Universität Stuttgart
Institut für Fahrzeugtechnik Stuttgart

Vertraulich

Automatisierte Vermeidung von Straßenschäden mit Hilfe von Reinforcement Learning-Konzepten

Automated Road Damage Avoidance using Reinforcement Learning Concepts

Masterarbeit

von

cand. tema Qiuyi Cao

Matr. Nr. 3492217

Institutsbetreuer:	Ralf Sauerwald, M.Sc.
Zweiter Betreuer:	Lukas Lang, M.Sc.
Industriebetreuer:	Kailin Tang, Virtual Vehicle Research GmbH
Prüfer:	Prof. Dr.-Ing. H.-C. Reuss

Vorgelegt an der Universität Stuttgart
Institut für Fahrzeugtechnik Stuttgart
Lehrstuhl Kraftfahrzeugmechatronik

«Abgabe_real»

Universität Stuttgart
Institut für Fahrzeugtechnik Stuttgart

Lehrstuhl
Kraftfahrzeugmechatronik
Prof. Dr.-Ing. H.-C. Reuss

Tel. (+49) 711 685 – 68501
Fax (+49) 711 685 – 68533
info@ifs.uni-stuttgart.de

18.07.2022

Masterarbeit

für

Frau cand. tema Qiuyi Cao

Matr. Nr. 3492217

Thema: Automatisierte Vermeidung von Straßenschäden mit Hilfe von Reinforcement Learning-Konzepten

Automated Road Damage Avoidance using Reinforcement Learning Concepts

Road damage is a problem that not only reduces driving comfort but sometimes also threatens driving safety. At present, the information about road damage (e.g., position, shape, and severity) can be transmitted to highly automated vehicles via infrastructure to vehicle (I2V) communications. Road damage has irregular shapes, which presents a challenge for conventional trajectory planning algorithms. Reinforcement Learning (RL) has obtained more attention recently as it does not rely on preprogramed logics and has the potential to solve more general decision-making problems. As part of a dynamic and international team and within the scope of an international research project, the thesis student shall help with the development of a first-of-a kind use case for an automated driving vehicle concept for avoidance of road damages using RL concepts and will have chance to contribute to dissemination of the implementation results.

Universität Stuttgart
Institut für Fahrzeugtechnik Stuttgart

Schutzvermerk:	Die Arbeit ist bis zum 17.01.2028 vertraulich zu behandeln
Institutsbetreuer:	Ralf Sauerwald, M.Sc.
Zweiter Betreuer:	Lukas Lang, M.Sc.
Industriebetreuer:	Kailin Tang, Virtual Vehicle Research GmbH
Prüfer:	Prof. Dr.-Ing. H.-C. Reuss
Beginn:	18.07.2022
Abgabedatum:	17.01.2023

_____

Prof. Dr.-Ing. H.-C. Reuss

Universität Stuttgart
Institut für Fahrzeugtechnik Stuttgart

Erklärung

Hiermit versichere ich, Qiuyi Cao, dass ich die vorliegende Arbeit, bzw. die darin mit meinem Namen gekennzeichneten Anteile, selbständig verfasst und bei der Erstellung der Arbeit die einschlägigen Bestimmungen, insbesondere zum Urheberrechtsschutz fremder Beiträge, eingehalten und nur die angegebenen Quellen und Hilfsmittel benutzt habe.

Soweit meine Arbeit fremde Beiträge (z. B. Bilder, Zeichnungen, Textpassagen) enthält, erkläre ich, dass ich diese Beiträge als solche gekennzeichnet (z. B. Zitat, Quellenangabe) habe und dass ich eventuell erforderliche Zustimmungen der Urheber zur Nutzung dieser Beiträge in meiner Arbeit eingeholt habe.

Für den Fall der Verletzung Rechter Dritter durch meine Arbeit erkläre ich mich bereit, der Universität Stuttgart einen daraus entstehenden Schaden zu ersetzen bzw. die Universität Stuttgart von eventuellen Ansprüchen Dritter freizustellen.

Die Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Ferner ist sie weder vollständig noch in Teilen bereits veröffentlicht worden. Das elektronische Exemplar stimmt mit den anderen Exemplaren überein.

Stuttgart, den 17.01.2023

Qiuyi Cao

Contents

Acknowledgement

This work was conducted in the scope of the ESRIUM Project, which has received funding from the European Union Agency for the Space Programme (EUSPA) under the European Union's Horizon 2020 research and innovation programme and under grant agreement No 101004181. The content of this thesis reflects only the author's view. Neither the European Commission nor the EUSPA is responsible for any use that may be made of the information it contains. The thesis was written at Virtual Vehicle Research GmbH in Graz, which is partially funded within the COMET-K2 Competence Centers for Excellent Technologies Programme by the Austrian Federal Ministry for Climate Action (BMK), the Austrian Federal Ministry for Digital and Economic Affairs (BMDW), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the K2 programme management.

Acronyms

VOC	Vehicle Operation Cost
ESRIUM	EGNSS-enabled Smart Road Infrastructure Usage and Maintenance
EGNSS	European Global Navigation Satellite System
ADAS	Advanced Driver Assistance Systems
CAV	Connected Automated Vehicles
OEDR	Object and Event Detection and Response
ODD	Operational Design Domain
ADS	Autonomous Driving System
CV	Computer Vision
V2I	Vehicle to Infrastructure
RL	Reinforcement Learning
MDP	Markov Decision Process
DP	Dynamic Programming
MC	Monte-Carlo methods
TD	Temporal-Difference Learning
AC	Actor-Critic

Symbols

Symbol	Unit	Description
π	-	Policy
r	-	Reward
s	-	State
a	-	Action
v	-	Value
Q	-	Q-Value
Pr	-	Probability
E	-	Expectation
γ	-	Discount factor
T	-	Transition probability
G	-	Return
α	-	Learning rate
ϵ	-	Probability parameter
θ	-	Parameters
J	-	A scalar performance measure
μ	-	On-Policy distribution
b	-	Baseline or Behaviour function

List of Figures

List of Tables

Kurzfassung

ESRIUM ist ein EU-Projekt mit Schwerpunkt auf Straßenschäden und daraus resultierenden Problemen. In dieser Arbeit werden Konzepte des Reinforcement Learning zur Vermeidung von Straßenschäden untersucht.

