exists one trajectory of $\tilde{\mathcal{U}}^a(\mathbf{x}, \mathcal{B}_1)$ which does not collide with $\mathcal{B}_2$, the state $\mathbf{x}$ is not an ICS according to Def. 5, because one trajectory $\tilde{u}$ exists which does not collide with $\mathcal{B}_1$ or $\mathcal{B}_2$, so

$$\forall t \; A(\tilde{u}(\mathbf{x}, t)) \cap \mathcal{B}(t) = \emptyset.$$

Since $\tilde{\mathcal{U}}^a(\mathbf{x}, \mathcal{B})$ is the set of trajectories which does not collide with $\mathcal{B}$ for the state $\mathbf{x}$ the same can be done for each possible state

$$\mathrm{ICO}(\mathcal{B}, \tilde{\mathcal{U}}) = \bigcup_{\mathbf{x} \in \mathcal{X}} \mathrm{ICS}(\mathbf{x}, \mathcal{B}, \tilde{\mathcal{U}}).$$

Hence, it is shown that (3.4) is valid. For the general case, the union is computed as

$$\mathrm{ICO}(\mathcal{B}) = \bigcup_{i=1}^{N_\mathrm{b}} \mathrm{ICO}(\mathcal{B}_i, \tilde{\mathcal{U}}_i^a),$$

where

$$\mathcal{U}_i^a = \begin{cases} \tilde{\mathcal{U}}, & i = 1 \\ \tilde{\mathcal{U}}^a(\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}), & i \neq 1. \end{cases}$$

□

Additionally, it can be shown with ICO Prop. 1 that

$$\mathrm{ICO}(\mathcal{B}_2) \subseteq \mathrm{ICO}(\mathcal{B}_2, \tilde{\mathcal{U}}^a(\mathcal{B}_1))$$

since $\tilde{\mathcal{U}}^a(\mathcal{B}_1) \subset \tilde{\mathcal{U}}$. This property allows one to calculate ICO in a sequential manner. For the union of ICO, it is sufficient to determine the intersection sets of the admissible trajectory sets $\tilde{\mathcal{U}}^a$ if an additional object appears:

$$\mathrm{ICO}(\mathcal{B}) = \{\mathbf{x} \in \mathcal{X} | \bigcap_{i=1}^{N_\mathrm{b}} \tilde{\mathcal{U}}^a(\mathbf{x}, \mathcal{B}_i) = \emptyset\}$$

Based on this property, two novel ICS-Checkers are presented in Sec. 3.9.1.

3.5. Motion Safety Regarding Unexpected Objects

When robots traverse through partially-known environments unforeseen objects can appear caused by imperfect perception capabilities and occlusions. These situations are in particular dangerous if the robot is not able to avoid a collision with the unexpected object due to: too less space to bypass the object; the robot is too fast to brake in time. In [60] this problem is addressed by adopting the velocity profile of the robot for a given path. The velocity of the robot is controlled such that the robot can come to a standstill before colliding with any mobile object possibly intercepting its future path. Therefore, the maximum possible velocity of the objects and the limited field of view of the robot is taken into account. In [16] a similar problem, the safety of a robot system with limited field of view in dynamic and uncertain environments, is discussed. Therefore, the concept of braking inevitable collision states is presented. If the robot system is in a braking inevitable collision state, it will collide with an object and will not be at rest at the time point of the collision. Conversely, if the navigation algorithm guarantees that the robot system will never be in such a state, the robot system will always be in rest if a collision occurs. This is the main idea of this approach and it is called passive motion safety.

In this section, no deterministic but a simple probabilistic model of the perception system of the robot is used. Based on this model, the problem of preventing a collision with randomly appearing objects beyond the planning horizon is addressed. These objects should represent typical arising situations, which occur due to an imperfect perception system of the robot, such as: occluded objects; not detected objects due to noisy sensor data; dynamic objects which have been classified as static ones. In the following, it is shown how to increase the safety of the robot assuming that an unforeseen object appears.

The workspace contains the known objects $\mathcal{B}$ and the current state of the robot system is $\mathbf{x} \notin \mathrm{ICO}(\mathcal{B})$. Additionally, one unexpected static objects $\mathcal{B}_\mathrm{u}$ exists which position is uniformly distributed in $\mathcal{W}$. The probability of collision at the time t with object $\mathcal{B}_i$, as presented in Sec. A.1.2,

$$P_i(C|\tilde{u}(\mathbf{x},t)) := P(\mathcal{A}(\tilde{u}(\mathbf{x},t)) \cap \mathcal{B}_i \neq \emptyset) = \int\limits_{\mathcal{A}^{b_i}(\tilde{u}(\mathbf{x},t))} f_i(\mathbf{p},t)\,\mathrm{d}\mathbf{p}$$

is equal for every time step, since $f(\cdot)$ is a uniform distribution. Thus, the collision probability for every trajectory is equal. It is shown, that the probability of being in an ICS regarding the known objects and the unforeseen one can be reduced by increasing the maneuverability of the robot system. The robot has a finite set of trajectories $\mathcal{I}$ and its maneuverability is proportional to the number of collision-free trajectories regarding the known objects.

Definition 7 (Maneuverability). *The maneuverability $M(\mathbf{x},\mathcal{B})$ of a robot state $\mathbf{x}$ is defined as*

$$M(\mathbf{x},\mathcal{B}) = \frac{N^\mathrm{a}_\mathcal{B}(\mathbf{x})}{N^\mathrm{a}(\mathbf{x})},\ M(\mathbf{x},\mathcal{B}) \in [0,1],\ N^\mathrm{a}_\mathcal{B}(\mathbf{x}) \leq N^\mathrm{a}(\mathbf{x}),$$

where $N^\mathrm{a}_\mathcal{B}(\mathbf{x})$ is the number of admissible trajectories regarding all objects $\mathcal{B}$ and $N^\mathrm{a}(\mathbf{x})$ are the number of all admissible trajectories in $\mathcal{I}$ at state $\mathbf{x}$ regarding kinematic and dynamic constraints of the robot.

The robot system is not in an ICS, if there exists one trajectory $\tilde{u}$ which does not collide either with $\mathcal{B}$ or $\mathcal{B}_\mathrm{u}$. According to (3.4) it is sufficient to calculate $\mathrm{ICO}(\mathcal{B}_u, \mathcal{I}^\mathrm{a}(\mathcal{B}))$ for determining the ICS status, because $\mathbf{x} \notin \mathrm{ICO}(\mathcal{B})$. Since every trajectory $\tilde{u} \in \mathcal{I}^\mathrm{a}(\mathcal{B})$ has the same probability to collide with the unexpected object $P_\mathrm{u}(C|\tilde{u})$ and the collision events regarding the different trajectories are independent, the probability that the robot system is in an ICS (all trajectories lead to a collision) is

$$P(\mathbf{x} \in \mathrm{ICO}(\mathcal{B}_\mathrm{u}, \mathcal{I}^\mathrm{a}(\mathcal{B}))) = P_\mathrm{u}(C|\tilde{u})^{N^\mathrm{a}_\mathcal{B}},$$

where $N^\mathrm{a}_\mathcal{B}$ is the number of admissible trajectories of $\mathcal{I}^\mathrm{a}(\mathcal{B})$. This is equivalent to the probability that all trajectories, which are not colliding with $\mathcal{B}$, collide with $\mathcal{B}_\mathrm{u}$ meaning that there exists no trajectory that is collision-free for $\mathcal{B}$ and $\mathcal{B}_\mathrm{u}$. The probability can be decreased by increasing the number of admissible trajectories $N^\mathrm{a}_\mathcal{B}$ resulting in a higher maneuverability. Loosely speaking, if an unexpected object appears in the workspace, the probability of being in an ICS can be reduced by increasing the maneuverability regarding the known objects. In Sec. 3.10 simulations results confirm the idea of the robot maneuverability.

3.6. Probabilistic Collision State

The concept of ICS is limited to deterministic environments and is not directly applicable to uncertain environments. The reason for this is, that the future states of dynamic objects can only be predicted with uncertainty. It is assumed that the robot system still behaves deterministically. A naive approach would be to transform the probabilistic modeled environment to a bounded uncertain representation as described in Sec. 1.2.2. This could lead to navigation results with unnecessary high costs, since objects are bypassed with a huge distance. Especially in environments with a high density of dynamic objects, this may lead to the freezing robot problem (FRP) [117]. A robot which is caught in a FRP cannot find any valid trajectory to proceed to its goal location. The reason for this is, that in such environments the bounded uncertain representation of the objects may overlap and occupy too much space and thus there is not enough space left for the robot to find a collision-free trajectory.

This problem is addressed by using a stochastic modeled representation of the environment. Hence, the collision probability is used as a risk indicator instead of a binary collision assessment which leads to the probabilistic collision state (PCS). The PCS is a probabilistic generalization of the ICS concept. Instead of verifying the existence of at least one collision-free trajectory leading to a safe state, the collision probabilities of the trajectories are evaluated. Therefore, it is necessary to calculate the probability $P_i(C|\tilde{u}(\mathbf{x}))$ that the robot system has a collision with the ith object applying the trajectory $\tilde{u}$ starting at $\mathbf{x}$. The collision probability regarding all objects, assuming that the collision events of the objects are independent, is calculated as

$$P(C|\tilde{u}(\mathbf{x})) = 1 - \prod_{i=1}^{N_\mathrm{b}} (1 - P_i(C|\tilde{u}(\mathbf{x}))).$$

Since the robot system can choose any input trajectory from the set of possible input trajectories $\tilde{U}$, the input trajectory causing the minimum collision probability allows one to define the probability of an inevitable collision state.

Definition 8 (Probabilistic Collision State). *The probability of a state* $\mathbf{x}$ *leading to a collision is defined as the minimum collision probability under the best possible input trajectory:*

$$\mathrm{PCS}(\mathbf{x}) := \min_{\tilde{u}\in\tilde{\mathcal{U}}} P(C|\tilde{u}(\mathbf{x}))$$

with $\tilde{u} : t \in [0, \infty) \mapsto \mathcal{U}$ *is defined for an infinite time horizon.*

In other words, for validating the safety of a state $\mathbf{x}$, the collision probability of all possible future trajectories need to be evaluated. The trajectory with the smallest collision probability is equal to the probability that the state is an ICS. In the rest of this work, the equivalent notation $\mathrm{PCS}(\tilde{u})$ is used which refers to the PCS value of the last state $\mathbf{x}$ when executing $\tilde{u}$.

In the following it is shows that $\mathrm{PCS}(\mathbf{x}) = 1 \Leftrightarrow \mathrm{ICS}(\mathbf{x}) = 1$ when computing $\mathrm{PCS}(\mathbf{x})$ according to Def. 8. In a deterministic scenario, the position of all objects is known such that the probability distribution of all objects is a Dirac impulse δ:

$$f_i(\mathbf{p}, t) = \begin{cases} \delta, \text{ if } \mathbf{p}(t) = \mathbf{c}_i(t) \\ 0, \text{ otherwise,} \end{cases}$$

where $\mathbf{c}_i(t)$ is the reference point of object $\mathcal{B}_i$ as introduced in Sec. A.2. From this follows directly that

$$\mathcal{A}(\tilde{u}(\mathbf{x}, t)) \cap \mathcal{B}_i(t) \neq \emptyset \Leftrightarrow P_i(C|\tilde{u}(\mathbf{x}), t) = 1.$$

Thus, using Def. 5, the statement $\mathrm{ICS}(\mathbf{x}) = 1$ can be reformulated to

$$\forall \tilde{u}(\mathbf{x}) \in \tilde{\mathcal{U}}, \exists t, \exists i : P_i(C|\tilde{u}, t) = 1.$$

Using the computations from Sec. A.2, it is shown that this statement is equivalent to $\mathrm{PCS}(\mathbf{x}) = 1$:

$$\begin{aligned}
&\Leftrightarrow \forall \tilde{u}(\mathbf{x}) \in \tilde{\mathcal{U}}, \exists t, \exists i : P_i(C|\tilde{u}(\mathbf{x}), t) = 1 \\
&\Leftrightarrow \forall \tilde{u}(\mathbf{x}) \in \tilde{\mathcal{U}}, \exists k, \exists i : P_i(C|\tilde{u}(\mathbf{x}), [t_k, t_{k+1})) = 1 \\
&\Leftrightarrow \forall \tilde{u}(\mathbf{x}) \in \tilde{\mathcal{U}}, \exists k : P(C|\tilde{u}(\mathbf{x}), [t_k, t_{k+1})) = 1 \\
&\Leftrightarrow \forall \tilde{u}(\mathbf{x}) \in \tilde{\mathcal{U}} : P(C|\tilde{u}, [0, \infty)) = 1 - \prod_{k=0}^{\infty} (1 - P(C|\tilde{u}(\mathbf{x}), [t_k, t_{k+1}))) = 1 \\
&\Leftrightarrow \mathrm{PCS}(\mathbf{x}) = \min_{\tilde{u}\in\tilde{\mathcal{U}}} P(C|\tilde{u}(\mathbf{x}), [0, \infty)) = 1.
\end{aligned}$$

In the next section, the concept of PCS is used to define the overall collision probability (OCP) which defines the collision probability of a trajectory during and beyond the planning horizon.

3.7. Overall Collision Probability

In Sec. 1.2.4 the PMP approach is briefly described which uses the concept of ICS to verify that the partial trajectories will not end in a collision beyond the planning horizon. In the previous

section, the probabilistic extension of ICS, the PCS concept is introduced which allows extending the complete PMP approach to a probabilistic setting. Therefore, the safety of a trajectory $\tilde{u}$ is determined by the collision probability of the trajectory $P(C|\tilde{u})$ during the planning horizon and the PCS value of its last state $\mathrm{PCS}(\tilde{u})$. The collision probability of the trajectory considers the time interval $I_t = [0, T_h]$ assuming that the trajectory starts at $t = 0$ and that T_h is the planning horizon. The PCS calculation indicates the future collision probability for the time interval $I_t^+ = (T_h, \infty)$ beyond the planning horizon. Both collision probabilities are combined into the overall collision probability P^∞ which represents the collision probability of a trajectory for an infinite time horizon:

Definition 9 (**Overall Collision Probability).** *The probability of a trajectory leading to a collision during or after the trajectory, is defined as*

$$P^\infty(C|\tilde{u}) = 1 - (1 - \underbrace{P(C|\tilde{u})}_{t \in I_t})(1 - \underbrace{\mathrm{PCS}(\tilde{u})}_{t \in I_t^+}).$$

In the literature, to the best knowledge of the author, safety assessment of a robot trajectory has never be done in a probabilistic fashion including reasoning about the future collision probability beyond the planning horizon. It should be noted that it is possible to introduce different weights for both collision probabilities. Depending on the confidence of both estimated collision probabilities, the weights could be adopted. This is especially relevant for real-world applications.

3.8. Probabilistic Collision Costs

In general, the aim of the robot is to reach its goal state while minimizing a tradeoff between navigation costs and the collision probability. Regarding an uncertain environment, it is impossible to guarantee a collision-free trajectory, since in most cases a small collision probability remains due to uncertainties. But not all possible occurring collisions are equal with respect to their risk or possible harm: a collision with a very fast and heavy object has a very high potential of danger compared to a slow and light one. Therefore, the risk function

$$\mathrm{risk}_{\mathrm{coll}}(\tilde{u}) = P(C|\tilde{u}) \ \mathrm{cost}_{\mathrm{coll}}$$

is introduced to rank each collision, as proposed by [66]. The cost of collision function $\mathrm{cost}_{\mathrm{coll}}$ (severity of crash), which only depends on the square of the vehicle speeds, is used in [66]. This is replaced by the *internal energy* assuming an inelastic impact

$$\mathrm{cost}_{\mathrm{coll}} = \underbrace{\frac{1}{2} \frac{\mathrm{mass}_i \, \mathrm{mass}_j}{\mathrm{mass}_i + \mathrm{mass}_j} \|\mathbf{v}_r\|^2}_{\text{internal energy}},$$

where $\mathbf{v}_r$ is the relative velocity and $\mathrm{mass}_i, \mathrm{mass}_j$ are the weights of the colliding objects i and j. The maximum internal energy, which could occur between any objects, is used to normalize

the collision cost, thus

$$\overline{\text{cost}}_{\text{coll}} \in [0, 1].$$

The normalized collision cost is used to calculate the probabilistic collision cost (PCC). Therefore, the indicator function (A.8) from Sec. A.3 is replaced by

$$\text{Ind}_{\text{cost}}(C|\tilde{u}, \mathbf{u}_i) = \begin{cases} \overline{\text{cost}}_{\text{coll}}, & \text{collision detected} \\ 0, & \text{collision free} \end{cases}$$

and leads to the probabilistic collision cost

$$\text{PCC}_i(\tilde{u}) = \int_{\hat{\tilde{u}}} \text{Ind}_{\text{cost}}(C|\tilde{u}, \mathbf{u}_i) f(\mathbf{u}_i)\, d\mathbf{u}_i, \;\; \mathbf{u}_i \in \hat{\tilde{\mathcal{U}}}.$$

This integral is approximated by Monte Carlo sampling as described in Sec. A.3. The difference between the pure collision probability and the probabilistic collision cost is evaluated by simulations in Sec. 3.10.5.

3.9. Implementations

In this section example implementations of the above presented approaches and definitions are shown. Two novel ICS checkers based on Sec. 3.4 are presented. They are also used in the novel ICS based navigation algorithm which considers the robot maneuverability from Sec. 3.5. Furthermore, an example implementation for the probabilistic collision state checker based on Sec. 3.6 is shown which is also used for the evaluation of the overall collision probability from Sec. 3.7. Afterwards, these implementations are evaluated by different simulation scenarios in Sec. 3.10.

3.9.1. Inevitable Collision State Checkers

The ICS Def. 5 is not directly implementable for two reasons: there is an infinite number of input trajectories and an unlimited time horizon. The infinite number of input trajectories $\tilde{u}$ of the robot is approximated by computing a finite subset of input trajectories. According to ICS Prop. 1, this leads to a conservative computation of ICS.

The problem of computing with an infinite time horizon can be solved by applying only maneuvers that come to a standstill after a finite time horizon. Since the computational effort increases with time, the main focus lies on braking maneuvers which come to a standstill within a reasonable time horizon. So only a subset $\mathcal{I} \subset \tilde{\mathcal{U}}$ is used for the computation. In the following two ICS-Checker algorithms are presented. The first one considers only the trajectory set $\mathcal{I}^{\text{a}}$ that contains all admissible trajectories $\tilde{u}$ which are not colliding with the known objects $\mathcal{B}$. Therefore, the set $\mathcal{B}^{\text{a}}$ is introduced which combines all objects which have already been used for the ICS evaluation. An overview of the sequential ICS-Checker is given in Alg. 2.

Additional to the ICS property of a state $\mathbf{x}$, the algorithm also determines all trajectories of $\mathcal{I}$ which are collision-free regarding all objects $\mathcal{B}$. The aim of the second ICS-Checker is to determine as fast as possible the ICS status of the state $\mathbf{x}$. Hence, it checks sequentially or

Algorithm 2: ICS Checker

Input : $\mathbf{x}, \mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{N_\mathrm{b}}\}$
Output : ICS flag, $\mathcal{I}^\mathrm{a}$

Initialize: Select $\mathcal{I} \subset \tilde{\mathcal{U}}$, $\mathcal{I}^\mathrm{a} \leftarrow \mathcal{I}$, $\mathcal{B}^\mathrm{a} \leftarrow \emptyset$
for $i \leftarrow 1$ **to** N_b **do**
  $\mathcal{I}^\mathrm{a}(\{\mathcal{B}_i, \mathcal{B}^\mathrm{a}\}) \leftarrow \mathrm{ICS}(\mathbf{x}, \mathcal{B}_i, \mathcal{I}^\mathrm{a}(\mathcal{B}^\mathrm{a}))$
  $\mathcal{B}^\mathrm{a} = \{\mathcal{B}_i, \mathcal{B}^\mathrm{a}\}$
  if $\mathcal{I}^\mathrm{a}(\mathcal{B}^\mathrm{a}) = \emptyset$ **then**
    return true, $\emptyset$
return false, $\mathcal{I}^\mathrm{a}$

parallel all trajectories $\tilde{u}$ if one exists which is collision-free regarding all objects $\mathcal{B}$. If such a trajectory is found, the state $\mathbf{x}$ is not an ICS and the algorithm returns. Compared to the previous algorithm, only one trajectory is known to be collision-free and not the set $\mathcal{I}^\mathrm{a}$. An overview of the ICS-Checker is given in Alg. 3.

Algorithm 3: ICS Checker 2

Input : $\mathbf{x}, \mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{N_\mathrm{b}}\}$
Output : ICS flag, $\tilde{u}^\mathrm{a}$

Initialize: Select $\mathcal{I} \subset \tilde{\mathcal{U}}$
foreach $\tilde{u} \in \mathcal{I}$ **do**
  collfree = true
  for $i \leftarrow 1$ **to** N_b **do**
    if $\mathrm{ICS}(\mathbf{x}, \mathcal{B}_i, \tilde{u})$ **then**
      collfree = false
      break
  if *collfree* $=$ *true* **then**
    return false, $\tilde{u}^\mathrm{a} = \tilde{u}$
return true, $\emptyset$

Compared to all other known ICS implementations [23, 83, 84, 90], these two are more efficient since only collision-free trajectories are considered. In Sec. 3.10.2, simulation results of the sequential computation of ICS sets according to Sec.3.4 are presented.

3.9.2. Probabilistic Collision State Checker

The implementation of the PCS Def. 8 suffer from the same problems as the implementation of ICS: there is an infinite number of input trajectories and an unlimited time horizon. The infinite number of input trajectories $\tilde{u}$ of the robot, is solved by computing with a finite subset of input trajectories. This leads to a conservative computation of PCS. The problem of computing with an infinite time horizon can be solved by applying only maneuvers that come to a standstill after a finite time horizon. Since the computational effort increases with time, the main focus lies

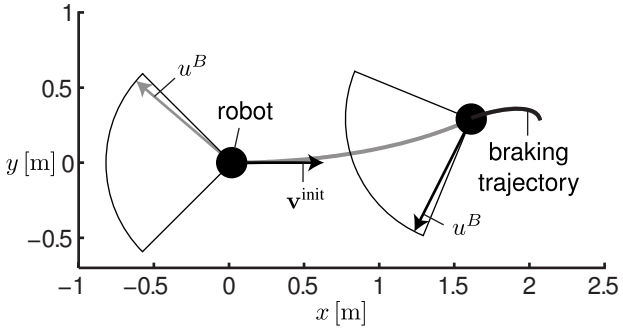


Fig. 3.4.: Example generation of a braking trajectory by two different acceleration directions. The direction of the acceleration is given in the relative coordinate system of the object.

on braking maneuvers which come to a standstill in a reasonable time horizon. A PCS checker is introduced which is based on Monte Carlo simulation allowing one to investigate multiple braking trajectories of each object. As described in Sec. 3.6, the collision probability $P(C|\tilde{u}(\mathbf{x}))$

$$\mathrm{PCS}(\mathbf{x}) := \min_{\tilde{u}\in\tilde{\mathcal{U}}} P(C|\tilde{u}(\mathbf{x}))$$

is used to calculate the PCS of a robot state $\mathbf{x}$. Therefore, a subset of input trajectories $\tilde{u}$ needs to be generated and then the collision probability $P(C|\tilde{u}(\mathbf{x}))$ is estimated for each trajectory. For estimating $P(C|\tilde{u}(\mathbf{x}))$, Monte Carlo simulation is used as explained in Sec. A.3. The robot and object trajectories need to be generated in order to get discretized object states $\mathbf{s}_i(\mathbf{u}_i)$ based on the discretized control inputs $\mathbf{u}_i$.

Generation of Braking Trajectories

Trajectories for the objects need to be generated in order to get the discretized object states $\mathbf{s}_i(\mathbf{u}_i)$. The SIS algorithm presented in Sec. A.3 is used for generating the braking trajectories for the robot and the workspace objects. Since only braking trajectories are used, the reduced control input set $\mathcal{U}^B \subset \mathcal{U}$ is applied and the time horizon T_h is replaced by the braking time T_b. The time interval in which all objects come to a standstill is denoted by $I_t = [0, T_b]$. The control inputs are given in the relative coordinate system of the object and not in the global workspace coordinates. The control input $u^B \in \mathcal{U}^B$ is given in polar coordinates $[u_r^B, u_\theta^B]$ with azimuth $u_\theta^B \in (\frac{3}{4}\pi, -\frac{3}{4}\pi)$ and the radius $u_r^B \in [a^{\min}, a^{\max}]$. An example braking trajectory is illustrated in Fig. 3.4. Under the assumption of $a^{\min} > 0$, the object comes to a standstill within a finite time horizon. Furthermore, it is assumed that the weights of the sampled trajectories are equal

$$f(\mathbf{u}) = \text{const}, \quad \forall \mathbf{u} \in \hat{\tilde{\mathcal{U}}}^B, \text{ with}$$

$$\hat{\tilde{\mathcal{U}}}^B = \underbrace{\mathcal{U}^B \times, \ldots, \times \mathcal{U}^B}_{N_\mathrm{c} \text{ Cartesian products}}.$$

meaning that there is no preferred trajectory. Hence, the resulting collision probability $P_i(C|\tilde{u})$ for one robot trajectory $\tilde{u}$ is the ratio of the number of trajectories leading to a collision and the number of all evaluated trajectories of the ith object.

It is also possible to generate not only pure braking trajectories in order to get a less conserva-

tive approximation of PCS. The only requirement is, that all objects come to a standstill within a finite time horizon. But simulating longer trajectories increases the computational effort which could be used for the safety assessment of more trajectory candidates. Sec. 3.10.3 evaluates the influence of the number of samples on the PCS calculation.

3.10. Simulations

In this section simulation scenarios are used to evaluate the example implementations from Sec. 3.9. First, two novel ICS checkers based on Sec. 3.9.1 are evaluated together with the influence of the robot maneuverability to the collision probability regarding unexpected objects. Therefore, a simulation setup as in [84] is used. Afterwards, different sampling strategies are applied to the presented PCS checker from Sec. 3.9.2. This allows one to compare the OCP to the collision probability of a trajectory by random scenarios containing multiple objects.

3.10.1. Inevitable Collision State Checkers

Robot Model The state $\mathbf{x}$ of the robot is represented by its position $\mathbf{p}$ and its velocity $\boldsymbol{v}$. The dynamics of the robot is determined by the nonlinear differential equation $\dot{\mathbf{x}}(t) = m(\mathbf{x}(t), u(t))$

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix}}_{\dot{\mathbf{x}}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ u_1 \\ u_2 \end{bmatrix}}_{u} a^{\max} \tag{3.5}$$

with respect to the velocity constraint $\sqrt{v_x^2 + v_y^2} \leq v^{\max}$ and the acceleration constraint $\sqrt{a_x^2 + a_y^2} \leq a^{\max}$. For the simulation, $v^{\max} = 3\,\frac{\mathrm{m}}{\mathrm{s}}$ and $a^{\max} = 2\,\frac{\mathrm{m}}{\mathrm{s}^2}$ was used. The disc-shaped robot has a radius of $2\,\mathrm{m}$.

Workspace Description The workspace $\mathcal{W}$ has a size of $100\,\mathrm{m} \times 100\,\mathrm{m}$ and is populated with 15 moving objects. These are ignoring dynamic objects as described in Sec. 1.2.2. Identical to [84] the objects' trajectories are modeled as closed B-splines with 10 random control knots. The disc-shaped objects stir with a random constant velocity between $1 \rightarrow 2\frac{\mathrm{m}}{\mathrm{s}}$ and their radius is $2\,\mathrm{m}$. A snapshot of the simulation environment is depicted in Fig. 3.5.

To prevent the robot to drive and stay at corner points, where usually no object trajectories disturb the robot, the workspace of the robot $\mathcal{W}_R$ is limited to $50\,\mathrm{m} \times 50\,\mathrm{m}$ and centrally positioned in $\mathcal{W}$.

Additional to the known workspace objects, 5 random static objects with radius $2\,\mathrm{m}$ are placed every $5\,\mathrm{s}$ at a random position in $\mathcal{W}_R$ which has a minimum distance of $6\,\mathrm{m}$ to the robot to prevent instantaneous collisions.

Navigation Algorithm The robot is applied with the ICS-Avoid algorithm from [84]. An overview is given in Alg. 4. The robot system has a fixed number of control inputs $\mathcal{J}$ and the resulting states $\mathbf{x}$ are checked one after another for $\mathbf{x} \in \mathrm{ICO}(\mathcal{B})$. The trajectories are generated

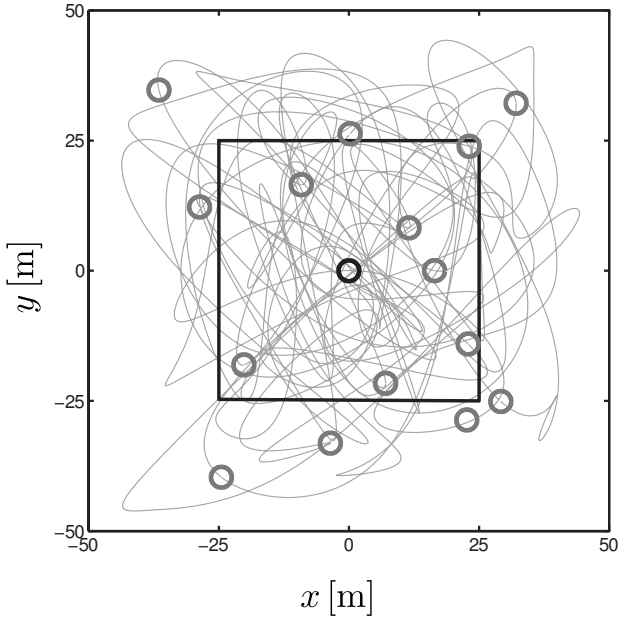


Fig. 3.5.: Snapshot of the workspace, gray circles depict the 15 objects and the gray lines depict the associated trajectories. The robot is shown as a black circle and the inner black square illustrates the workspace of the robot.

with the sampling time $T_s = 0.1\,\mathrm{s}$ and the control inputs are constant for $T_c = 1.0\,\mathrm{s}$. The set $\mathcal{J}$ contains five different control inputs $[u_1, u_2] = \{[0,0],[1,0],[-1,0],[0,1],[0,-1]\}$. The set of collision-free trajectories $\mathcal{K}$ of the ICS-Checker is defined as the safe control kernel. For the next time step, the safe control kernel is added to the set of control inputs $\mathcal{J}$. Thus, ICS-Avoid can guarantee a fallback mechanism due to the safe control kernel.

Algorithm 4: ICS-Avoid [84]

Input : $\mathbf{x}(t), \mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{N_\mathrm{b}}\}, \mathcal{I}, \mathcal{J}(t)$
Output : $u, \mathcal{J}(t+T_c)$

foreach $u \in \mathcal{J}(t)$ **do**
  $\mathbf{x}(t+T_c) = \mathbf{x}(t) + \int_t^{t+T_c} m(\mathbf{x}(t), \mathbf{u})\, dt$
  if $\mathrm{ICS}(\mathbf{x}(t+T_c)) = 0$ **then**
    $\mathcal{K} = \{u \in \mathcal{I} |\, \mathrm{ICS}(\mathbf{x}(t+T_c), \mathcal{B}, u) = 0\}$
    $\mathcal{J}(t+T_c) = \{\mathcal{I}, \mathcal{K}\}$
    return $u, \mathcal{J}(t+T_c)$

As shown in Sec. 3.5, it is possible to decrease the probability of being in an ICS regarding an unexpected object by increasing the robot maneuverability. Therefore, the ICS-Avoid algorithm is extended so that the maneuverability of the robot is taken into account. The algorithm is shown in Alg. 5. The robot chooses the control input $\mathbf{u} \in \mathcal{J}$ which maximizes its maneuverability, thus increasing its motion safety. While the robot is navigating in the workspace by applying ICS-Avoid, the performance of three different ICS-Checkers are evaluated: ICS-Checker by [83], ICS-Checker from Alg. 2 and ICS-Checker from Alg. 3. All algorithms use the same set of

Algorithm 5: ICS-Avoid including maneuverability

Input : $\mathbf{x}(t), \mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{N_\mathrm{b}}\}, \mathcal{I}, \mathcal{J}(t)$
Output : $\mathbf{u}^*, \mathcal{J}(t + T_c)$

Initialize: $\mathcal{K} \leftarrow \emptyset$
foreach $\mathbf{u} \in \mathcal{J}(t)$ **do**
  $\mathbf{x}(t + T_c) = \mathbf{x}(t) + \int_t^{t+T_c} m(\mathbf{x}(t), u)\, dt$
  if $\mathrm{ICS}(\mathbf{x}(t + T_c)) = 0$ **then**
    Compute Maneuverability $M(\mathbf{x}(t + T_c), \mathcal{B})$
$u^* = \arg\max_u M(\mathbf{x}(t + T_c), \mathcal{B})$
$\mathbf{x}^* = \mathbf{x}(t) + \int_t^{t+T_c} m(\mathbf{x}(t), u^*)\, dt$
$\mathcal{K} = \{u \in \mathcal{I} | \, \mathrm{ICS}(\mathbf{x}^*, \mathcal{B}, u) = 0\}$
$\mathcal{J}(t + T_c) = \{\mathcal{I}, \mathcal{K}\}$
return $u^*, \mathcal{J}(t + T_c)$

Tab. 3.1.: ICS-Checker performance

Algorithm	**Checks**		**Mean** 10^{-3}[s]
	$\overline{N}_C$	Δ to [83] [%]	
ICS-Checker [83]	140.00	-	17.93
ICS-Checker Alg. 2	74.64	46.69	10.37
ICS-Checker Alg. 3	58.15	58.46	7.74

trajectories which contains only pure braking trajectories. The braking trajectories are generated based on a constant deceleration in the robot coordinate frame. As presented in the previous section (Sec. 3.9.2), the robot comes to a standstill within a finite time horizon, if the direction of the resulting acceleration vector is chosen from the interval $(\frac{3}{4}\pi, \frac{5}{4}\pi)$ and its magnitude is greater than zero. For this evaluation a fixed control input is applied till the robot comes to a standstill. 7 different directions are considered with an equidistant step size of 0.2 beginning at $\frac{3}{4}\pi$ and the corresponding magnitude is chosen, such that the duration of the longest braking time lasts not longer than $5\,\mathrm{s}$. The workspace was populated by 20 known objects and their trajectories are known for a time horizon of $5\,\mathrm{s}$. For comparison, the mean number of trajectory checks $\overline{N}_C$ and the mean computation times are listed in Tab. 3.1 for 1654 ICS calculations. The standard ICS-Checker from [83] needs to evaluate 7 braking trajectories for all 20 workspace objects. Hence, 140 checks are needed to determine the ICS property of one state. The ICS-Checker form Alg. 2 needs 46.69% less collision checks compared to the ICS-Checker presented in [83], despite both algorithms have the same result containing the ICS property of the state and the set of admissible trajectories. The ICS-Checker Alg. 3 needs even less collision checks, since only the ICS property is computed. In the following section, random workspaces are generated to evaluate the two different ICS-Avoid algorithms according to their motion safety.

Tab. 3.2.: Evaluation of ICS-Avoid

Alg.	Run	Collisions						Maneuverability					
		$T_h = 1\text{s}$		$T_h = 3\text{s}$		$T_h = 5\text{s}$		$T_h = 1\text{s}$		$T_h = 3\text{s}$		$T_h = 5\text{s}$	
		$\mathcal{B}$	$\mathcal{B} \cup \mathcal{B}_u$	$\mathcal{B}$	$\mathcal{B} \cup \mathcal{B}_u$	$\mathcal{B}$	$\mathcal{B} \cup \mathcal{B}_u$	$\mathcal{B}$	$\mathcal{B} \cup \mathcal{B}_u$	$\mathcal{B}$	$\mathcal{B} \cup \mathcal{B}_u$	$\mathcal{B}$	$\mathcal{B} \cup \mathcal{B}_u$
Alg. 4	1	25	26	4	10	4	9	0.50	0.42	0.67	0.53	0.54	0.48
	2	27	33	3	7	0	4	0.40	0.38	0.68	0.63	0.67	0.55
	3	14	38	6	15	5	15	0.55	0.37	0.58	0.46	0.62	0.40
	4	25	35	4	15	4	10	0.47	0.43	0.66	0.46	0.55	0.42
	5	18	31	7	12	1	12	0.57	0.40	0.59	0.47	0.64	0.43
	Mean	**21.8**	**32.6**	**4.8**	**11.8**	**2.8**	**10.0**	**0.50**	**0.50**	**0.63**	**0.51**	**0.60**	**0.46**
Alg. 5	1	19	29	5	5	5	3	0.55	0.47	1.00	0.80	0.78	0.87
	2	12	15	1	9	0	6	0.70	0.65	0.95	0.70	0.90	0.73
	3	16	15	3	17	2	7	0.66	0.63	0.86	0.56	0.84	0.77
	4	19	22	5	12	1	6	0.63	0.51	0.77	0.60	0.98	0.68
	5	12	28	0	7	2	4	0.73	0.55	0.97	0.80	0.85	0.75
	Mean	**15.6**	**21.8**	**2.8**	**10.0**	**2.0**	**5.2**	**0.65**	**0.56**	**0.91**	**0.69**	**0.87**	**0.76**

3.10.2. Motion Safety Regarding Unexpected Objects

For validating the concept of Sec. 3.5, improving safety by increasing the maneuverability of the robot, the two different ICS-Avoid algorithms are evaluated in the same simulation environment. Therefore, 5 different random workspaces (runs) are generated which are identical for both algorithms. As presented in [83], the future trajectories of the objects is only provided for a fixed time horizon T_h. Three different time horizons are considered: 1, 3 and $5\,\text{s}$. Each run was evaluated according to the number of collisions regarding the known objects $\mathcal{B}$, while ignoring the unexpected objects $\mathcal{B}_u$, and according to the total number of collisions regarding all objects $\mathcal{B} \cup \mathcal{B}_u$. The number of collisions considering only the unexpected objects $\mathcal{B}_u$ is not discussed, since the robot may collide with a known object while avoiding an unknown one. Furthermore, the maneuverability is determined for each run and the simulation results are summarized in Tab. 3.2.