Die Arbeit führt zunächst in das Problem der Straßenschäden ein, das in der Forschung zum autonomen Fahren nicht ernst genommen wird. Obwohl in dieser Arbeit nur die Bewegungsplanung auf der Grundlage bekannter Straßenschäden betrachtet wird, ist das Problem der Straßenschadensvermeidung kein eigenständiges Thema, sondern bedarf der Unterstützung durch viele andere Techniken. Daher wird der technische Hintergrund einschließlich des allgemeinen autonomen Fahrsystems, der Computer Vision zur Erkennung von Straßenschäden und der I2V-Techniken erläutert und die einschlägige Literatur gesichtet. Für das eigentliche Bewegungsplanungsproblem werden die jüngsten Forschungen zu Bewegungsplanungsalgorithmen und Reinforcement Learning zusammengefasst.

Herkömmliche Bewegungsplanungsalgorithmen werden dann erläutert und mit Algorithmen des Reinforcement Learning verglichen. Anhand der Benchmarks können wir die Vorteile des RL-Ansatzes für das Problem der Vermeidung von Straßenschäden erkennen. die Details des Reinforcement-Learning-Konzepts wird dann erläutert und die theoretische Grundlage für die technische Umsetzung ist geliefert. Abschließend werden der Entwurf des Szenarios und die Aufbereitung des Trainingsdatensatzes beschrieben. Die abstrakte Problemformulierung des Straßenschadenproblems erfolgt ebenfalls im Rahmen des Reinforcement Learning.

Zusammenfassend erklärt die Arbeit die Notwendigkeit der Forschung auf dem Gebiet der Straßenschäden, gibt einen Überblick über die Literatur im Bereich der Bewegungsplanungsalgorithmen, formuliert das Problem der Straßenschadenvermeidung in einem Reinforcement Learning Rahmen und stellt den Trainingsrahmen auf.

Abstract

The ESRIUM project, which is centered on the issue of road damage, serves as the foundation for the thesis. This thesis investigates the application of reinforcement learning theory to the problem of road damage avoidance.

The thesis begins by outlining the issue of road damage, which has received little attention in studies on autonomous driving. Although the road damage avoidance problem is not a separate topic and requires the support of many other strategies, this thesis simply takes into account mobility planning based on knowledge about known road damage. This is followed by an explanation of the technical background, which included a general autonomous driving system, computer vision for spotting road damages, and Infrastructure-to-Vehicle(I2V) approaches, as well as a study of pertinent literature. The most current studies on motion planning algorithms and reinforcement learning are also summarized, for the motion planning problem itself.

Then, conventional motion planning algorithms are described and contrasted with methods for reinforcement learning. The benchmark studies show the potential advantages of applying reinforcement learning solution to the road damage avoidance challenge. The thesis then goes into further depth on the idea of reinforcement learning and offers the theoretical underpinnings for its technical application. Lastly, a description of the scenario design and training dataset preparation is provided. The road damage problem's abstract formulation is also carried out within a reinforcement learning framework.

To sum up, the thesis discusses the importance of studying road damage, reviews the research on motion planning algorithms, formulates the challenge of preventing road damage within the context of reinforcement learning, and establishes the training environment.

1 Introduction

Walking into the 2020s, self-driving cars are no more a fantasy but a future in coming. An autonomous driving system is a complex system with massive sub-topics to research. With the development of AI technology like deep learning, the amount of research applying learning approaches to autonomous driving systems shows significant growth. In this thesis, the possibility of using reinforcement learning in motion planning under the scenario of road surface damage is discussed. The work is based on the project ESRIUM, so it inherits some settings to maintain coherence.

This chapter will introduce the use case of road damage and the project ESRIUM, along with the structure of the remaining parts of the thesis.

1.1 Road Damage

Road damage is a very common phenomenon of infrastructure destruction. Transportation systems, which play the most important role in the daily activities of human beings, are under enormous operational pressure. The wear and tear over the years make road surface damage inevitable. Not to mention that natural causes such as landslides, weathering, and oxidation will definitely make the pavement more fragile. While most crashes are due to human factors, which is one of the reasons why autonomous driving systems were introduced, poor road conditions are also a threat to passengers' safety. A study from the Pacific Institute for Research and Evaluation highlights that roadway condition is a contributing factor in more than half (52.7%) of the nearly 42,000 American deaths resulting from motor vehicle crashes each year and cost the U.S. economy more than $217 billion each year [1].

Although there is a dedicated team responsible for pavement maintenance, the road network is so large that it is costly to achieve comprehensive road damage detection by traditional methods. The maintenance work itself will disrupt traffic and is very expensive, making it difficult to maintain the road surface in good condition at all times.

In fact, the cost to maintain a mile of road per year in the U.S. is in the range of $782-$208,736, and the average cost for highways is $28,020 [2]. And the costs of road damage go beyond maintenance. The concept of Vehicle Operation Cost (VOC) should be introduced in the problem. VOC by definition are the costs associated with operating a motor vehicle. It is composed of fuel, oil, tires, repairs and maintenance, and interest and depreciation costs. VOC depends on the quality of the road surface

as measured by its roughness [3]. When a vehicle damages the road surface and increases its roughness, it thereby increases the vehicle operating costs of subsequent vehicles. Normally VOC is 10-100 times to maintenance cost [4]. Consequently, road damage will have a huge and long-term impact on the economy.

It has been mentioned previously that road conditions, as well as human factors, are both contributing factors to car crashes. In spite of the fact that human error is the major cause of car accidents, it is unrealistic to place hopes on improving road users' driving abilities or regulating their driving behavior in order to ensure driving safety. However, with the technology we have today, we can avoid road damage, maintain it in a timely manner, and provide algorithms to assist drivers to make reasonable and safe decisions when they encounter road damage. Therefore, research on road damage is necessary to improve road driving safety.

Tarmac and asphalt are typically durable pavements. Asphalt surfaces are usually very robust and most of the damage you see such as potholes, alligator cracks and uneven surfaces are often the cumulative result of a range of causes. Heavy vehicles can put a lot of pressure on the surface itself. The consistent stress can cause weaknesses to emerge in the road surface, which as a result causes cracking. Leaking oil, which is in small amounts and cleaned quickly is not always an issue, but if sits on the asphalt longer, it will seep in and ruin the top layer, as well as being exceptionally difficult to repair and remove. Besides human factors, nature also plays a role. Water is one of the biggest causes of road surface damage. Normally tarmac seals and pavement preservation can avoid significant damage to the road surface from water, but if the surface is cracked, water seeps in and weakens the base course layer, which causes depressions. What we need to pay more attention to is that if that base layer is damaged, the road will continually have problems with traffic load and be more vulnerable to cracks and potholes until the road is completely repaved. Since asphalt depends on a binder to hold together the rocks, aggregates and sand that make up the surface, it could face the risk of the dissolution of the binder due to ultraviolet light. Over time, the road will resemble the loose arrangement of gravel and starts raveling. Resurfacing then will be necessary. Oxidation can not only age people but also road surfaces, making them less flexible and more susceptible to being cracked under heavy loads. The consistent movement of tectonic plates, along with earthquakes and other natural phenomena, can cause a surface to shift. Even though it will settle over time, cracks, sinkholes, and other major damage will be inevitable. The main causes of road damage are shown in Fig. 1.1 [5].

Road damage differs from other on-road static obstacles in the feature of different severity levels. In [6], four levels of road damage severity are identified with qualitative descriptions. Four road damage severity levels due to landslides are listed from level A

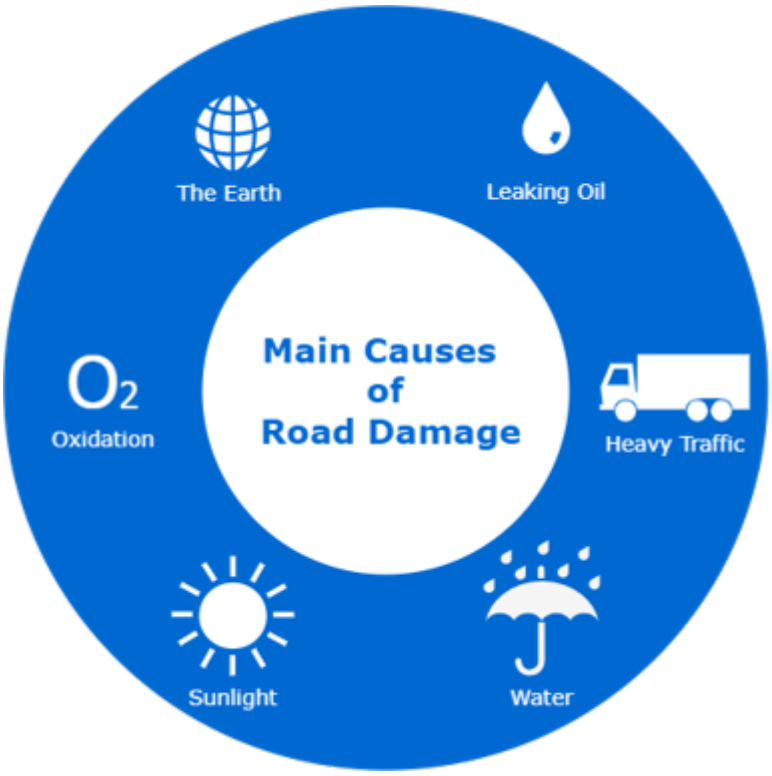

Figure 1.1: Main causes of road damage [5]

to level D. Under damage level A, there are rarely cracks and deformations, and no speed reduction of the vehicle is required; under damage level B, damage includes pavement deformation and cracks and/or roadside destruction, without affecting the functioning of the road; under damage level C, pavement deformation is so substantial that traffic lanes and/or the roadside are affected, and necessary restrictions such as alternative pass and traffic light regulations are required; under damage level D, displacements are of the same order of magnitude as for level C, but with the vertical component prevailing and providing the possibility for complete destruction and loss of the pavement continuity, and the functioning of the road is seriously affected. The demonstration is shown in Fig. 1.2.

In summary, although road damage is common and unavoidable, it has an important impact on traffic safety, comfort and economy. Therefore, road damage is a use case that deserves to be studied and to have state-of-the-art technology applied to it. This is also the starting point of the project ESRIUM.

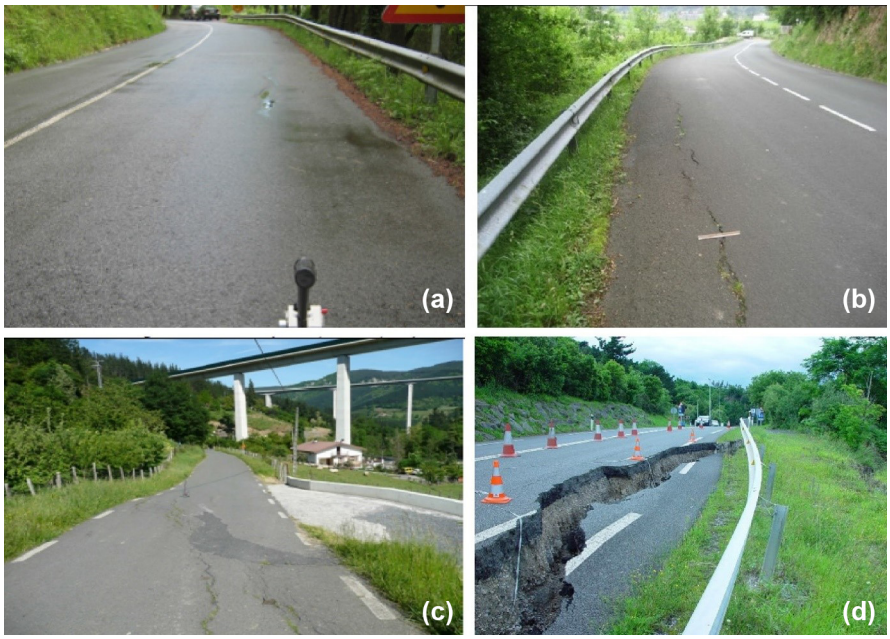

Figure 1.2: Road damage severity classification: (a) no/slight damage, (b) moderate damage, (c) severe damage, and (d) partial/total destruction [6]

1.2 The project ESRIUM

ESRIUM is the abbreviation for "EGNSS-enabled Smart Road Infrastructure Usage and Maintenance", which is a Horizon2020 project aiming at increasing the safety and energy efficiency of transport on European roads. Its key in the novation is a digital map of road surface and road wear. The mission is an EGNSS-based digital road wear map generating routing recommendations based on road damage locations, road damage type, recent repair interventions and road damage prediction [7].