The ICS-Avoid algorithm maximizing the maneuverability of the robot reduces the average number of collisions compared to the other one. Regarding the known objects $\mathcal{B}$, the average number of collision for Alg. 4 is 9.80 and for Alg. 5 is 6.80 which is a relative difference of 30.61%. The difference for the unknown objects $\mathcal{B}_u$ is more significant. The average number of collisions for Alg. 4 is 18.13 and for Alg. 5 is 12.33 which is a relative difference of 31.99%. The reason for the difference regarding the known objects is because of the limited prediction horizon T_h, thus no significant difference can be observed for objects which prediction is known for longer time horizons. The time horizon T_h has less influence on the difference between both algorithms regarding all workspace objects $\mathcal{B} \cup \mathcal{B}_u$. The biggest relative difference of the mean

Tab. 3.3.: Simulation parameters for different sampling strategies.

strategy	$a^{\min}\ \left[\frac{\mathrm{m}}{\mathrm{s}^2}\right]$	$T_{\mathcal{U}}\ [\mathrm{s}]$
hard braking trajectories	0.4	$-$
soft braking trajectories	0.01	$-$
various trajectories	0.4	1.0

Tab. 3.4.: Initial states of the objects. The velocity is given in the object coordinate frame.

object	$\mathbf{p}(t_0)\ [\mathrm{m}]$	$\boldsymbol{v}(t_0)\ \left[\frac{\mathrm{m}}{\mathrm{s}}\right]$
black	$[0.0,\ \ 0.0]$	$[1.5, 0.0]$
gray	$[1.8, -0.2]$	$[1.5, 0.0]$

values is 48.00% ($10.0 \rightarrow 5.2$) and occurs with the longest time horizon $T_h = 5\mathrm{s}$. This is mainly because most collisions occur due to the unforeseen objects which confirms the influence of the maneuverability to the motion safety of the robot regarding unexpected objects.

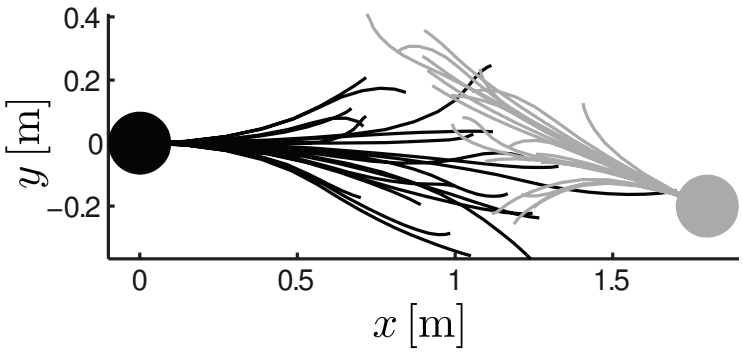
3.10.3. Probabilistic Collision State Checker

The PCS checker presented in Sec. 3.9.2 is designed to assess the safety of a robot state in the presence of multiple uncertain objects. The quality of the PCS analysis depends on the sampling algorithm and on the number of samples. Three different sampling strategies are evaluated. The first two (hard, soft braking) are the same as described in Sec. 3.9.2, with different values for the minimum applied acceleration $a^{\min}$. The third one (various trajectories) uses the complete input space $\mathcal{U}$ instead of pure braking trajectories $\mathcal{U}^B$. Since it is still necessary that all workspace objects come to a standstill, the complete input space $\mathcal{U}$ is only applied for a fixed time horizon $T_{\mathcal{U}}$ and afterwards a braking trajectory is generated by using the first strategy. In Fig. 3.6 one example scenario for each strategy is depicted and Tab. 3.3 shows the simulation parameters.

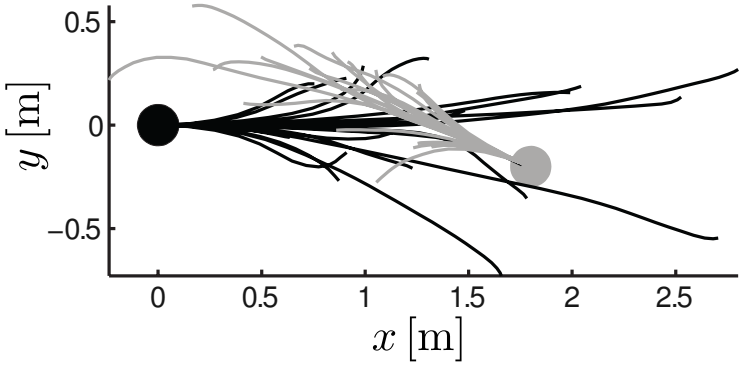
The number of samples N_s are varied between 10 and 200 samples. Each configuration was repeated 10 times to obtain the variance of the estimations. The initial states of the two objects are shown in Tab. 3.4. The results of the PCS analysis for the different strategies are shown in Fig. 3.7.

The final values of the PCS computations are in the same range for all three strategies ($0.01 \rightarrow 0.02$). But the approaches have a big difference, if only few samples are used. One reason for this is, that the final states of the objects are very close for the first strategy due to the initial state of the objects and the used $a^{\min}$. This is relaxed by the second strategy, applying a lower $a^{\min}$. The third strategy has the largest result interval $0.13 \rightarrow 0.02$ which is caused by the biggest input space. As discussed already in Sec.3.9.2, the strategy with the biggest input space using enough samples will have the most representative result. But instead of using many and long[1] samples for the PCS computation, one can evaluate more trajectory candidates. Finally, these results are not general, since there is a big dependency on the investigated scenario and the PCS computation is usually not used isolated. It is still an open problem, how the available

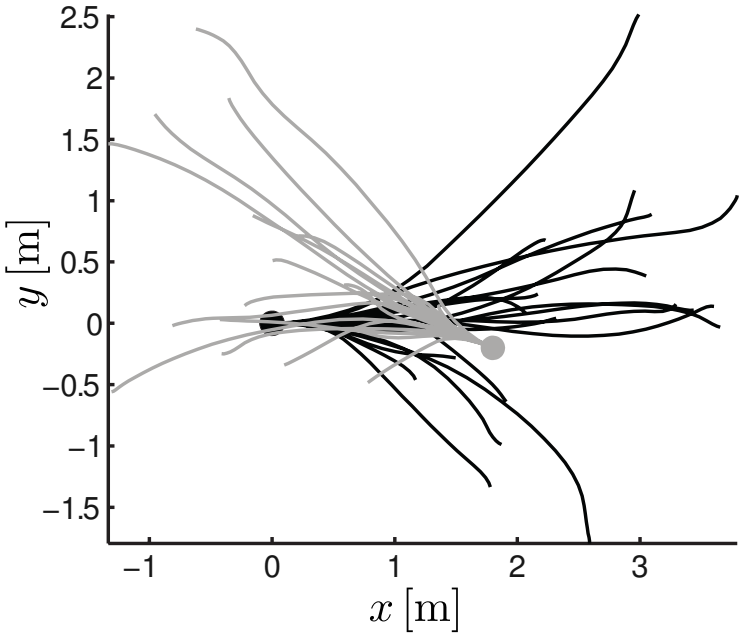
[1] in the sense of time



(a) Hard braking trajectories

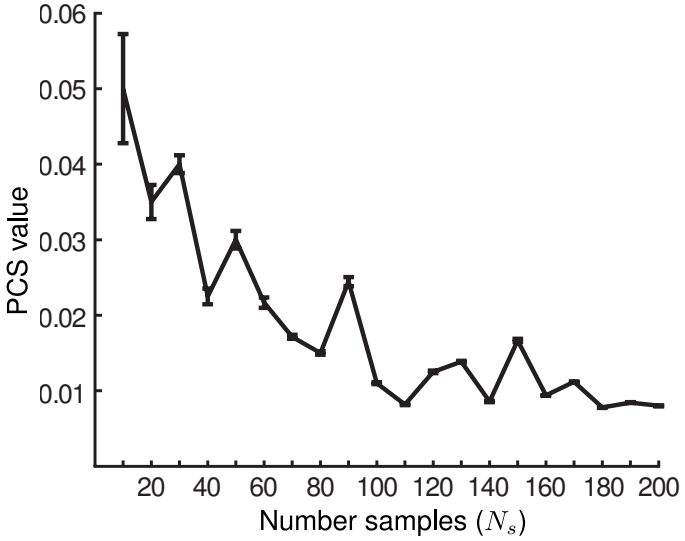


(b) Soft braking trajectories

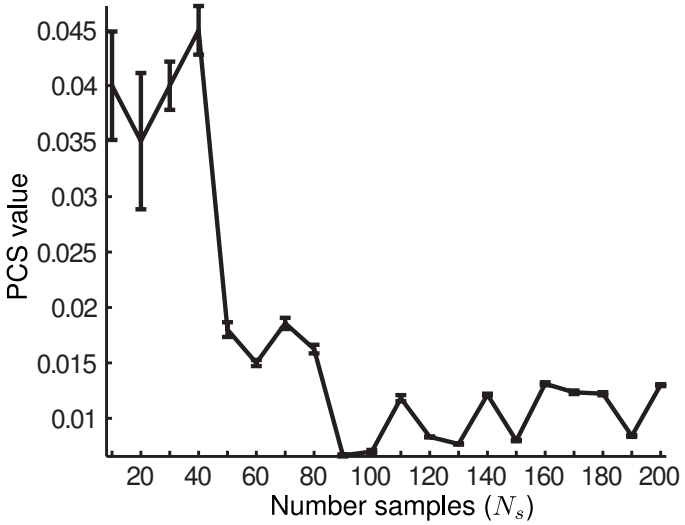


(c) Various trajectories

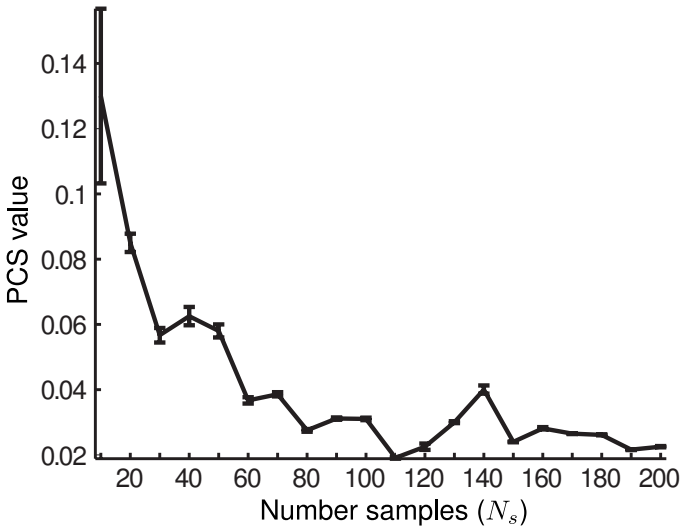
Fig. 3.6.: Example scenarios used to evaluate the three PCS strategies. Each image shows 20 samples of each object.



(a) Hard braking trajectories



(b) Soft braking trajectories



(c) Various trajectories

Fig. 3.7.: Illustrating the result of the PCS analysis for the three different sampling strategies. The error bars indicate the standard error of 10 iterations.

Tab. 3.5.: Object parameters

30 regions for $x[\mathrm{m}]$	1 region for $y[\mathrm{m}]$	Init $\sphericalangle \boldsymbol{v}[\mathrm{rad}]$	Init $\|\boldsymbol{v}\|[\frac{\mathrm{m}}{\mathrm{s}}]$	$v^{\max}[\frac{\mathrm{m}}{\mathrm{s}}] a^{\max}[\frac{\mathrm{m}}{\mathrm{s}^2}]$		$\Sigma(t_0)$			
$[0.4, 0.6] \cdots$ $[6.2, 6.4]$	$[-1.0, 1.0]$	$[\frac{3}{4}\pi, \frac{5}{4}\pi]$	$[1.0, 2.0]$	2.0	2.0	$\begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$			

Tab. 3.6.: Parameters for trajectory generation.

$P(C\|\tilde{u})$				$\mathrm{PCS}(\tilde{u})$	
T_s [s]	T_c [s]	$T_h[s]$	N_s	N_s	$a^{\min}[\frac{\mathrm{m}}{\mathrm{s}^2}]$
0.025	0.25	1.0	20	5	1.0

computation time is divided for the generation of the trajectories and their corresponding PCS analysis.

3.10.4. Overall Collision Probability

In order to show the usefulness of the overall collision probability (Sec. 3.7), random scenarios are generated and the difference between the overall and the trajectory collision probability is determined. Despite the workspace objects, the initial state of the robot is fixed and has the initial state $\mathbf{x} = \begin{bmatrix} 0\mathrm{m} & 0\mathrm{m} & 1.5\frac{\mathrm{m}}{\mathrm{s}} & 0\frac{\mathrm{m}}{\mathrm{s}} \end{bmatrix}^T$. The objects are placed randomly in front of the robot facing towards it. The workspace objects are placed randomly in one of thirty predefined adjacent regions which are partitioned in x-direction. The evaluated regions and other parameters for the objects are listed in Tab. 3.5. Furthermore, 10 random robot trajectories are generated for each scenario. Each of the scenarios is evaluated 50 times. An example scenario using the listed parameters is shown in Fig. 3.8. For verification, the mean $\overline{D}$ and the maximum difference $D_{\max}$ of $P^{\infty}(C|\tilde{u}) - P(C|\tilde{u})$ are obtained from all scenarios and the results are shown in Fig. 3.9. It can be seen that there is a significant difference between the collision probability of the trajectory and the overall collision probability. The maximum achieved difference is 86%. It can also be seen that the improvement depends on the distance to the objects when assuming the velocity range and direction as listed in Tab. 3.5 for the robot and the objects.

3.10.5. Probabilistic Collision Cost

In Sec. 3.8 the probabilistic collision cost is introduced which is an alternative for the collision probability. Instead of checking if a collision occurs, the strength of the collision is checked, also called collision cost. In Fig. 3.10 and 3.11 two different scenarios are evaluated with two different cost functions. The applied simulation parameters are shown in Tab. 3.7. The first cost function only consists of $P(C|\tilde{u})$ and second one of $\mathrm{PCC}(\tilde{u})$. There is no big difference in the first scenario, since the possible collisions occur with nearly maximum relative speed. The mean difference is 0.2. In the second scenario the difference is larger, since the objects are facing in a

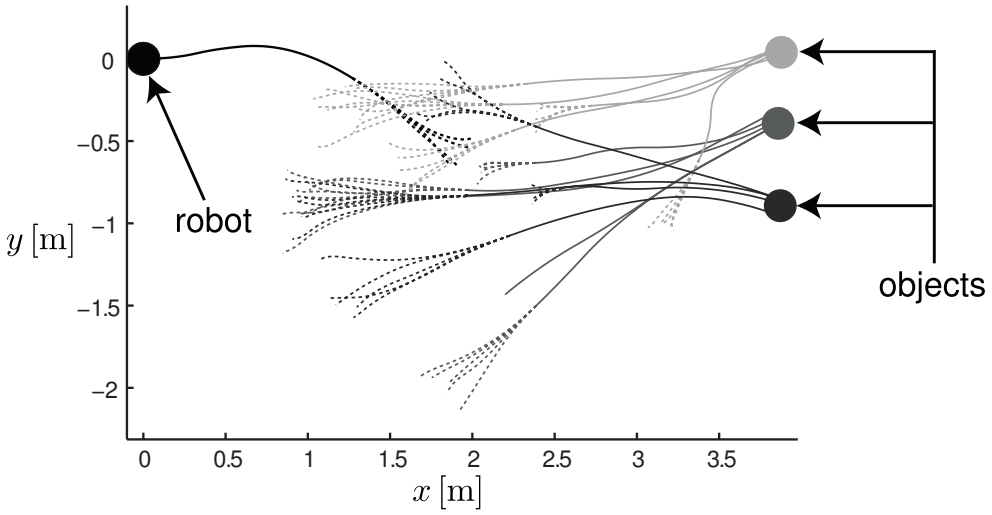


Fig. 3.8.: Random scenario for region 18 in x-direction: the solid lines depict the trajectories within the planning phase and the dashed lines depict the braking trajectories for the PCS calculation.

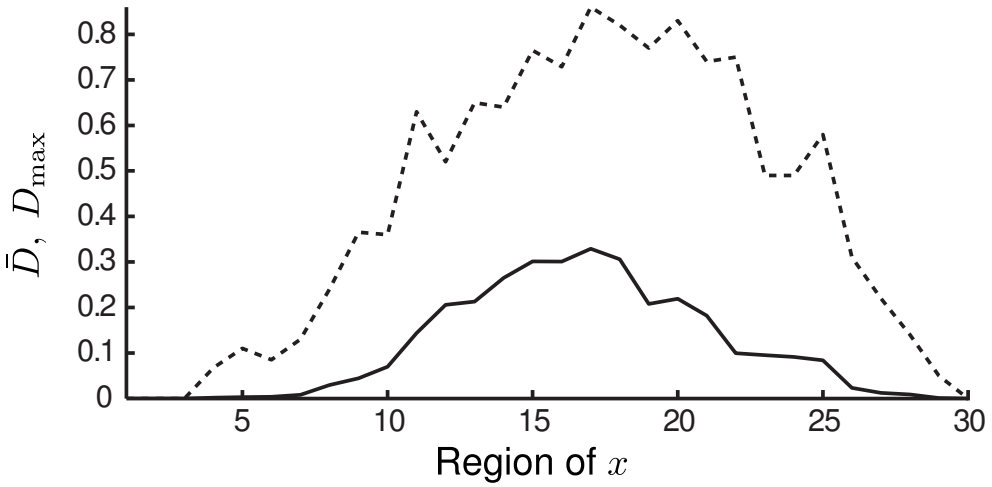
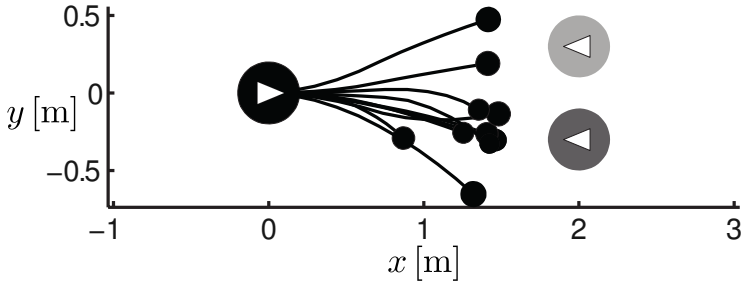


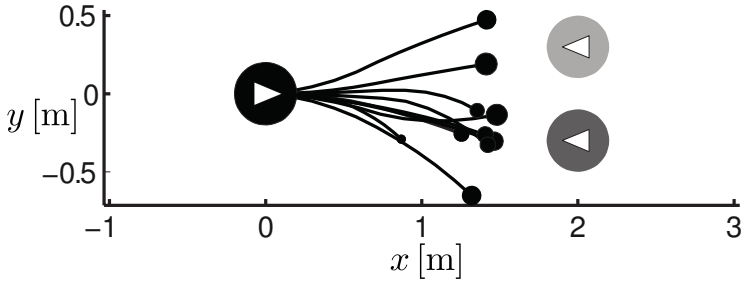
Fig. 3.9.: The solid line depicts the mean difference and the dashed line depicts the maximum difference of one x - region between $P(C|\tilde{u})$ and $P^{\infty}(C|\tilde{u})$.

Tab. 3.7.: Initial states of the objects. The velocity is given in the object coordinate frame.

object	simulation 1		simulation 2	
	$\mathbf{p}(t_0)\,[\mathrm{m}]$	$\boldsymbol{v}(t_0)\,\left[\frac{\mathrm{m}}{\mathrm{s}}\right]$	$\mathbf{p}(t_0)\,[\mathrm{m}]$	$\boldsymbol{v}(t_0)\,\left[\frac{\mathrm{m}}{\mathrm{s}}\right]$
black	$[0.0,\ \ 0.0]$	$[1.5,\ \ 0.0]$	$[0.0,\ \ 0.0]$	$[\ \ 1.5, 0.0]$
light gray	$[0.5,\ \ 1.0]$	$[0.6, -0.6]$	$[2.0,\ \ 0.3]$	$[-1.5, 0.0]$
dark gray	$[0.5, -1.0]$	$[0.6,\ \ 0.6]$	$[2.0, -0.3]$	$[-1.5, 0.0]$



(a) Collision probability $P(C|\tilde{u})$.



(b) Probabilistic collision costs $\mathrm{PCC}(\tilde{u})$.

Fig. 3.10.: The same scenario is evaluated by two different cost functions. The diameter of the circles is proportional to the cost of each trajectory.

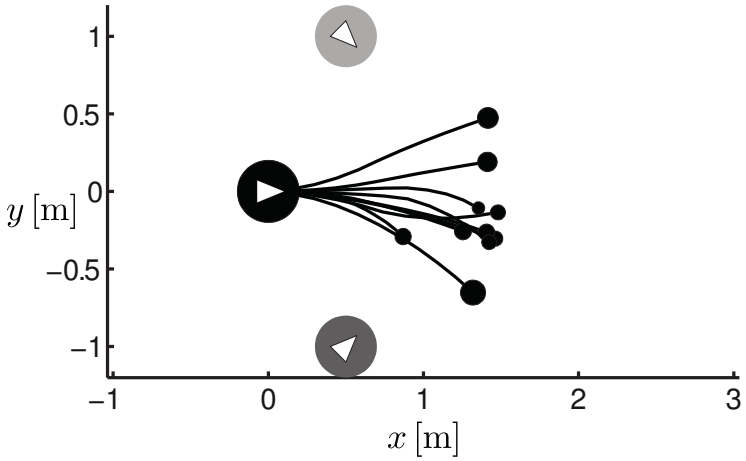
similar direction and thus the relative speed is much lower, resulting in lower collision costs. The mean difference is 0.46 resulting in the relative difference of 44%. The probabilistic collision cost seems to be promising for crowded scenarios if the workspace objects can bear up against weak[2] collisions.

3.11. Discussion

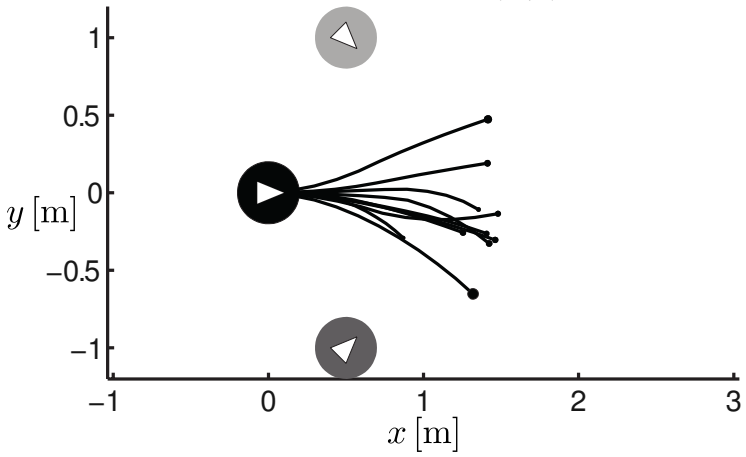
In this chapter, different novel definitions and approaches for assessing the safety of partial trajectories beyond the planning horizon are presented. They comprise methods for deterministic and stochastic modeled environments. Moreover, the motion safety regarding unexpected objects is discussed addressing the problem of imperfect perception capabilities of the robot system.

Union of Inevitable Collision States The sequential computation for unions of ICS sets was presented. This allows one to reuse the computed ICO of all objects if an additional object

[2] in the sense of a small impact



(a) Collision probability $P(C|\tilde{u})$.



(b) Probabilistic collision costs $\mathrm{PCC}(\tilde{u})$.

Fig. 3.11.: The same scenario is evaluated by two different cost functions. The diameter of the circles is proportional to the cost of each trajectory.

appears in the workspace. Thereby, a major drawback of the ICO computation is addressed. Furthermore, two novel ICS-Checker algorithms are introduced allowing a more efficient computation than former implementations. Simulation results validate the novel sequential computation of ICO and showed a significant reduction of computational effort for the ICS calculations.

Safety regarding unexpected Objects The concept of the robot maneuverability for increasing motion safety was presented. It has been shown, that a higher robot maneuverability decreases the probability to collide with randomly appearing objects in the workspace. However, this approach cannot consider uncertainties in the states or future states of the objects including the robot system. The extension of the robot maneuverability to a complete probabilistic setting would be a future step to assess the safety beyond the planning horizon by considering all possible sources of uncertainties. For the evaluation of the robot maneuverability, a novel ICS-Avoid algorithm considering the maneuverability of the robot was used. Simulation results validate this concept and showed a significant reduction of collisions especially regarding unexpected objects or objects which future motion is only known for a limited time horizon.

Probabilistic Collision State The novel definition for the probabilistic computation of Inevitable Collision States was presented. The proposed definition allows reasoning about the safety of planned trajectories in uncertain and dynamic environments. Furthermore, it is shown that this definition is a generalization of the inevitable collision state approach. The presented method is especially useful in dynamic environments where the future motion of other objects is highly uncertain. The presented computation of PCS preserves the three criteria mentioned in the Sec. 1.2.1. However, this approach does not consider the uncertainty in the future states of the robot system. But this is necessary in order to consider all sources of uncertainty. The example implementation of this definition was evaluated by simulation scenarios and showed that it is efficient and thus applicable to real world scenarios.

Probabilistic Collision Costs Furthermore, the concept of the probabilistic collision cost was presented which is an alternative safety criteria instead of the collision probability. This concept is especially useful in environments with a high density of dynamic objects which tolerate collisions with a certain intensity. The probabilistic collision cost allows one to assess the danger of a collision rather than only its probability. Simulation scenarios confirmed that this safety criteria is promising for uncertain and densely packed dynamic environments.

Overall Collision Probability The presented definition of the overall collision probability assesses the safety for a partial trajectory during and beyond the planning horizon. This novel definition for safety assessment of partial trajectories allows one to apply the PMP framework to probabilistic modeled environments. Since this approach is based on the probabilistic collision state approach, it has the same shortcomings. Simulation scenarios showed the relevance of the new definition.

4. Interactive Assessment

Summary This chapter discusses the problem of assessing the safety of future motions of robotic systems while considering the avoidance behavior of other objects in the workspace. The avoidance behavior originates from the interaction between reactive or controllable dynamic objects. In this context, interaction describes the reactive behavior between objects resulting from the assumption that they incorporate the future motion of the surrounding objects into their own motion planning and expect similar anticipation from them. Therefore, the definitions of ICS and PCS are extended in order to consider this kind of interaction. These extended definitions are applied to car scenarios assessing the safety of highway lanes, and to mobile robot scenarios, considering the avoidance behavior of humans in the assessment process. Finally, simulation results show the relevance of these safety assessment approaches and point out that the consideration of cooperative behavior of reactive objects results in a less conservative and thus more reliable safety assessment.

The outline of this chapter is as follows: Sec. 4.1 is motivating this chapter and states the problem. Sec. 4.2 provides an overview of related work. Then, the definition and properties of the cooperative inevitable collision state (cICS) is presented in Sec. 4.3. This leads to two different definitions of the cooperative probabilistic collision state (cPCS) in Sec. 4.4. The following Sec. 4.5 shows possible implementations of cICS and cPCS. The corresponding simulation results are presented in Sec. 4.6. Finally, a discussion of this chapter is given in Sec. 4.7.

4.1. Motivation and Problem Formulation

The common approach to robot navigation in dynamic environments is to decompose the problem into motion prediction and motion planning. The former provides the robot with the predicted future states of objects in its workspace. Depending on the model of the environment, the prediction can be either deterministic or probabilistic. The latter tries to find a robot trajectory which fulfills the objective of the robot while minimizing certain running costs. Additionally, the motion planner needs to respect certain constraints such as kinematic and dynamic constraints of the robotic system and constraints preventing the robot from collisions with other objects. In deterministic environments the objects are represented as forbidden regions in the configuration space and in stochastic environments as chance constraints [15] (i.e. the collision probability of the trajectory must be below a predefined threshold) or the objects must be considered in the cost function by means of the collision risk.

In real-world scenarios this separation can lead the robot to a freezing robot problem (FRP) as stated in [117]. This means, that the robot cannot find any valid trajectory and freezes. This problem mainly appears if the density of dynamic objects together with the amount of uncertainty in their prediction exceeds a certain level. However, even if the future motion of all objects is exactly known, the FRP can occur. Two different FRP scenarios are depicted in Fig. 4.1. Since

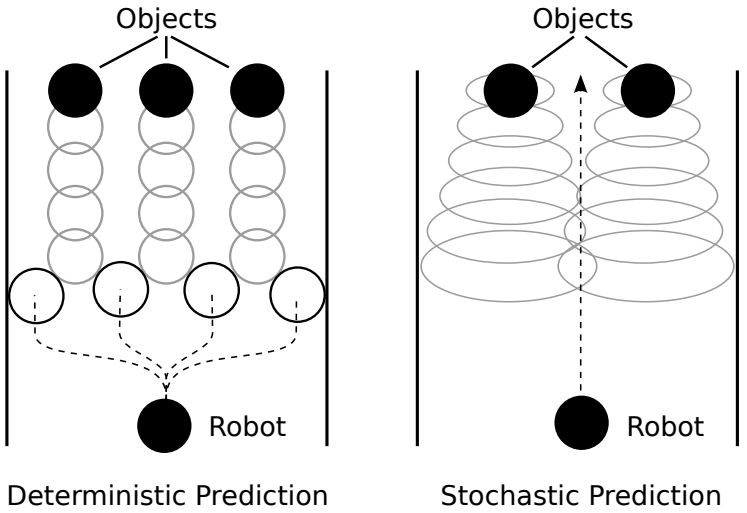


Fig. 4.1.: The freezing robot problem is illustrated for a deterministic and a stochastic modeled environment. The goal of the robot is to navigate through the corridor in the opposite direction of the other objects. The gray $2-\sigma$ ellipsoids represent the uncertain position (assuming normal distribution) of the objects for different time steps. The deterministic prediction of the objects is depicted by gray circles (Inspired by figures of [117]). In the deterministic case, the robot cannot find any collision-free trajectory and in the stochastic case all possible trajectories of the robot have a high probability of collision.

better or even exact models for motion prediction cannot solve the FRP, the question is how is it possible to successfully navigate under these conditions. For instance, people avoid each other (called joint collision avoidance) by adopting their trajectories [50] which gives each person more space for navigation. This behavior can mitigate the FRP and is illustrated in Fig. 4.2 and leads to the idea of interactive safety assessment.

A robot trajectory is safe, if the robot is not colliding with any object in the environment. The challenge of safety assessment in environments populated with reactive objects is to consider the avoidance behavior of these objects. But the assessment needs to take into account, that the avoidance of the objects does not lead to a collision with another object. The problem is stated as:

Interactive Safety *The state of the workspace of the robot (state of all objects including the robot) is safe, if there exists at least one trajectory for each object and the robot ending in a state which is guaranteed to be safe for an infinite time horizon.*

This problem is very similar to the problem of multi-robot motion planning (e.g. [74]). The reason for this is that the avoidance behavior of reactive objects is considered which in turn needs to be assessed concerning its safety. Otherwise, the avoidance behavior can lead to collisions between other dynamic objects which would contradict the definition of reactive objects from Sec. 1.2.2. In the next section, a brief overview of related work regarding interactive safety assessment is given.

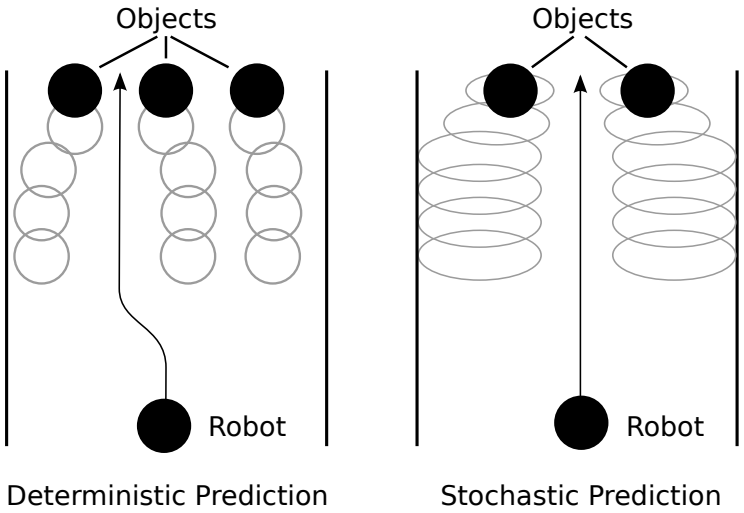


Fig. 4.2.: If the avoidance behavior of the objects is considered, the freezing robot problem can be avoided. The gray $2-\sigma$ ellipsoids represent the uncertain position (assuming normal distribution) of the objects for different time steps. The deterministic prediction of the objects is depicted by gray circles (Inspired by figures of [117]).

4.2. Related Work

The problem of assessing the safety while incorporating avoidance behavior of other objects was investigated in [18, 29] for traffic scenarios. In [18] a threat assessment approach is presented which estimates the PDF representing the future motion of all vehicles in the scene. This is achieved by generating as many future trajectories as possible which do not cause any collision for the complete traffic scene. The collision-free scenarios are ranked by a goal function modeling the preferred actions of the drivers. Each scenario contains one collision-free trajectory for every traffic participant. This work was extended in [29] by new sampling strategies in order to be more efficient and by adding a visibility model for modeling the driver's attention.

Another related field is joint collision avoidance. An algorithm that uses interacting Gaussian processes is presented in [117]. Every trajectory is modeled as a Gaussian process to represent the uncertain future positions of the objects and the interaction is modeled by an interaction potential which is proportional to the Euclidean distance of the corresponding agents. Only few navigation algorithms consider the avoidance possibilities of workspace objects. In [59] the reflective navigation approach is presented which is based on the recursive probabilistic velocity obstacles (extension of the probabilistic velocity obstacles). The robot reflects the environment and incorporates the reactive behavior of the other moving objects into its own planning.

In crowd simulations, where hundreds or even thousands of agents are simultaneously simulated, joint collision avoidance plays an important role as well. Due to the high number of agents, the computational efficiency of the navigation algorithms is crucial. A precomputed roadmap [69] is used by [119] to obtain a high-level cognitive map of the environment. This is used for the global navigation of the agents. During the execution of the agents, the reciprocal velocity obstacle (RVO) approach is used for the collision avoidance behavior of the agents. The RVO approach is an extension of the velocity obstacles method [30] allowing joint collision avoidance. Instead of the RVO, a social force model [49] is applied by [110] to continuously

update the roadmap based on the inter-agents interaction. A more recent approach is the optimal reciprocal collision avoidance presented in [121] that overcomes some limitations of the RVO approach. It can guarantee collision-free navigation for multiple robots assuming a deterministic environment. Some extensions [1, 109] exist which also incorporate the kinematic and dynamic constraints of the robotic system. However, none of these approaches fulfills all three safety criteria from Sec. 1.2.1. Hence, the concept of *cooperative inevitable collision state* is presented in the following considering all three safety criteria. It identifies states that will eventually lead to a collision in environments populated by reactive objects.

4.3. Cooperative Inevitable Collision State

In the following, the definition of the cooperative inevitable collision state (cICS) is given that allows one to perform interactive safety assessment beyond the planning horizon. It is an extension of the ICS concept presented in Sec. 3.3. Instead of only considering the safety of the robot system, it also considers the safety of all dynamic objects. The avoidance behavior originates from the definition of reactive objects in Sec. 1.2.2. Therefore, the workspace state $\mathbf{x}_\mathcal{W}$ is introduced containing the states of all objects in the workspace

$$\mathbf{x}_\mathcal{W} = \{\mathbf{x}_\mathcal{A}, \mathbf{x}_1, \dots, \mathbf{x}_{N_\mathrm{b}}\}.$$

Definition 10 (Cooperative Inevitable Collision States)**.** *The workspace state* $\mathbf{x}_\mathcal{W}$ *is an* cICS *iff*

$$\mathrm{cICS}(\mathbf{x}_\mathcal{W}, \tilde{\boldsymbol{\mathcal{U}}}) = \begin{cases} 1, & \forall \tilde{\mathbf{u}}(\mathbf{x}_\mathcal{W}) \in \tilde{\boldsymbol{\mathcal{U}}} \;\; \exists t \;\; \exists (i,j | i \neq j) \;\; \mathcal{B}_i^A(\tilde{u}_i) \cap \mathcal{B}_j^A(\tilde{u}_j) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Where $\mathcal{B}^\mathcal{A}$ is a set containing the occupancy of all objects and the robotic system. The future trajectories of $\mathcal{B}^\mathcal{A}$ are $\tilde{\mathbf{u}} = \{\tilde{u}_\mathcal{A}, \tilde{u}_1, \dots, \tilde{u}_{N_\mathrm{b}}\}$ and the set of all possible trajectories is denoted as $\tilde{\boldsymbol{\mathcal{U}}} = \tilde{\mathcal{U}}_\mathcal{A} \times \tilde{\mathcal{U}}_1 \times \cdots \times \tilde{\mathcal{U}}_{N_\mathrm{b}}$. The operator $\times$ denotes the Cartesian product. Compared to the ICS Def. 5, a valid control input for all objects must be found instead of one control input for the robotic system. The definition of cICS leads to the definition of the cooperative inevitable collision state set (cICSS):

Definition 11 (**Cooperative Inevitable Collision State Set).** *The Cooperative Inevitable Collision State Set is defined as*

$$\mathrm{cICSS}(\tilde{\boldsymbol{\mathcal{U}}}) = \{\mathbf{x}_\mathcal{W} \in \mathcal{X}_\mathcal{W} | \, \mathrm{cICS}(\mathbf{x}_\mathcal{W}, \tilde{\boldsymbol{\mathcal{U}}}) = 1\},$$

where $\mathcal{X}_\mathcal{W} = \mathcal{X}_\mathcal{A} \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_{N_\mathrm{b}}$.