With the development of autonomous driving systems, Advanced Driver Assistance Systems (ADAS) are widely equipped on modern vehicles to reduce the driver's cognitive load. Although the concept of ego-vehicle intelligence is popular, depending on the on-board sensors to achieve a higher autonomous level is still challenging. But through introducing infrastructure assistance into the decision-making process of automated vehicles, a higher level and more robust autonomy can be achieved to ensure a safer drive. Infrastructure communications can be used to convey dynamic traffic and hazard information ahead and give real-time routing and driving recommendations to Connected Automated Vehicles (CAV) beyond the range of on-board sensors.

The decision-making process of automated vehicles is fully based on the on-board sensor information nowadays. Due to the limitation of the information sources, more automated vehicles especially heavy trucks driving on the road will lead to more road

damage in the form of rutting, because they may all choose to drive in the middle line. Inspired by this, ESRIUM investigates infrastructure assisted routing recommendations utilizing C-ITS(Cooperative Intelligent Transport Systems) communications. [8] presents four use cases: 1. AI-based road damage prediction to support enhanced road maintenance planning, 2. Routing Recommendations based on the road wear map, provided via C-ITS messages (Manoeuvres within lane and between lanes) 3. C-ITS Message 'GNSS-correction data' provision. 4. Wear map content provision. [9] and [10] study two use case scenarios: a in-lane off-set recommendation to avoid accumulative pressure which causes rutting and a strategic lane change and lane utilization information to help CAVs to avoid road damage.

With the existing achievements, this thesis will explore the possibility of using the reinforcement learning concept to avoid road damage with different severity levels by giving recommendations of deceleration and off-set.

1.3 Structure of the thesis

Chapter 2 will introduce the technical background of the thesis, including the literature review in the field of autonomous driving systems, computer vision in road damage detection, I2V and reinforcement learning. Chapter 3 lists the commonly used motion planning algorithms like A* search, CL-RRT, convex optimization and potential field and analyses their pros and cons in the road damage avoidance problem as a benchmark to the method of reinforcement learning. Chapter 4 will explain the concept of reinforcement learning along with relevant algorithms. With the theoretical foundation, a technical realization of reinforcement learning in road damage avoidance will be implemented in chapter 5, including the scenario design, road damage generation and problem formulation in the reinforcement learning framework. Chapter 6 draws a conclusion in the end, concluding the research and discussing potential future works.

2 Technical Background

During the last decades, advances in artificial intelligence and control systems have allowed self-driving cars to become possible. ADAS has already significantly improved road safety [11]. The assistance of infrastructure can help to further enhance road safety and driving comfort [12]. This thesis is about using the reinforcement learning concept to realize an accurate road damage avoidance according to the damage severity level. Before we actually implement the approach, there are still several questions to be solved:

- If the autonomous driving system technology of the current level can realize the vision for road damage avoidance?
- What kind of technologies are needed in this process?
- If the road damage can be detected?
- How does the vehicle get the road damage information?
- Why should we use reinforcement learning?
- Can't conventional motion planning algorithms solve the problem?

In this chapter, we will take a glimpse of the development of relevant technologies to answer these questions.

2.1 Autonomous Driving System

Road traffic crashes cause approximately 1.3 million people's death annually according to the report of WHO in 2018. Between 20 and 50 million people suffer non-fatal injuries, many of which induce a disability in the end. With such a big live cost, road traffic injuries are the 8th leading cause of death among all ages, while other causes are all diseases. For the death of children and young adults aged 5-29 years, road traffic injuries are even the leading cause [13]. Meanwhile, as the result of the survey from NHTSA, 94% of all road accidents are caused by human error, like speeding, drunk driving, distracted driving and so on [14]. Looking at all these brutal statistics from another point of view, we can see that the road traffic crashes are not only the deadliest killer but also an opening for human intervention, through preventing human error to avoid car crashes, leading to an extension of human lifespan. Autonomous vehicles can also benefit specific groups who cannot drive themselves, for example, both young

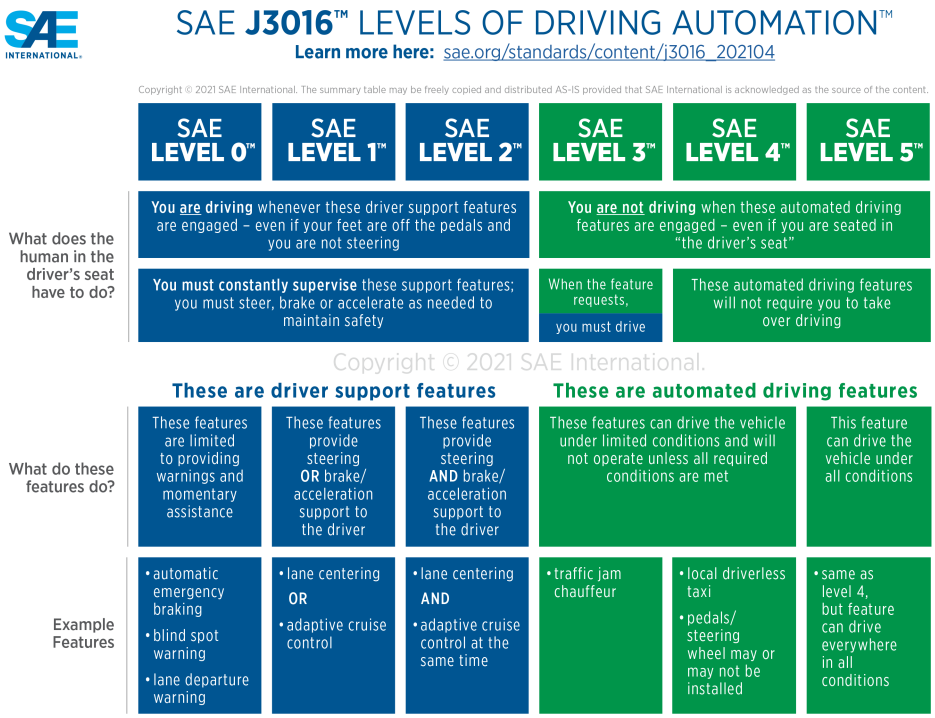

Figure 2.1: SAE Levels of driving automation [15]

and old people and people with disabilities. Autonomous driving systems can also optimize driving behavior to reduce fuel consumption and reduce traffic congestion through planning.

The commonly used taxonomy of autonomous driving system is defined by the SAE Standard, under which ADS has six levels of driving automation, ranging from no driving automation (Level 0) to full driving automation (Level 5), based on if it is equipped with the autonomous capabilities like automated lateral/longitudinal control, OEDR (Object and Event Detection and Response) and complete/restricted ODD (Operational Design Domain) [15]. A detailed explanation of each level is shown in Fig. 2.1. The function studied in this work assists the driver in changing both speed and steering angle, so it is a Level 2 autonomous driving system function according to the SAE standard.

Nowadays, ADAS are commonly equipped on vehicles providing assistance for drivers. ADAS corresponds to the level 0 to level 2 autonomous driving system in the SAE standard, and it can provide functions including adaptive cruise control, lane keeping assistance, and automatic emergency braking. But the pursuit of a higher autonomous level or fully autonomous vehicles continues unabated. In 2020, Waymo, formerly known as the Google self-driving car project, operated a commercial self-driving taxi service in USA and expanded the service to the public, which drags the vision of

introducing ADS into people's daily life closer.

With the increase in public attention and the influx of capital, a great deal of research on autonomous driving systems continues to be conducted. A common approach to system architecture has been established over the years. Most autonomous driving systems divide the immense task of autonomous driving into several subcategories and employ a range of sensors and algorithms on different modules. Recently, end-to-end driving has started to emerge as an alternative to the modular approach. Deep learning models have come to dominate many of these tasks. At the application level, the individual core functions are usually divided into localization, mapping, perception, planning, vehicle control and human-machine interface. This thesis uses the learning method to deal with the motion planning problem.

2.2 Road damage detection and recognition

The automated vehicles are normally equipped with LiDAR(Light Detection and Ranging), cameras and radars to detect the surrounding environment. Road damage can be captured by on-board sensors like laser line-scan cameras and 3D cameras, which create images with high quality and resolution. This method is used by some agencies to perform pavement condition surveys. However, such imaging equipment mounted on dedicated vehicles is expensive and difficult to extend to local agencies with a limited budget. A low-cost method capable of comprehensively surveying road surfaces is required. The images from smartphones and drive cameras reduce the cost of collecting data, but also presents challenges to computer vision [16].

To meet this challenge, in 2018, the Japanese research team hosted the IEEE BigData Cup to evaluate the contemporary methods towards this problem. 54 teams from all over the world participated in the challenge, and several novel methods were proposed for improving the accuracy of automatic road damage detection system. These methods were utilized by several municipalities in Japan after and challenge. The practical use and the feedback of government agencies suggested that more robust algorithms are needed. Moreover, most of the methods are limited to road conditions in a single country. To solve these problems, Global Road Damage Detection Challenge (GRDDC) is held in 2020. In addition to the road damage images from Japan, datasets from India and Czech Republic were also augmented, which leads to the data volume tripled, comprising 26620 images [8].

The main tasks of GRDDC'2020 is classification of road damage and detection of the location of road damage. Out of the 121 teams, team IMSC won the first prize with

a method based on ultralytics-YOLO (u-YOLO)[17]. YOLO is an abbreviation for 'you only look once', it's a state-of-the-art, real-time object detection system. The proposed approach synthesized images using Python Augmenter and input them along with the existing images to the trained u-YOLO to get multiple predictions about the road damage classification and location, of every test image. By filtering the predictions, an improved accuracy is achieved. Regarding the u-YOLO, they employed the ensemble learning method, using different combinations of hyperparameters to generate different trained models. The best performed models are selected, and each image is passed through all the selected models. During this process, the prediction variance is reduced and the accuracy is improved [18]. This combination of the Ensemble Model and Ensemble Prediction providing the highest accuracy with the cost of speed of detection. Most of the other winning methods executed data augmentation and applied ensemble learning. During the training process, YOLO models and variant R-CNN models are most used, which both are state-of-the-art algorithms accomplished by the development of computer vision. The GRDCC will in coming years introduce more tasks like severity analysis and the road damage classes will also be extended.

The development of computer vision in road damage detection offers a potential method, by which the data is easier and cheaper to get. That would extend the scenarios wider to other countries and regions and thus extend the application scope of road damage avoidance algorithms.

2.3 Vehicle-to-Infrastructure

In addition to the autonomous driving system solution, connected cars are also an important solution to improve road safety and increase traffic efficiency. In contrast to self-driving cars that need to be equipped with sophisticated sensors, connected cars only need to be equipped with communication devices that allow them to access the Internet or communicate with surrounding wireless devices. Connected vehicle is an Internet of Things(IoT) technology with broad implications from connected entertainment systems that connect with the driver's mobile phone to Internet-connected vehicles that have bi-directional communication with other vehicles, mobile devices and city intersections, which is the key concept of V2X [19]. Vehicle-to-Everything(V2X) or Car-to-Everything(C2X) refers to the communication of connected cars with any other factors. Vehicle-to-Infrastructure is a subtype of V2X communication, which enables connected cars to communicate wirelessly with devices in their environment.

V2I has a wide range of use cases. [20] uses I2V integration of Adaptive Cruise Control (ACC) and the Variable Speed Limit (VSL), a roadside control method, to reduce the

Figure 2.2: A use case of I2V communication [25]

risk of rear-end collisions. In the scenario of intersections, [21] uses the I2V system to analyse the trajectories of cars, bicycles and pedestrians within an intersection to determine a potential collision and update the information with drivers to guide them take maneuvers to avoid an accident. [22] presents a complete traffic sign recognition system based on on-board vision sensor with the help of I2V communication, using a RF particle filter tracking system, to improve the recognition accuracy. The recent research works on the combination of V2V and V2I, and applies it to energy management for hybrid vehicles[23], and to road weather and traffic condition information exchange[24]. In short, with the support of infrastructure, our transportation can become safer and smarter. Fig.2.2 demonstrates a digital infrastructure on the motorway, showing a comprehensive usage of information from infrastructure.