Loosely speaking, the cICSS is defined as the set of all workspace states which are an cICS. If only a subset of trajectories $\boldsymbol{\mathcal{I}} \subset \tilde{\boldsymbol{\mathcal{U}}}$ is used, this is denoted as $\mathrm{cICSS}(\boldsymbol{\mathcal{I}})$. If the trajectory set is not specified, the complete set is considered. The ICO property Prop. 1 of Sec. 3.3 applies also for cICSS, thus a conservative approximation of cICSS can be calculated by using only a subset of all possible future trajectories.

Property 4 (cICSS Approximation).

$$\mathrm{cICSS}(\tilde{\boldsymbol{\mathcal{U}}}) \subseteq \mathrm{cICSS}(\boldsymbol{\mathcal{I}}),\ \textit{with}\ \boldsymbol{\mathcal{I}} \subset \tilde{\boldsymbol{\mathcal{U}}} \tag{4.2}$$

Proof.

$$\begin{aligned}
&\mathrm{cICSS}(\tilde{\boldsymbol{\mathcal{U}}}) \\
&\quad\Leftrightarrow \{\mathbf{x}_\mathcal{W} \in \mathcal{X}_\mathcal{W} | \forall \tilde{\mathbf{u}}(\mathbf{x}_\mathcal{W}) \in \tilde{\boldsymbol{\mathcal{U}}} \quad \exists t\ \exists (i,j|i\neq j)\ \mathcal{B}_i^A(\tilde{u}_i(t)) \cap \mathcal{B}_j^A(\tilde{u}_j(t)) \neq \emptyset\} \\
&\quad\Leftrightarrow \{\mathbf{x}_\mathcal{W} \in \mathcal{X}_\mathcal{W} | \forall \tilde{\mathbf{u}}(\mathbf{x}_\mathcal{W}) \in \boldsymbol{\mathcal{I}} \cup (\overline{\boldsymbol{\mathcal{I}}} \cap \tilde{\boldsymbol{\mathcal{U}}}) \ \exists t\ \exists (i,j|i\neq j)\ \mathcal{B}_i^A(\tilde{u}_i(t)) \cap \mathcal{B}_j^A(\tilde{u}_j(t)) \neq \emptyset\} \\
&\quad\Leftrightarrow \{\mathbf{x}_\mathcal{W} \in \mathcal{X}_\mathcal{W} | \forall \tilde{\mathbf{u}}(\mathbf{x}_\mathcal{W}) \in \boldsymbol{\mathcal{I}} \quad \exists t\ \exists (i,j|i\neq j)\ \mathcal{B}_i^A(\tilde{u}_i(t)) \cap \mathcal{B}_j^A(\tilde{u}_j(t)) \neq \emptyset\} \ \cap \\
&\qquad \{\mathbf{x}_\mathcal{W} \in \mathcal{X}_\mathcal{W} | \forall \tilde{\mathbf{u}}(\mathbf{x}_\mathcal{W}) \in \overline{\boldsymbol{\mathcal{I}}} \cap \tilde{\boldsymbol{\mathcal{U}}} \quad \exists t\ \exists (i,j|i\neq j)\ \mathcal{B}_i^A(\tilde{u}_i(t)) \cap \mathcal{B}_j^A(\tilde{u}_j(t)) \neq \emptyset\} \\
&\quad\Leftrightarrow \quad \mathrm{cICSS}(\boldsymbol{\mathcal{I}}) \cap \mathrm{cICS}(\overline{\boldsymbol{\mathcal{I}}} \cap \tilde{\boldsymbol{\mathcal{U}}})
\end{aligned}$$

□

The same property holds for cICS, thus the calculation of cICS with a reduced input space leads to a conservative result. In the next section, the concept of cICS is extended to stochastic environments.

4.4. Cooperative Probabilistic Collision State

In order to apply the presented cICS concept to stochastically modeled environments, the cooperative probabilistic collision state (cPCS) is introduced. Therefore, Def. 10 is extended to a probabilistic setting. Instead of evaluating the workspace state $\mathbf{x}_\mathcal{W}$ representing a deterministic environment, the PDF

$$\begin{aligned}
&f(\mathbf{x}_\mathcal{W}) = f(\mathbf{x}_\mathcal{A}, \mathbf{x}_1, \ldots, \mathbf{x}_{N_\mathrm{b}}) \text{ with} \\
&\int_{\mathcal{X}_\mathcal{A}} \int_{\mathcal{X}_1} \ldots \int_{\mathcal{X}_{N_\mathrm{b}}} f(\mathbf{x}_\mathcal{A}, \mathbf{x}_1, \ldots, \mathbf{x}_{N_\mathrm{b}})\, \mathrm{d}\mathbf{x}_\mathcal{A}\, \mathrm{d}\mathbf{x}_1 \ldots\, \mathrm{d}\mathbf{x}_{N_\mathrm{b}} = \\
&\int_{\mathcal{X}_\mathcal{W}} f(\mathbf{x}_\mathcal{W})\, \mathrm{d}\mathbf{x}_\mathcal{W} = 1, \text{ with } \mathcal{X}_\mathcal{W} = \mathcal{X}_\mathcal{A} \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_{N_\mathrm{b}}
\end{aligned}$$

is now evaluated which represents the uncertainty of the workspace state $\mathbf{x}_\mathcal{W}$ by this joint probability distribution. Two definitions for the probabilistic extension of cICS are given which differ in the available information to the agents. The first definition cPCS^d assumes that the objects perceive the environment without any uncertainty, however, the robot is unsure about the present scenario due to sensor noise. The second definition cPCS^u assumes that the objects try to minimize the expected collision probability regarding all possible scenarios.

4.4.1. Definition of cPCS^d

The definition of cPCS^d assumes optimal behavior of all objects in the environments in order to avoid any collision, although the initial scenario is not exactly known.

Definition 12 (Cooperative Probabilistic Collision State $\mathrm{cPCS}^{\mathrm{d}}$)**.** *The probability that an uncertain workspace state $\mathbf{x}_{\mathcal{W}}$ is an* cICS *is*

$$\mathrm{cPCS}^{\mathrm{d}}(f(\mathbf{x}_{\mathcal{W}})) := \int_{\mathcal{X}_{\mathcal{W}}} \mathrm{cICS}(\mathbf{x}_{\mathcal{W}}) f(\mathbf{x}_{\mathcal{W}}) \, \mathrm{d}\mathbf{x}_{\mathcal{W}}.$$

Compared to Sec. A.3, the definition can be seen as a Monte Carlo simulation, whereby $\mathrm{cICS}(\mathbf{x}_{\mathcal{W}})$ is the indicator function.

4.4.2. Definition of $\mathrm{cPCS}^{\mathrm{u}}$

The definition of $\mathrm{cPCS}^{\mathrm{u}}$ assumes optimal behavior of all objects in the environments in order to avoid any collision regarding the distribution of all possible scenarios. The minimum collision probability allows defining the probability of an inevitable collision state for reactive objects.

Definition 13 (Cooperative Probabilistic Collision State $\mathrm{cPCS}^{\mathrm{u}}$)**.** *The probability of a workspace state $\mathbf{x}_{\mathcal{W}}$ leading to a collision assuming reactive objects is defined as the collision probability under the best possible input trajectories $\tilde{\mathbf{u}}$ regarding all possible scenarios $\mathcal{X}_{\mathcal{W}}$:*

$$\mathrm{cPCS}^{\mathrm{u}}(f(\mathbf{x}_{\mathcal{W}})) := \min_{\tilde{\mathbf{u}}} P(C|f(\mathbf{x}_{\mathcal{W}}), \tilde{\mathbf{u}})$$

where $P(C|f(\mathbf{x}_{\mathcal{W}}), \tilde{\mathbf{u}})$ is the collision probability that at least one collision occurs when applying $\tilde{\mathbf{u}}$ to all objects.

4.4.3. Discussion

Both definitions consider the avoidance behavior of reactive objects for safety assessment. The difference of the definitions is in the optimization of the future control inputs of the objects. The control inputs are optimized separately for *each* possible scenario in Def. 12, whereas in Def. 13 the control inputs are optimized for *all* possible scenarios. This can be interpreted in the following way. Objects in Def. 12 behave like they exactly know the current scenario, but the robotic system is unsure about the current scenario due to sensor noise. According to Def. 13, all objects are unsure about the current scenario and behave in order to minimize the expected collision risk considering all possible scenarios. For better understanding some properties of both definitions are presented.

Property 5 (Conservative Property of $\mathrm{cPCS}^{\mathrm{u}}$)**.**

$$\forall f(\mathbf{x}_{\mathcal{W}}), \quad \mathrm{cPCS}^{\mathrm{u}}(f(\mathbf{x}_{\mathcal{W}})) \geq \mathrm{cPCS}^{\mathrm{d}}(f(\mathbf{x}_{\mathcal{W}}))$$

Proof. Def. 13 can be rewritten with (A.3) as

$$\mathrm{cPCS}^{\mathrm{u}}(f(\mathbf{x}_{\mathcal{W}})) = \min_{\tilde{\mathbf{u}} \in \tilde{\boldsymbol{\mathcal{U}}}} \int_{\mathcal{X}_{\mathcal{W}}} \mathrm{Ind}(C|\mathbf{x}_{\mathcal{W}}, \tilde{\mathbf{u}}) f(\mathbf{x}_{\mathcal{W}}) \, \mathrm{d}\mathbf{x}_{\mathcal{W}}$$

with

$$\mathrm{Ind}(C|\mathbf{x}_\mathcal{W}, \tilde{\mathbf{u}}) = \begin{cases} 1, & \text{if } \exists t\ \exists (i,j|i \neq j)\ \mathcal{B}_i^A(\tilde{u}_i(t)) \cap \mathcal{B}_j^A(\tilde{u}_j(t)) \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases}$$

With the definition of the optimal control inputs

$$\tilde{\mathbf{u}}^*(f(\mathbf{x}_\mathcal{W})) := \arg\min_{\tilde{\mathbf{u}} \in \tilde{\boldsymbol{\mathcal{U}}}} \int_{\mathcal{X}_\mathcal{W}} \mathrm{Ind}(C|\mathbf{x}_\mathcal{W}, \tilde{\mathbf{u}}) f(\mathbf{x}_\mathcal{W})\, \mathrm{d}\mathbf{x}_\mathcal{W}$$

the equation can be rewritten as

$$\mathrm{cPCS}^\mathrm{u}(f(\mathbf{x}_\mathcal{W})) = \int_{\mathcal{X}_\mathcal{W}} \mathrm{Ind}(C|\mathbf{x}_\mathcal{W}, \tilde{\mathbf{u}}^*(f(\mathbf{x}_\mathcal{W}))) f(\mathbf{x}_\mathcal{W})\, \mathrm{d}\mathbf{x}_\mathcal{W}.$$

We define the optimized control inputs for one scenario $\tilde{\mathbf{u}}^*(\mathbf{x}_\mathcal{W})$ as

$$\tilde{\mathbf{u}}^*(\mathbf{x}_\mathcal{W}) := \arg\min_{\tilde{\mathbf{u}} \in \tilde{\boldsymbol{\mathcal{U}}}} \mathrm{Ind}(C|\mathbf{x}_\mathcal{W}, \tilde{\mathbf{u}}),$$

meaning, that the control inputs avoid a collision if it is possible. Since the optimized control inputs $\tilde{\mathbf{u}}^*(\mathbf{x}_\mathcal{W})$ have a higher chance to avoid a collision than the control inputs $\tilde{\mathbf{u}}^*(f(\mathbf{x}_\mathcal{W}))$ which are optimized regarding all possible scenarios $\mathbf{x}_\mathcal{W} \in \mathcal{X}_\mathcal{W}$, it follows

$$\begin{aligned}
& \{\mathbf{x}_\mathcal{W} \in \mathcal{X}_\mathcal{W} |\, \mathrm{Ind}(C|\mathbf{x}_\mathcal{W}, \tilde{\mathbf{u}}^*(f(\mathbf{x}_\mathcal{W})))\} && \supseteq && \mathrm{cICSS}(\tilde{\mathbf{u}}^*(\mathbf{x}_\mathcal{W})) \\
\forall f(\mathbf{x}_\mathcal{W}) \quad & \int_{\mathcal{X}_\mathcal{W}} \mathrm{Ind}(C|\mathbf{x}_\mathcal{W}, \tilde{\mathbf{u}}^*(f(\mathbf{x}_\mathcal{W}))) f(\mathbf{x}_\mathcal{W})\, d\mathbf{x}_\mathcal{W} && \geq && \int_{\mathcal{X}_\mathcal{W}} \mathrm{cICS}(\mathbf{x}_\mathcal{W}, \tilde{\mathbf{u}}^*(\mathbf{x}_\mathcal{W})) f(\mathbf{x}_\mathcal{W})\, \mathrm{d}\mathbf{x}_\mathcal{W} \\
\forall f(\mathbf{x}_\mathcal{W}) \quad & \mathrm{cPCS}^\mathrm{u}(f(\mathbf{x}_\mathcal{W})) && \geq && \mathrm{cPCS}^\mathrm{d}(f(\mathbf{x}_\mathcal{W}))
\end{aligned}$$

□

Property 6 (Upper bound of cPCS)**.**

$$\mathrm{cPCS}^\mathrm{d}(f(\mathbf{x}_\mathcal{W})) = 1 \Leftrightarrow \mathrm{cPCS}^\mathrm{u}(f(\mathbf{x}_\mathcal{W})) = 1$$

Proof. If $\mathrm{cPCS}^\mathrm{d}(f(\mathbf{x}_\mathcal{W})) = 1$ it follows $\mathrm{cPCS}^\mathrm{u}(f(\mathbf{x}_\mathcal{W})) = 1$ with

$$\forall \tilde{\mathbf{u}}^*(\mathbf{x}_\mathcal{W})\ \ \exists t\ \ \exists (i,j|i \neq j)\ \ \mathcal{B}_i^A(\tilde{u}_i(t)) \cap \mathcal{B}_j^A(\tilde{u}_j(t)) \neq \emptyset.$$

Thus no $\tilde{\mathbf{u}}$ exists which can prevent a collision with any possible scenario $\mathbf{x}_\mathcal{W}$

$$\forall \tilde{\mathbf{u}}^*(f(\mathbf{x}_\mathcal{W}))\ \ \exists t\ \ \exists (i,j|i \neq j)\ \ \mathcal{B}_i^A(\tilde{u}_i(t)) \cap \mathcal{B}_j^A(\tilde{u}_j(t)) \neq \emptyset.$$

This implies

$$\mathrm{cPCS}^\mathrm{u}(f(\mathbf{x}_\mathcal{W})) = 1.$$

The other direction, $\mathrm{cPCS}^\mathrm{u}(f(\mathbf{x}_\mathcal{W})) = 1$ follows $\mathrm{cPCS}^\mathrm{d}(f(\mathbf{x}_\mathcal{W})) = 1$ is analogous. □

Property 7 (Lower bound of cPCS)**.**

$$\mathrm{cPCS}^\mathrm{u}(f(\mathbf{x}_\mathcal{W})) = 0 \rightarrow \mathrm{cPCS}^\mathrm{d}(f(\mathbf{x}_\mathcal{W})) = 0.$$

This follows directly from Prop. 5. In the next section, example implementations for cICS,

$\mathrm{cPCS}^{\mathrm{u}}$ and $\mathrm{cPCS}^{\mathrm{d}}$ are presented.

4.5. Implementations

In this section an example implementation for cICS and for both definitions of cPCS are shown. Therefore, one implementation for automotive applications – navigation on a highway – and one implementation for mobile robot applications – navigation in densely packed environments – are presented.

4.5.1. Automotive Application

In this section, cICS and $\mathrm{cPCS}^{\mathrm{d}}$ are used for assessing the safety of road scenes. To make use of the structured environment of roads, the safety of each road lane is assessed separately. For this application, the $\mathrm{cPCS}^{\mathrm{d}}$ definition is used since the PCS definition (Def. 8) would result in an unrealistic assessment of the motion safety. As explained by Fuller [37]:

> *" [...] the experience of subjective risk is aversive and so drivers are motivated to escape from situations which elicit the experience or to avoid those situations. "*

Thus, the driver behaves to avoid collisions in the present situation. For this implementation a nonlinear one-dimensional vehicle model is used which is described in the next section.

Vehicle Model

In most traffic situations the human driver plans the vehicle's lateral movement relative to the lanes rather than to the absolute ground [125]. Imitating this approach, most trajectory generation approaches use the so-called Frenet frame. Since the assessment is performed for each lane separately, the state $\mathbf{x}(t) = [s(t), v(t)]$ of the vehicle at time t is represented by the vehicle position in s and its velocity v in the direction of the lane. The initial state is denoted as $\mathbf{x}(0)$. The reference point of the vehicle position is its volumetric center point. A nonlinear model is used as presented in [3], since it shows a good trade-off between accuracy and complexity. It was validated against the high fidelity simulation veDYNA [24]. The longitudinal dynamics of the vehicles are described by the equations

$$\dot{s} = v, \quad \dot{v} = \begin{cases} u\, a^{\max}, & u \leq 0 \ \vee \ 0 < v \leq v^{\mathrm{sw}} \\ u\, a^{\max} \frac{v^{\mathrm{sw}}}{v}, & u > 0 \ \wedge \ v > v^{\mathrm{sw}} \\ 0, & v \leq 0 \end{cases} \tag{4.3}$$

with respect to the constraints

$$u \in [-1, 1], \quad v \leq v^{\max}, \tag{4.4}$$

where $v^{\max}$ is the maximum velocity of the vehicle. The model prevents backward driving. For positive accelerations, the dynamics of the model switch at velocity v^{sw}. For velocities $0 < v \leq v^{\mathrm{sw}} \vee u \leq 0$ the acceleration is limited due to tire friction. For faster velocities $v > v^{\mathrm{sw}}$, the acceleration is limited due to the available engine power. The acceleration constraints model

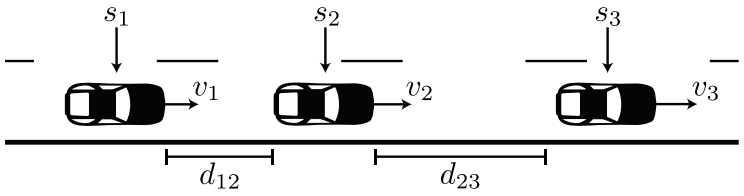


Fig. 4.3.: The state of the lane is shown for one example scenario.

the maximum possible tire friction according to Kamm's circle $a^{\max}$. The acceleration consists of the lateral and longitudinal, respectively normal a_{N} and tangential acceleration a_{T}:

$$a^{\max} = \sqrt{a_{\mathrm{N}}^2 + a_{\mathrm{T}}^2}, \quad a_{\mathrm{N}} := \frac{v^2}{\rho(s)}, \quad a_{\mathrm{T}} := \dot{v}.$$

The function $\rho(s)$ transforms the path coordinate s to the radius of the curve. The tangential acceleration a_{T} and the normal acceleration a_{N} is derived from the radius of curvature.

Assuming constant u, the analytic solution for $u > 0$ and $v > v^{\mathrm{sw}}$ is

$$\mathbf{x}(t) = \begin{cases} s(t) = & s(0) + \frac{(v(0)^2 + 2a^{\max} v^{\mathrm{sw}} ut)^{\frac{3}{2}} - v(0)^3}{3a^{\max} v^{\mathrm{sw}} u}, \\ v(t) = & \sqrt{v(0)^2 + 2a^{\max} v^{\mathrm{sw}} ut}. \end{cases} \tag{4.5}$$

For the other cases the analytic solutions are analogous.

Problem Statement

For this problem, the workspace state is identical with the complete state $\mathbf{x}_{\mathcal{W}}$ of a lane containing N_{v} vehicles. It is described by the initial velocities $\boldsymbol{v}(0) = [v_1(0), \ldots, v_{N_{\mathrm{v}}}(0)]$ and the initial position along the lane $\boldsymbol{s}(0) = [s_1(0), \ldots, s_{N_{\mathrm{v}}}(0)]$. An example road scene is illustrated in Fig. 4.3. In order to classify a road lane as safe the cICS Def. 10 is used. This entails that instead of evaluating the safety of a single object, the safety of a group of objects is evaluated. Loosely speaking, to fulfill Def. 10, one trajectory $\tilde{u}$ must be determined for each vehicle which will be collision-free regarding an infinite time horizon. For this implementation, we assume constant control inputs for all vehicles $\tilde{u} \to u$. Thus, the goal is to find the set of control inputs

$$\begin{aligned} &\mathbf{u} = [u_1, \ldots, u_{N_{\mathrm{v}}}] \text{ subject to} \\ &\forall (i, j | i > j) \;\; \forall t \in [0, \infty) \quad \mathrm{dist}_{ij}(t) \geq 0. \end{aligned} \tag{4.6}$$

The function $\mathrm{dist}_{ij}(t)$ measures the free space between the vehicles i and j along $s(t)$

$$\mathrm{dist}_{ij}(t) = s_i(t) - s_j(t) - \frac{l_i}{2} - \frac{l_j}{2},$$

where l_i and l_j are the lengths of the ith and jth vehicle. In the next section, the problem described in (4.6) is reformulated as a set of nonlinear programming sub-problems.

Pairwise Safety Assessment

The complexity of (4.6) depends on the number of cars on the lane. In order to break down the problem, it is reduced to a set of sub-problems. Thus, only two cars are considered and the maximum control input $u_{\mathrm{r}}^{\max}$ of the rear vehicle is determined which will never collide with the front vehicle for the given control input u_{f}. The index f refers to the front vehicle and the index r to the rear vehicle. This can be formulated as a nonlinear programming problem with the objective function $\mathrm{cost}(u_{\mathrm{r}})$:

$$\begin{aligned} &\mathrm{cost}(u_{\mathrm{r}}) = u_{\mathrm{r}}, \quad \max_{u_{\mathrm{r}} \in \mathcal{U}_r} \mathrm{cost}(u_{\mathrm{r}}) \text{ subject to} \\ &\mathrm{dist}_{\mathrm{fr}}(t, u_{\mathrm{r}}, u_{\mathrm{f}}) \geq 0, \ \forall t \in [0, \infty). \end{aligned} \tag{4.7}$$

The function $\mathrm{dist}_{\mathrm{fr}}(t, u_{\mathrm{r}}, u_{\mathrm{f}})$ determines the distance between the front vehicle and the rear vehicle at time t, when the control inputs u_{r} and u_{f} are applied. The valid sets of control inputs for the rear and the front vehicle are denoted as $\mathcal{U}_r$ respectively $\mathcal{U}_f$. The calculated maximum control input u_{r} is the control input for the front car of the next pair of vehicles. The control input of the very first car is set to $u_{N_{\mathrm{v}}} \equiv 1$. The reason for this is, that the relative distance between two cars $\mathrm{dist}_{\mathrm{fr}}(\cdot, \cdot, u_{\mathrm{f}})$ is a monotonic function regarding u_{f}, thus the maximum allowed control input will lead to a solution for u_{r} iff one exists. In the following, it is shown that there exists a solution regarding (4.7), if there exists a solution according to (4.6) and vice versa.

Proof. The vector $\mathbf{u}^{\mathrm{s}}$ contains a solution of the nonlinear programming problem (4.7) and $\mathbf{u}^{\mathrm{g}}$ is a solution of the global problem (4.6). First it is shown, that every solution $\mathbf{u}^{\mathrm{s}}$ is also a solution of the global problem. Since a vehicle can only collide either with the front or the rear vehicle, it is sufficient to make sure that no rear vehicle collides with the front car. This is explicitly modeled by the constraint in (4.7). Next, it is shown that every solution $\mathbf{u}^{\mathrm{g}}$ is also a solution of the nonlinear programming problem. For each solution of $\mathbf{u}^{\mathrm{s}}$, it is valid that

$$\forall i \ u_i^{\mathrm{g}} \leq u_i^{\mathrm{s}}, \text{ with } u_i^{\mathrm{g}} \in \mathbf{u}^{\mathrm{g}}, u_i^{\mathrm{s}} \in \mathbf{u}^{\mathrm{s}},$$

since the nonlinear programming problem determines the largest possible u_i which satisfies the constraints. It is shown by the counter-evidence, that it is not possible that there exists a global solution $\mathbf{u}^{\mathrm{g}}$ which is not found by the pairwise nonlinear programming approach. This means, there exists one control input for which $u_i^{\mathrm{g}} > u_i^{\mathrm{s}}$ is valid. Since u_i^{s} is the maximum possible control input, one can see that $u_{i+1}^{\mathrm{g}} > u_{i+1}^{\mathrm{s}}$ is true. This is carried on till $u_{N_{\mathrm{v}}}^{\mathrm{g}} > u_{N_{\mathrm{v}}}^{\mathrm{s}}$. This cannot be true, since $u_{N_{\mathrm{v}}}^{\mathrm{s}} \equiv 1$ and a larger control input violates the model constraints in (4.4). □

In order to reduce computational demand and to guarantee deterministic response times, the solution of the nonlinear programming problem is stored in a 4D lookup table (LUT). The input of the LUT are both initial velocities $v_{\mathrm{r}}(0)$, $v_{\mathrm{f}}(0)$, the initial free space $\mathrm{dist}_{\mathrm{fr}}(0)$ and the constant control input u_{f}. The output of the LUT is the maximum possible control input $\hat{u}_{\mathrm{r}}^{\max}$ of the rear vehicle. Due to the discretization of the inputs, the resulting control input

$$\hat{u}_{\mathrm{r}}^{\max} = \mathrm{LUT}(\mathrm{dist}_{\mathrm{fr}}, v_{\mathrm{r}}, v_{\mathrm{f}}, u_{\mathrm{f}})$$

may violate the constraint in (4.7). In order to ensure a conservative approximation of $u_{\mathrm{r}}^{\max}$, the

input parameters are over $(\bar{\cdot})$ or respectively under approximated $(\underline{\cdot})$

$$\hat{\underline{u}}_{\mathrm{r}}^{\max} = \mathrm{LUT}(\underline{\mathrm{dist}}_{\mathrm{fr}}, \underline{v}_{\mathrm{f}}, \overline{v}_{\mathrm{r}}, \underline{u}_{\mathrm{f}}).$$

Based on the LUT, an cICS based and a $\mathrm{cPCS}^{\mathrm{d}}$ based safety assessment algorithm are presented in the following sections. It is noted that it is also possible to solve the nonlinear programming problem directly instead of using the LUT.

Lane Safety Assessment

In this section a cICS and a $\mathrm{cPCS}^{\mathrm{d}}$ checker for road lanes are presented. Both algorithms are based on the pairwise nonlinear programming problem discussed above.

Cooperative Inevitable Collision State Checker Compared to known ICS checkers, the cICS checker presented in this section uses the continuous input space of the system instead of a finite subset. Thus, it can ensure to find a solution if one exists. The basic idea of the cICS checker is, that it performs an ICS check for each rear car state regarding the front vehicle. By applying the ICS check pairwise to all vehicles the lane is classified as safe if a valid control input is found for each vehicle. The control input of the front car is the maximum possible control input u_{r} of the previously evaluated pair of vehicles. The definition of cICS Def. 10 requires to set the control input for the first car in the lane to one. This guarantees that a valid control input of each car can be found if one exists. However, this implementation allows one to use an arbitrary control input of the first car, whereby the assessment becomes more conservative. The control input of the first car can be set to the last observed value, this corresponds to constant velocity prediction. Another possibility is to simulate the worst case by setting the control input to minus one which means full braking of the front car. The complete algorithm is shown in Alg. 6. Furthermore, the cICS checker determines the minimum allowed control input u_{m} of all

Algorithm 6: cICS based assessment

Input : $\boldsymbol{s}(0)$, $\mathbf{v}(0)$, $u_{N_{\mathrm{v}}}$
Output : cICS flag, u_{m}

foreach $i = N_{\mathrm{v}} : 2$ **do**
  $\underline{v}_f \leftarrow v_i$
  $\overline{v}_r \leftarrow v_{i-1}$
  $\underline{\mathrm{dist}}_{\mathrm{fr}} \leftarrow s_i, s_{i-1}$
  $\underline{u}_f \leftarrow \hat{\underline{u}}_i^{\max}$
  $\hat{\underline{u}}_{i-1}^{\max} = \mathrm{LUT}(\underline{\mathrm{dist}}_{\mathrm{fr}}, \underline{v}_f, \overline{v}_r, \underline{u}_f)$
  if $\hat{\underline{u}}_{i-1}^{\max} < -1$ **then**
    return true, 0
$u_{\mathrm{m}} = \min\{\hat{\underline{u}}_{N_{\mathrm{v}}-1}^{\max}, \ldots, \hat{\underline{u}}_1^{\max}\}$
return false, u_{m}

vehicles. It can be interpreted as the safety margin of the lane state. Additionally, the difference of the allowed control inputs with and without the ego car can be used to measure the disturbance

caused by the ego car. This can be used as an additional cost factor for a navigation algorithm in order to minimize the disturbance of the ego vehicle.

Cooperative Probabilistic Collision State Checker For evaluating cICS, the state of all vehicles in the lane and the control input of the first car in each lane must be known without any uncertainty. Since this is an unrealistic assumption for real-world situations due to sensor noise and the hardly predictable human drivers, the $\mathrm{cPCS}^{\mathrm{d}}$ approach is used. For this implementation, independent variables are assumed, thus the joint probability distribution $f(\mathbf{x}_{\mathcal{W}})$ can be computed as

$$f(\mathbf{x}_{\mathcal{W}}) = f_{\mathcal{A}}(\mathbf{x}) f_1(\mathbf{x}) \cdots f_{N_{\mathrm{b}}}(\mathbf{x}).$$

The PDFs of all vehicles representing their initial state comply to the uncertain initial state of the entire road scene $f(\mathbf{x}_{\mathcal{W}}(0))$. It is noted, that the assumption of independent variables does not mean that the vehicles also behave independently. The uncertain state of each object is represented by the uncertain velocity and the uncertain position along the lane

$$f_i(\mathbf{x}) = f_i(s) f_i(v), \text{ with } i \in \{\mathcal{A}, 1, \ldots, N_{\mathrm{b}}\}$$

assuming that the random variables v and s are independent. The PDFs $f_i(s(0))$ and $f_i(v(0))$ represent the uncertain initial position and the initial velocity of the ith vehicle, respectively. Additionally, a probabilistic driver model is used for the first car in the lane and is represented by the PDF $f_{N_{\mathrm{v}}}(u)$. For solving the $\mathrm{cPCS}^{\mathrm{d}}$ calculation, Monte Carlo simulation is used as described in Sec. A.3. Therefore, N_{s} samples are drawn from the distributions. As presented in [135] importance sampling is used that allows one to calculate the $\mathrm{cPCS}^{\mathrm{d}}$ value by the ratio between the number of collision scenarios N_{C} and all simulated scenarios N_{s}. An overview of the complete algorithm is given in Alg. 7. The $\mathrm{cPCS}^{\mathrm{d}}$ method is especially suitable for probabilistic predictions of the other vehicles, while the cICS approach is applicable to deterministic predictions.

4.5.2. Mobile Robot Application

In this section, an example implementation of $\mathrm{cPCS}^{\mathrm{u}}$ is used for assessing the safety in densely packed dynamic environments. A typical scenario for this application is an autonomous mobile robot navigating in environments populated with humans. The definition of $\mathrm{cPCS}^{\mathrm{u}}$ allows arbitrary workspaces and has no restriction on the shape, kinematics or dynamics of the object, however, in this section a $\mathrm{cPCS}^{\mathrm{u}}$ checker is presented for disk-shaped objects in a two-dimensional workspace with position coordinates x and y.

In contrast to the previous section, the $\mathrm{cPCS}^{\mathrm{u}}$ definition is used instead of the $\mathrm{cPCS}^{\mathrm{d}}$. The reason for this is, that the $\mathrm{cPCS}^{\mathrm{d}}$ approach would exceed the computational resources and the limited computing time for a 2D workspace. According to Prop. 5 the assessment by $\mathrm{cPCS}^{\mathrm{u}}$ becomes more conservative compared to $\mathrm{cPCS}^{\mathrm{d}}$, however, it is less conservative than PCS which does not consider any interaction between the objects. In order to perform safety assessment based on the $\mathrm{cPCS}^{\mathrm{u}}$ definition, the model for the motion prediction of reactive objects is described after the general idea of this implementation. Then, the optimal input trajectory of the robot is obtained which is then used to determine the probability density function for reactive

Algorithm 7: $\mathrm{cPCS}^{\mathrm{d}}$ based assessment

Input : $f(\mathbf{x}_{\mathcal{W}})\ f_{N_{\mathrm{v}}}(u)$
Output : $\mathrm{cPCS}^{\mathrm{d}}$ value

Initialize: $N_{\mathrm{C}} = 0$
foreach $i = 1 : N_{\mathrm{s}}$ **do**
  {Sample from distributions}
  $u_{N_{\mathrm{v}}} \sim f_{N_{\mathrm{v}}}(u)$
  foreach $j = 1 : N_{\mathrm{v}}$ **do**
    $s_j \sim f_j(s)$
    $v_j \sim f_j(v)$
  foreach $j = N_{\mathrm{v}} : 2$ **do**
    $\underline{v}_f \leftarrow v_j$
    $\overline{v}_r \leftarrow v_{j-1}$
    $\underline{\mathrm{dist}}_{\mathrm{fr}} \leftarrow s_j, s_{j-1}$
    $\underline{u}_f \leftarrow \hat{\underline{u}}_j^{\max}$
    $\hat{\underline{u}}_{j-1}^{\max} = LUT(\underline{\mathrm{dist}}_{\mathrm{fr}}, \underline{v}_f, \overline{v}_r, \underline{u}_f)$
    if $\hat{\underline{u}}_{j-1}^{\max} < -1$ **then**
      $N_{\mathrm{C}} = N_{\mathrm{C}} + 1$
      Break
return $\mathrm{cPCS}^{\mathrm{d}} = \frac{N_{\mathrm{C}}}{N_{\mathrm{s}}}$

objects. Finally, the collision probabilities are computed resulting in the $\mathrm{cPCS}^{\mathrm{u}}$ evaluation.

General Idea

The definition of $\mathrm{cPCS}^{\mathrm{u}}$ requires to solve the optimization problem described in Def. 13. This problem can be seen as a multi-robot motion planning problem where the objective is to minimize the collision probability of the entire system. In addition, this needs to be done for an infinite time horizon (see Sec. 3.1). Since the presented $\mathrm{cPCS}^{\mathrm{u}}$ checker provides an additional safety check to extend the safety assessment of common motion planning approaches, a computational efficient implementation is crucial. Thus, a two-stage optimization is used to improve the tradeoff between computational complexity and quality of the result.

In a first step, the robot trajectory $\tilde{u}_{\mathcal{A}}^*$ is determined which minimizes the PCS value assuming ignoring objects. In the second step, the reactive objects will modify their future motion in order to minimize their collision probability. In other words, the objects react on the robot trajectory to minimize their collision risk. The ignoring objects are modeled with a PDF $f(\mathbf{x}, t)$ which is given by a motion prediction algorithm. The reactive objects are represented by a PDF $f(\mathbf{x}, t, \tilde{u}_{\mathcal{A}}^*)$ which depends on the robot trajectory $\tilde{u}_{\mathcal{A}}^*$.

Motion Model

For this implementation, a constant acceleration model is used for the prediction of the reactive workspace objects. The dynamic system of the constant acceleration model is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ a_x \\ a_y \end{bmatrix},$$

where the absolute value of the acceleration $\sqrt{a_x^2 + a_y^2} \leq a^{\max}$ is limited. The velocity is indirectly limited by the initial velocity since only braking trajectories are considered as discussed in Sec. 3.9.2. After a time discretization with $t_k = k\,T$, where $k \in \mathbb{N}^+$ has been introduced as the time step and $T \in \mathbb{R}^+$ is the time step size, the dynamic model can be exactly transformed to the discrete time form:

$$\underbrace{\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}(t_{k+1})}_{\mathbf{x}(t_{k+1})} = \underbrace{\begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}(t_k)}_{\mathbf{x}(t_k)} + \underbrace{\begin{bmatrix} a_x \frac{T^2}{2} \\ a_y \frac{T^2}{2} \\ a_x T \\ a_y T \end{bmatrix}}_{u}. \tag{4.8}$$

It is further assumed that the initial state of the objects has a multivariate Gaussian distribution $\mathbf{x}(0) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean value $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. From the multiplication rule and the addition rule of independent random variables with Gaussian distributions, it follows that the mean value and the covariance of the state $\mathbf{x}$ in (4.8) are updated as

$$\begin{aligned} \boldsymbol{\mu}(t_{k+1}) &= A\boldsymbol{\mu}(t_k) + u \\ \boldsymbol{\Sigma}(t_{k+1}) &= A\boldsymbol{\Sigma}(t_k)A^T, \end{aligned} \tag{4.9}$$

where A^T is the transpose of the system matrix A. Note that the input u has no influence on the covariance matrix because this input is deterministic. For predicting the reactive objects, the control input u must be specified depending on the robot trajectory. For this implementation, ignoring objects can be seen as a special case of reactive objects. Their control input is set to $u = [0, 0, 0, 0]$ resulting in a constant velocity prediction. The optimization of the robot trajectory is described in the next section.

Input Trajectory of the Robot

Under the assumption that the workspace objects move independently of the robot, i.e. $\forall i : f_i(\mathbf{x}, t, \tilde{u}_{\mathcal{A}}) \rightarrow f_i(\mathbf{x}, t)$, the input trajectory $\tilde{u}^*$ that minimizes $\mathrm{PCS}(\mathbf{x})$ for the robot state $\mathbf{x}$ according to Def. 8 must be determined

$$\begin{aligned} \tilde{u}^* &:= \underset{\tilde{u} \in \tilde{\mathcal{U}}^B}{\arg\min}\, P(C|\tilde{u}(\mathbf{x}), \mathcal{B}) \text{ with} \\ P(C|\tilde{u}^*(\mathbf{x}), \mathcal{B}) &= \mathrm{PCS}(\mathbf{x}), \end{aligned}$$

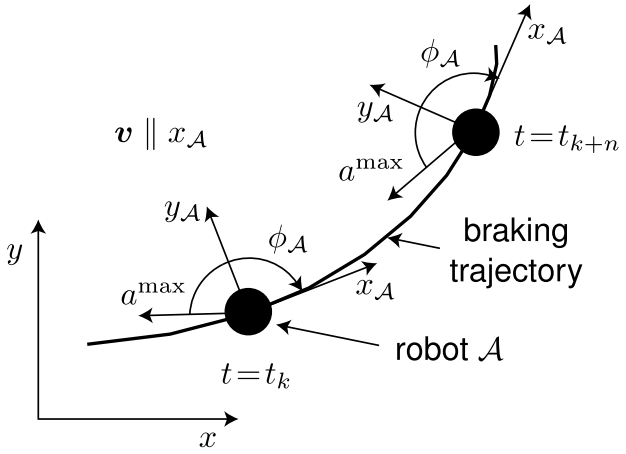


Fig. 4.4.: Braking trajectory of the robot. The direction of the acceleration is constant in the relative coordinate system of the robot and the magnitude is constant over time.

where $\mathcal{B}$ is represented by $\{f_1(\mathbf{x}, t), \ldots, f_{N_\mathrm{b}}(\mathbf{x}, t)\}$ assuming ignoring dynamic objects. Therefore, a finite set of possible input trajectories $\tilde{\mathcal{U}}^B$ of braking trajectories, i.e. trajectories for which the velocity is constantly decreasing, is generated. The finite set of braking trajectories is chosen as follows: use the maximum possible absolute acceleration such that $\sqrt{a_x^2 + a_y^2} = a^{\max}$, where the maximum absolute acceleration is limited through the contact friction of the robot. The parameter that is varied is the direction $\phi_\mathcal{A}$ of the acceleration, where the subscript $\mathcal{A}$ emphasizes that the direction is given in the relative coordinate system of the robot and not in the global workspace coordinates. The scalar product of the velocity vector and the direction vector $\phi_\mathcal{A}$ (both in relative coordinates) is always negative to ensure braking trajectories, see Fig. 4.4. The state trajectories are computed from the input trajectories with the constant acceleration model of (4.8). The collision probability for the robot trajectories is computed by discretizing the workspace as described in Sec. A.2.

Next, the computed input trajectory $\tilde{u}_\mathcal{A}^*$ minimizing $\mathrm{PCS}(\mathbf{x})$ is used to adapt the probability distribution $f_i(\mathbf{x}, t)$ for reactive objects such that it depends on $\tilde{u}_\mathcal{A}^*$: $f_i(\mathbf{x}, t) \rightarrow f_i(\mathbf{x}, t, \tilde{u}_\mathcal{A}^*)$.

Reactive Object Modeling

One of the difficulties in the implementation is that the probability distribution $f_i(\mathbf{x}, t, \tilde{u}_\mathcal{A})$ of reactive workspace objects depends on the input trajectory $\tilde{u}_\mathcal{A}$, while the choice of the robot trajectory $\tilde{u}_\mathcal{A}$ depends on the probability distribution $f_i(\mathbf{x}, t, \tilde{u}_\mathcal{A})$. This mutual dependency is broken up by first assuming that the probability distribution of the workspace objects is independent of the robot trajectory $\tilde{u}_\mathcal{A}$. In this section, the distribution of the reactive objects is described based on the optimized robot trajectory $\tilde{u}_\mathcal{A}^*$ that is defined in the previous section. In order to distinguish the trajectories of the workspace objects $\mathcal{B}_i$ from the ones of the robot, they are denoted by $\tilde{u}_i$ which are element of $\tilde{\mathcal{U}}_i$. A finite number of input trajectories $\tilde{u}_{i,k} \in \tilde{\mathcal{U}}_i$ is generated for each object as presented in the previous section. Therefore, the direction of the acceleration ϕ is varied while the maximum absolute acceleration $a^{\max}$ is applied. The additional index k in $\tilde{u}_{i,k}$ indicates the kth input trajectory for the ith object. The input trajectory causing the smallest value for $\mathrm{cPCS}^\mathrm{u}(f(\mathbf{x}_\mathcal{W}^i))$ is denoted by $\tilde{u}_i^*$. Here, the workspace state $\mathbf{x}_\mathcal{W}^i$ contains only the state of the robot and the ith object, meaning that collisions between the workspace objects are

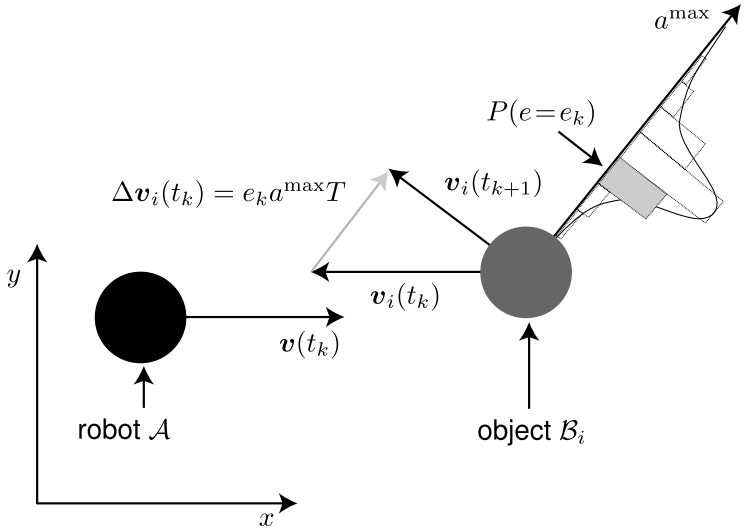


Fig. 4.5.: Acceleration applied to an object in order to avoid collision with the robot.

neglected for this implementation to reduce the computational complexity. The optimal input trajectories $\tilde{u}_i^*$ model the case when reactive workspace objects try to avoid a collision with maximum effort or willingness. However, reactive objects may not react with maximum effort to the trajectory of the robot. For this reason, the PMF $q(e)$ is introduced, where e is the effort varying in the interval $(0, 1]$ [1]. The applied absolute acceleration for the optimal acceleration direction of $\tilde{u}_i^*$ is obtained as $\sqrt{a_x^2 + a_y^2} = e\, a^{\max}$. If $e = 0$, the acceleration of the object is $a_x = a_y = 0$ and if $e = 1$, the full acceleration for avoiding the robot is applied as for $\tilde{u}_i^*$. But the direction of the acceleration is always the one used for generating $\tilde{u}_i^*$. Since only a finite number of values of e is used, the kth values is denoted by e_k. The PMF of e and the acceleration direction are depicted in Fig. 4.5. The final probability distribution is computed as

$$f_i(\mathbf{x}, t, \tilde{u}^*) = \sum_{k=1}^{N_e} q(e_k) f_{i,k}(\mathbf{x}, t, \tilde{u}^*),$$

where N_e is the number of considered values of e and $f_{i,k}(\mathbf{x}, t, \tilde{u}^*)$ is the distribution of the ith object when applying the acceleration direction of $\tilde{u}_i^*$ with the effort e_k. This motion prediction of the objetcs allows one to compute the collision probability for a robot trajectory leading to the $\mathrm{cPCS^u}$. In the next section, the evaluation of the cICS and $\mathrm{cPCS^d}$ implementation for the automotive application and the implementation of $\mathrm{cPCS^u}$ for the mobile robot applications based on simulation scenarios are presented.

4.6. Simulations

In this section simulation scenarios are used to evaluate the definitions of cICS, $\mathrm{cPCS^d}$ and $\mathrm{cPCS^u}$. Therefore, the two example implementations for automotive and service robotic applications are used.

[1] The interval $[-1, 1]$ would also consider hostile objects which do not fulfill the definition of reactive objects from Sec.1.2.2.

4.6.1. Automotive Applications

In order to check collisions during the planning horizon and to evaluate the safety of the final state, the future state of the other traffic participants need to be predicted. For the following simulation scenarios a constant velocity (CV) prediction is used. It is noted, that any deterministic prediction can be used for the cICS approach and any probabilistic prediction can be applied for the $\mathrm{cPCS}^{\mathrm{d}}$ approach.

In order to apply the cICS or $\mathrm{cPCS}^{\mathrm{d}}$ checker, the control input u_{N_v} of the first car in the road lane needs to be predefined. One possibility is to use the current control input which is similar to a constant acceleration prediction. For conservative results, the control input can be set to $u_{N_\mathrm{v}} = -1$ meaning full braking. For the following simulation scenarios the control input was set to zero assuming constant velocity for the first car.

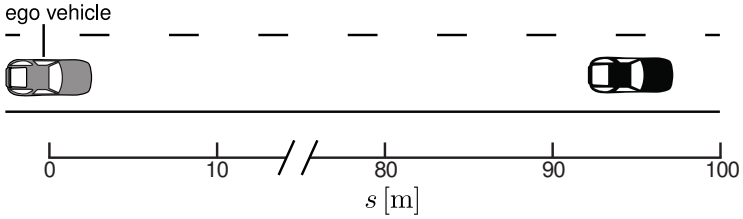
Two different common driving situations are considered. In the first scenario, the ego vehicle has a much higher velocity than the other vehicle driving ahead of it. Driving towards the end of a traffic jam is a typical example for such kind of scenario. The second scenario represents a lane change situation, at which the ego vehicle wants to change to the right-hand lane after overtaking.

End of a Traffic Jam

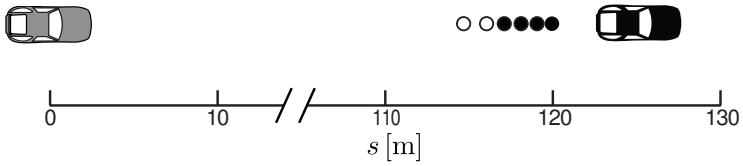
The first simulation setup represents the danger resulting from driving towards the end of a traffic jam. The initial state of the ego vehicle is $[s(0), v(0)] = [0\,\mathrm{m}, 30\,\mathrm{m/s}]$ and the initial state of the front vehicle is $[55\,\mathrm{m}, 8\,\mathrm{m/s}]$. For the generation of possible trajectory candidates for the ego vehicle, the motion planning approach described in [125] is used. The planning horizon and the time span of the trajectories of the motion planner are set to 4.0 s and the replanning time to 1.5 s. In Fig. 4.6a the scenario with the final states of the generated trajectories is illustrated. In the following, the concept of cICS and $\mathrm{cPCS}^{\mathrm{d}}$ are applied to this scenario to assess the safety of the final states of the trajectories beyond the planning horizon.

cICS During the planning horizon all considered trajectories are collision-free, without considering the result of the cICS evaluation. The resulting positions of the ego vehicle at time 4.0 s are shown in Fig. 4.6b as circles. The apparently best collision-free trajectory (most comfortable) is the one resulting in the rightmost state, since it does not change its velocity of 30 m/s and thus has zero jerk (maximum comfort). All other states result in a slower velocity. In Fig. 4.6c the replanning result after 1.5 s is shown after applying the apparently best trajectory. It can be seen, that all generated trajectories are leading to a collision with the front vehicle and no collision-free trajectory can be found when lane changes are not considered. The result of the cICS evaluation is depicted in Fig. 4.6b. It can be seen, that only braking trajectories with high deceleration are classified as safe by the cICS evaluation that will prevent the collision beyond the planning horizon.

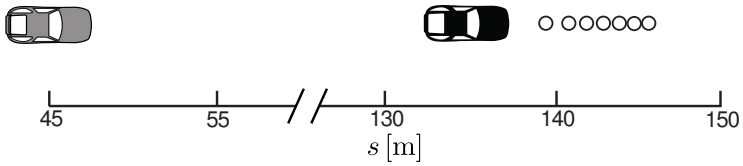
cPCS$^{\mathrm{d}}$ In order to evaluate this scenario with the $\mathrm{cPCS}^{\mathrm{d}}$ checker from Alg. 7 the PDF functions $f_{N_\mathrm{v}}(u), f_i(s), f_i(v)$ need to be specified. They are modeled as Gaussian distributions with the mean value of the CV prediction and the corresponding standard deviations are $\sigma_u = 0.2$,



(a) The ego vehicle drives towards a much slower vehicle.



(b) Planning result at $t = 0\,\mathrm{s}$.



(c) Planning result at $t = 1.5\,\mathrm{s}$. The final states of the trajectories are not cICS since the trajectories are already colliding during the planning horizon.

Fig. 4.6.: The images illustrate the scenario of a car driving towards the end of a traffic jam. The ego vehicle is shown on the left and the slower driving vehicle on the right. The circles depict the final states of the planning result, where solid circles represent states classified as cICS states.

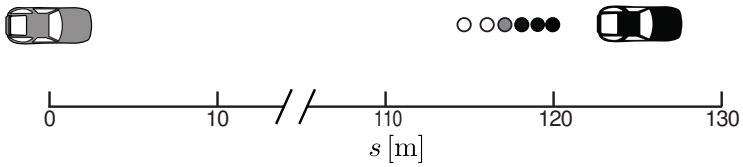
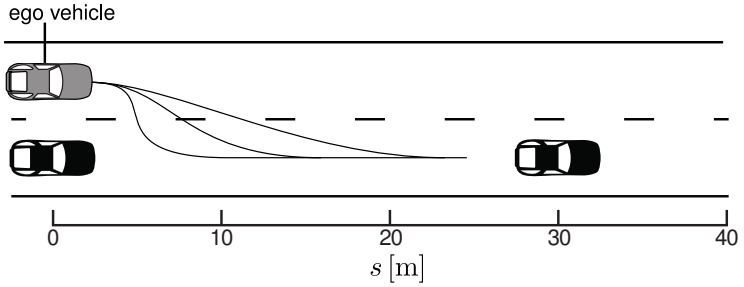


Fig. 4.7.: The image shows the result of the $\mathrm{cPCS}^{\mathrm{d}}$ evaluation of the traffic jam scenario shown in Fig. 4.6. Circles depict the final states of the planning result. The color indicates the value of the $\mathrm{cPCS}^{\mathrm{d}}$ evaluation. Black meaning 1.0 and white meaning 0.0.

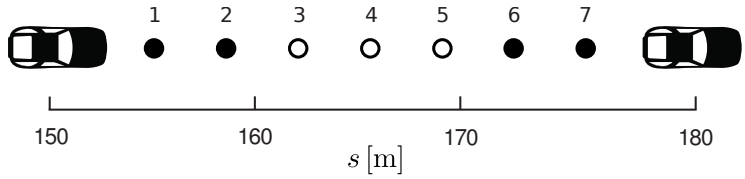
$\sigma_s = 5.0$ and $\sigma_v = 2.0$. For evaluating this scenario, 100 samples were generated according to these PDF functions. In Fig. 4.7 the result is depicted. The quantitative evaluation is $\mathrm{cPCS}^{\mathrm{d}} = [0.00, 0.05, 0.59, 0.91, 0.99, 1.00]$ starting from the left state. It can be seen, that the $\mathrm{cPCS}^{\mathrm{d}}$ checker is able to relax the binary evaluation of the cICS checker and allows one to consider the uncertainties from the prediction of other vehicles.

Lane Change Scenario

The aim of this simulation scenario is to show the result of the presented method during a lane change scenario. The ego vehicle starts on the left lane, while two other cars are driving on the right one. The aim is to find a safe trajectory that allows the ego vehicle to merge between the other vehicles. The simulation scenario is depicted in Fig. 4.8a. Using the navigation algorithm from [125], seven trajectories are generated that lead the ego vehicle into the desired gap. All



(a) The ego vehicle drives on the left lane and the other vehicles drive with a constant gap of $30\mathrm{m}$.



(b) The final states are depicted by circles, where solid ones are classified as cICS states.

Fig. 4.8.: The images illustrate the scenario of a car merging on another lane and the corresponding cICS evaluation.

seven final states have different velocities, where the state 1 has the lowest and state 7 the highest velocity. The seven states are evaluated by the cICS algorithm and the result is illustrated in Fig. 4.8b. Only three states are non cICS states, while the others would eventually lead to a collision beyond the planning horizon. The velocity of the left cICS states $(1, 2)$ is too slow, thus the rear vehicle cannot prevent a collision. In contrast to the right cICS states $(6, 7)$, where the velocity is too high, thus the ego vehicle will eventually collide with the front vehicle.

As mentioned in Sec. 4.5.1, the minimum determined control input u_{m} and the difference of the control inputs Δu_{r} regardless of the ego vehicle are provided. Without considering the ego vehicle, the maximum control input of the rear vehicle is 0.0, since the front vehicle has an control input of $u_{\mathrm{f}} = 0$ and the same velocity. The difference of the control inputs of the rear car considering the ego vehicle are $\Delta u_{\mathrm{r}} = [-0.51, -0.32, -0.25]$ for the states 3 to 5. As might be expected, the state 5 results in the least braking force (disturbance) for the rear vehicle. It is be mentioned, that this leads to tailgating with the front car and this behavior is not considered in this evaluation.

Discussion

The first simulation scenario demonstrates that the cICS check is indispensable to prevent a collision beyond the planning horizon. A shorter replanning time or a longer planning horizon may also prevent the collision, but there is no guarantee for this and it will result in a higher computational effort. Additionally, unnecessary motion oscillations of the vehicle can be avoided which are caused by a misleading safety assessment. For instance, during the lane change scenario an unsafe trajectory is applied which forces the ego vehicle to abort the maneuver or at least to adopt its trajectory by replanning. Furthermore, the proposed approach can also be used to validate the safety of possible navigation goals. Therefore, the cICS check is performed first and then the

Tab. 4.1.: Simulation parameters for the obstacles

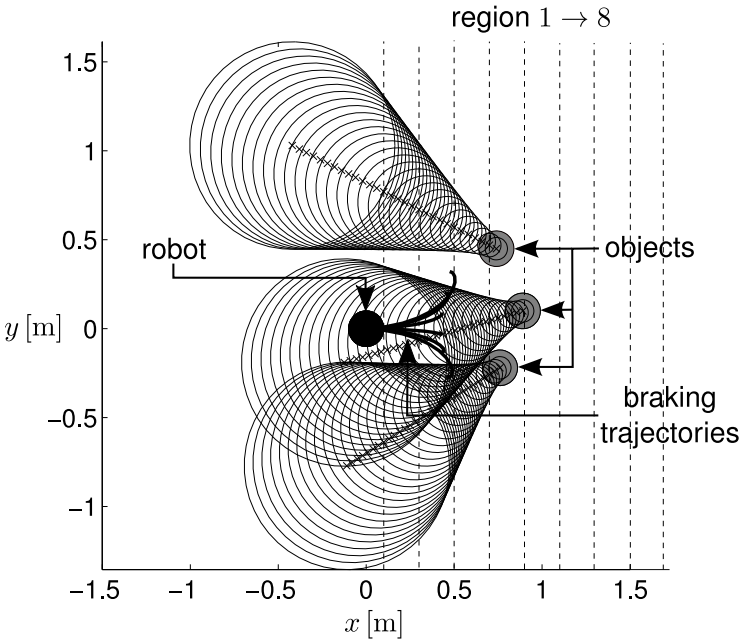
Eight regions for $x[\mathrm{m}]$	$[0.1, 0.3] \cdots [1.5, 1.7]$
One region for $y[\mathrm{m}]$	$[-0.5, 0.5]$
Initial velocity direction $\sphericalangle\mathbf{v}[\mathrm{rad}]$	$[\frac{3}{4}\pi, \frac{5}{4}\pi]$
Initial absolute velocity $\|\mathbf{v}\|$ $[\frac{\mathrm{m}}{\mathrm{s}}]$	$[0.0, 0.5]$
Acceleration direction $\sphericalangle\mathbf{a}[\mathrm{rad}]$	$[\frac{3}{4}\pi, \frac{5}{4}\pi]$
Absolute acceleration $[\frac{\mathrm{m}}{\mathrm{s}^2}]$:	$\{0.1, 0.3, 0.5, 0.7, 0.9\}$
Initial covariance $\Sigma(0)$ (see (4.9))	$\begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$

trajectories are generated. This order can reduce the calculation time if the used motion planner has a high computational complexity compared to the complexity of the cICS check.

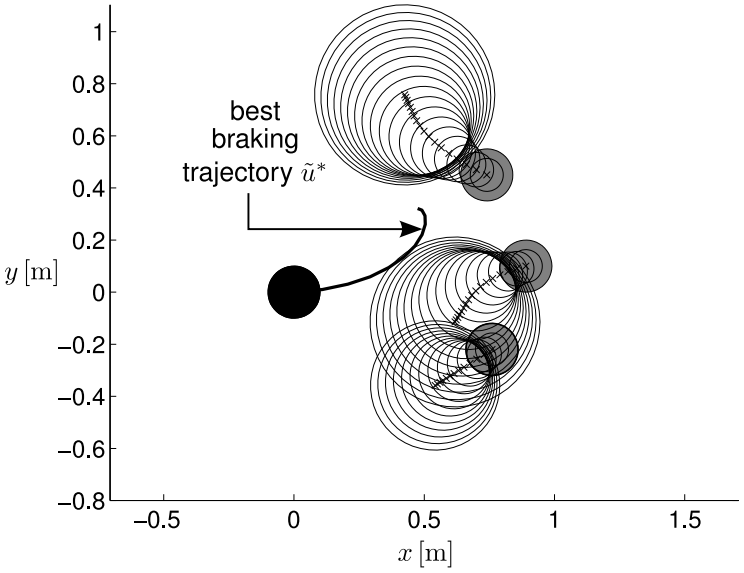
4.6.2. Mobile Robot Application

In this section, simulations are used to validate the $\mathrm{cPCS}^{\mathrm{u}}$ concept. The following simulations show the influence of the probabilistic effort $q(e)$ (for avoiding the robot) on the result of $\mathrm{cPCS}^{\mathrm{u}}(\mathbf{x}_{\mathcal{A}})$ for reactive objects. In order to show the usefulness of computing with a distribution of the effort for avoiding the robot, random scenarios are generated and evaluated. Despite the workspace objects, the initial state of the robot is fixed and has the initial state $\mathbf{x} = \begin{bmatrix} 0\mathrm{m} & 0\mathrm{m} & 0.5\frac{\mathrm{m}}{\mathrm{s}} & 0\frac{\mathrm{m}}{\mathrm{s}} \end{bmatrix}^T$. The objects are placed randomly in front of the robot facing towards it. Each scenario consists of one robot and three workspace objects. The workspace objects are placed randomly in one of eight predefined adjacent regions which are partitioned in x-direction. The regions and other parameters for the objects are listed in Tab. 4.1.
An example scenario using the listed parameters is shown in Fig. 4.9. In this scenario, the collision probability is reduced by 45% when considering reactive objects. For each of the eight regions 100 initial states are randomly generated for the objects. For this implementation, the ignoring and reactive objects are modeled as:

- Ignoring objects, i.e. objects that are not trying to avoid the robot. Thus, the probability distribution for the effort is $q(e) = \begin{cases} 0, \text{ if } e \in (0, 1] \\ \delta, \text{ if } e = 0 \end{cases}$, where δ is the Dirac impulse.
- Reactive objects, i.e. objects that are trying to avoid the robot. For the simulations, a Gaussian distribution for $q(e)$ is used with mean value $\mu = 0.5$ and standard deviation $\sigma = 0.2$. In order to obtain a finite number of effort values e_k, the Gaussian distribution is discretized.



(a) Motion prediction assuming ignoring objects, the associated collision probability is $\mathrm{PCS}(\mathbf{x}_{\mathcal{A}}) = 40\%$.



(b) Motion prediction assuming reactive objects, the associated collision probability is $\mathrm{cPCS}^{\mathrm{u}}(f(\mathbf{x}_{\mathcal{W}})) = 18\%$.

Fig. 4.9.: Random scenario for $x[\mathrm{m}]$ region 4: Gaussian distributions are illustrated by 2σ-ellipsoids.

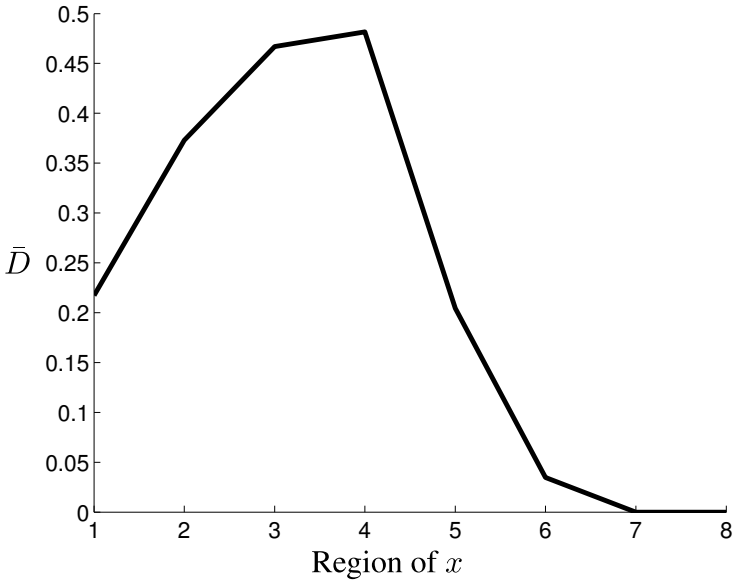


Fig. 4.10.: Relative difference between PCS (ignoring objects) and $\mathrm{cPCS}^{\mathrm{u}}$ (reactive objects).

In order to obtain relevant results, randomly generated situations with a collision probability of less than 0.01 are discarded. To verify the usefulness of modeling the avoidance capabilities of the objects, the mean relative difference

$$\overline{D} = \frac{1}{N_{\mathrm{s}}} \sum^{N_{\mathrm{s}}} 1 - \frac{\mathrm{cPCS}^{\mathrm{u}}}{\mathrm{PCS}}$$

obtained from all N_{s} scenarios is shown in Fig. 4.10. It can be seen that there is a significant difference between reactive and ignoring workspace objects. The maximum achieved difference is 48%. It can also be seen that the improvement depends on the distance of the robot to the objects when considering the velocity range and direction as listed in Tab. 4.1. There is no difference for greater distances than 1.3m since the collision probabilities are zero for reactive and ignoring workspace objects. During the evaluation, PCS was calculated 600 times for three workspace objects. The algorithm is implemented in Matlab and was executed on a AMD Phenom with 2.5 GHz. The mean computational time of $\mathrm{cPCS}^{\mathrm{u}}$ for one workspace state is $0.1\mathrm{s}$.

Discussion

The simulation results show a clear improvement of the safety assessment using $\mathrm{cPCS}^{\mathrm{u}}$ instead of PCS. Therefore, a suboptimal two-stage approach is used which brakes up the mutual dependency of the robot and object trajectories. However, it is not possible to give an upper error bound for this approach due to the decoupling of the original problem. The focus of these implementations is on an on-line capable algorithm which is suitable to be integrated into a navigation framework improving its safety assessment. The original $\mathrm{cPCS}^{\mathrm{u}}$ problem can be more complex than the classic motion planning problem. Since the $\mathrm{cPCS}^{\mathrm{u}}$ can be seen as an additional safety check, the response time of the algorithm is more important than the quality of the result as long as the result of $\mathrm{cPCS}^{\mathrm{u}}$ is conservative.

4.7. Discussion

In this chapter, the problem of assessing the safety of scenarios containing reactive or controllable objects is addressed. The main contributions of this chapter are novel definitions for safety assessment that reason over an infinite time horizon while taking into account the cooperative behavior of all reactive objects in the workspace. These includes the extension of the ICS concept, cICS for deterministic environments, and the two novel concepts $\mathrm{cPCS}^{\mathrm{d}}$ and $\mathrm{cPCS}^{\mathrm{u}}$ for stochastic environments. These concepts can be used as an additional safety assessment for PMP planners as described in Sec. 1.2.4. The following paragraphs briefly discuss the concept of cICS and cPCS.

Cooperative Inevitable Collision State The definition of cICS is suited for deterministic environments which are populated by reactive or controllable objects. The aim of this approach is to determine a safe future motion for all object including the robotic system, thus the motion safety can be guaranteed for an infinite time horizon. In other words, this concept assesses the states of all objects so that no one is an ICS state, while taking into account the cooperative behavior of the workspace objects. This concept is especially useful for densely packed environments that can cause the FRP [117]. Taking into account the reciprocal collision avoidance of these objects will lead to a less conservative assessment and is the only solution to relax the FRP.

An example implementation is presented for autonomous driving on highways. The computational complexity can be reduced by assessing each lane separately resulting in an 1D optimization problem for each vehicle. However, this leads to a conservative assessment since lane changes or variable control inputs are not considered in this implementation. If lane changes and variable control inputs for all vehicles would be taken into account, this may lead to an even more complex task than the original motion planning problem of the ego vehicle. Thus, this kind of extension would not be appropriate for an on-line application for autonomous driving.

Simulation scenarios showed the relevance of this safety assessment approach. In addition to the pure safety assessment, it is also possible to measure the disturbance of the other vehicles caused by the ego vehicle. In challenging situations, like a high density of vehicles, the cICS implementation can also be used to evaluate promising goal states before wasting computational time by generating trajectories to unsafe states.

For real-world applications, it is necessary to consider the reaction time of human drivers and their limited perception capabilities. A model for the perception system of the human driver will prevent the algorithm to react to vehicles which the driver cannot perceive.

The proposed concept can be applied to other motion planning methods in order to preserve the three criteria for motion safety mentioned in Sec. 1.2.1. This is particularly the case for the third criteria: reasoning over an infinite time horizon.

Cooperative Probabilistic Collision State The proposed definitions of $\mathrm{cPCS}^{\mathrm{u}}$ and $\mathrm{cPCS}^{\mathrm{d}}$ allow reasoning about the safety of planned motions in uncertain dynamic environments beyond the planning horizon. The presented definitions are especially useful in densely packed environments where the future motion of other objects in the workspace is highly uncertain. Due to the high uncertainty the collision risk of future trajectories of the robot increases which may lead to a FRP. To address this problem for stochastic modeled environments, cPCS is applied

considering the cooperative behavior of reactive objects. Two different definitions of cPCS are given which differ in the behavior of the objects. While objects according to the definition of $\mathrm{cPCS}^{\mathrm{d}}$ behave optimal regarding the present scenario, the objects behave optimal regarding all possible scenarios according to the definition of $\mathrm{cPCS}^{\mathrm{u}}$.

Two different applications, autonomous driving on highways and robot navigation in densely packed environments, are used to evaluate the presented concepts. For the automotive application the $\mathrm{cPCS}^{\mathrm{d}}$ definition is used which is based on the definition of cICS. It allows one to relax the binary assessment and considers uncertainty in the state of all objects.

For the mobile robot application in densely packed environments, the $\mathrm{cPCS}^{\mathrm{u}}$ definition is used which is less computational complex than the $\mathrm{cPCS}^{\mathrm{d}}$ definition. Moreover, the required optimization for estimating $\mathrm{cPCS}^{\mathrm{u}}$ is decomposed into: an optimization of the robot trajectory assuming ignoring objects; an optimization for the reactive trajectories of the objects based on this robot trajectory. This is done in order to avoid the mutual dependency of the robot and object trajectories. But this results in a non-optimal solution and in turn leading to a more conservative implementation of $\mathrm{cPCS}^{\mathrm{u}}$. In this example implementation, the willingness of objects to avoid the robot was considered which showed a big impact on the collision risk. Hence, the simulation results show that the consideration of the cooperative behavior of reactive objects is necessary to avoid the FRP. The simulations of the $\mathrm{cPCS}^{\mathrm{u}}$ checker showed that it is efficient and thus applicable to real-world scenarios.

5. Integration into Motion Planning

Summary The purpose of this chapter is to present the integration of the safety assessment concepts from previous chapters into motion planning approaches. The concept of cICS from Chap. 4 is integrated into a navigation framework for autonomous navigation on highways. It allows one to overcome former limitations regarding its safety assessment by guaranteeing motion safety during and beyond the planning horizon. The idea of closed-loop assessment from Chap. 2 is used to develop two motion planners which generate multiple motion possibilities for the robot to reach its goal. Thereby, the robotic system is able to react to changes in the environment improving the safety as well as the expected cost for the navigation task. The motion planners are evaluated by simulation scenarios from the field of mobile robot navigation. In Chap. 4 the problem of safety assessment considering the avoidance behavior of reactive objects is discussed. This idea is used in an interactive motion planning approach that considers the avoidance behavior as well as the goal directness of reactive objects.

The outline of this chapter is as follows: Sec. 5.1 provides the motivation and a general problem formulation for motion planning. Then Sec. 5.2 provides a brief overview of related work on motion planning. The integration of the cICS concept into a navigation framework for autonomous driving on highways is presented in Sec. 5.3. In Sec. 5.4 a motion planner for on-line and off-line planning incorporating replanning possibilities is presented. Inspired by the interactive safety assessment, a sampling-based motion planner considering the interaction and goal-directness of all objects in the workspace is described in Sec. 5.5. Sec. 5.6 presents example implementations of the proposed navigation approaches. The corresponding simulation results are presented in Sec. 5.7. Finally, a discussion of this chapter is given in Sec. 5.8.

5.1. Motivation and Problem Formulation

The novel safety assessment approaches from the previous chapters can be used to evaluate the safety of trajectories in a subsequent step after the trajectory generation. In the following sections, the integration of these safety assessment approaches into motion planning are presented allowing for a more efficient computation and a wider application to motion planning problems. First, the motivation and problem formulation for two different classes of motion planning problems are presented: motion planning in deterministic and in uncertain environments.

For deterministic environments the motion planning task is formulated as a continuous-time optimal control problem. In order to show the advantages of integrating the idea of safety assessment beyond the planning horizon into optimal control, a brief overview of related work on motion planning approaches for deterministic environments with the focus on autonomous driving is given in the next section.

For navigation in uncertain environments, the motion planning task is formulated as a discrete-time optimal control problem. Two variants are presented for considering dynamic objects in the

problem formulation: chance constraints and cost functions including the collision probability of the robot. Additionally, a brief overview of related work on motion planning approaches for uncertain environments is presented.

5.1.1. Motion Planning in Deterministic Environments

In this section the problem of navigation in deterministic and dynamic environments is discussed. In order to formally define the problem, the optimal control problem for deterministic environments is introduced [57]:

Problem 1 (Optimal Control Problem)**.** *The optimal control problem is to find the control function $u_c^*(t)$ which causes the system*

$$\dot{\mathbf{x}} = m(\mathbf{x}(t), u(t), t),\ \mathbf{x}(0) = \mathbf{x}_0$$

to minimize the cost function

$$\begin{aligned} \text{cost}(u(t)) &= \phi(\mathbf{x}(T), T) + \int_{t_0}^{T} L(\mathbf{x}(t), u(t), t)\,\mathrm{d}t \\ u_c^*(t) &:= \arg\min \text{cost}(u(t)) \end{aligned}$$

subject to

$$\begin{aligned} &\mathbf{x}(T) = \mathbf{x}_g \\ &\forall t\ \mathbf{x}(t) \in \mathcal{X}^{\mathrm{a}}(t) \textit{ with } \mathcal{X}^{\mathrm{a}}(t) = \{\mathbf{x} \in \mathcal{X} | \mathcal{A}(\mathbf{x}(t)) \cap \mathcal{B}(t) = \emptyset\} \\ &\forall t\ u(t) \in \mathcal{U}(t). \end{aligned} \tag{5.1}$$

The term T represents the time duration of the optimal trajectory $u_c^*(t)$ and $\mathbf{x}(t)$ describes the state along the trajectory. The subscript c indicates that a cost function is optimized instead of the pure collision probability. The admissible control set $\mathcal{U}(t)$ and the admissible state space $\mathcal{X}^{\mathrm{a}}(t)$ at time t ensure the kinematic and dynamic constraints of the robot. The admissible state space is time dependent in order to take into account dynamic objects in the workspace. In order to ensure that the robot reaches the goal state $\mathbf{x}_g$ at the final time T, the constraint $\mathbf{x}(T) = \mathbf{x}_g$ is introduced. The cost function contains the two terms $\phi(\cdot)$ and $L(\cdot)$ which are the endpoint cost and the trajectory cost, respectively.

This kind of formulation allows one to exactly define the motion planning problem: find a solution which starts at the initial state $\mathbf{x}_0$ and reaches the goal state $\mathbf{x}_g$. Many applications demand not only to reach the goal but also to choose the controls such that a predefined cost function is minimized. The cost function depends on the environment and task of the robot. Length, time duration, energy consumption or jerk of a trajectory are common criteria that are minimized. Furthermore, constraints are introduced to ensure that the optimized trajectories are executable by the robot and to define forbidden regions in order to prevent the robot from collisions. Thus, optimal control provides a natural formulation for motion planning in deterministic environments.

5.1.2. Motion Planning in Uncertain Environments

In this section, the problem of navigation in uncertain and dynamic environments is discussed. The challenges of motion planning in uncertain environments rise from the different sources of uncertainty [77]:

- Uncertainty in robot sensing (RS)
- Uncertainty in robot predictability (RP)
- Uncertainty in environment sensing (ES)
- Uncertainty in environment predictability (EP)

In this chapter, the focus is on applications that arise from mobile robot navigation in human populated environments. The occurring uncertainty originates mostly from the sources ES and EP in such kind of environments. Motion prediction of humans or human controlled systems is a difficult challenge, because it depends on internal states and intentions that cannot be measured directly. For example, the goal of the human is crucial information which is needed to predict his long-term motion. Since the uncertainty caused by the robot is usually much smaller than the one caused by the environment, we neglect the sources RS and RP to reduce the complexity of the motion planning problem. The same assumptions hold for autonomous car navigation in the presence of other human controlled vehicles or pedestrians [36].