I2V technology C-ITS is also used in the project ESRIUM. C-ITS means Cooperative Intelligent Transport Systems, it is a WLAN-based radio system for the exchange of safety-relevant information between vehicles and the road(e.g. accident report, traffic jam, breakdown, natural events, etc.) [25]. The communication process is shown in Fig.2.3. During the process of I2V, the road sends information, comparable to a radio signal. Anyone who receives it can use it. There is no direct connection between the receiver and the transmitter. Receiving the information is completely anonymous. Anyone who wants to communicate in this way must register. Every message sent contains a certificate from the sender. This ensures that you always know who is sending the information and that you can trust the content of the messages. In the process of V2I, the vehicle sends information, the road receives it. Vehicles must also register and attach a certificate to the messages. Special protective measures are used to ensure that data is completely anonymized and no inference to personal data is possible. The central station is responsible for traffic management and other information/data processing, it can transmit data in a bidirectional manner with roadside

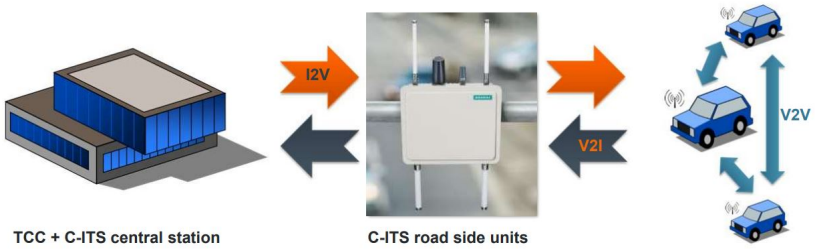

Figure 2.3: I-CTS structure [26]

units, depending on the needs of I2V or V2I. In the project ESRIUM, EGNSS positon will be received by the central station and the EGNSS-correction data will be sent to end-users to correct the vehicle position that the on-board ADAS system is based on, in order to ensure the high-precision positioning and that ADAS systems are working correctly. Besides, C-ITS will also provide routing recommendations to vehicles.

2.4 Motion Planning Algorithms

The core functions of autonomous vehicles include localization, mapping, perception, planning, vehicle control and human-machine interface, and each function needs the cooperation of software and hardware. An architecture of autonomous vehicle consisting of both software and hardware parts comprises these functions, and splits each function into smaller tasks. A general hierarchical scheme of autonomous vehicle can be found in [27], [28] and [29]. [30] provides a simplified architecture, dividing it in to three layers: (1) perception layer, (2) planning layer, and (3) trajectory control layer [31]. These three layers are sequentially connected in the order as mentioned, as shown in Fig.2.4. Among them, the planning layer is the core layer of the software architecture of AV. This layer is responsible to decide best driving behavior and generates a collision free local trajectory to follow at each time instant. There are three main features of the planning layer, (1)route planning, (2) behavior planning, and (3) motion planning (path planning and trajectory planning). A route can be defined as a trip from initial position to the final destination through the road network. Driving behavior planner decides safe driving actions and generates a safe, comfort and feasible trace to follow. A path is a sequence of way-points of independent attributes like position, orientation, linear velocity, angular velocity, acceleration, and steering angel etc, but without considering mechanical limitation of vehicle. While a trajectory can be seen as a sequence of feasible spatio-temporal states of vehicles (time varying way-points), taking kinematic constraints into account.

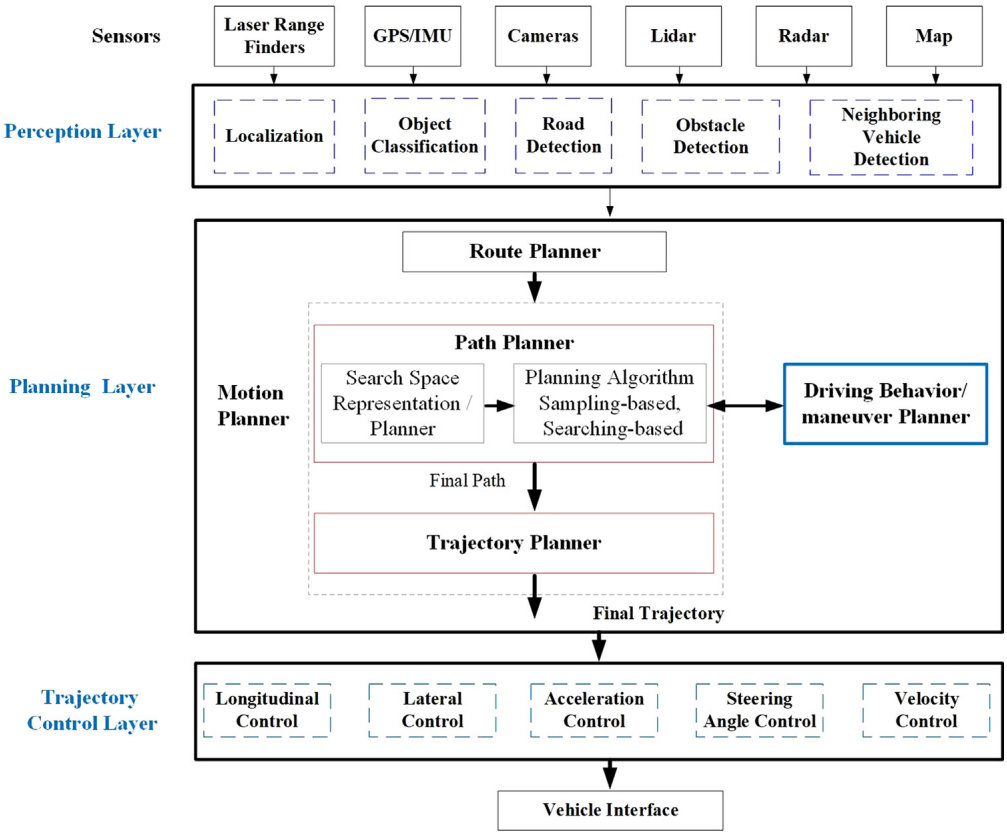

Figure 2.4: Software structure in autonomous vehicles [30]

The Motion Planner subsystem is responsible for computing a trajectory from the current self-driving car's State to the current Goal. The initial state and final goal can be defined according to demand, with the combination of independent attributes like position, orientation, linear velocity, and angular velocity etc. This trajectory must follow the Path defined by the Behavior Selector subsystem as closely as possible, while satisfying car's kinematic and dynamic constraints, and providing safety and comfort to the passengers. Several motion planning algorithms have been proposed in the literature. In this thesis, we will focus on the on-road motion planning aiming at planning trajectories that follow the route, which differs from unstructured motion planning, in which there are no lanes and, thus, trajectories are far less constrained [32].