In order to formally define the problem, the discrete-time optimal control problem without considering any other objects is introduced:

Problem 2 (Discrete-Time Optimal Control)**.** *The optimal control problem is to find a set of inputs* $\tilde{u}_c^* = \{u_c^{*,1}, u_c^{*,2}, \ldots, u^{*,N_\mathrm{T}}\}$ *which causes the system*

$$\mathbf{x}^{k+1} = m(\mathbf{x}^k, u^k),\ \mathbf{x}^0 = \mathbf{x}_0 \tag{5.2}$$

to minimize the cost function

$$\begin{aligned} \mathrm{cost}(u^0, u^1, \ldots, u^{N_\mathrm{T}-1}) &= \phi(\mathbf{x}^{N_\mathrm{T}}) + \sum_{k=0}^{N_\mathrm{T}-1} L(\mathbf{x}^k, u^k) \\ \tilde{u}_c^* &:= \arg\min \mathrm{cost}(u^0, u^1, \ldots, u^{N_\mathrm{T}-1}) \end{aligned} \tag{5.3}$$

subject to

$$\begin{aligned} \forall k\ \mathbf{x}^k &\in \mathcal{X}^k \\ \forall k\ u^k &\in \mathcal{U}^k \\ \mathbf{x}^{N_\mathrm{T}} &= \mathbf{x}_g. \end{aligned} \tag{5.4}$$

The variable t_k with $k \in \{0, 1, \ldots, N_\mathrm{T} - 1\}$ describes the discrete time points and N_T is the horizon or number of applied control inputs. $\mathbf{x}^k$ describes the state and u^k the control input at time t_k. The set $\mathcal{U}^k$ is the admissible input space and $\mathcal{X}^k$ is the admissible state space at time t_k for the robot. In order to ensure that the robot reaches the goal at the final stage N_T, the constraint $\mathbf{x}^{N_\mathrm{T}} = \mathbf{x}_g$ is introduced. The cost function (5.3) contains the two terms $\phi(\cdot)$ and $L(\cdot)$ which are the endpoint costs and the trajectory costs, respectively. This formulation does

not consider any uncertain objects in the workspace of the robot. In principle there are two possibilities to consider uncertain objects in optimal control based motion planners: introducing chance constraints or adapting the cost function to consider the collision probability of the robot. Chance constraints are defined as

$$\forall t \; P(C|\tilde{u}(\mathbf{x}(0), t)) < \tau_{\text{coll}}, \tag{5.5}$$

where $P(C|\tilde{u}(\mathbf{x}(0), t))$ is the collision probability that the robot will collide at time t while executing the trajectory $\tilde{u}$ starting at $\mathbf{x}(0)$. For integration into Prob. 2, (5.5) is added to the constraints in (5.4). This chance constraint ensures that at each time point the collision probability of the robot trajectory is below a predetermined threshold τ_{coll}. The upper bound of the resulting collision probability of an entire trajectory is

$$\bar{P}(C|\tilde{u}_c^*) = 1 - \big(1 - \tau_{\text{coll}}\big)^{N_{\text{T}}}.$$

In [15] chance constraints are introduced for static objects which positions are exactly known without any noise. This approach is extended by [8] allowing chance constraints to be used for uncertain dynamic objects.

Instead of using chance constraints, a different cost function can be used which depends on the trajectory costs and the collision probability, also called probability of failure of the trajectory,

$$\text{cost}(\tilde{u}) = h\big(P(C|\tilde{u}(\mathbf{x}(0))), L(\tilde{u})\big). \tag{5.6}$$

For integration into Prob. 2 the original cost function (5.3) is replaced by this cost function (5.6). The function $h(\cdot, \cdot)$ can, for instance, be a weighted sum or a product and adjust the tradeoff between both criteria. The trajectory costs $L(\cdot)$ are now calculated for the entire trajectory rather than an additive cost function for each stage.

In order to illustrate the difference between both possibilities, an illustrative example is used. The scenario is sketched in Fig. 5.1. The task of the robot is to reach a predefined goal location while minimizing the length of the trajectory. As can be seen, with chance constraints the robot prefers to navigate between the noisy objects assuming a probability of collision close to the predefined threshold τ_{coll}. Integrating the objects in a cost function, e.g. $h = P(C|\tilde{u}(\mathbf{x}(0))) + l(\tilde{u}(\mathbf{x}(0)))$ with $l(\cdot)$ is the length of $\tilde{u}$, results in a trajectory which is slightly longer than the other solution but with a clearly smaller collision probability. This example shows that the solutions from chance constrained optimal control may lead to unnecessary risky trajectories. The reason for this is, that the chance constraints do not allow optimizing the tradeoff between trajectory cost and the collision probability but rather define an upper bound of collision probability. In many applications, especially in the presence of humans, the robot should minimize the tradeoff between efficiency of the trajectory and the collision probability. If the threshold for the chance constraints is too low, the robot may often get stuck since no trajectory can be found that fulfills the constraints. Otherwise, choosing a higher threshold may result in unnecessary risky trajectories in many situations, as shown in Fig. 5.1. Thus, in this chapter the motion planning approaches for uncertain and dynamic environments consider the objects in the cost function by the collision probability of the robot trajectory. In the next section, a brief overview of related work on motion planning in deterministic and uncertain environments

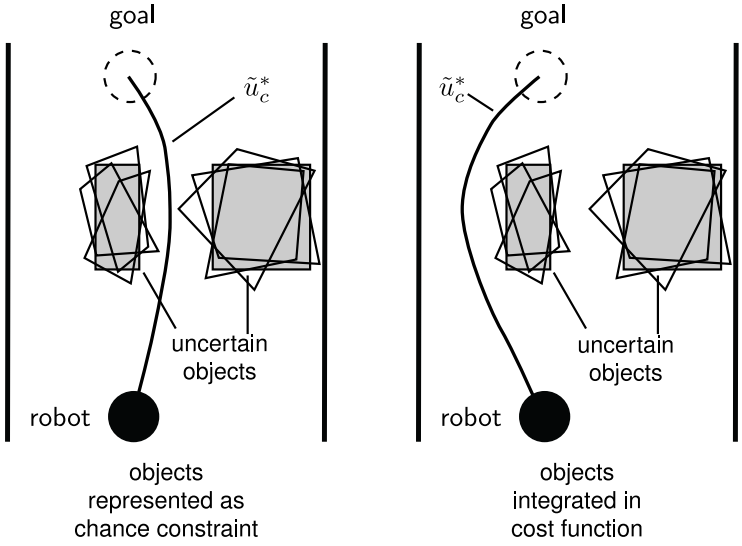


Fig. 5.1.: This example scenario of motion planning in uncertain environments sketch the different result if the objects are considered as chance constraints or in the cost function by the collision probability of the trajectory.

is given. Therefore, the focus is set on autonomous driving and on mobile robot navigation in populated environments as an example application for deterministic environments and uncertain environments, respectively.

5.2. Related Work

In order to allow a robotic system to fulfill a certain navigation task, a motion planning problem must be solved. Motion planning or its related problem path planning are known to be complex tasks. In [97] it was shown that the problem of finding a path for a robot represented by several polyhedral parts in environments with polyhedral objects is PSPACE-hard. An algorithm for non-polyhedral objects in the environments is presented in [105]. Its time complexity is twice exponential in the dimension of the configuration space. A more efficient algorithm is presented in [19] which has single exponential complexity regarding the configuration space.

In the last decades, many different solutions for a variety of motion planning problems are developed. For a general and detailed overview of motion planning approaches the reader is referred to the well-known monograph of Latombe [69] which presents most of the basic approaches and their properties for various motion planning problems. The books of LaValle [73] and Laugier and Chatila [70] include more recent methods.

5.2.1. Autonomous Navigation of Vehicles

Many approaches for autonomous driving belong to the class of sampling-based motion planners. These include rapidly exploring random trees (RRT) [72] and state lattices planner [94]. The state lattice approach can be seen as a generalization of a grid and allows performing efficient constrained motion planning as a heuristic search. In the construction phase the state lattice planner generates a grid of lattice nodes and adds connections to it if there exists a feasible path

between any two nodes. Inverse trajectory generation is used for computing the edges between the lattice nodes. The lattice grid is overlaid by a cost map to include objects in the motion planning. Then, heuristic graph search algorithms are applied to find the optimal sequence of edges in the lattice to a predefined goal location. The RRT algorithm belongs to the class of probabilistic motion planners. States in the configuration space of the robot are randomly sampled and connected to the closest node in the tree. It is not required that these two states are exactly connected which makes this approach also applicable to systems with difficult constraints. Moreover, the RRT planner satisfies the criteria for probabilistic completeness: the probability that a trajectory is found connecting the start and goal approaches 1 if the number of samples approaches infinity. Both approaches, the RRT and the lattice planner, have been applied successfully in full-scaled autonomous vehicles [63, 82]. As stated in [125], these methods are most useful for combinatorial difficult problems encountered in environments such as parking lots, but they cannot guarantee to find the optimal solution in finite time or to provide solutions to a set of specified goals.

Another related field is model predictive control (MPC) also called receding horizon control (RHC). Instead of solving the complete motion planning task at once, it iteratively generates partial trajectories using a dynamic model of the robotic system to predict its future states. Most approaches use open-loop MPC which generates a sequence of control inputs. The algorithm directly takes into account the kinematic and dynamic constraints of the robot as well as the environment constraints. At the same time the control inputs are optimized to minimize a specified cost function. Due to its incremental character, this approach is especially useful for dynamic environments. In [104] a RHC problem is formulated as a mixed-integer linear programming problem for multi-vehicle planning. Due to the integer variables, hard objects and collision avoidance constraints can be integrated in the RHC approach. In [6] a MPC approach is used in a semi-autonomous control framework. The MPC generates continuously an optimal trajectory which is used to obtain the minimum threat posed to the vehicle. Based on the result of the threat assessment, the level of intervention is adjusted to prevent the vehicle from leaving a safe corridor. Due to the high complexity of the MPC-based trajectory generation, this method is not suitable for generating trajectories for long time horizons.

The two capabilities, considering a set of alternative goal states and replanning with high frequencies, are important requirements for motion planning at high-speeds including object avoidance. Therefore, in [52] and [88] a strategy is suggested which takes advantage of the structure in the road environment by considering multiple final states. Based on this idea, an optimal control motion planner is presented in [125] using quintic polynomials. It generates optimal trajectories to a variety of goal states resulting in a so called fan-shaped trajectory set. An unconstrained optimal control approach is used to generate this set of trajectories which is checked for feasibility in a subsequent step. This allows a replanning cycle time of less than $100\,\mathrm{ms}$ whereby a maximum of more that 1000 alternative trajectories are generated. The alternative trajectories in combination with the high replanning frequency enable the system to react to unforeseen traffic changes.

Recently, the project *HAVEit* compared two approaches for high-speed driving in structured environments in [98]. As a result, the above mentioned quintic polynomial based approach outperformed the other method based on searching the optimal solution in the discretized command space. The special demands for high-speed driving are: smooth maneuvers, high level of safety,

hard real-time constraints and a long foresight. Hence, the approach of [125] is applied to this problem in Sec. 5.3. The purpose is to show the integration of the safety assessment concept from Chap. 3 since safety beyond the planning horizon was originally not considered by this approach.

5.2.2. Mobile Robot Navigation in Populated Environments

In [13] a navigation approach is presented based on motion patterns of people. These motion patterns are learned from recorded human motions in the environment allowing for predicting the future motion of humans. The predicted patterns of the humans are mapped in a 2D occupancy grid storing the probability of occupancy of each cell. Based on this map, an A* search is applied to find a minimum-cost path to the goal of the robot. If the robot path intersects with a human, the robot changes its velocity or stops to allow the human to pass the robot. The work of [101] uses a similar method for predicting the future motion of humans, however, the robot uses also the same motion patterns for its navigation. Thus, the robot behaves more human like and chooses motion patterns that minimize the risk of disturbing the humans. These two approaches are typical examples predicting the future motion of humans and applying common path or motion planners to find an optimal trajectory which minimizes the disturbance for humans as well as the navigation cost for the robot. Other similar approaches are e.g. [68] and [111]. More recent approaches for human motion prediction, such as [9, 129] and [87], use inverse optimal control to determine the cost function explaining the motion of humans.

Due to the highly uncertain motion prediction of humans, the approaches mentioned before may generate trajectories with high collision probabilities. One reason for this is, that these approaches ignore the collection of new information about the environment while executing a trajectory. There are only few navigation approaches taking into account multiple future distributions of objects representing their future states. These distributions can also be seen as unknown future measurements. In [116] a stochastic dynamic programming formulation is given which needs to solve the optimal control problem for all possible measurements resulting in an optimal feedback policy. In a stochastic environment the set of possible measurements is infinite. That is why most common approaches ignore the possible measurements beyond the current state of the robot. This is called an open-loop receding horizon control problem. As stated in [127] this leads to conservative results. However, a closed-loop receding horizon control (CLRHC) problem is in most cases not applicable due to the infinite set of possible future measurements. This argument induced the authors of [116] to introduce their partially closed-loop receding horizon control (PCLRHC) approach. Instead of considering all possible future measurements, only the most likely measurement is assumed. But this may lead to a non-conservative result, meaning that this approach can violate some constraints of the receding horizon control problem such as chance constraints. For instance, if the most likely measurement and the real measurements vary considerably for a certain time horizon, the real collision probability for the future robot motion may be higher than the predicted one.

The work of [117] claims that even the perfect motion prediction without any uncertainty can still lead the robot to an FRP. The reason for this is that the interaction between the objects and the robot is ignored in the common motion planning algorithms for populated environments. Hence, it is not valid to decouple the motion planning problem into motion prediction and trajectory generation. Crowd simulation is a related field of motion planning in populated environments

considering reciprocal collision avoidance. During a crowd simulation hundreds or even thousands of agents are simulated simultaneously, thus the computational efficiency of the navigation algorithm is crucial. A precomputed roadmap [69] is used in [119] to obtain a high-level cognitive map of the environment. During the execution of the agents, the reciprocal velocity obstacle (RVO, extension of the velocity obstacle approach [30]) approach is used for the collision avoidance behavior of the agents. Instead of the RVO, a social force model [49] is applied by [110] to update the roadmap continuously based on the inter-agents interaction forces. A more recent approach is the optimal reciprocal collision avoidance presented by [121] which overcomes some limitations of the RVO approach. It can guarantee collision-free navigation for multiple robots assuming a complete model of the environment. Some extensions exist which also incorporate the kinematic and dynamic constraints of the robot model as done in the work of [109] and [1]. If the future motion of the surrounding objects is only available through stochastic motion prediction, the problem becomes more challenging. Only few navigation algorithms consider the avoidance possibilities of workspace objects in uncertain environments. One of the first approaches was published by [58]. It is called reflective navigation and uses recursive probabilistic velocity obstacles which are an extension of the probabilistic velocity obstacles. The robot reflects on its environment and incorporates the avoiding behavior of the other moving objects into its own planning. An algorithm for joint collision avoidance, which uses interacting Gaussian processes, is presented by [117]. Every trajectory of the objects is modeled as a Gaussian process and the interaction between all objects including the robot is represented by an interaction potential which is proportional to the Euclidean distance of the corresponding agents.

In the following, two novel motion planning algorithms for uncertain environments based on the closed-loop assessment from Chap. 2 are shown in Sec. 5.4 and an interactive motion planner based on Chap. 4 is presented in Sec. 5.5.

5.3. Optimal Control Considering Safety Beyond the Planning Horizon

In this section the cICS concept from Chap. 4 is integrated in an optimal control approach with focus on high-speed driving in structured environments like country roads as well as highways. It is noted, that the integration is not limited to autonomous driving and can be used for various motion planning tasks formulated as optimal control problems. The special demands for high-speed autonomous driving are: smooth maneuvers, high level of safety, hard real-time constraints and a long foresight. As mentioned in Sec. 5.2.1, the approach described in [125] is appropriate for this kind of task. This approach makes use of quintic polynomials but instead of generating trajectories in world coordinates they are represented in lane coordinates of the road. This allows for generating comfortable minimum jerk maneuvers at a wide speed range. Therefore, an optimal control approach is used excluding other vehicles and kinodynamic constraints which is presented in the following subsections. This approach, which is presented in detail in [125], is combined with the cICS concept from Sec. 4.3 addressing the problem of safety beyond the planning horizon. Since no on-line optimization is included in this motion planning framework, it has a bounded response time and is on-line capable. The following brief description of the optimal control approach is taken from [137].

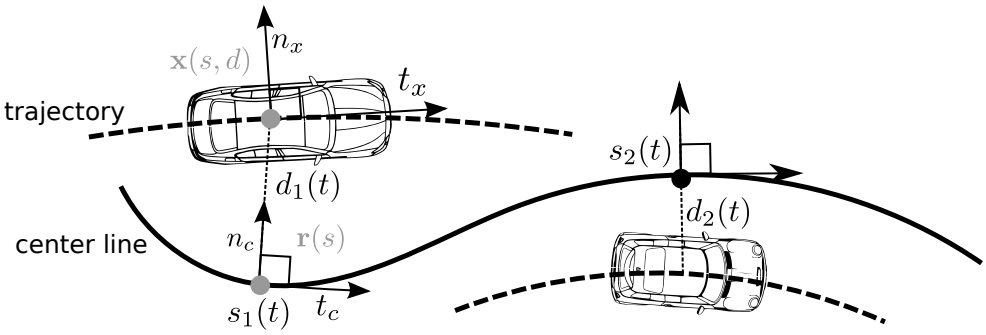


Fig. 5.2.: Vehicles represented in the Frenet Frame.

Street Relative Coordinates and Cost Decomposition

In most traffic situations the human driver plans the vehicle's lateral movement relative to the lanes rather than to the absolute ground. Imitating this approach, the trajectory generation problem is formulated in the so-called Frenet frame $[\mathbf{n}_c, \mathbf{t}_c]$ of the street, shown in Fig. 5.2. Here, the offset to the lane center is denoted by $d(t)$ and $s(t)$ describes the covered arc length of the frame's root point $\mathbf{r}(s)$ along the center line. Next, it is assumed that the trajectory cost cost may be separated into a lateral and a longitudinal component, cost_d and cost_s, according to the weighted sum $\mathrm{cost}(d,s) = \mathrm{cost}_d(d) + w_s\,\mathrm{cost}_s(s),\ w_s > 0$. It can be shown (see e.g. [125]) that the unconstrained movement (no restrictions such as objects) of $d(t)$ that transfers the vehicle from the initial state $[d(0), \dot{d}(0), \ddot{d}(0)]$ to a given end state $[d(T), \dot{d}(T), \ddot{d}(T)]$ while minimizing the cost function (Jerk) is a fifth-order polynomial. This gives us the general shape of the lateral trajectory, such that only the end state and the end time is left for optimization. The target application narrows a priori a set of reasonable solutions. The vehicle should generally progress along the road and not crosswise, thus, $d(t)$'s first and second derivative at time T is constrained to be zero. As for the choice of the terminal time, reaching the end state too early might lead to uncomfortable, energetically wasteful actions, whereas a too late arrival implies lagging movements. Since these issues are also strongly coupled with the end state, the goal is to find the best trade-off by defining the terminal cost to be

$$\phi(d(T), T) := (w_T\, t + \tfrac{1}{2} w_d [d(t) - d_{\text{ref}}]^2)_T$$

with $w_T, w_d > 0$ which penalize both slow convergence and final deviations from the reference trajectory with e.g. $d_{\text{ref}} = 0$. In order to reduce the number of end states, the lateral trajectory is only allowed to arrive at certain points in absolute time as well as with certain discrete distances to the reference trajectory d_{ref}. Consequently, all admissible polynomials form an entire fan-shaped trajectory set evenly covering the maneuver space as shown on the left of Fig. 5.3. Analog considerations lead to a set of polynomial movements for the longitudinal velocity $\dot{s}(t)$ which can be seen on the right of Fig. 5.3.

Compared to the optimal-control approach from Sec. 5.1.1, this approach does not consider any kinematic or dynamic constraints of the robot nor other objects in the workspace. Therefore, the longitudinal set is crosswise superimposed with the lateral set and back-transformed to global coordinates to compute parameters such as accelerations, curvature and velocities of the trajectories. Afterwards, trajectories which do not satisfy the constraints of the vehicle are ne-

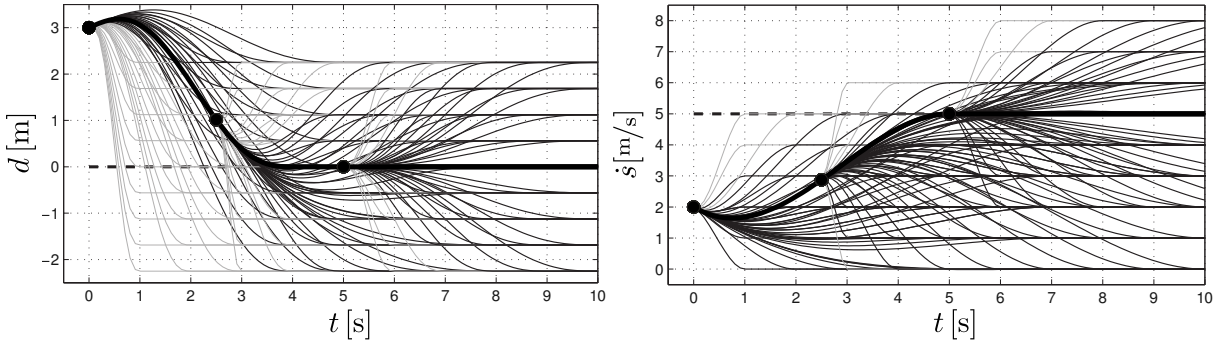


Fig. 5.3.: Simulation of an optimal transfer to the dashed reference by replanning [137]. In each step the thick line is the optimal trajectory, black are the valid and gray the invalid alternatives.

glected from the fan-shaped trajectory set. Furthermore, trajectories which are colliding during the planning horizon are identified by the approach described in [78] and are also removed from the trajectory set. Loosely speaking, the constrained optimal control approach is decoupled into unconstrained trajectory generation and a subsequent constraint check. In the following, the integration of cICS is presented which identifies trajectories that are colliding with objects beyond the planning horizon of the trajectories.

cICS Integration

The presented trajectory generation approach can be characterized as a partial motion planning (PMP) [92] method since the trajectories have a limited time horizon and do not reach the global navigation goal. Thereby, the problem arises that the final state of the vehicle may have a non-zero velocity. As pointed out in [92] it is indispensable to assess the safety of partial trajectories by using ICS [33] for evaluating the final state of the trajectory. For the integration in the framework of [125] the cICS approach is used considering the reactive behavior of the surrounding vehicles. This results in a less conservative assessment compared the the ICS assessment as discussed in Chap. 4. The improved framework fulfills all three safety criteria presented in 1.2.1. The presented cICS checker from Sec. 4.3 is used to evaluate the safety of the final state of each trajectory in the fan-shaped trajectory set. It is recalled that a workspace state $\mathbf{x}_{\mathcal{W}}$ is an Cooperative Inevitable Collision State (cICS) iff

$$\text{cICS} \;\Leftrightarrow\; \forall \tilde{\mathbf{u}}(\mathbf{x}_{\mathcal{W}}) \in \tilde{\boldsymbol{\mathcal{U}}} \;\; \exists t \;\; \exists (i,j|i \neq j) : \;\; \mathcal{B}_i^A(\tilde{u}_i(t)) \cap \mathcal{B}_j^A(\tilde{u}_j(t)) \neq \emptyset.$$

For integration into the optimal control formulation the following constraint is added to (5.1) in the optimal control problem Prob. 1:

$$\text{cICS}(\mathbf{x}_{\mathcal{W}}(T)) = 0, \text{ with } \mathbf{x}_{\mathcal{W}}(T) = \{\mathbf{x}_{\mathcal{A}}(T), \mathbf{x}_{\mathcal{B}_1}, \ldots, \mathbf{x}_{\mathcal{B}_{N_\text{b}}}\}.$$

The state $\mathbf{x}_{\mathcal{A}}$ is equivalent to the state $\mathbf{x}$ in the optimal control formulation. This constraint replaces the former end state constraint $\mathbf{x}(T) = \mathbf{x}_g$, since for the application of driving on a highway there is no real goal state. Furthermore, the time T describes the planning horizon of the motion planner rather than the end time. Considering the same motion planning problem, but assuming ignoring objects in the environments, the ICS approach can be used by adding the

constraint

$$\mathrm{ICS}(\mathbf{x}(T)) = 0.$$

This extended motion planning approach for autonomous driving on highways including safety assessment beyond the planning horizon is evaluated by simulations in Sec. 5.7.1. Therefore, the implementation described in Sec. 4.5.1 is used to evaluate each final state of the fan-shaped trajectory set if it is an cICS state.

5.4. Motion Graph Planning

In this section, two possibilities are shown to integrate the safety assessment approach from Chap. 2 into motion planning algorithms. Two different kind of integration possibilities are given: an *on-line* and an *off-line* integration. The integration into roadmap-based planners is used as an example for off-line planning. The trajectories are generated without any information about the dynamic objects. The on-line integration uses a given solution to the motion planning problem and improves this result by generating multi-edges allowing the route to be replanned during execution. The multi-edges are optimized by minimizing the expected cost between two vertices taking into account the motion prediction of the dynamic objects. The first proposed integration is the *on-line* and the second the *off-line* approach.

5.4.1. On-line Planning

The problem of motion planning in uncertain and dynamics environments with a deterministic robot can be stated as an optimal control problem (Sec. 5.1.2). In this case, the objects $\mathcal{B}$ of the workspace are considered in the cost function (5.6). The cost function contains the term $P(C|\tilde{u})$ which penalizes the collision cost regarding the objects $\mathcal{B}$. The second term of the cost function, $L(\cdot)$, defines the trajectory cost of the robot. Common cost functions are length, duration or energy consumption of the trajectory. The purpose of the function $h(\cdot)$ is to adjust the tradeoff between trajectory cost and safety (collision probability) of the optimal trajectory $\tilde{u}_c^*$. The solution to the optimal control problem is the optimal trajectory $\tilde{u}_c^*$ between the initial and goal state of the robot.

However, the idea of closed-loop assessment from Chap. 2 shows that it is possible to reduce the expected collision probability by generating additional motion possibilities. Thus, it is possible to generate a motion graph which has a lower expected collision probability than the optimal trajectory $\tilde{u}_c^*$ that ignores future observations. The goal of motion planning in uncertain environments is not to reduce the expected collision probability but the expected cost. To make this possible, it is necessary to adapt some definitions from Chap. 2. Therefore, the same environmental model from Sec. 2.4.3 is assumed and Def. 2 (minimum collision trajectory) is adopted:

Definition 14 (Minimum cost trajectory $\tilde{u}_c^*$ between two vertices $\mathbf{v}_i$, $\mathbf{v}_j$ regarding one distribution)**.**

$$\tilde{u}_c^*(\mathbf{v}_i, \mathbf{v}_j, f^{t'}(\mathbf{x}, t|\boldsymbol{\theta}_k)) := \underset{\tilde{u}\in\mathbf{e}_{ij}}{\arg\min}\, \mathrm{cost}(\tilde{u}, f^{t'}(\mathbf{x}, t|\boldsymbol{\theta}_k)).$$

This definition allows one to adopt the Def. 3 (collision probability between vertices connected by multi-edges) to determine the expected cost between two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ regarding multi-edges.

Definition 15 (Expected cost between two adjacent vertices $\mathbf{v}_i$, $\mathbf{v}_j$ with multi-edges)**.**

$$\begin{aligned}\operatorname{cost}(\mathbf{e}_{ij}, \mathbf{b}_{t_k}) :&= \mathrm{E}_{\boldsymbol{\theta}(t_k)}\left[\operatorname{cost}\left(\tilde{u}_c^*(\mathbf{v}_i, \mathbf{v}_j, f^{t_i}(\mathbf{x}, t|\boldsymbol{\theta}))\right)\right] \\ &= \int \operatorname{cost}\left(\tilde{u}_c^*(\mathbf{v}_i, \mathbf{v}_j, f^{t_i}(\mathbf{x}, t|\boldsymbol{\theta}))\right) f\left(\boldsymbol{\theta}(f^{t_k}(\mathbf{x}, t_i))\right) \mathrm{d}\boldsymbol{\theta},\end{aligned}$$

with $t_i > t_k$ and $\tilde{u}_c^*$ is the trajectory with the lowest cost.

According to Prop. 1 and Prop. 2 it can be shown that

$$\operatorname{cost}(\mathbf{e}_{ij}, \mathbf{b}_{t_k}) \leq \min_{\tilde{u} \in \mathbf{e}_{ij}} \operatorname{cost}(\tilde{u}, \mathbf{b}_{t_k}).$$

Thus, it is possible to generate a motion graph with lower expected cost than the optimal trajectory $\tilde{u}_c^*$. In the following, the graph reduction algorithm from Sec. 2.7.2 is adapted to compute the navigation cost of an entire motion graph instead of the pure collision probability. Therefore, the equations (2.6) and (2.7) need to be adopted. For every pair of vertices connected by multi-edges, Def. 3 is applied to determine the function

$$g_c(\mathbf{e}_{ij}, \mathbf{b}_{t_k}) \rightarrow [\pi_c^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}), P(C|\pi_c^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}), \mathbf{b}_{t_k}), L(\pi_c^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}), \mathbf{b}_{t_k})] \tag{5.7}$$

which identifies the optimal policy π_c^* for a given believe $\mathbf{b}_{t_i}$, the collision probability between the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$, and the trajectory cost for a given $\mathbf{b}_{t_k}$. The optimal policy is defined as

$$\pi_c^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}) := \arg\min_{\pi} h\big(P(C|\pi(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i})), L(\pi(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}))\big),$$

and $\pi(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}))$ returns the optimal trajectory given the belief $\mathbf{b}_{t_i}$. Note, that π_c^* and the collision probability as well as the trajectory cost can depend on different observation times. After merging all multi-edges in $\mathcal{H}$ (subgraph according to Def. 4), rule (2.5) is applied to all vertices $\mathbf{v}_k$ with $\deg^-(\mathbf{v}_k) \geq 1$ and $\deg^+(\mathbf{v}_k) = 1$ which are connected to the goal vertex $\mathbf{v}_g$. Therefore, the two functions $g_c(\mathbf{e}_{ik}, \mathbf{b}_{t_l})$ and $g_c(\mathbf{e}_{kj}, \mathbf{b}_{t_l})$ need to be merged by

$$\begin{aligned} g_c(\mathbf{e}_{ik}, \mathbf{b}_{t_l}) \times g_c(\mathbf{e}_{kj}, \mathbf{b}_{t_l}) &\rightarrow g_c(\mathbf{e}_{ij}, \mathbf{b}_{t_l}) \\ \pi_c^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}) &= \{\pi_c^*(\mathbf{v}_i, \mathbf{v}_k, \mathbf{b}_{t_i}), \pi_c^*(\mathbf{v}_k, \mathbf{v}_j, \mathbf{b}_{t_k})\} \\ P(C|\pi_c^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}), \mathbf{b}_{t_l}) &= P(C|\pi_c^*(\mathbf{v}_i, \mathbf{v}_k, \mathbf{b}_{t_i}), \mathbf{b}_{t_l}) \\ &\quad + (1 - P(C|\pi_c^*(\mathbf{v}_i, \mathbf{v}_k, \mathbf{b}_{t_i}), \mathbf{b}_{t_l}))P(C|\pi_c^*(\mathbf{v}_k, \mathbf{v}_j, \mathbf{b}_{t_k}), \mathbf{b}_{t_l}) \\ L(\pi_c^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}), \mathbf{b}_{t_l}) &= L(\pi_c^*(\mathbf{v}_i, \mathbf{v}_k, \mathbf{b}_{t_i}), \mathbf{b}_{t_l}) + L(\pi_c^*(\mathbf{v}_k, \mathbf{v}_j, \mathbf{b}_{t_k}), \mathbf{b}_{t_l}), \end{aligned}$$

where $\{\pi_c, \pi_c'\}$ is the concatenation of two policies. The collision probability and the trajectory cost are stored separately to allow for a fast calculation of the overall navigation cost $\operatorname{cost}(\cdot)$ if the function $h(\cdot)$ is changed. These two adopted rules allow for computing the cost for an entire graph by the graph reduction algorithm from Sec. 2.7 and in turn to define the problem of optimal motion graph planning.

Problem 3 (Optimal Motion Graph)**.** *Given a cost function*

$$\text{cost}(\tilde{u}, \mathbf{b}_{t_s}) = h\big(P(C|\tilde{u}, \mathbf{b}_{t_s}), L(\tilde{u})\big)$$

the optimal motion graph problem is to find a graph $\mathcal{G}^* = \{\mathcal{V}^*, \mathcal{E}^*\}$ *which enables the system*

$$\mathbf{x}^{k+1} = m(\mathbf{x}^k, u^k),\ \mathbf{x}^0 = \mathbf{x}_0,$$

to minimize the cost function by determining the optimal policy $\pi_c^*(\mathbf{v}_s, \mathbf{v}_g, \mathbf{b}_{t_s})$ *traversing the graph form the initial vertex* $\mathbf{v}_s$ *to* $\mathbf{v}_g$ *given the initial belief* $\mathbf{b}_{t_s}$ *of the environment. The optimal policy returns the optimal trajectory in the graph for a certain belief*

$$\pi_c^*(\mathbf{v}_i, \mathbf{v}_j, \mathbf{b}_{t_i}) \ \rightarrow\ \tilde{u}_c^*(\mathbf{v}_i, \mathbf{v}_j),\ \tilde{u}_c^* := \underset{\tilde{u} \in \mathbf{e}_{ij}}{\arg\min}\, \text{cost}(\tilde{u}, \mathbf{b}_{t_i}),\ \mathbf{e}_{ij} \in \mathcal{E}^*$$

subject to

$$\begin{aligned} &\forall \tilde{u} \in \mathcal{E} \wedge \tilde{\mathcal{U}} \\ &\mathbf{v}_g \in \mathcal{V},\ \mathbf{v}_g = \{\mathbf{x}_g, t_g\} \\ &\mathbf{v}_s \in \mathcal{V},\ \mathbf{v}_s = \{\mathbf{x}_0, t_0\} \\ &|\mathcal{E}^*| = N_{\mathcal{E}} \\ &|\mathcal{V}^*| = N_{\mathcal{V}}. \end{aligned}$$

The time points t_0 and t_g indicated the time when the robotic system is at the start state and the goal state, respectively. This motion planning problem requires to solve a combined optimization of the vertices $\mathcal{V}^*$ and edges $\mathcal{E}^*$ to obtain the optimal motion graph $\mathcal{G}^*$. It is obvious that this optimization problem is considerable more complex than the optimal control problem Prob. 2. In order to ensure the requirements for an on-line motion planner some simplifications and heuristics are introduced in the following.

The definition of the expected cost between two adjacent vertices allows one to define the optimal additional edge between two vertices:

Definition 16 (Optimal edge between two adjacent vertices $\mathbf{v}_i$, $\mathbf{v}_j$)**.** *The optimal additional trajectory* $\tilde{u}_c^+$ *for reducing the cost between the vertices* $\mathbf{v}_i$ *and* $\mathbf{v}_j$ *is defined as*

$$\tilde{u}_c^+(\mathbf{e}_{ij}, \mathbf{b}_{t_k}) := \underset{\tilde{u} \in \tilde{\mathcal{U}}(\mathbf{v}_i, \mathbf{v}_j)}{\arg\min}\ \text{cost}(\mathbf{e}_{ij} \cup \{\tilde{u}\}, \mathbf{b}_{t_k}). \tag{5.8}$$

This definition allows one to perform an incremental optimization of edges between two vertices and is the centerpiece of the following incremental optimization approach.

The optimal motion graph planning Prob. 3 requires to solve a coupled optimization of the set of vertices $\mathcal{V}^*$ and Edges $\mathcal{E}^*$. As already mentioned, the original problem is to complex for an on-line optimization. Hence, an algorithm is presented which incrementally optimizes a given solution to the motion planning problem by constructing a finite motion graph. The initial solution is denoted as $\tilde{u}_c^0$ with $\tilde{u}_c^0(t_0) = \mathbf{x}_0,\ \tilde{u}_c^0(t_g) = \mathbf{x}_g$. This novel motion planning algorithm allows improving any given solution to the motion planning problem by adding multi-edges allowing one to replan the route during execution. In Alg. 8 the overview of the algorithm is given. First, the given solution $\tilde{u}_c^0$ is transformed to a graph structure. Therefore, the start

and final state of the trajectory are stored as vertices and N_v additional states are sampled from the trajectory. The partial trajectories between the vertices are the edges of the graph. These edges are stored in the $\mathrm{edge\_list}$ and represent the candidates for the on-line optimization. In order to reduce the expected cost of the existing plan, additional trajectories are generated for these edges to obtain multi-edges according to Def. 16. Since the possible improvement gained by an additional edge is not known before the optimization, a heuristic is applied to determine the order of the optimization for the edges in the $\mathrm{edge\_list}$. In this implementation the edge with the highest cost is used as an heuristic. This step is repeated until the improvement is below the predefined threshold τ_c, then the edge is removed from the $\mathrm{edge\_list}$. The algorithm terminates when the $\mathrm{edge\_list}$ is empty. One could also specify a maximum number of iterations or abort the algorithm after a certain time duration resulting in an anytime algorithm.
The principal steps of this algorithm for three iterations are illustrated in Fig. 5.4 for one example

Algorithm 8: On-line Planning

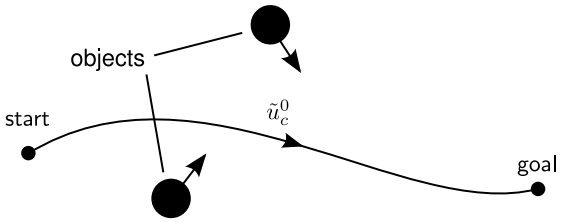
Input : $\tilde{u}_c^0$
Output : π_c^*

Initialize: Sampling N_v vertices
$\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N \mid \mathbf{v}_i \in \tilde{u}_c^0\}$,
Edges are the associated parts of $\tilde{u}_c^0$
$\mathrm{edge\_list} = \mathcal{E} = \{\mathbf{e}_{12}, \ldots, \mathbf{e}_{N-1N}\}$
while *edge_list* $\neq \emptyset$ **do**
  Get cost of all edges
  foreach $\mathbf{e}_{ij} \in$ *edge_list* **do**
    $\mathrm{cost}(\mathbf{e}_{ij}, \mathbf{b}_{t'})$
  Find edge with the highest cost
    $\mathbf{e}_{ij}^{\max} = \arg\max(\mathrm{cost}(\mathbf{e}_{ij}, \mathbf{b}_{t'}))$
  Generate optimal edge
    $\tilde{u}_c^+(\mathbf{e}_{ij}, \mathbf{b}_{t'})$ (5.8)
  Determine improvement
    $\Delta\,\mathrm{cost} = \mathrm{cost}(\mathbf{e}_{ij}, \mathbf{b}_{t'}) - \mathrm{cost}(\mathbf{e}_{ij} \cup \{\tilde{u}_c^+\}, \mathbf{b}_{t'})$
  if $\Delta\,\mathrm{cost} < \tau_c$ **then**
    $edge\_list = edge\_list \setminus \{\mathbf{e}_{ij}^{\max}\}$
  Append to edges
    $\mathbf{e}_{ij} = \mathbf{e}_{ij} \cup \{\tilde{u}_c^+\}$
Determine optimal policy π_c^* (adopted Alg. 1)
return π_c^*

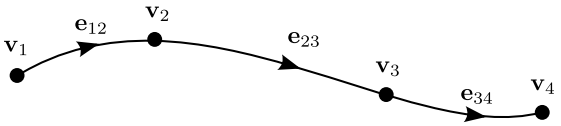
scenario. An example implementation for this novel algorithm is presented in Sec. 5.6.1.

5.4.2. Off-line Planning

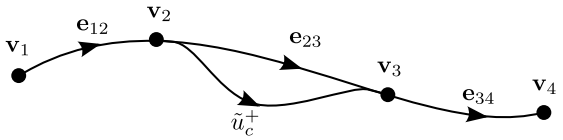
Instead of optimizing a given solution to the motion planning problem, the safety assessment concept from Chap. 2 is also integrated into a roadmap-based planner. The idea of roadmap-based planners [55] is to divide the original motion planning problem into two subproblems:



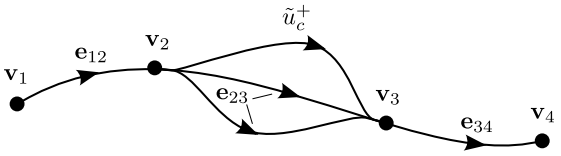
(a) Example scenario with two moving objects and initial solution $\tilde{u}_c^0$.



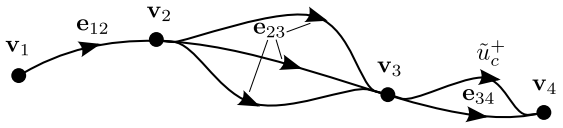
(b) Initialize step: sampled vertices with resulting edges.



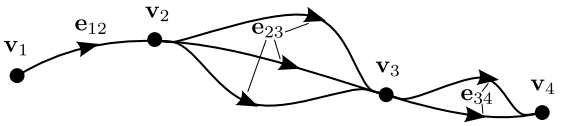
(c) Optimized edge $\tilde{u}_c^+$ added to edge $\mathbf{e}_{23}$ since it has the highest cost.



(d) Another edge is added since edge $\mathbf{e}_{23}$ has still the highest cost and improvement was bigger than τ_c.



(e) Optimized edge $\tilde{u}_c^+$ added to edge $\mathbf{e}_{34}$ since it has the highest cost.



(f) Resulting multigraph after three iterations.

Fig. 5.4.: Figures depicting single steps of Alg. 8 for an example scenario, until three optimization steps are performed.

construction of the roadmap and the query problem. The roadmap generated off-line provides a graph of vertices (sampled robot states) which are connected by edges (trajectories). In this phase the kinematic and dynamic constraints as well as static objects of the environment are usually considered. This means that the robot is able to follow any route in the roadmap without colliding with any static object. However, dynamic objects are not considered in the construction phase, this is done by the query phase. In the query phase, graph search algorithms are applied to find the optimal trajectory which minimizes a predefined cost function. For integrating the novel graph reduction algorithm from Sec. 5.4.1 can be used in the query phase to determine the optimal policy rather than the optimal route. Additionally, another possible integration for the construction phase is presented in the following.

The aim of this integration is to generate additional edges between vertices to reduce the collision probability regarding dynamic objects. In this case, the cost is not optimized since the trajectory cost is not so sensitive to multi-edges than the collision probability. For instance, the length of trajectories between two vertices which are relatively close together, may vary between $\pm 10\%$. The influence on the collision probability can be much more sensitive, in some cases an additional edge can reduce a collision probability of 20% close to zero, as shown in Sec. 2.9.2. In other words, the goal is to construct a roadmap which minimizes the expected collision probability while avoiding to generate a very large number of vertices. Since no information about dynamic objects is available during the construction phase, the trajectories cannot be optimize based on the motion prediction of dynamic objects. However, the trajectories are generated in such a way, that vertices are connected with multi-edges consisting of trajectories with a low similarity. The proposed similarity between trajectories is based on the intersection of the enlarged occupancy (see Sec. A.2) of the robot while executing the trajectories.

Definition 17 (Similarity of two trajectories)**.** *The similarity S between two trajectories $\tilde{u}_i$ and $\tilde{u}_j$ is defined as*

$$S(\tilde{u}_i, \tilde{u}_j) = \frac{\int |\Delta \mathcal{A}^b_{ij}(t)| \, \mathrm{d}t}{\int |\mathcal{A}^b(t)| \, \mathrm{d}t},$$

where $|\mathcal{A}^b(t)|$ is the area of the enlarged robot system and

$$|\Delta \mathcal{A}^b_{ij}(t)| = |\mathcal{A}^b(\tilde{u}_i(t)) \cap \mathcal{A}^b(\tilde{u}_j(t))|$$

is the common area of the two trajectories.

Proposition 8. *If no information is given about the future positions of the dynamic objects, it is optimal to generate trajectories with minimal similarity between two vertices in order to minimize the expected collision probability.*

Proof. Let $\tilde{u}_i$ be a given trajectory and $\tilde{u}^*_j$ the trajectory that minimizes the common collision probability

$$\tilde{u}^*_j = \arg\min_{\tilde{u}_j} P(C_{\tilde{u}_i} \cap C_{\tilde{u}_j}),$$

where $C_{\tilde{u}_i}$ and $C_{\tilde{u}_j}$ are the collision events, that the robot collides during the execution of the

trajectory $\tilde{u}_i$ and $\tilde{u}_j$, respectively. It can be derived

$$\begin{aligned}\min_{\tilde{u}_j} P(C_{\tilde{u}_i} \cap C_{\tilde{u}_j}) &= \min_{\tilde{u}_j} \int P(C_{\tilde{u}_i}(t) \cap C_{\tilde{u}_j}(t)) \,\mathrm{d}t \\ &= \min_{\tilde{u}_j} \int \int_{\Delta\mathcal{A}_{ij}^b(t)} f(\mathbf{p}, t) \,\mathrm{d}\mathbf{p} \,\mathrm{d}t.\end{aligned}$$

Since only $\Delta\mathcal{A}_{ij}^b$ is depending on $\tilde{u}_j$, this term can just be minimized by reducing the common occupancy area of the robot for both trajectories, thus

$$\begin{aligned}\tilde{u}_j^* &= \arg\min_{\tilde{u}_j} P(C_{\tilde{u}_i} \cap C_{\tilde{u}_j}) \\ \Leftrightarrow \tilde{u}_j^* &= \arg\min_{\tilde{u}_j} \int |\Delta\mathcal{A}_{ij}^b(t)| \,\mathrm{d}t \\ \Leftrightarrow \tilde{u}_j^* &= \arg\min_{\tilde{u}_j} S(\tilde{u}_i, \tilde{u}_j).\end{aligned}$$

Therefore, the common collision probability is minimized iff the similarity is minimized. □

This approach allows generating a roadmap which has a lower expected collision probability during the query phase. Multi-edges should be generated especially in regions which have a higher collision risk, such as narrow corridors or regions with a poor visibility. An example implementation for generating multi-edges with minimum similarity is presented in Sec. 5.6.2.

5.5. Interactive Motion Planning

In this section, a navigation approach is presented that is based on the idea of interactive safety assessment from Chap. 4. This approach is developed for addressing the freezing robot problem (FRP). It was shown in [117] that the FRP cannot be solved without considering the avoidance possibilities of the objects. In addition, [59] claims that a robot which is navigating to defensively will surely get stuck in dense pedestrian traffic. It is recalled from Chap. 4, that the idea of interactive safety is to find at least one safe trajectory for all objects in the workspace in order to guarantee that there exists a solution to prevent any collision. To transfer this idea to motion planning, the goal of each object needs to be taken into account.

Interactive Motion Planning *The goal is to find a collision-free trajectory for the robot which brings it closer to its goal while minimizing a cost function which depends on the future motion of all objects in its environment. Furthermore, the future motion of the reactive objects depends on their goal state and on the future motion of all other objects including the robot resulting in a mutual dependency.*

The interaction describes the mutual dependency of the objects on their motion to reach their individual goal. Therefore, it is assumed that reactive objects incorporate the future motion of surrounding objects into their own motion planning and expect similar anticipation of the other objects. One possibility to give a mathematical formulation for this problem is to use

game theory, more precisely discrete-time infinite dynamic games [11]. This kind of motion planning problem is clearly more complex than the common optimal control problem for one robot assuming ignoring objects in its workspace. Since in this case, $N_\mathrm{b} + 1$ coupled optimal control problems (one for each object including the robot) need to be solved. The coupled optimal control problem shows that it is not possible to decouple the problem of interactive motion planning into motion prediction of the objects and trajectory planning based on this prediction as done in most common motion planning approaches.

The purpose of this section is to present a motion planner which incorporates the interaction between the workspace objects. The resulting algorithm can be seen as a preliminary study to evaluate the potential of interactive motion planning compared to the common motion planning approach assuming ignoring objects in the workspace. Therefore, the workspace of one robotic system which is populated by reactive objects (see Sec. 1.2.2) is assumed. The proposed algorithm to address the problem of interactive motion planning is based on three assumptions:

1. Reactive objects avoid the robot if they are aware of it
2. Reactive objects move in order to avoid collisions with favored trajectories of the robot
3. Reactive objects move in order to come closer to their goal state

Since the cost function of the robot is not known to the other objects and it is not directly possible to measure if the other objects are aware of the robot, the future motion of the reactive objects are represented by PDFs. The PDF describing the future motion of the ith object is denoted as $f_i(\tilde{u}, \tilde{u}_\mathcal{A})$, where $\tilde{u}$ is a possible future trajectory of the object and $\tilde{u}_\mathcal{A}$ is the trajectory of the robot. The subscript $\mathcal{A}$ is added for a clear distinction between trajectories of the robot and other objects.

The main idea of the algorithm is to consider the interaction by defining two collision events as presented by [29]: $C_\mathcal{A}$, collision between any object and the robot and $C_\mathcal{B}$, collision between any objects, excluding the robot. Both events are the two essential elements for the cost function of the robot. The goal is to find the robot trajectory $\tilde{u}^*_{\mathcal{A},c}$ such that it minimizes the following cost function

$$\begin{aligned} \mathrm{cost}(\tilde{u}_\mathcal{A}) &= w_\alpha P(C_\mathcal{A}|\tilde{u}_\mathcal{A}) + w_\beta P(C_\mathcal{B}|\tilde{u}_\mathcal{A}) + w_\gamma (1 - \overline{L}(\tilde{u}_\mathcal{A})) \\ \tilde{u}^*_{\mathcal{A},c} &:= \underset{\tilde{u}_\mathcal{A} \in \tilde{\mathcal{U}}_\mathcal{A}}{\arg\min}\, \mathrm{cost}(\tilde{u}_\mathcal{A}), \end{aligned} \tag{5.9}$$

where $w_\alpha, w_\beta, w_\gamma$ are weight factors, $P(C_\mathcal{A}|\tilde{u}_\mathcal{A})$ is the collision probability that the robot will collide with at least one object, $P(C_\mathcal{B}|\tilde{u}_\mathcal{A})$ is the collision probability that at least two objects (excluding the robot) will collide and $\overline{L}(\tilde{u}_\mathcal{A})$ is the normalized trajectory cost of the robot also called goal function. This goal function

$$\overline{L}(\tilde{u}_\mathcal{A}) \in [0, 1] \tag{5.10}$$

represents the performance of the trajectory $\tilde{u}$ in order to achieve the navigation goal. It is normalized over all considered trajectories. In order to determine the two collision probabilities $P(C_\mathcal{A}|\tilde{u}_\mathcal{A})$ and $P(C_\mathcal{B}|\tilde{u}_\mathcal{A})$ the PDFs of the reactive objects need to be specified. Therefore, all possible trajectories of the objects are ranked by their goal functions. These goal functions represent their preferred motion assuming free-space, meaning that surrounding objects are not

considered. This behavior is equivalent to ignoring workspace objects. In contrast the distributions of the reactive objects are defined as

$$f_i(\tilde{u}, \tilde{u}_{\mathcal{A}}) = n_i e^{-L_i^a(\tilde{u}, \tilde{u}_{\mathcal{A}})}, \quad \text{with } n_i \text{ being a normalization constant}$$

and with

$$L_i^a(\tilde{u}, \tilde{u}_{\mathcal{A}}) = w^a(\tilde{u}_{\mathcal{A}}, \tilde{u}) L_i(\tilde{u}), \quad w^a \in [0, 1] \tag{5.11}$$

being the goal function for reactive objects. The weight w^a is introduced which models the avoidance behavior of reactive objects by adopting the results from the goal function $L_i(\cdot)$ assuming ignoring objects

$$w^a(\tilde{u}_{\mathcal{A}}, \tilde{u}_i) = 1 - (\underbrace{\overline{L}(\tilde{u}_{\mathcal{A}})}_{\substack{\text{predictability} \\ \in [0,1]}} \underbrace{D_i(\tilde{u}_{\mathcal{A}})}_{\substack{\text{detectability} \\ \in [0,1]}}). \tag{5.12}$$

It depends on the predictability of the robot trajectory (assumption 2) and its detectability (assumption 1). Predictable trajectories are those trajectories which are reasonable for the robot (e.g. trajectories that are smooth and lead to its goal) and thus are predictable for other objects. Detectable trajectories are those trajectories that allow other objects to observe the robot during the execution of its trajectory. Assumption 3 (goal directness) is modeled by the goal function of the object. This navigation approach is not directly implementable since it needs to consider an infinite set of trajectories for each object including the robot. In Sec. 5.6.3 an example implementation is presented.

5.6. Implementations

In this section, example implementations of the proposed approaches from the previous sections are presented. First, one possible implementation for the off-line and on-line integration of the motion planning approach from Sec. 5.4 are shown. Afterwards, an algorithm for interactive motion planning based on the idea of Sec. 5.5 is described.

5.6.1. On-line Motion Graph Planning

The algorithm from Sec. 5.4.1 allows one to improve *any* given solution to the motion planning problem by adding multi-edges allowing one to replan the route during execution. This even holds if the motion planning algorithm finds the optimal trajectory based on the information available at the initial state. This approach requires to solve the optimization problem from (5.8) which is not directly implementable due to the infinite number of possible PDFs representing the future position of the objects. Thus, as described in Sec. 2.8, a belief tree containing only a finite set of possible distributions is used to estimate $\hat{\text{cost}}(\cdot) \approx \text{cost}(\cdot)$. The optimization problem results in

$$\begin{aligned} \tilde{u}_c^+(\mathbf{e}_{ij}, \mathbf{b}_{t_k}^{t_i}) &:= \underset{\tilde{u} \in \tilde{\mathcal{U}}(\mathbf{v}_i, \mathbf{v}_j)}{\arg\min}\ \hat{\text{cost}}(\mathbf{e}_{ij} \cup \{\tilde{u}\}, \mathbf{b}_{t_k}^{t_i}), \text{ with} \\ \hat{\text{cost}}(\mathbf{e}_{ij}, \mathbf{b}_{t_k}^{t_i}) &:= \frac{1}{N_\text{s}} \sum_{s=1}^{N_\text{s}} \text{cost}\left(\tilde{u}_{c,s}^*(\mathbf{v}_i, \mathbf{v}_j, f_s^{t_i}), \mathbf{b}_{t_k}^{t_i}\right), f_s^{t_i} \in \mathbf{b}_{t_k}^{t_i} \end{aligned} \tag{5.13}$$

where N_s is the number of sampled distributions for one object representing its belief. Using the trajectory generation approach described in Sec. 2.8.4 the optimization problem becomes an one-dimensional nonlinear programming problem

$$\alpha^*(\mathbf{e}_{ij}, \mathbf{b}_{t_k}) := \underset{\alpha\in[-1,1]}{\arg\min}\, \hat{\mathrm{cost}}(\mathbf{e}_{ij} \cup \{\tilde{u}(\alpha)\}, \mathbf{b}_{t_k}).$$

However, this optimization problem is still challenging due to its nonlinear characteristic and the possible existence of local minima. That is why a generic random search algorithm [96] is applied with a fixed collection of candidates which are uniformly sampled from the complete interval of $\alpha \in [-1, 1]$. For optimizing each candidate, the discrete Newton algorithm is used. The result of the global optimization is the candidate with the smallest cost. This approach is evaluated by simulation results in Sec. 5.7.2.

5.6.2. Off-line Motion Graph Planning

For implementing the off-line integration presented in Sec. 5.4.2 one has to solve the optimization problem

$$\tilde{u}_j^* := \underset{\tilde{u}_j\in\tilde{\mathcal{U}}}{\arg\min}\, S(\tilde{u}_i, \tilde{u}_j).$$

Using the trajectory generation approach described in Sec. 2.8.4 it results in a nonlinear programming problem

$$\alpha^* := \underset{\alpha\in[-1,1]}{\arg\min}\, S(\tilde{u}_i, \tilde{u}_j(\alpha)).$$

Since this problem also suffers from possible local minima, the same numerical optimization approach described above in Sec. 5.6.1 is used. This numeric optimization allows generating roadmaps with multi-edges resulting in a lower expected collision probability regarding the entire roadmap as stated in Prop. 8. In Sec. 5.7.2 this proposition is verified by simulations using the described optimization in order to minimize the similarity of multi-edges.

5.6.3. Interactive Motion Planning

In this section, an example implementation for the interactive motion planning approach from Sec. 5.5 is shown. This approach is not directly implementable since it requires to consider the infinite trajectory space of all objects and the robot in order to determine the trajectory which minimizes (5.9). Hence, a finite set of robot candidate trajectories is generated and Monte Carlo simulation is used to estimate the collision probabilities for each candidate. The sampled trajectories are generated as described in Sec. A.3. Thus, all trajectories are now described by a vector of control inputs $\tilde{u} \rightarrow \mathbf{u}$. The implementation for reactive objects is descried first followed by the overall algorithm.

The probability distribution $f(\cdot)$ is calculated based on a goal function $L(\cdot)$ which models the fact that objects have preferred trajectories. As presented by [29], the preference depends on the goal of the objects, the preferred velocity and smooth trajectories (avoiding high accelerations and fast changes in the motion direction). Hence a goal function is introduced which evaluates each sample according to this behavior. Since the goal function of [29] was designed for traffic scenarios, the path deviation is replaced by a goal directness and the lateral and longitudinal

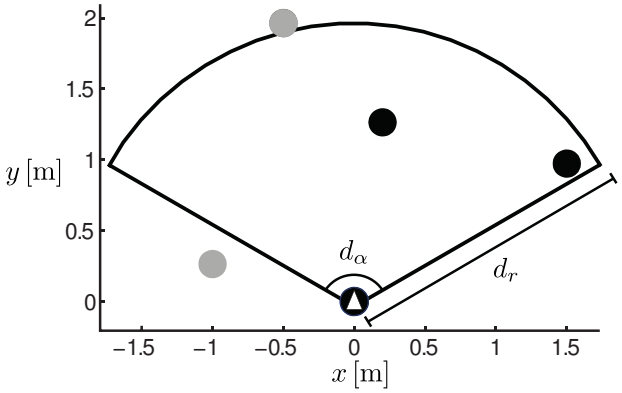


Fig. 5.5.: The arc depicts the field of view (FOV) model of the considered object. The black objects are inside the FOV and the gray ones are outside.

accelerations are replaced by the absolute acceleration. Each trajectory of the robot or the objects is ranked according to goal directness, smoothness or collision risk by

$$\begin{aligned} L(\mathbf{u}_i) = w_{\text{dist}} \underbrace{\text{dist}(\mathbf{x}_i^{\text{g}}, \mathbf{x}_i(T))}_{\text{goal directness}} + \\ \int [w_{\text{vel}} \underbrace{(v(t) - v_d)^2}_{\text{favors desired velocity } v_d} + \\ w_{\text{acc}} \underbrace{a(t)^2}_{\text{favors smooth trajectories}}] \, \text{d}t, \end{aligned} \tag{5.14}$$

where $\text{dist}(\cdot)$ is a distance function between the final state of the object trajectory $\mathbf{x}_i(T)$ and its goal state $\mathbf{x}_i^{\text{g}}$. The distance between two states $\mathbf{x}_1$ and $\mathbf{x}_2$

$$\text{dist}(\mathbf{x}_1, \mathbf{x}_2) = w_e \|\mathbf{p}_1 - \mathbf{p}_2\| + w_v |\mathbf{v}_1 - \mathbf{v}_2|$$

is calculated by the Euclidean distance for position and velocity difference, with the corresponding weights w_e and w_v. This goal function is used to determine the density function $f(\cdot)$

$$f(\mathbf{u}_i) = n e^{-L(\mathbf{u}_i)},$$

where n is a normalizing constant. The goal function (5.14) is suited for ignoring objects since it is independent of the motion of other objects in the workspace. In the following the calculation of the weight w^a for transforming the prediction of ignoring objects to reactive objects is shown. In order to integrate the avoidance behavior of workspace objects, an additional weight w^a is used to decrease the goal function value. Therefore, the detectability and the predictability of the robot trajectory need to be measured. In order to get a measure for the detectability, a deterministic field of view (FOV) model is used which is described by an angular and linear constraint. The FOV model has a certain opening angle d_α and a limited range d_r which results in an arc. In Fig. 5.5 an example FOV model is depicted. An object is detected if its center point is closer than d_r and if the relative angle between the center points of the objects is smaller than d_α. The

detectability D_i that the ith object detects the robot during its trajectory $\mathbf{u}_\mathcal{A}$ is defined as

$$\mathrm{D}_i(\mathbf{u}_\mathcal{A}) = \frac{1}{N_\mathrm{d}} \sum_{n=1}^{N_\mathrm{d}} \mathrm{Ind}(D|\mathbf{u}_\mathcal{A}, \mathbf{x}_i^n), \tag{5.15}$$

with $\mathbf{x}_i^n$ is the nth time-equidistantly sampled state along the trajectory of the ith object and the indicator function is defined as

$$\mathrm{Ind}(D|\mathbf{u}_\mathcal{A}, \mathbf{x}_i^n) = \begin{cases} 1, & \text{robot detected} \\ 0, & \text{robot not detected.} \end{cases}$$

Loosely speaking, for every sampled state $\mathbf{x}_i^n \in \mathbf{u}_i$, the FOV model is evaluated and the number of successful detections is normalized by the number of tests N_d. The detectability is calculated for each trajectory of the ith object. In order to determine the cost for one robot trajectory according to (5.9), the two collision probabilities $P(C_\mathcal{A}|\mathbf{u}_\mathcal{A})$ and $P(C_\mathcal{B}|\mathbf{u}_\mathcal{A})$ need to be calculated.

For the calculation of $P(C_\mathcal{A}|\mathbf{u}_\mathcal{A})$ the collision probability between the robot trajectory $\mathbf{u}_\mathcal{A}$ and the ith object

$$P_i(C_\mathcal{A}|\mathbf{u}_\mathcal{A}) = \int\limits_{\mathcal{U}_i^{C_\mathcal{A}}} \mathrm{Ind}(C_\mathcal{A}|\mathbf{u}_\mathcal{A}, \mathbf{u}_i) f(\mathbf{u}_i)\, \mathrm{d}\mathbf{u}_i \tag{5.16}$$

must be calculated, where $\mathrm{Ind}(\cdot)$ is an indicator function which is one if the trajectory of the robot $\mathbf{u}_\mathcal{A}$ collides with the trajectory of the object $\mathbf{u}_i$ and zero otherwise. The set $\mathcal{U}_i^{C_\mathcal{A}}$ contains all control inputs of the ith object which does not collide with another object excluding the robot. The collision probability considering all objects

$$P(C_\mathcal{A}|\mathbf{u}_\mathcal{A}) = 1 - \underbrace{\prod_{i=1}^{N_b} (1 - P_i(C_\mathcal{A}|\mathbf{u}_\mathcal{A}))}_{\text{probability of no direct collision}} .$$

is defined by the probability that no object collides with the robot.

The calculation of $P(C_\mathcal{B}|\mathbf{u}_\mathcal{A})$ is similar to the calculation of $P(C_\mathcal{A}|\mathbf{u}_\mathcal{A})$. Only the subset $\mathcal{U}^{C_\mathcal{B}}$ of the trajectories of the objects which do not collide with the robot are investigated.

$$\mathcal{U}_i^{C_\mathcal{B}} \subset \mathcal{U}_i, \quad \forall \mathbf{u}_i \in \mathcal{U}_i^{C_\mathcal{B}}\ \mathrm{Ind}(C_\mathcal{A}|\mathbf{u}_\mathcal{A}, \mathbf{u}_i) = 0$$

The probability of collision excluding the robot is determined by considering all trajectories

$$\boldsymbol{\mathcal{U}}^{C_\mathcal{B}} = \mathcal{U}_1^{C_\mathcal{B}} \times \mathcal{U}_2^{C_\mathcal{B}} \times \cdots \times \mathcal{U}_{N_\mathrm{b}}^{C_\mathcal{B}}$$

which do not collide with the robot

$$P(C_\mathcal{B}|\mathbf{u}_\mathcal{A}) = \int\limits_{\boldsymbol{\mathcal{U}}^{C_\mathcal{B}}} \mathrm{Ind}(C_\mathcal{B}|\mathbf{u}_\mathcal{B}) f(\mathbf{u}_\mathcal{B})\, \mathrm{d}\mathbf{u}_\mathcal{B},\ \mathbf{u}_\mathcal{B} = [\mathbf{u}_1, \ldots, \mathbf{u}_{N_\mathrm{b}}]. \tag{5.17}$$

Where $\mathrm{Ind}(\cdot)$ is an indicator function which is 1 if a collision between any trajectory of the

objects occurs and 0 if no collision occurs. Both integrals (5.16) and (5.17) cannot be calculated directly due to the infinite number of possible trajectories, hence, Monte Carlo simulation is used to estimate the collision probabilities. This is done as explained in Sec. A.3. Therefore, a finite set of $N_{\tilde{\mathrm{u}}}^{\mathcal{B}}$ trajectories $\tilde{\mathcal{U}}_{\mathcal{B}_i}^{C}$ is generated for each object and each trajectory is weighted as described above. So far, the cost for any robot trajectory can be estimated considering the behavior of reactive objects in the workspace. In order to determine the optimal robot trajectory, the infinite set of robot trajectories is approximated by a finite set $\tilde{\mathcal{U}}_{\mathcal{A}}^{C}$ containing $N_{\tilde{\mathrm{u}}}^{\mathcal{A}}$ trajectory candidates. The trajectories are generated with the same method as the object trajectories for the Monte Carlo simulation. The finite set of trajectories is also used for determining the normalized goal function $\overline{L}(\cdot) \in [0,1]$ which is used to measure the predictability of the robot trajectory. The overview of the resulting algorithm is shown in Alg. 9. This implementation is evaluated

Algorithm 9: Interactive Motion Planning

Input : $\mathbf{x}_{\mathcal{A}}(0), f_1(\mathbf{x}, t_0), \ldots, f_{N_{\mathrm{b}}}(\mathbf{x}, t_0)$
Output : $\mathbf{u}_{\mathcal{A},c}^{*}$

Initialize: Generate robot trajectories (see Sec. A.3)
$\tilde{\mathcal{U}}_{\mathcal{A}}^{C} = \{\mathbf{u}_{\mathcal{A},1}, \ldots, \mathbf{u}_{\mathcal{A},N_{\tilde{\mathrm{u}}}^{\mathcal{A}}}\}$
Determine normalized predictability of each $\mathbf{u}_{\mathcal{A}}$
$\overline{L}(\mathbf{u}_{\mathcal{A}})$ according to (5.10) and (5.14)
Generate object trajectories for each object (see Sec. A.3)
$\tilde{\mathcal{U}}_{\mathcal{B}_i}^{C} = \{\mathbf{u}_{i,1}, \ldots, \mathbf{u}_{i,N_{\tilde{\mathrm{u}}}^{\mathcal{B}}}\}$
Determine weights assuming ignoring objects
$L(\mathbf{u}_i)$ according to (5.14)

foreach $\tilde{u}_{\mathcal{A}} \in \tilde{\mathcal{U}}_{\mathcal{A}}^{C}$ **do**
  Determine detectability of each object trajectory
  $\mathrm{D}_i(\mathbf{u}_{\mathcal{A}})$ according to (5.15)
  Determine weights for each object trajectory
  w^a according to (5.12)
  Determine cost for robot trajectory
  $\mathrm{cost}(\mathbf{u}_{\mathcal{A}})$ according to (5.9)

Determine optimal trajectory
$\mathbf{u}_{\mathcal{A},c}^{*} = \arg\min_{\mathbf{u}_{\mathcal{A}}} \mathrm{cost}(\mathbf{u}_{\mathcal{A}})$

return $\mathbf{u}_{\mathcal{A},c}^{*}$

by simulation scenarios in Sec. 5.7.3.

5.7. Simulations

In the following sections various simulation scenarios are presented to evaluate the proposed algorithms from this chapter by their implementations described in Sec. 5.6. First, simulation scenarios for motion planning in dynamic uncertain environments are presented to evaluate the usefulness of motion graphs that allow the robot to replan its route during execution. Therefore, the off-line (Sec. 5.4.2) and the on-line approach (Sec. 5.4.1) are evaluated separately. After-

wards, simulation scenarios which could lead to a FRP are generated to evaluate the idea of interactive motion planning from Sec. 5.5.

5.7.1. Optimal Control

In this section, simulations are provided to evaluate the extended motion planning approach from Sec. 5.3. First, the individual steps of the complete approach through the use of an example scenario representing a common highway situation is presented. The ego vehicle (black) is driving in the middle lane and generates trajectories for all three lanes. In order to avoid illustration difficulties, only trajectories for the current velocity are drawn. The snapshot of the scenario as well as the generated trajectories and the results of the safety assessment are illustrated in Fig. 5.6. The motion of the other vehicles are predicted assuming constant velocity in their current lane. All trajectories leading to collision during or beyond the planning horizon are correctly

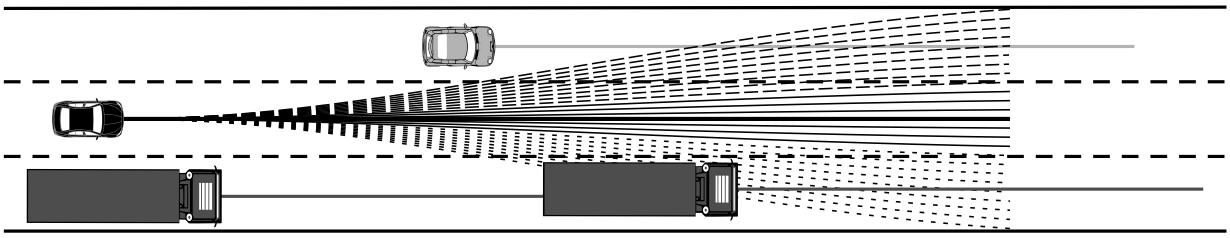


Fig. 5.6.: The resulting fan-shaped trajectory set of the ego vehicle is shown. The thick line depicts the best collision-free trajectory and the solid lines are collision-free during and beyond the planning horizon. Trajectories leading to collision during the planning horizon are illustrated by short dashed lines, long dashed lines represent trajectories leading to an cICS. The alleged best trajectory is shown by the thick line.

identified: the ego vehicle will collide with the trucks on the left during the planning horizon. Furthermore, it will collide with the vehicle on its left lane beyond the planning horizon according to Def. 10 (cICS), since the velocity of the ego vehicle is higher than the velocity of the purple one. Ignoring constraints by traffic regulations and aiming to keep the current velocity, the trajectory with the lowest cost is the constant velocity trajectory.

In order to evaluate the computational performance of the presented algorithm, a 6:23 min simulation highway drive is used. The current C++ implementation was tested on a Intel core I5-2500 using only a single core. The average response time was 0.010 s with an average of 469.26 generated trajectories for the ego vehicle and 9.25 surrounding vehicles. At the worst instant of our scenario, the algorithm had to check 677 trajectories with 17 vehicles for collision resulting in a worst case response time of 0.121 s. The performance of the presented algorithm has been exhaustively tested for hours of collision-free driving under various conditions by simulation scenarios.

5.7.2. Motion Graph Planning

In this section simulation results of the motion planning algorithms presented in Sec. 5.4.1 and Sec. 5.4.2 are demonstrated.

Tab. 5.1.: Evaluation of Alg. 8

Scenario	α**-Parameters**		$\text{cost}(\mathbf{e}_{24}, f)$		
	Rand α_1,α_2	Opt α_2	Rand	Opt	Δ
1	$0.20, -0.99$	-0.81	1.10	1.02	7.5%
2	$-0.93, \quad 0.80$	0.34	2.08	1.79	13.8%
3	$-0.75, \quad 0.65$	0.39	0.84	0.73	12.7%
4	$0.69, \quad 0.38$	-0.46	1.37	0.75	45.4%
5	$-0.91, \quad 0.27$	0.28	1.11	1.10	1.3%

Optimization for On-line Planning

For evaluating the algorithm from Sec. 5.6.1 the same simulation environment from Sec. 2.9.2 is used. Instead of three multi-edges between vertex $\mathbf{v}_2$ and $\mathbf{v}_4$ only one vertex is randomly generated and the second one is either randomly generated or optimized according to (5.13). For the evaluation, multiple scenarios are generated with different edges. The one given edge $\mathbf{e}_{24}$ is randomly generated with $\alpha \in [-1, 1]$. The same interval for α is used for the second random edge and the optimized edge. For evaluating the expected collision probability a belief tree with $N_\text{s} = 100$ is generated to represent the future position of the object. The cost function for optimizing the edge is

$$\text{cost}(\tilde{u}) = h(P(C|\tilde{u}), L(\tilde{u})) = \frac{1}{1 - P(C|\tilde{u})} l(\tilde{u}),$$

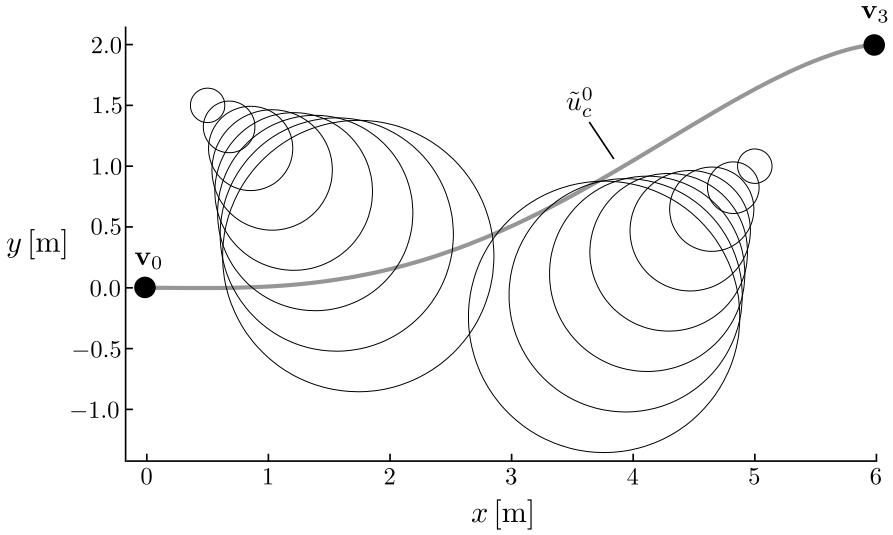
with $L(\tilde{u}) = l(\tilde{u})$, representing the length of the trajectory. This cost functions has the following properties

$$\text{cost}(\tilde{u}) = \begin{cases} \infty, & P(C|\tilde{u}) = 1, \\ l(\tilde{u}), & P(C|\tilde{u}) = 0, \\ l(\tilde{u}) < \text{cost} < \infty, & \text{else.} \end{cases}$$

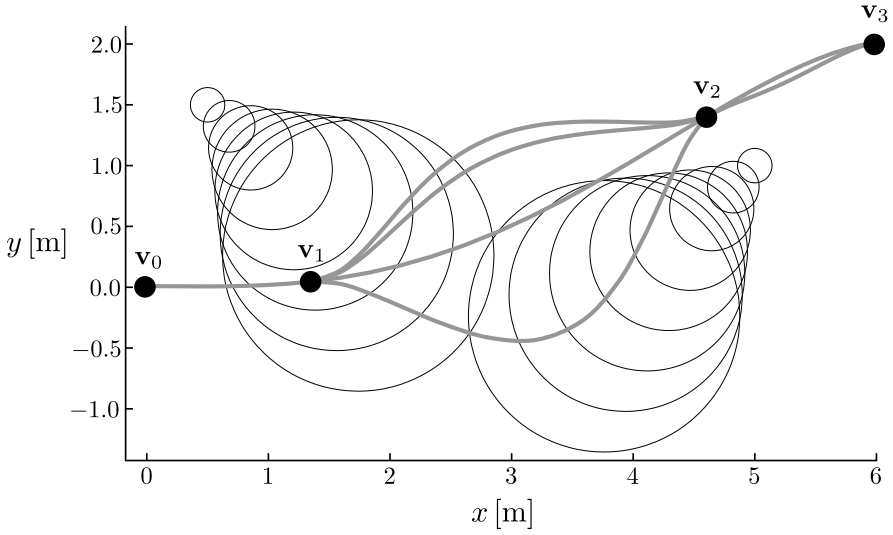
The maximum relative improvement of the collision probability of the optimized edge is 45.4% compared to the randomly generated edge. The minimum improvement is 1.3% and the mean is 16.14%. The individual results are shown in Tab. 5.1. This result shows that additional edges which are optimized according to the distribution of the object can clearly reduce the expected collision probability.

On-line Planning

For the evaluation of Alg. 8 a simulation scenario with two moving objects is used. The scenario is depicted in Fig. 5.7. The initial mean values and the diagonal covariance matrix representing



(a) The initial motion plan is shown in gray. The solid black circles show the 2-σ ellipsoids of the objects at every 5th time step.