Methods for motion planning can be mainly categorized into four classes: graph search based, sampling based, interpolating curve based, and numerical optimization based methods [33][34].

In motion planning, the basic idea of traveling from the current state to the current goal is to traverse a state space, which is often represented as an occupancy grid or lattice that depicts where objects are in the environment. Thus, graph searching algorithms can be implemented, visiting the different states in the grid, to find a solution or not to the planning problem [33]. A basic graph searching algorithm is Dijkstra Algorithm. It can find single-source shortest path in the graph. [35] implemented it in multi-vehicles simulations. Then an extension of Dijkstra Algorithm, called A* Algorithm, is developed to enable a fast node search applying heuristics. Several applications in mobile robotics have used A* Algorithm as basis for improvement, such as the dynamic A^* (D^*)[36], Field D^* [37], θ^* [38], Anytime repairing A^* (ARA^*) and Anytime D^* (AD^*) [39], among others. A hybrid A^* algorithm served as part of the DARPA Urban Challenge in the Standford automated Vehicle Junior [40]. State Lattice Algorithm is another representative graph search algorithm, which is based in local queries from a set of lattices or primitives containing all feasible features, allowing vehicles to travel from an initial state to several others. [41] decomposed environment in a local variable grid, depending on the complexity of the maneuver, and [42] used the spatio-temporal lattices to execute the graph searching.

Another motion planning method is sampling based. A sampling based planner tries to solve timing constrains in high dimensional spaces. The approach executes a random sampling in the state space, looking for connectivity inside it. The drawback of this method is that the solution is suboptimal. Rapidly-Exploring Random Tree (RRT) is a in motion planning field commonly used sampling based algorithm. It consider the non-holonomic constraints, such as maximum turning radius and momentum of the vehicle. MIT team uses RRT at DARPA Urban Challenge [43], but the resulting path is

not optimal, it's jerky and not curvature continuous. [44] developed a new approach, named RRT^*, to converge an optimal solution.

Interpolating Curve Planners take the advantage of Computer Aided Geometric Design(CAGD) in path smoothing, inserting a new set of data within the range of previously known set and generating a new set of data (a smoother path), to improve the trajectory continuity along with considering the vehicle constraints and the dynamic environment the vehicle navigates. Line and circles[45], Clothoid curves[46], polynomial curves[47], Bézier curves[48] and spline curves[49] are used to smooth the path.

Numerical Optimization methods aim to minimize or maximize a function subject to different constrained variables. [50] and [51] uses numerical optimization respectively to smooth the previously computed trajectories, and to compute trajectories for kinematic constrains.

Besides these traditional algorithms, various Machine Learning and Deep Learning Algorithms are more often used in Autonomous Driving Architectures for different tasks. In the field of motion planning, [52] employed the Deep Deterministic Policy Gradient (DDPG) algorithm for calculating the acceleration of the vehicle; [53] uses a deep-wide neural network, called ShufflePointNet, to learn local depictions for point cloud data; [54] applies a nueral network algorithm named Light Gated Recurrent Unit (Li-GRU) for trajectory estimation. Machine learning's capability to process large amounts of data improves the accuracy and efficiency of motion planning.

2.5 Reinforcement Learning

The human nature of learning is realized by interacting with our environment, which is also the basic idea of reinforcement learning(RL). The key point of reinforcement learning is learning how to map situations to actions, so as to maximize a numerical reward signal [55]. As a core component of artificial intelligent methods, RL techniques are being widely applied for designing artificial agents to mimic human tasks in playing games,objects/box grabbing and fetching, driving cars, and so on.

In the field of autonomous driving system, the domain of reinforcement learning has become a powerful learning framework now capable of learning complex policies in high dimensional environments, just as seen in an intersection scenario in dynamic environments and varying vehicles dynamics. [56] applied deep reinforcement learning algorithm(DRL) using a full-sized autonomous vehicle. DRL incorporates deep learning into the solution, allowing the agent to make decisions from unstructured input without manual engineering of the state space. The system is first trained in simulation and

then in real time using on board computers, and succeeded in the real-world task of following lanes. [57] encodes traffic rules as constraints of the optimization, building the RL model based on constrained policy optimization to improve traffic rule compliance of motion planners for autonomous vehicles. [58] extended reinforcement learning with a safety layer using set-based prediction to guarantee a safe autonomous lane changing. [59] builds a configurable reinforcement learning environment for motion planning of autonomous vehicles called CommonRoad-RL[60]. This platform greatly alleviates the work burden of the attempts to apply RL methods in motion planning problems.

2.6 Development Environment

The project's chosen environment is Conda in PyCharm. The first part of technical realization is creating the scenarios, inserting road damages and outputting XML files for reinforcement learning. The second part is setting up the reinforcement learning environment. We will use the tools from CommonRoad, a collection of composable benchmarks for motion planning on roads. It uses OpenAI-Gym environment as its reinforcement learning framework and builds an on-road autonomous driving environment. Gym is an open-source Python library for developing and comparing reinforcement learning algorithms by providing a standard API for communication between learning algorithms and environments, and a standard set of environments.

The training process can be directly implemented in CommonRoad-RL. An alternative choice is using Stable-baselines3. Stable-baselines3 is a collection of reliable implementations of reinforcement learning algorithms in PyTorch that are compatible with OpenAI-Gym environment. Stable baselines3 offers a training framework with different algorithm choices, the user only needs to build a custom environment specifying the problem.