(b) Vertex $\mathbf{v}_1$ and $\mathbf{v}_2$ are extracted from the initial motion plan. Between $\mathbf{v}_1$ and $\mathbf{v}_2$ three additional edges were generated and one between $\mathbf{v}_2$ and $\mathbf{v}_3$.

Fig. 5.7.: Simulation scenario for evaluating Alg. 8 containing two ignoring objects.

the initial states of the objects are

$$\begin{aligned}\boldsymbol{\mu}_1(t_0) &= [5.0\,\mathrm{m}, 1.0\,\mathrm{m}, -0.35\,\mathrm{m/s}, -0.35\,\mathrm{m/s}]\\ \boldsymbol{\mu}_2(t_0) &= [1.5\,\mathrm{m}, 0.5\,\mathrm{m}, \quad 0.35\,\mathrm{m/s}, -0.35\,\mathrm{m/s}]\\ \boldsymbol{\Sigma}(t_0) &= \mathrm{diag}[0.01\,\mathrm{m}^2, 0.01\,\mathrm{m}^2, 0.05\,\mathrm{m^2/s^2}, 0.05\,\mathrm{m^2/s^2}].\end{aligned}$$

The initial trajectory is a solution for the motion planning problem for navigating between the vertices

$$\begin{aligned}\mathbf{v}_0 &= [0\,\mathrm{m}, 0\,\mathrm{m}, 0\,\mathrm{m/s}, 0\,\mathrm{m/s}, 0\,\mathrm{s}]\\ \mathbf{v}_3 &= [6\,\mathrm{m}, 2\,\mathrm{m}, 0\,\mathrm{m/s}, 0\,\mathrm{m/s}, 4\,\mathrm{s}].\end{aligned}$$

The initial trajectory $\tilde{u}_c^0$ is calculated by the trajectory optimization described in Sec. 5.6.1 which is also used for optimizing the additional multi-edges. The cost for the motion plan is 15.94.

As described in Alg. 8, first, two vertices are extracted from the motion plan by sampling time-equidistant states from the initial trajectory.

$$\begin{aligned}\mathbf{v}_1 &= [1.35\,\mathrm{m}, 0.04\,\mathrm{m}, 2.07\,\mathrm{m/s}, 0.27\,\mathrm{m/s}, 1.3\,\mathrm{s}]\\ \mathbf{v}_2 &= [4.58\,\mathrm{m}, 1.40\,\mathrm{m}, 1.95\,\mathrm{m/s}, 1.14\,\mathrm{m/s}, 2.6\,\mathrm{s}].\end{aligned}$$

The number of samples for each distribution in the sampling tree is set to $N_s(k) = 20$. The threshold for the necessary improvement of each additional edge is set to $\tau_c = 0.5$. After determining the costs between all adjacent vertices, the edge $\mathbf{e}_{12}$ is identified as the one causing the highest cost. Thus, an additional edge between vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ is calculated. Since the improvement was bigger than the threshold, the next two edges were also generated between vertex v_1 and v_2. Afterwards, one edge is generated between $\mathbf{v}_2$ and $\mathbf{v}_3$ causing an improvement below the threshold, since the collision probability has nearly no effect on the cost of this edge. The final cost of the optimized graph is 7.79 which is an improvement of 51.13% comparing to the cost of the initial plan.

For reasons of completeness, the cost of the optimal route in this graph is 15.11 meaning an improvement of 5.2% compared to the initial solution. This improvement results from the sampled vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ allowing a different shape of the trajectory between $\mathbf{v}_0$ and $\mathbf{v}_3$ than the initial solution.

Off-line Planning

The goal is to evaluate the correlation of the collision probability of multi-edges with its similarity as stated in Prop. 8. Therefore, graphs are generated according to Sec. 2.8.4 which consist of the two vertices

$$\begin{aligned}\mathbf{v}_1 &= [0\,\mathrm{m}, 2\,\mathrm{m}, 0\,\mathrm{m/s}, 0\,\mathrm{s}]\\ \mathbf{v}_2 &= [4\,\mathrm{m}, 2\,\mathrm{m}, 2\,\mathrm{m/s}, 3\,\mathrm{s}]\end{aligned}$$

and one edge generated with $\alpha = 1$. A total of 10 different second edges are generated with $\alpha \in \{-1.0, -0.8, \ldots, 1.0\}$ resulting in a similarity interval of $\mathcal{S}(\tilde{u}_1, \tilde{u}_2) \in [0.33, 1.0]$. The graph

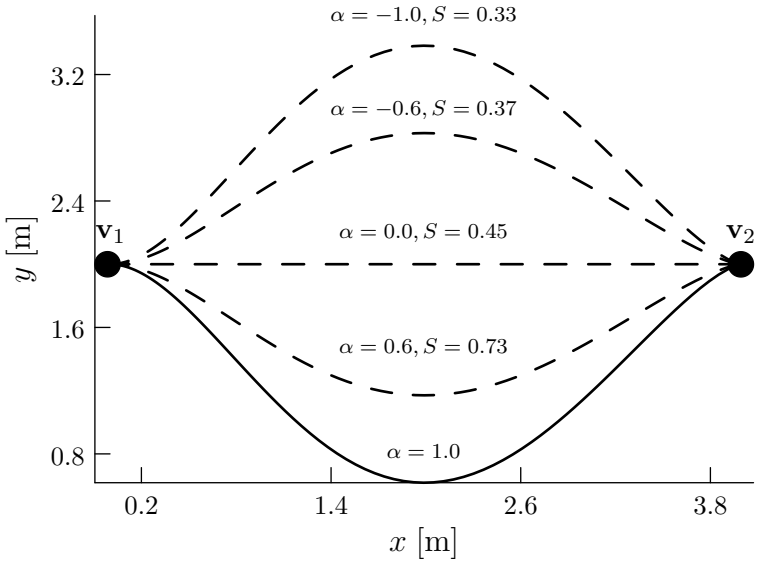


Fig. 5.8.: The solid line illustrates the given edge and the dashed lines depict the investigated second edges with their associated similarity regarding the first edge.

with some example second edges is shown in Fig. 5.8. In order to obtain results that do not represent only one scenario, randomized positions of the object in the workspace are generated. For the evaluation, 20 random beliefs $\mathbf{b}_{t_0}^{t_1}(f^{t_0})$ are generated and for each $\mathbf{b}$ 100 random distributions $f^{t_1} \sim \mathbf{b}_{t_0}^{t_1}$ are sampled. Therefore, 20 random distribution $f^{t_0}(t_1) = \mathcal{N}(\boldsymbol{\mu}_{t_1}, \boldsymbol{\Sigma}_{t_1})$ representing the position of the object in the workspace at time t_1 are generated based on the information available at time t_1 with

$$\boldsymbol{\Sigma}(t_1) = \begin{bmatrix} 0.4305 & 0 & 0.1450 & 0 \\ 0 & 0.4305 & 0 & 0.1450 \\ 0.1450 & 0 & 0.0500 & 0 \\ 0 & 0.1450 & 0 & 0.0500 \end{bmatrix}$$

and $\boldsymbol{\mu}_{t_1}$ is uniformly distributed in $\mathcal{W} : x \in [0, 5],\ y \in [0, 4]$. The belief $\mathbf{b}_{t_0}^{t_1}(f^{t_0}(t_1))$ is represented by 100 samples. At time t_1 each sample $f_s^{t_1} = \mathcal{N}(\boldsymbol{\mu}^+, \boldsymbol{\Sigma}(t_0)),\ f_s^{t_1} \in \mathbf{b}_{t_0}^{t_1}$ has the initial covariance

$$\boldsymbol{\Sigma}(t_0) = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0.05 \end{bmatrix}$$

and their mean value is distributed as $\boldsymbol{\mu}^+ \sim \mathcal{N}(\boldsymbol{\mu}(t_1), \boldsymbol{\Sigma}(t_1) - \boldsymbol{\Sigma}(t_0))$ according to Sec. 2.8.2. The collision probability $P(C|\mathbf{e}_{12}, \mathbf{b}),\ \mathbf{e}_{12} = \{\tilde{u}_1, \tilde{u}_2\}$ for all 10 $\tilde{u}_2$ are estimated by Alg. 1. For each $\tilde{u}_2$ the mean collision probability is calculated and illustrated with respect to their similarity measure $S(\tilde{u}_1, \tilde{u}_2)$ in Fig. 5.9. As expected, the results show: the greater the similarity, the lower is the collision probability for traversing between two vertices. This result is in accordance with Prop. 8.

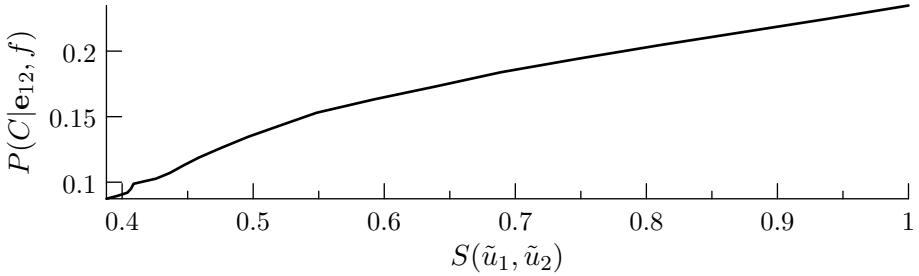


Fig. 5.9.: Estimated collision probability subject to the similarity of the edges.

Tab. 5.2.: Goal and cost function parameters. Parameters for goal function from [18].

goal function $L(\cdot)$			cost function $C(\cdot)$		
w_{dist}	w_{vel}	w_{acc}	w_α	w_β	w_γ
0.0085	$0.05/T_h/(1+v_d^2)$	$0.05/T_h/a_{\text{max}}^2$	0.5	0.5	0.4

5.7.3. Interactive Navigation

In order to demonstrate that the proposed navigation algorithm from Sec. 5.5 is useful for populated environments, an example scenario containing 4 reactive objects is used. The scenario is illustrated in Fig. 5.10. The scenario has been chosen to contain a few typical problems regarding motion planning in populated environments. Object 2 and 3 need to cross their path to reach their goals. Furthermore, object 1 prefers to take a trajectory between both objects to reach its goal. The highest risk in this scenario originates from the objects $1, 2$ and 3. Object 4 represents the case of free-space motion since its goal and motion should not be influenced by the other objects in the environment. The motion planning Alg. 9 is applied to each object and the results are depicted in Fig. 5.10. The used parameters for the trajectory generation are listed in Tab. 3.6 and the trajectories are generated for a fixed time duration of $T_h = 2\,\text{s}$. The weights for the collision probabilities are higher compared to the weight of the goal function since the focus is set on safe trajectories.

In summary, the four trajectories minimizing the cost function of all four objects are collision free during this time interval. Each of the four trajectories has a collision probability of $P(C_\mathcal{A}) < 5\%$. Object 4 chooses a trajectory which is close to a straight constant velocity trajectory which is expected due to the goal function and the fact that no interaction between other objects occurs. The trajectories of object $1, 2$ and 3 show a different behavior. As can be seen, object 3 and 4 avoids object 1 by delaying their change of direction towards their goal. Thereby, object 1 gains enough space to choose a trajectory between both objects. It is noted, that these four trajectories are independently generated for each object, although the result seems to be generated by a centralized approach at a first glace. Due to the consideration of the interaction between the objects by both collision probabilities $P(C_\mathcal{A})$, $P(C_\mathcal{B})$ and the heuristics predictability and detectability this result is achieved.

Determining the preferred trajectories of all objects only according to their goal function $L(\cdot)$, leads to a collision between the objects 2 and 3.

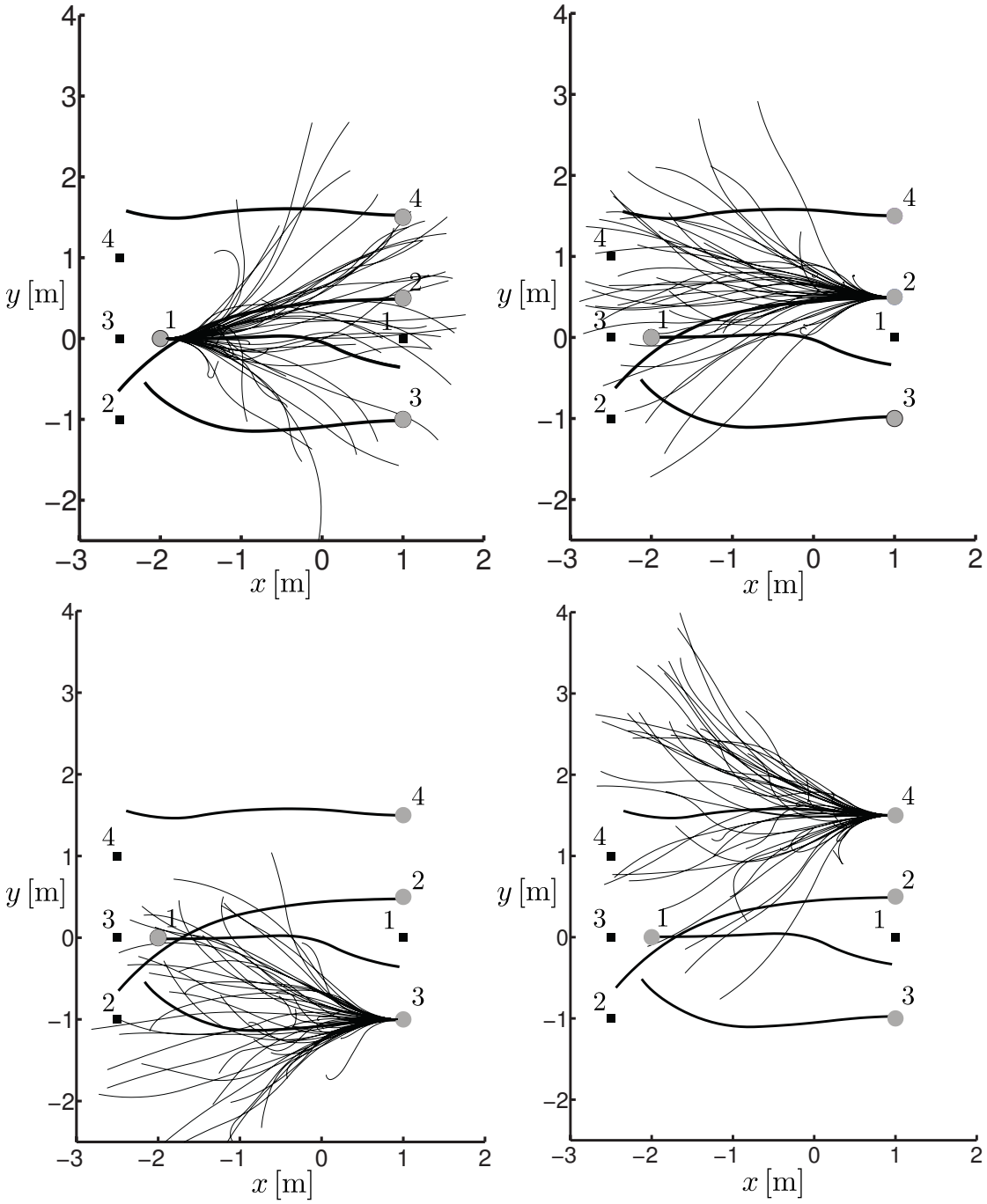


Fig. 5.10.: The results for each workspace object with the proposed navigation Alg. 9 are illustrated. The tiny lines depict all trajectory candidates and the thick line is the most promising one. The squares illustrate the goal position for each object.

Tab. 5.3.: Simulation Results

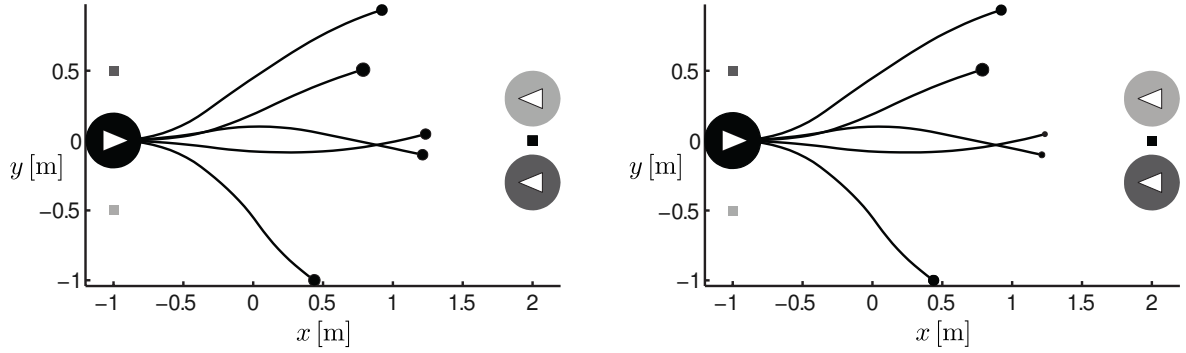
N_s	$N_{\tilde{u}^C}$	mean $\left\lvert\frac{P(C_\mathcal{B}\lvert\tilde{u}^C)-P(C_\mathcal{B})}{P(C_\mathcal{B})}\right\rvert$	max $\left\lvert\frac{P(C_\mathcal{B}\lvert\tilde{u}^C)-P(C_\mathcal{B})}{P(C_\mathcal{B})}\right\rvert$
1000	171	10.5%	39.3%

Collision Excluding the Robot

In order to show the necessity of considering the collision probability between workspace objects excluding the robot $P(C_\mathcal{B}|\mathbf{u})$, random scenarios are generated. Therefore, the same setup as in Sec. 3.10.4 is used. However, the workspace objects are only generated for the 10th x-region and 100 scenarios including 10 different robot trajectories are investigated. Then, the difference of $P(C_\mathcal{B}|\tilde{u})$ and $P(C_\mathcal{B})$ is determined for the 1000 robot trajectories $\tilde{u}$. $P(C_\mathcal{B})$ is the collision probability that a collision between any object excluding the robot occurs, when the robot is ignored. In order to estimate the necessity for this calculation, the number of robot trajectories leading to a higher collision probability between the workspace objects are determined. These trajectories are denoted as $\tilde{u}^C$. Furthermore, the difference of both probabilities are obtained by calculating the mean relative difference of $P(C_\mathcal{B}|\tilde{u}^C)$ and $P(C_\mathcal{B})$ and the maximum relative difference for all scenarios. The simulation results are shown in Tab. 5.3. It can be seen that there is a significant difference in 17% of all cases with a mean value of 10.5% and a maximum observed difference of 39.3%. This shows, that the robot needs to take into account the collision probability of all objects in the workspace, otherwise the avoidance behavior of reactive objects may lead to a collision between objects excluding the robot. However, this behavior contradicts the definition of reactive objects from 1.2.2.

Collision Avoidance Behavior

In Sec. 5.5 a goal function is presented which models the collision avoidance behavior of reactive objects. The purpose of this goal function is to relax the FRP as described by [117]. In Fig. 5.11 an example scenario is evaluated by the goal function for ignoring and reactive objects. The sampled trajectories of the robot and the objects are generated as described in Sec. 5.5 and allow the robot to navigate to its goal location and to avoid the other objects. The result assuming ignoring objects shows that there exists no clear preference for the robot. The trajectories leading to its goal location have a slightly higher collision probability, but the trajectories avoiding the other objects are not leading to its goal. Depending on the maximum accepted collision probability, the robot can be in a FRP since no acceptable trajectory exists. The assessment of the trajectories is different if the avoidance behavior of the reactive objects is taken into account. Since the objects are facing towards the robot, the detectability (5.15) of the trajectories of the objects is very high. As a result, the collision probability of the two trajectories of the robot leading through the reactive objects is clearly reduced (25%) and thereby relaxing the FRP for this scenario. The collision probability of the other trajectories of the robot is nearly not changing, since the corresponding predictability is very low.



(a) Collision probability $P(C_{\mathcal{A}}|\tilde{u})$ without collision avoidance model.

(b) Collision probability $P(C_{\mathcal{A}}|\tilde{u})$ with collision avoidance model.

Fig. 5.11.: Diameter of circles depict the probability of collision for each trajectory candidate and the squares illustrate the goal locations for each object.

5.8. Discussion

This chapter presented the integration of safety assessment concepts from previous chapters into motion planning algorithms: integration of safety assessment beyond the planning horizon into optimal control; incremental optimization for improving trajectories by motion graphs; approach for interactive motion planning.

Optimal Control The optimal control based motion planner [125] was extended to address former limitations regarding safety guarantees. By integrating the cICS checker from Chap. 4, this approach fulfills all three safety criteria introduced in Sec. 1.2.1. The presented safety assessment guarantees motion safety during and beyond the planning horizon. This novel framework for autonomous driving is especially suitable for high-speed navigation on freeways. The definition of cICS and its integration into optimal control is applicable to various motion planning problems such as motion planning for helicopters and unmanned aerial vehicles.

In order to make this motion planning framework able to handle many different traffic situations, it must be capable to handle stochastic motion prediction of all traffic participants. The prediction of other traffic participants in real-world experiments is a critical point since it embodies the basis of the motion planning approach. Furthermore, the current implementation of the cICS checker evaluates each trajectory independently. It should be possible, however, to use the result from one cICS evaluation to assess other states at the same time. For instance, if one trajectory leads to an cICS due to the resulting high velocity, the same holds for trajectories with higher velocities on the same lane. The computing time of the algorithm could be reduced by making use of these properties.

Motion Graph Planning The approach for motion graph planning extends the idea of closed-loop safety assessment from Chap. 2 to motion planning. Instead of addressing the original problem of generating the optimal motion graph with a finite set of vertices and edges, the presented approach is incrementally improving a given solution to the motion planning problem. In a first step, an appropriate common motion planner is used to generate a trajectory which solves the motion planning problem. In a second step, additional edges and vertices are gener-

ated to reduce the expected navigation cost of the initial solution. This is done by incrementally optimizing additional edges between vertices with high cost. Although this approach is not generating the optimal motion graph, this approach allows one to incrementally improve any given solution to the motion planning problem, even if the given solution is the optimal trajectory. Due to its incremental characteristic, the presented algorithm can be interrupted at any time. This kind of algorithm is called an anytime algorithm. An example implementation is used to evaluate this approach by simulation scenarios and the simulation results show a clear improvement already after few iterations compared to the the initial solution.

Drawbacks of this approach are, that it is based on heuristics and on the initial solution to the motion planning problem. Thus, it is not likely that this approach will find the optimal motion graph for a specified number of vertices and edges. Furthermore, the gained improvement by the motion graph algorithm compared to the initial solution cannot be predicted, however, the improvement can be guaranteed.

Interactive Motion Planning Furthermore, the problem of motion planning in dynamic uncertain environments populated by reactive objects – interactive motion planning – was discussed in this chapter. It was shown that it is not possible to decouple the problem of interactive motion planning into motion prediction of the objects and trajectory planning based on this prediction as done in most common motion planning approaches. A novel motion planner was presented incorporating the interaction between the workspace objects and the robot. Therefore, assumptions and heuristics are introduced which model the avoidance behavior of reactive objects regarding the future motion of the robot. The resulting algorithm can be seen as a preliminary study to evaluate the potential of interactive motion planning compared to the common motion planning approach. An example implementation has shown by simulation studies that interactive motion planning is a promising research direction to address the freezing robot problem.

However, due to the use of heuristics the approach does not explicitly consider all necessary constrains such as collision-free motion of all objects. Hence, a more detailed evaluation especially by real-world scenarios is necessary.

6. Conclusions

Summary This chapter provides a summary and discussion of the presented approaches and methods of each chapter from this thesis. In addition, possible directions of future research work are outlined.

6.1. Summary

Nowadays, robots are used in industrial settings to improve productivity and to perform tasks that are potentially harmful to humans. Their benefits compare to human workers are their high precision, non-stop operation, and high moving speed. More recently, attention has turned to the idea of the robotic co-worker e.g. [47]. In environments such as homes and offices the robots should aid and support humans. A typical field of application arises from the aging population in developed worlds entailing a strong demand on aging care. Robots should support and reveal nurses in the aging care sector to prevent rising health care costs.

Safe and reliable motion planning capabilities in the presence of humans are one of the key capabilities a robotic system must have to meet the goal of the robotic co-worker. Safe hardware design, safe motion planning and safe motion execution must be investigated to ensure a safe robotic system. In this thesis, the problem of safe motion planning, especially the involved safety assessment, of robotic system is addressed. The following paragraphs briefly describe the main contributions of this thesis.

Closed-loop Assessment In Chap. 2 a novel safety assessment approach is presented that estimates the collision probability of a motion graph also called roadmap. Instead of defining the safety of a roadmap by the route with the minimum collision probability, the optimal policy is determined allowing the robot to navigate through the roadmap with the smallest expected collision probability. The optimal policy considers the possibility of the robot to replan its route during execution depending on new received information about the environment. That means, if new received measurements allow a more precise prediction of the environment, the robot is able to adopt its route in order to minimize its collision probability regarding the new information. Since the future measurements are not known at the time of the safety assessment, a finite set of possible future measurements are simulated so the optimal policy and the associated expected collision probability can be estimated. Therefore, a graph reduction algorithm is presented that allows assessing the safety of any motion graph. Furthermore, it was shown that the collision probability of the entire graph is always smaller than the collision probability of any route in the graph. This safety assessment algorithm can be applied to any roadmap based planner for uncertain environments.

Two different implementations, one for mobile robot applications and one for automotive applications, are presented. Both implementations are verified by various simulation scenarios

and showed that the expected collision probability could be clearly reduced. In summary, the consideration of replanning possibilities leads to a more reliable safety assessment in uncertain environments.

Assessment Beyond Planning Horizon In Chap. 3 the problem of assessing the safety of partial trajectories beyond their planning horizon is addressed. Therefore, the concept of ICS is recalled from literature and the novel generalization PCS is introduced which is applicable to uncertain environments. The PCS value represents the probability of the robot being in an ICS regarding the uncertain information about the environment. In order to retrieve a safety criterion that considers the collision probability during the planning horizon of a trajectory and their probability ending in an ICS, the overall collision probability was introduced. This allows for the first time to extend the idea of partial motion planning to stochastic modeled environments. Additionally, the probabilistic collision cost is introduced as an alternative safety criterion taking into account the severity of the collision. Simulation results showed, that this criterion is especially useful for crowded environments assuming that objects can bear up against weak collisions. Furthermore, the problem of determining unions of ICS sets is addressed by presenting a novel iterative algorithm that allows a more efficient calculation of ICS than previous ICS checkers. This novel algorithm leads to the definition of the robot maneuverability which represents the number of future motion possibilities preventing the robot to end in an ICS. It was shown that a higher maneuverability minimizes the probability of collision with unforeseen changes or objects in the environment.

Interactive Assessment In Chap. 4 the problem of interactive safety assessment was defined and addressed. Therefore, extensions of the ICS and PCS concepts were introduced considering the avoidance behavior of reactive objects in the environment. For deterministic environments the cICS definition is introduced which aims to find at least one trajectory for each object preventing it from ending in an ICS. Two concepts for uncertain environments are presented that differ in the available information to the robot and the other objects about the environment. These novel definitions allow to assess the safety beyond the planning horizon while taking into account the reciprocal avoidance behavior of reactive objects. Thereby, the FRP is addressed since the consideration of reactive behavior gives each object more space for navigation resulting in lower collision probabilities. This is also verified by several simulation results of the presented implementations for mobile robot and automotive applications.

Integration into Motion Planning In Chap. 5 some of the safety assessment approaches from the previous chapters are integrated into or extended to motion planning algorithms. First of all, the integration of the cICS concept into optimal control based motion planners is presented allowing to guarantee safety for an infinite time horizon. An implementation for autonomous driving on highways is used to demonstrate the necessity of assessing the safety beyond the planning horizon. Based on the idea of closed-loop assessment, the idea of motion graph planning is presented. A given solution (trajectory) for the motion planning problem is improved by generating additional trajectories resulting in a roadmap. It was shown that any given solution can be optimized by a roadmap allowing the robot to replan its route. Finally, based on the concept of interactive assessment an interactive motion planning algorithm is presented. It considers the reciprocal collision avoidance of reactive objects by reasoning about the safety of all objects.

6.2. Discussion and Future Directions

The concepts presented in this thesis give rise to various topics for future research. The following paragraphs describe selected research projects:

Semi-autonomous control Semi-autonomous control of vehicles or robotic systems can be seen as an intermediate step towards fully autonomous systems. The interplay of the human operator and the control approach is one of the critical research questions. One possibility to address this problem is to apply invariance control [126], a control approach for constrained nonlinear systems. Invariance control is already applied to various problems such as trajectory supervision and haptic rendering [102]. The proposed control concept consists of two separate controllers: the nominal and the invariance controller. The purpose of the nominal controller is to achieve the main control objectives without considering the constraints, while the role of the invariance controller is to ensure stability and feasibility of the complete system. In the case of semi-autonomous vehicles, the human driver takes the role of the nominal controller and the invariance controller ensures that the driver is not entering a state which will lead to a collision. A reliable safety assessment approach is used to identify the admissible space for the considered vehicle. Therefore, other dynamic objects, safety beyond the planning horizon, and uncertain changes in the environment need to be taken into account. Thus, semi-autonomous control is a well-suited field of application for the safety assessment concepts from this thesis.

Game-theory based Interactive Motion Planning Results from Chap. 5 on interactive motion planning have shown, that the consideration of the reactive behavior of objects has a considerable effect on the quality of the navigation result. In order to extend and continue this work, the problem of motion planning in the presence of reactive objects (e.g. humans or human-driven cars) can be formulated using differential game theory [11]. In this sense, interaction describes the mutual avoidance behavior between humans resulting from the assumption that humans incorporate the future motion of the surrounding persons into their own motion planning and expect similar anticipation from them. Game theory can be used to model different agents with own interests and the corresponding multi-decision making process. It assumes that all agents aim to minimize their cost by anticipating actions of other agents. The solutions may consider the individual goals of the agents, their kinematic and dynamic constraints, and their limited perceptual capabilities. Important issues in this context are: type of game, single shot or multi-shot games; convergence criteria of the algorithm; approximation techniques for on-line capable algorithms. In order to include uncertainties in the current state of the objects or in their payoff, the problem can be extended to stochastic games [89].

Combination of Reachability Analysis and Inevitable Collision Obstacles Another important field of safety assessment is reachability analysis e.g. [64] that is mainly used for automatic verification tools. Reachability analysis is also used to verify the safety of various applications such as autonomous driving [4]. As stated in [2], reachability analysis determines the set of all states that a system can reach given a set of control inputs and the set of initial states of the system. The purpose of the ICO concept described in Chap. 3 is to determine all states that will inevitably lead to a collision with a certain object. The reachability analysis

and the calculation of ICO sets are similar problems. Existing algorithms for calculating or approximating the ICO sets [130] can only be used for a limited class of robotic systems and polygon-shaped objects. Hence, the algorithms for calculating the ICO sets could be improved by reachability analysis tools in terms of computational efficiency and applicability.

A. Estimation of Collision Probability

Summary Uncertain environments pose a particular challenge for the safety assessment of robot trajectories. Due to the uncertainty in the future position of other objects, it is not possible to perform a binary assessment: safe or unsafe trajectory. Hence, the expectation of a possible collision is calculated as a safety criterion. This is expressed as the collision probability of a certain trajectory. The aim of this chapter is to present two approaches for estimating the collision probability of a robot trajectory that are used in this thesis to assess the safety of trajectories in uncertain environments.

The outline of this chapter is as follows: Sec. A.1 gives the problem formulation and two different methods for estimating the collision probability for an entire trajectory are presented. In Sec. A.2 the collision probability is estimated by discretizing the workspace using a probabilistic occupancy grid. Sec. A.3 presents a method based on Monte Carlo simulation. Finally, a discussion of this chapter is given in Sec. A.4.

A.1. Problem Formulation

This chapter addresses the problem of assessing the safety of a given robot trajectory in uncertain environments. The uncertainty occurs in the state of all workspace objects and in their future states. Whereby, in real-world scenarios, the uncertainty associated with the robot states is considerably smaller than the uncertainty in the states of the other objects. This is in particular true for dynamic objects, since the quality of the motion prediction depends on the reliability of the stochastic motion model, whereby its uncertainty usually grows over time.

It is assumed, that the robot and all objects in the workspace are rigid bodies. The objects or the robot may have arbitrary shape that changes over time, and their uncertainty can be represented by an arbitrary PDF.

Collision Probability *Given the future motion of the robot system and the stochastic motion prediction of all workspace objects, compute the probability that the robot will collide with at least one object during its intended trajectory.*

In the following, some notion is introduced and two possible approaches are presented to estimate the collision probability of a robot trajectory.

A.1.1. Notation

The workspace of the robot system $\mathcal{A}$ is denoted by $\mathcal{W}$ and the subset of the workspace occupied by $\mathcal{A}$ in state $\mathbf{x}(t)$ is expressed as $\mathcal{A}(\mathbf{x}(t)) \subset \mathcal{W}$. The state $\mathbf{x} = [\mathbf{p}, \boldsymbol{v}]$ is represented by its position $\mathbf{p}$ and its velocity $\boldsymbol{v}$. The state $\mathbf{x}(t)$ and input $u(t)$ of the considered robot system $\mathcal{A}$ (for a

point in time t) can take values from the state space $\mathcal{X}$ and the control space $\mathcal{U}$. An input trajectory of the robot system is denoted by $\tilde{u}$ which maps the time t to the input space: $[0, \infty) \to \mathcal{U}$. The set of input trajectories is denoted by $\tilde{\mathcal{U}}$. The workspace occupancy generated from the input trajectory at time t is denoted by $\mathcal{A}(\tilde{u}(t))$ and the occupancy of the robot at position $\mathbf{p}(t)$ is denoted as $\mathcal{A}(\mathbf{p}(t))$. The occupancy of the ith object at time t in the workspace is denoted by $\mathcal{B}_i(t)$. The unified occupancy of all objects is written in short notation as $\mathcal{B} = \bigcup_{i=1,\dots,N_\mathrm{b}} \mathcal{B}_i$, where N_b is the number of workspace objects. Due to the lack of a perfect model of the environment, the states of the objects are represented by probabilistic density functions (PDF). The PDF describing the state of the ith object $\mathcal{B}_i$ at time t is denoted by $f_i^{t'}(\mathbf{x}, t),\ t \geq t'$ and their position uncertainty is expressed as $f_i^{t'}(\mathbf{p}, t)$. The superscript t' indicates the time point of the information which is used for the prediction at time t. This time point is called the observation time, since it represents the time of the information at which the object was last detected or observed. Since one can only formulate a probability distribution for a random vector and not an occupancy set, f_i represents the probability distribution of a reference point $\mathbf{c}$ of $\mathcal{B}_i$. The PDF describing the state of the robot is denoted by $f_\mathcal{A}(\mathbf{x}, t)$.

A.1.2. Collision Probability for a Single Time Point

First, the calculation of the collision probability for one point in time regarding one object is shown. It is based on the definition of [115] which can be seen as an extension of the definition of [29]. Toit and Burdick [115] extended the former definition to allow joint distributions of the robot and object, describing interaction between them. The collision probability that the robot is in collision with the ith object at time t is expressed as

$$P_i(C|t) = \int_\mathcal{W} \int_\mathcal{W} \mathrm{Ind}(\mathbf{p}_\mathcal{A}(t), \mathbf{p}_i(t)) f_\mathcal{A}(\mathbf{p}_\mathcal{A}, t) f_i(\mathbf{p}_i, t)\, \mathrm{d}\mathbf{p}_\mathcal{A}\, \mathrm{d}\mathbf{p}_i \tag{A.1}$$

with

$$\mathrm{Ind}(\mathbf{p}_\mathcal{A}, \mathbf{p}_i) = \begin{cases} 1, & \text{if } \mathcal{A}(\mathbf{p}_\mathcal{A}) \cap \mathcal{B}_i(\mathbf{p}_i) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \tag{A.2}$$

It is noted, that instead of the position, the state of the objects needs to be considered if their shape changes over time. In this work, it is assumed that the shape of all objects is constant. In real-world applications, the uncertainty of the future states of the robot is negligible compared to the much higher uncertainty of the future states of the objects. Hence, it is assumed that the future trajectory of the robot is without any uncertainty and that the workspace occupied by the robot at a certain time point is deterministic. This assumption allows to reformulate (A.1) in two possible ways. However, even for the simplified equations, no closed-form expression is known for arbitrary PDFs. The first one

$$P_i(C|\mathbf{p}_\mathcal{A}, t) = \int_\mathcal{W} \mathrm{Ind}(\mathbf{p}_\mathcal{A}, \mathbf{p}_i) f_i(\mathbf{p}, t)\, \mathrm{d}\mathbf{p} \tag{A.3}$$

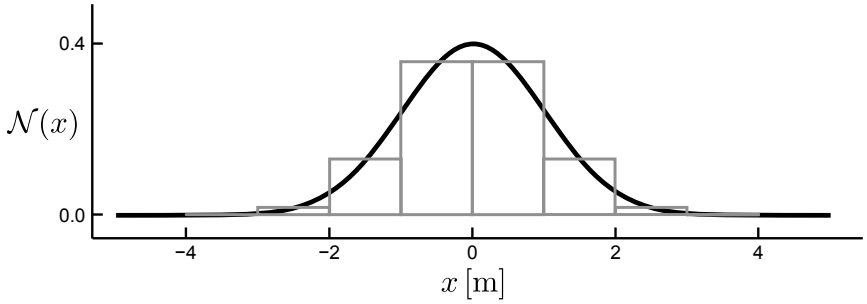


Fig. A.1.: Discretized Gaussian PDF for the bounded support $[-4, +4]$.

will be solved by Monte Carlo simulation in Sec. A.3 The second possibility formulates the problem by enlarging the robot system $\mathcal{A} \rightarrow \mathcal{A}^{\mathrm{b}_i}$.

$$P_i(C|\mathbf{p}_\mathcal{A}) = \int\limits_{\mathcal{A}^{\mathrm{b}_i}(\mathbf{p}_A)} f_i(\mathbf{p}, t)\,\mathrm{d}\mathbf{p}. \tag{A.4}$$

The enlargement is performed by Minkowski addition, so that the indicator function can be omitted, this is explained in Sec. A.2. In the following, both approaches are presented to estimate the collision probability for a complete trajectory.

A.2. Workspace Discretization

Occupancy grid maps are a common approach for generating maps with noisy and uncertain sensor measurements [112]. The map discretizes the robot workspace in evenly spaced cells and its value represents the probability that this cell is occupied by an obstacle. The idea of discretization can also by used to represent arbitrary PDFs in the workspace of the robot as a set of cells with an associated probability of occupancy. For instance, this approach is used in [35] and [4] to estimate the collision probability for autonomous vehicles.

In order to estimate the collision probability according to (A.4), the integral is approximated by the sum of a finite set. This is achieved by discretizing the workspace into evenly spaced cells assuming uniform distribution in the cells and by enlargement of the robot system allowing to model other objects as point masses. If the PDF has an infinite support, it is transformed to a truncated PDF having bounded support to ensure a finite set representing the PDF. The approximation of a Gaussian function is sketched in Fig. A.1. The approximation for a multidimensional PDF is analogous. The enlargement is performed by Minkowski addition of the area $(-\mathcal{B}_i + \mathbf{c}_i)$ to $\mathcal{A}$, so that the new occupancy of the robot is $\mathcal{A}^{\mathrm{b}_i} = \mathcal{A} \oplus (-\mathcal{B}_i + \mathbf{c}_i)$, that is explained in more detail in [81]. The Minkowski addition of the occupancy sets is visualized in Fig. A.2 for a two-dimensional workspace with position coordinates x and y.

The probability that a certain region R of the workspace is occupied by an object $\mathcal{B}_i$ is obtained by integration:

$$P(R \cap \mathcal{B}_i \neq \emptyset, t) = \int\limits_{R \oplus (-\mathcal{B}_i + \mathbf{c}_i)} f_i^{t'}(\mathbf{p}, t)\,\mathrm{d}\mathbf{p}.$$

The probability that the robot system $\mathcal{A}$, applying the input trajectory $\tilde{u}$, has a collision C with

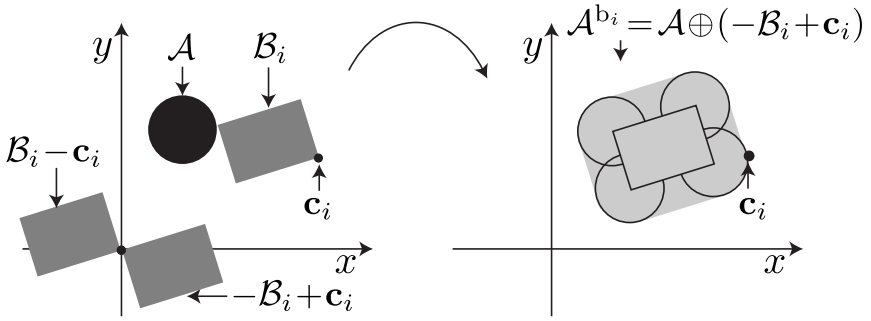


Fig. A.2.: Minkowski addition of the workspace occupancy of the robot and another object.

another probabilistic object $\mathcal{B}_i$ at time t arises from

$$P_i(C|\tilde{u}(t), t') = P(\mathcal{A}(\tilde{u}(t)) \cap \mathcal{B}_i \neq \emptyset | t') = \int\limits_{\mathcal{A}^{\mathrm{b}_i}(\tilde{u}(t))} f_i^{t'}(\mathbf{p}, t)\, \mathrm{d}\mathbf{p},$$

where the index i of the collision probability refers to the ith workspace object, $\mathbf{p}$ is a position in the workspace, and t' is the observation time. Additionally, the probability that a collision occurs within a time interval $[t_k, t_{k+1})$ is considered, where $t_k = k\,T$, $k \in \mathbb{N}^+$ is a time step and $T \in \mathbb{R}^+$ is the step size. The set occupied in the workspace for a time interval is denoted by $\mathcal{A}\big(\tilde{u}([t_k, t_{k+1}))\big)$ such that the collision for a time interval is obtained by

$$P_i(C|\tilde{u}([t_k, t_{k+1})), t') = \int\limits_{\mathcal{A}^{\mathrm{b}_i}(\tilde{u}([t_k, t_{k+1})))} f_i^{t'}\big(\mathbf{p}, [t_k, t_{k+1})\big)\, \mathrm{d}\mathbf{p}. \tag{A.5}$$

Due to the discretization of the workspace, this equation can be rewritten as

$$P_i(C|\tilde{u}([t_k, t_{k+1})), t') = \sum_{\text{cell} \in \mathcal{C}^{\mathcal{A}}([t_k, t_{k+1}))} P(\mathbf{p} \in \text{cell}), \tag{A.6}$$

where $\mathcal{C}^{\mathcal{A}}([t_k, t_{k+1}))$ is the set of cells which are occupied by the enlarged robot system during the time interval $[t_k, t_{k+1})$ and cell is one cell of the occupancy grid. The probability that the position $\mathbf{p}$ of the object is inside a cell is expressed as

$$P(\mathbf{p} \in \text{cell}_j) = a f_i^{t'}(\gamma_j, t), \quad a \in \mathbb{R}^+,$$

where γ_j is the center of the cell cell_j and a is the area of the cell. Loosely speaking, the probability of a collision is finally obtained by summing up the probabilistic occupancies $P(\mathbf{p} \in \text{cell}_j)$ for all cells which are occupied by the robot. An example of the deterministic occupancy of the robot and the probabilistic occupancy of another workspace object is visualized in Fig. A.3. The collision probability during a time interval considering all N_b objects $\mathcal{B}_i$ is derived as

$$P(C|\tilde{u}([t_k, t_{k+1})), t') = 1 - \prod_{i=1}^{N_\mathrm{b}} \big(1 - P_i(C|\tilde{u}([t_k, t_{k+1})), t')\big),$$

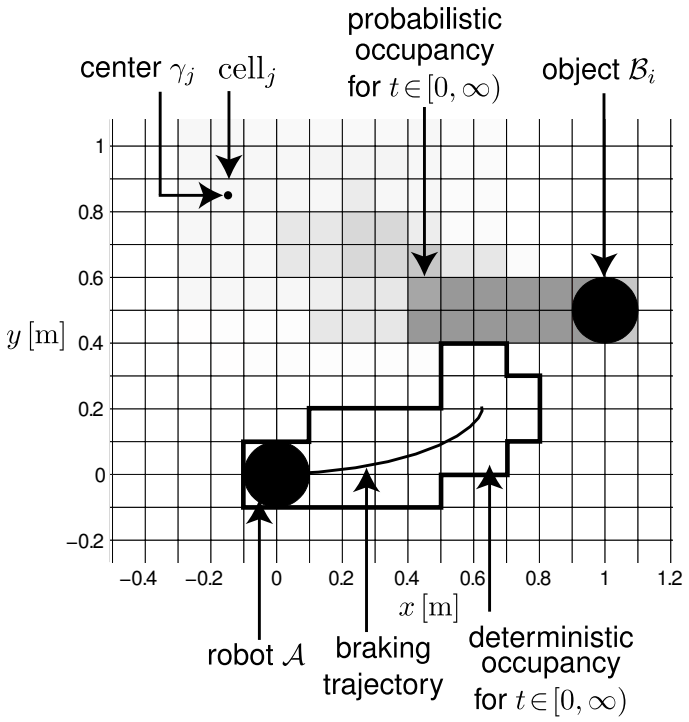


Fig. A.3.: Numerical integration of the collision probability by workspace discretization.

assuming that the collision probability of the time intervals are independent. Finally, the collision probability for the complete trajectory results in

$$P(C|\tilde{u}, t') = 1 - \prod_{k=0}^{N_\mathrm{k}-1} \big(1 - P(C|\tilde{u}([t_k, t_{k+1})), t')\big), \tag{A.7}$$

where N_k is the number of all time intervals for $\tilde{u}$.

A.3. Monte Carlo Approximation

In this section, a Monte Carlo based algorithm is presented for estimating collision probabilities. More precisely, finite-horizon simulation of dynamic objects is used as described in [100]. It is inspired by the work of [18, 25, 29] which use Monte Carlo simulation for threat analysis of road scenes. In [71] Monte Carlo simulation is used to estimate the collision probabilities between vehicles which future motions are modeled as Gaussian processes.

Compared to the occupancy grid approach, this method does not discretize the workspace of the robot. Furthermore, this approach does not need a stochastic motion model providing $f^{t'}(\mathbf{x}, t)$, instead it uses a so-called goal function that weights deterministic motions of the objects. The basic principle is to simulate a set of possible future trajectories of all workspace objects approximating their reachability set for the considered time horizon. These trajectories have to fulfill the kinematic and dynamic constraints of the object. In this work, the complete input space $\mathcal{U}$ of an object is used to generate the trajectories. In order to retrieve the probabilistic density function of their future motion, the trajectories are weighted by their goal function, representing the motion model of the objects. The trajectories together with their corresponding

weights represent the approximated density function of the future motion.

Trajectory Model The trajectory space $\tilde{\mathcal{U}}$ of an object contains an infinite number of input trajectories. In order to generate random trajectories from $\tilde{\mathcal{U}}_i$ for the ith object, N_c different control inputs are sampled from the input space of the object resulting in the the random vector $\mathbf{u}_i$

$$\mathbf{u}_i = [u^1, \ldots, u^{N_\mathrm{c}}], \quad u^i \in \mathcal{U}^i.$$

The resulting space of all random input vectors is denoted as

$$\hat{\tilde{\mathcal{U}}}_i = \mathcal{U}_i^1 \times \mathcal{U}_i^2 \times, \ldots, \times \mathcal{U}_i^{N_\mathrm{c}},$$

where $\times$ is the Cartesian product and the superscript denotes the time points at which the control inputs are sampled. Each control input is valid for a fixed time duration and the time interval I_t described the time span of the resulting trajectory. The time interval is denoted by $I_t = [0, T_h]$ and T_h is the finite time horizon of the simulation. For discretization, the sampling time T_c is used and during the time interval $I_c = [kT_c, (k+1)T_c]$ the control input is constant. Using the motion model $\dot{\mathbf{x}} = m(\mathbf{x}(t), u(t))$ of the object, the object states are obtained given the initial state $\mathbf{x}_i^0 = \mathbf{x}_i(0)$ for the ith object. The simulation time T_s ($T_s < T_c < T_h$) is defined for discretizing the object states along the resulting trajectory

$$\mathbf{s}_i(\mathbf{x}_i^0, \mathbf{u}_i) = [\mathbf{x}^1, \ldots, \mathbf{x}^{N_\mathrm{d}}], \quad \mathbf{x} \in \mathcal{X},$$

where $\mathbf{s}_i(\mathbf{u}_i)$ contains all N_d sampled states of the resulting object trajectory. In the following, the collision probability is defined and then approximated by Monte Carlo simulation. Therefore, random samples of $\mathbf{u}_i$ and $\mathbf{s}_i$ are generated, whereby the nth sample of the ith object is described by $\mathbf{u}_{i,n}$ and $\mathbf{s}_{i,n}$.

Estimation of Collision Probability For estimating the collision probability for the complete trajectory and not only one time point, (A.3) is extended by exchanging the state of the object and the robot by their complete trajectories. The collision probability is defined as

$$P_i(C|\tilde{u}) = \int_{\hat{\tilde{\mathcal{U}}}_i} \mathrm{Ind}(C|\tilde{u}, \mathbf{u}_i) f(\mathbf{u}_i)\, d\mathbf{u}_i, \tag{A.8}$$

where $f(\mathbf{u}_i)$ is the density function of the control inputs and $\mathrm{Ind}(C|\tilde{u}, \mathbf{u}_i)$ is an indicator function

$$\mathrm{Ind}(C|\tilde{u}, \mathbf{u}_i) = \begin{cases} 1, & \text{collision occurs} \\ 0, & \text{collision free.} \end{cases}$$

It is one, if a collision between the robot trajectory $\tilde{u}$ and the ith object trajectory $\mathbf{u}_i$ occurs and is zero otherwise. The density function $f(\mathbf{u}_i)$ represents the preferred control inputs vectors of the ith object.

Sampling Strategies The integral of (A.8) can be approximated by Monte Carlo sampling. An unbiased estimator for the collision probability is the *sample mean*

$$\hat{P}_i(C|\tilde{u}) = \frac{1}{N_\mathrm{s}} \sum_{n=1}^{N_\mathrm{s}} \mathrm{Ind}(C|\tilde{u}, \mathbf{u}_{i,n}),\ \mathbf{u}_{i,n} \sim f(\mathbf{u}_i),$$

where $\mathbf{u}_{i,n}$ is randomly sampled from $f(\mathbf{u}_i)$ as proposed by [100] and N_s is the number of samples. A goal function $L(\cdot)$ is used to rank each sample according to, e.g. smoothness or goal directness. The goal function is used to determine the density function

$$f(\mathbf{u}_i) = ne^{-L(\mathbf{u}_i)},$$

where n is a normalizing constant. Since it is not always possible to directly generate samples according to $f(\cdot)$, they are generated with a different distribution $p(\cdot)$. This method is called *importance sampling* [48]. This allows to write (A.8) as

$$P_i(C|\tilde{u}) = \int_{\hat{\tilde{\mathcal{U}}}_i} \mathrm{Ind}(C|\tilde{u}, \mathbf{u}_i) \frac{f(\mathbf{u}_i)}{p(\mathbf{u}_i)} p(\mathbf{u}_i)\, d\mathbf{u}_i.$$

This leads to the so called importance sampling estimator

$$\hat{P}_i(C|\tilde{u}) = \frac{1}{N_\mathrm{s}} \sum_{n=1}^{N_\mathrm{s}} \mathrm{Ind}(C|\tilde{u}, \mathbf{u}_{i,n}) \frac{f(\mathbf{u}_{i,n})}{p(\mathbf{u}_{i,n})}.$$

The samples are now generated with respect to the density function $p(\cdot)$ which is called the *importance sampling density*.

In this work, a uniform importance sampling density $p(\mathbf{u}_{i,n}) = \text{const}, \forall \mathbf{u}_{i,n} \in \hat{\tilde{\mathcal{U}}}$ is used which allows to define the weighted sample estimator

$$\hat{P}_i(C|\tilde{u}) = \sum_{n=1}^{N_\mathrm{s}} \mathrm{Ind}(C|\tilde{u}, \mathbf{u}_{i,n}) w_n,\ \text{with} \sum_{n=1}^{N_\mathrm{s}} w_n = 1 \tag{A.9}$$

as described by [100]. Where $w_n = \frac{\text{const}}{f(\mathbf{u}_{i,n})}$ is the weight of the random sample $\mathbf{u}_{i,n}$ which is based on the goal function $L(\cdot)$. The weighted samples $\{\mathbf{s}_{i,n}, w_n\}$ can be seen as an approximation of $f(\cdot)$, denoted as $\hat{f}(\cdot)$. The bias of the weighted sample estimator decreases if N_s is increased. By the strong law of large numbers, $\hat{P}(C|\tilde{u})$ will almost surely (a.s.) converge to $P(C|\tilde{u})$ with $N_\mathrm{s} \to \infty$ as presented by [7]. In other words, the probability that the robot trajectory $\tilde{u}$ will collide with an object, is the sum of the weights w_n of all trajectories leading to a collision.

Collision Detection In this thesis, all objects are either disc-shaped or their hull form a rectangle. Regarding disc-shaped objects, a collision between two objects is determined by checking the distance between them. If the distance is smaller than the sum of both radii, a collision occurs. The OBB-Tree algorithm [42] is used to efficiently determine collisions of objects with rectangular shape.

Trajectory Generation In order to calculate the collision probabilities it is necessary to generate trajectories for obtaining the random vectors $\mathbf{u}_i$ and $\mathbf{s}_i$. These contain the control inputs and corresponding states for one sample representing one possible trajectory of the ith object. For generating the trajectories the sequential Monte Carlo method (SMC) is used. An Overview of SMC methods is given by [20] and the first instance of modern SMC was proposed by [41].

The approach presented in the following is the *sequential importance sampling* (SIS). For SIS the random variable $\mathbf{u}$ and the sampling density must be decomposable [100]. Since uniform sampling is used for the sampling density $p(\cdot) = \text{const}$, the weighted sample estimator (A.9) is applied. First of all, N_s samples are generated according to the initial density function $f(\mathbf{x}, t_0)$ to obtain the initial states $\mathbf{x}^0$ and the initial weights are computed. Then, the samples are propagated according to $p(\cdot)$ and the new weights are computed. This is repeated, till the simulation reaches the desired time horizon. An overview of the SIS approach for one object is given in Alg. 10.

Algorithm 10: SIS

for $n = 1$ **to** N_s **do**
  Sample $\mathbf{x}_n^0 \sim f(\mathbf{x}, t_0)$
end for
{Propagation}
for $n = 1$ **to** N_s **do**
  for $k = 1$ **to** N_c **do**
    {Generate random inputs}
    $u^k \sim p(\cdot)$
  end for
  $\mathbf{u}_n = [u^1, \ldots, u^{N_\text{c}}]$
  for $k = 0$ **to** $N_\text{d} - 1$ **do**
    $\mathbf{x}^{k+1} = m(\mathbf{x}^k, u),\ u \in \mathbf{u}_n$
  end for
  $\mathbf{s}_n = [\mathbf{x}_n^0, \mathbf{x}^1, \ldots, \mathbf{x}^{N_\text{d}}]$
end for
{Compute normalized weight w_n}
{Compute final estimate}
$\hat{P}(C|\tilde{u}) = \sum_{n=1}^{N_\text{s}} \text{Ind}(C|\tilde{u}, \mathbf{u}_n) w_n$

A.4. Discussion

The advantages and disadvantages of both presented approaches are discussed in qualitative terms. The occupancy grid approach is especially suitable for objects predicted with a stochastic motion model and for objects with a complex shape, that can be efficiently considered by the Minkowski addition. However, given only the goal function for predicting future motion of the objects or high dimensional workspaces, the Monte Carlo approach is preferable. This is due to the fact, that the occupancy grid approach suffers from the curse of dimensionality.

The occupancy grid approach has a constant estimation error which results mainly due to the discretization of the workspace. This effect can be directly influenced by adjusting the resolution

of the occupancy grid. Furthermore, this approach has a deterministic execution time since no randomness is involved.

The estimation error from the Monte Carlo approach does not suffer from the discretization of the workspace but from the discretization of the control inputs and the sampling of trajectories from the infinite set of possible future trajectories. In addition, the estimation error depends mainly on the number of samples which makes it possible to control the estimation error during the calculation and individually for each object. For instance, a more accurate estimation can be made for close objects and objects with a high collision risk for the robot system. However, it is often not possible to get information about the estimation error of the calculation.

Another important aspect is, that the Monte Carlo approach identifies the trajectories that are leading to a collision. This aspect is important for the interactive assessment which is discussed in Chap. 4.

Bibliography

[1] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Proc. of the 10th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2010.

[2] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. PhD thesis, Technische Universität München, 2010.

[3] M. Althoff and A. Mergel. Comparison of Markov Chain abstraction and Monte Carlo simulation for the safety assessment of autonomous cars. *IEEE Trans. on Intelligent Transportation Systems*, 12(4):1237–1247, 2011.

[4] M. Althoff, O. Stursberg, and M. Buss. Model-based probabilistic collision detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 10:299–310, 2009.

[5] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *IEEE Transactions on Robotics & Automation*, pages 442–447, 2000.

[6] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *Int. Journal Vehicle Autonomous Systems*, 8:190–216, 2010.

[7] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[8] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35:51–76, 2013.

[9] G. Arechavaleta, J-P. Laumond, H. Hicheur, and A. Berthoz. Optimizing principles underlying the shape of trajectories in goal oriented locomotion for humans. In *IEEE-RAS International Conference on Humanoid Robots*, 2006.

[10] I. Asimov. *The Rest of the Robots*. Doubleday, 1964.

[11] T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, 1999.

[12] A. Bauer, K. Klasing, G. Lidoris, Q. Mühlbauer, F. Rohrmüller, S. Sosnowski, T. Xu, K. Kühnlenz, D. Wollherr, and M. Buss. The autonomous city explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics*, 1(2):127–140, 2009.

[13] M. Bennewitz. *Mobile Robot Navigation in Dynamic Environments*. PhD thesis, University of Freiburg, Department of Computer Science, 2004.

[14] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Transactions on Robotics*, 26(3):502–517, 2010.

[16] S. Bouraine, T. Fraichard, and H. Salhi. Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. *Autonomous Robots*, 32(3):267–283, 2012.

[17] G. E. P. Box and G. C. Tiao. *Bayesian Inference in Statistical Analysis*. Wiley, 1973.

[18] A. Broadhurst, S. Baker, and T. Kanade. Monte Carlo road safety reasoning. In *Proc. of the IEEE Intelligent Vehicle Symposium*, pages 319–324, 2005.

[19] J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, USA, 1988.

[20] O. Cappe, S.J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proc. of the IEEE*, 95(5):899–924, 2007.

[21] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces*. Pearson, 1976.

[22] E. P. F. Chan and Y. Yaya. Shortest path tree computation in dynamic graphs. *IEEE Transactions on Computers*, 58(4):541 –557, 2009.

[23] N. Chan, J. J. Kuffner, and M. Zucker. Improved motion planning speed and safety using regions of inevitable collision. In *17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, pages 103–114, 2008.

[24] C. Chucholowski, M. Vögel, O. Stryk, and T. M. Wolter. Real time simulation and online control for virtual test drives of cars. In *High Performance Scientific and Engineering Computing*, volume 8, pages 157–166. Springer Berlin Heidelberg, 1999.

[25] S. Danielsson, L. Petersson, and A. Eidehall. Monte Carlo based threat assessment: Analysis and improvements. In *Proc. of the IEEE Conference on Intelligent Vehicles Symposium*, pages 233–238, 2007.

[26] R. de Nijs, M. Julia, N. Mitsou, B. Gonsior, K. Kühnlenz, D. Wollherr, and M. Buss. Following route graphs in urban environments. In *Proc. of the IEEE Int. Symposium on Robot and Human Interactive Communication*, pages 363–368, 2011.

[27] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of a*. *Journal of the ACM*, 32(3):505–536, 1985.

[28] A. Doucet, N. De Freitas, and N. J. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[29] A. Eidehall and L. Petersson. Statistical threat assessment for general road scenes using Monte Carlo sampling. *IEEE Transactions on Intelligent Transportation Systems*, 9:137–147, 2008.

[30] P. Fiorini and Z. Shillert. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.

[31] T. Fraichard. A short paper about motion safety. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1140–1145, 2007.

[32] T. Fraichard and H. Asama. Inevitable collision states. a step towards safer robots? In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 388–393, 2003.

[33] T. Fraichard and H. Asama. Inevitable collision states. a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[34] C. Fulgenzi. *Autonomous navigation in dynamic uncertain environment using probabilistic models of perception and collision risk prediction*. PhD thesis, INRIA Rhône-Alpes, 2009.

[35] C. Fulgenzi, A. Spalanzani, and C. Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *Proc of the IEEE Int. Conf. on Robotics and Automation*, pages 1610–1616, 2007.

[36] C. Fulgenzi, A. Spalanzani, and C. Laugier. Probabilistic rapidly-exploring random trees for autonomous navigation among moving obstacles. In *Proc of the IEEE Int. Conf. on Robotics and Automation*, 2009.

[37] R. Fuller. A conceptualization of driver behavior as threat avoidance. *Ergonomics*, 27: 1139–1155, 1984.

[38] C. Garcia and M. Prett. Model predictive control: theory and practice. *Automatica*, 25(3): 335–348, 1989.

[39] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman and Hall, 2003.

[40] R. Geraerts and M. Overmars. Sampling techniques for probabilistic roadmaps planners. In *Proc. of the Int. Conf. on Intelligent Autonomous Systems*, pages 600–609, 2004.

[41] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F Radar and Signal Processing*, 140 (2):107–113, 1993.

[42] S. Gottschalk, M. C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proc. on ACM Siggraph 96*, 1996.

[43] M. Greytak and F. Hover. Motion planning with an analytic risk cost for holonomic vehicles. In *Proc of the 48th IEEE Conf. on Decision and Control*, pages 5655–5660, 2010.

[44] R. W. Grubbström and O. Tang. The moments and central moments of a compound distribution. *European Journal of Operational Research*, 170(1):106–119, 2004.

[45] L. J. Guibas, D. Hsu, H. Kurniawati, and E. Rehman. Bounded uncertainty roadmaps for path planning. In *Proc. of the Int. Workshop on the Algorithmic Foundations of Robotics*, 2008.

[46] S. Haddadin. *Towards Safe Robots: Approaching Asimov's 1st Law*. PhD thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen, 2011.

[47] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmüller, A. Albu-Schäffer, and G. Hirzinger. Towards the robotic co-worker. In *ISRR*, volume 70 of *Springer Tracts in Advanced Robotics*, pages 261–282. Springer, 2009.

[48] J. Handschin and D. Mayne. Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control*, 9(5):547–559, 1969.

[49] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51:4282–4286, 1995.

[50] D. Helbing, P. Molnár, I. J. Farkas, and K. Bolay. Self-organizing pedestrian movement. *Planning and Design*, 28:361–383, 2001.

[51] C. Hermes, C. Wohler, K. Schenk, and F. Kummert. Long-term vehicle motion prediction. In *IEEE Intelligent Vehicles Symposium*, pages 652–657, 2009.

[52] T. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *International Journal of Robotics Research*, 26(2):141–166, 2007.

[53] J. Jansson. *Collision Avoidance Theory with Application to Automotive Collision Mitigation*. PhD thesis, Linköping University, 2005.

[54] R. Platt Jr., R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proc. of Robotics Science and Systems VI*, 2010.

[55] L. E. Kavraki, J. C. Latombe P. Svestka, and M. H. Overmars. Probabilistic roadmaps for path planning in high-diemnsional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580, 1996.

[56] E. C. Kerrigan and J .M. Maciejowski. Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control. In *IEEE Conf. on Decision and Control*, 2000.

[57] D. E. Kirk. *Optimal Control Theory: An Introduction*. Dover Books on Electrical Engineering. Dover Publications, 2004.

[58] B. Kluge and E. Prassler. Reflective navigation: Individual behaviors and group behaviors. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 4172–4177, 2004.

[59] B. Kluge and E. Prassler. Recursive probabilistic velocity obstacles for reflective navigation. In Shinichi Yuta, Hajima Asama, Erwin Prassler, Takashi Tsubouchi, and Sebastian Thrun, editors, *Field and Service Robotics*, volume 24 of *Springer Tracts in Advanced Robotics*, pages 71–79. Springer Berlin / Heidelberg, 2006.

[60] K. Madhava Krishna, R. Alami, and T. Simeon. Safe proactive plans and their execution. *Robotics and Autonomous Systems*, 54:244–255, 2006.

[61] T. Kunz, U. Reiser, M. Stilman, and A. Verl. Real-time path planning for a robot arm in changing environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 5906–5911, 2010.

[62] H. Kurniawati, T. Bandyopadhyay, and N. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. In *Proc. of Robotics Science and Systems VII*, 2011.

[63] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How.

[64] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computation for families of linear vector fields. *Symbolic Computation*, 32:231–253, 2001.

[65] A. Lambert, D. Gruyer, and G. S. Pierre. A fast Monte Carlo algorithm for collision probability estimation. In *The 10th Int. Conf. on Control, Automation, Robotics and Vision*, 2008.

[66] A. Lambert, D. Gruyer, G. S. Pierre, and A. Ndjeng. Collision probability assessment for speed control. In *Proc. of the Int. IEEE Conf. on Intelligent Transportation Systems*, pages 1043–1048, 2008.

[67] A. Lambert, D. Gruyer, and G. St. Pierre. A fast Monte Carlo algorithm for collision probability estimation. In *Proc of the Int. Conf. on Control, Automation, Robotics and Vision*, pages 406–411, 2008.

[68] F. Large, D. A. Vasquez Govea, T. Fraichard, and C. Laugier. Avoiding Cars and Pedestrians using V-Obstacles and Motion Prediction. In *Proc. of the IEEE Intelligent Vehicle Symp.*, 2004.

[69] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[70] C. Laugier and R. Chatila. *Autonomous navigation in dynamic environments*. Springer Tracts in Advanced Robotics. Springer, 2007.

[71] C. Laugier, I. Paromtchik, M. Perrollaz, Y. Mao, J. Yoder, C. Tay, K. Mekhnacha, and A. Nègre. Probabilistic analysis of dynamic scenes and collision risk assessment to improve driving safety. *Intelligent Transportation Systems Journal*, 3(4):4–19, 2011.

[72] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, University of Illinois, Department of Computer Science, 1998.

[73] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[74] S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Trans. on Robotics and Automation*, 14(6), 1998.

[75] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1999.

[76] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The Int. Journal of Robotics Research*, 20:378–401, 2001.

[77] S. M. LaValle and R. Sharma. A framework for motion planning in stochastic environments: Modeling and analysis. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1995.

[78] A. Lawitzky, D. Wollherr, and M. Buss. Maneuver-based risk assessment for high-speed automotive scenarios. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1186–1191, 2012.

[79] A. Lazo and P. Rathie. On the entropy of continuous probability distributions. *IEEE Transactions on Information Theory*, 41(1):120–122, 1978.

[80] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. M. Stavens, A. Teichman, M. Werling, and S. Thrun. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium*, pages 163–168, 2011.

[81] J. Lien. Hybrid motion planning using minkowski sums. In *Proceedings of Robotics: Science and Systems IV*, 2008.

[82] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. *International Journal of Robotics Research*, 28(8):933–945, 2009.

[83] L. Martinez-Gomez and T. Fraichard. An efficient and generic 2d inevitable collision state-checker. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, pages 234–241, 2008.

[84] L. Martinez-Gomez and T. Fraichard. Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 100–105, 2009.

[85] H. Michalska and D.Q. Mayne. Robust receding horizon control of constrained systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633, 1993.

[86] P. E. Missiuro and N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *IEEE Int. Conf. on Robotics and Automation*, pages 1261–1267, 2006.

[87] K. Mombaur, A. Truong, and J.-P. Laumond. From human to humanoid locomotion – an inverse optimal control approach. *Autonomous Robots*, 23(3):369–383, 2009.

[88] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Hähnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

[89] A. Neyman and S. Sorin, editors. *Stochastic Games and Applications*, volume 570 of *Series C: Mathematical and Physical Sciences*. Kluwer Academic Publishers, 2003.

[90] R. Parthasarathi and T. Fraichard. An inevitable collision state-checker for a car-like vehicle. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3068–3073, 2007.

[91] S. Patil, J. van den Berg, and R. Alterovitz. Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty. In *In Proc. of Int. Conf. on Robotics and Automation*, pages 3238–3244, 2012.

[92] S. Petti and T. Fraichard. Safe motion planning in dynamic environments. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, pages 2210–2215, 2005.

[93] R. Philippsen. *Motion Planning and Obstacle Avoidance for Mobile Robots in Highly Cluttered Dynamic Environments*. PhD thesis, ETH Zürich, Institute of Robotics and Intelligent Systems, 2004.

[94] M. Pivtoraiko and A. Kelly. Efficient constrained path planning via search in state lattices. In *Proc. of the Int. Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2005.

[95] E. Prassler, J. Scholz, and M. Strobel. Maid: mobility assistance for elderly and disabled people. In *Proc. of the 24th Annual Conference of Industrial Electronics Society*, 1998.

[96] L. A. Rastrigin. The convergence of the random search method in the extremal control of a many parameter system. *Automation and Remote Control*, 24(10):1337–1342, 1963.

[97] J. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 224–241, 1979.

[98] P. Resende and F. Nashashibi. Real-time dynamic trajectory planning for highly automated driving in highways. In *Proc. of Intelligent Transportation Systems*, 2010.

[99] D. B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applies statistician. *Annals of Statistic*, 12(4):1151–1172, 1984.

[100] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. Wiley-Interscience, 1981.

[101] T. Sasaki and H. Hashimoto. Human observation based mobile robot navigation in intelligent space. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1044–1049, 2006.

[102] M. Scheint, J. Wolff, and M. Buss. Invariance control in robotic applications: Trajectory supervision and haptic rendering. In *Proc. of the American Control Conference*, pages 1436–1442, 2008.

[103] T. Schouwenaars. *Safe Trajectory Planning of Autonomous Vehicles*. PhD thesis, Massachusetts Institute of Technology, 2006.

[104] T. Schouwenaars, B. De Moor, E. Ferson, and J. How. Mixed integer programming for multi-vehicle path planning. In *Proc. European Control Conference*, pages 2603–2608, 2001.

[105] J. T. Schwartz and M. Sharir. On the "piano movers" problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure Applied Mathematics*, 36(3):345–398, 1983.

[106] R. Siegwart, K. O. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, R. Philippsen, R. Piguet, G. Ramel, G. Terrien, and N. Tomatis. Robox at expo.02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42:203–222, 2003.

[107] E. A. Sisbot and R. Alami. A human-aware manipulation planner. *IEEE Transactions on Robotics*, 28(5):1045–1057, 2012.

[108] T. Smith and R. G. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 542–549, 2005.

[109] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha. Smooth and collision-free navigation for multiple robots under differential-drive constraints. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2010.

[110] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha. Real-time navigation of independent agents using adaptive roadmaps. In *ACM Symposium on Virtual Reality Software and Technology*, 2007.

[111] S. Thompson, T. Horiuchi, and S. Kagami. A probabilistic model of human motion and navigation intent for mobile robot path planning. In *Proc. of Int. Conf. on Autonomous Robots and Agents*, pages 663–668, 2009.

[112] S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proc. of the National Conference on Artificial Intelligence*, 1996.

[113] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[114] N. E. Du Toit and J. W. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 966–973, 2010.

[115] N. E. Du Toit and J. W. Burdick. Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics*, 27(4):809–815, 2011.

[116] N. E. Du Toit and J. W. Burdick. Robotic motion planning in dynamic,uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2012.

[117] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2010.

[118] C. Urmson, C. Baker, J. M. Dolan, P. Rybski, B. Salesky, W. L. Whittaker, D. Ferguson, and M. Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, 30:17–29, 2009.

[119] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and Ming Lin. Interactive navigation of individual agents in crowded environments. In *Proc. of the Symposium on Interactive 3D Graphics and Games*, 2008.

[120] J. van den Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Proc. of Robotics Science and Systems VI*, 2010.

[121] J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, *Robotics Research*, volume 70 of *Springer Tracts in Advanced Robotics*, pages 3–19. Springer Berlin Heidelberg, 2011.

[122] D. Vasquez. *Incremental Learning for Motion Prediction of Pedestrians and Vehicles*. PhD thesis, Institut National Polytechnique de Grenoble, 2007.

[123] M. Werling and D. Liccardo. Automatic collision avoidance using model-predictive online optimization. In *IEEE Conf. on Decision and Control*, 2012.

[124] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. In *Proc. of Int. Conf. on Robotics and Automation*, 2010.

[125] M. Werling, S. Kammel, J. Ziegler, and L. Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *Int. Journal of Robotics Research*, 2011.

[126] J. Wolff and M. Buss. On stability of invariance controlled linear systems. In *Proc. of the European Control Conference*, pages 3281–3288, 2007.

[127] J. Yan and R. R. Bitmead. Incorporating state estimation into model predictive control and its application to network traffic control. *Automatica*, 41(4):595–604, 2005.

[128] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38 (4), 2006.

[129] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell amd M. Hebert, A. K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *IEEE Int. Conf. on Intelligent Robots and Systems*, 2009.

[130] M. Zucker. Approximating state-space obstacles for non-holonomic motion planning. Technical report, Carnegie Mellon University, Robotics Institute, 2006.

Own Publications

[131] M. Althoff, **D. Althoff**, D. Wollherr, and M. Buss. Safety verification of autonomous vehicles for coordinated evasive maneuvers. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010.

[132] **D. Althoff**, O. Kourakosand, M. Lawitzky, A. Mörtl, M. Rambowand F. Rohrmüller, D. Brščić, D. Wollherr, S. Hirche, and M. Buss. An architecture for real-time control in multi-robot systems , human centered robot systems. In H. Ritter, G. Sagerer, R. Dillmann, and M. Buss, editors, *Cognitive Systems Monographs*, pages 43–52. Springer, 2009.

[133] **D. Althoff**, M. Althoff, D. Wollherr, and M. Buss. Probabilistic collision state checker for crowded environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010.

[134] **D. Althoff**, C. N. Brand, D. Wollherr, and M. Buss. Computing unions of inevitable collision states and increasing safety to unexpected obstacles. In *In Proc. of Int. Conf. on Intelligent Robots and Systems*, 2011.

[135] **D. Althoff**, J. J. Kuffner, D. Wollherr, and M. Buss. Safety assessment of robot trajectories for navigation in uncertain and dynamic environments. *Springer Autonomous Robots*, SI Motion Safety for Robots, 2011.

[136] **D. Althoff**, D. Wollherr, and M. Buss. Safety assessment of trajectories for navigation in uncertain and dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2011.

[137] **D. Althoff**, M. Buss, A. Lawitzky, M. Werling, and D. Wollherr. On-line trajectory generation for safe and optimal vehicle motion planning. In P. Levi, O. Zweigle, K. Häußermann, and B. Eckstein, editors, *Autonomous Mobile Systems*, Informatik aktuell, pages 99–107. Springer Berlin Heidelberg, 2012.

[138] **D. Althoff**, M. Werling, N. Kaempchen, D. Wollherr, and M. Buss. Lane-based safety assessment of road scenes using Inevitable Collision States. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 2012.

[139] **D. Althoff**, B. Weber, D. Wollherr, and M. Buss. Closed-loop safety assessment of uncertain roadmaps: Incorporation of replanning possibilities. *Springer Autonomous Robots*, 2014, accepted.

[140] K. Klasing, **D. Althoff**, D. Wollherr, and M. Buss. Comparison of surface normal estimation methods for range sensing applications. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009.

[141] A. Lawitzky, **D. Althoff**, D. Wollherr, and M. Buss. Dynamic window approach for omnidirectional robots with polygonal shape. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010.

[142] A. Lawitzky, **D. Althoff**, C. F. Passenberg, G. Tanzmeister, D. Wollherr, and M. Buss. Interactive scene prediction for automotive applications. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 1028–1033, 2013.