3 Motion Planning Algorithms Benchmarking

Since the four types of motion planning algorithms mentioned in the previous chapter can already solve most of the motion planning problems in automated vehicles, what is the advantage of using a reinforcement learning algorithm in the road damage avoidance use case? This chapter will describe the concepts of four representative algorithms, A*, CL-RRT, Convex Optimization and Potential field, analyze their advantages and disadvantages generally and specifically in the road damage avoidance use case, and benchmark them with reinforcement learning.

3.1 A* Algorithm

A* Algorithm is a graph search based algorithm. It is the most popular search algorithm for path finding and an extension of Dijkstra's Algorithm and Greedy Best-First-Search.

The basic graph search algorithms are variants of Breadth-First-Search, which starts from some arbitrary node of a graph and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. We assume two-dimensional grids as the state space, where x and y indicate the horizontal and vertical positions. A frontier queue is built to store the grids as new starts for further exploration, and the reached grids will be stored in the reached dictionary to ensure no repeated exploration. The Best-First-Search algorithm idea is demonstrated below. The graph search algorithms vary the way the queue is used, switching from a first-in-first-out queue to a priority queue. In other words, before the search of the current depth is finished, Breadth-First-Search will always choose the grid of this depth to explore, even though new neighbors of deeper depth are added to the queue, while basic graph search algorithms will calculate the priority of the grids in the queue, and choose the grid with highest priority to explore.

Dijkstra's Algorithm works by visiting neighboring nodes in the graph starting with the object's starting point, checking if the visited one is the goal, and repeating the process from the not-yet-explored grids, to make sure all possible paths included with a certain number of iterations. It will expand outwards from the starting point until it reaches the goal. Dijkstra's Algorithm is guaranteed to find the shortest path from the starting point to the goal, as long as none of the edges have a negative cost. In Fig. 3.1, the pink square is the starting point, the blue square is the goal, and the teal areas show what areas Dijkstra's Algorithm scanned. The lightest teal areas are those farthest from the starting point, and thus form the "frontier" of exploration.

Algorithm 1 Best-First-Search algorithms

$frontier = Queue()$
$frontier.put(start)$
$reached = dict()$
$reached[start] = True$

while $not\ frontier.empty()$ **do**
  $current = frontier.get()$
  for $next\ in\ graph.neighbors(current)$ **do**
    $frontier.put(next)$
    $reached[next] = True$
  end for
end while

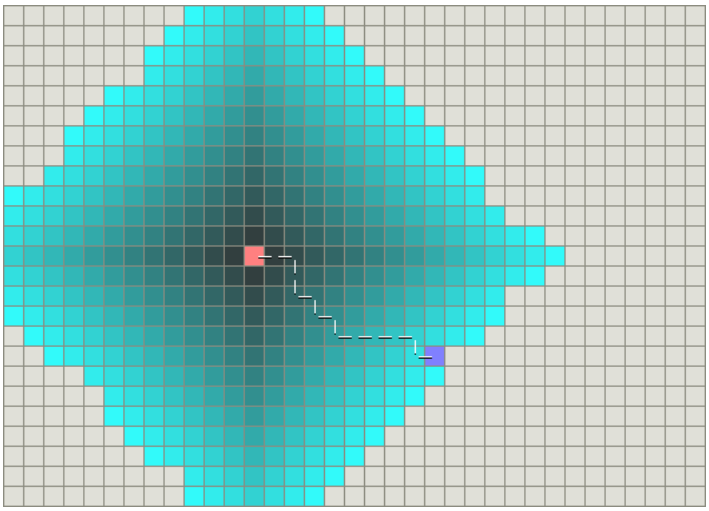

Figure 3.1: Path planning through Dijkstra's algorithm [61]

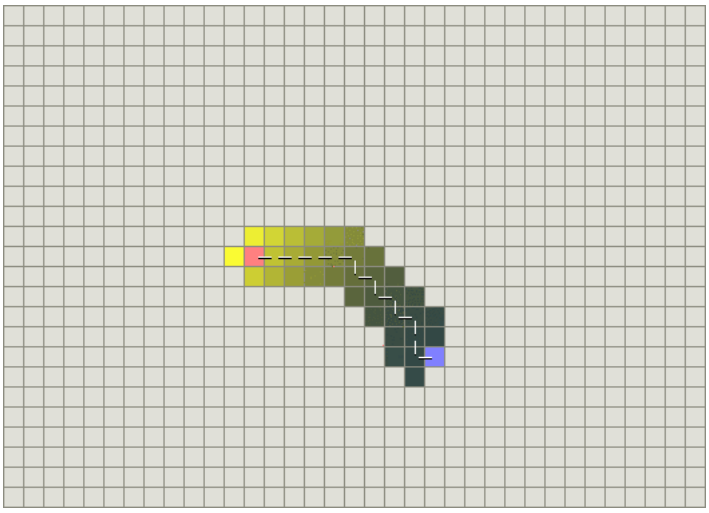

Figure 3.2: Path planning through Best-First-Search algorithm [61]

In the graph search problem, there are some things that we consider common sense, but the algorithms don't understand, for example, it takes longer to move from one state to another when the two states are getting farther apart; if the destination is to the east, it is more likely to find the best path by going east than by going west. And this information behind the relative relationship between the start and the goal can be used as an estimation, called a heuristic. And this approach is used in Greedy Best-First-Search algorithm. It estimates how far from the goal any grid is, and then in exploration, instead of selecting the closest grid, it selects the grid closest to the goal, which is represented with a low heuristic value. Greedy Best-First-Search is not guaranteed to find the shortest path. However, it runs much quicker than Dijkstra's Algorithm because it uses the heuristic function to guide its way toward the goal very quickly. Fig. 3.2 shows the result of the same demo used for testing Dijkstra's Algorithm. Yellow represents those nodes with a high heuristic value (high cost to get to the goal) and black represents nodes with a low heuristic value (low cost to get to the goal). Compared to the last Fig. 3.1, we can see that the scanned area of Greedy Best-First-Search is much less than Dijkstra's Algorithm, which means the computation time is also much less.

However, both of these examples illustrate the simplest case. When we introduce a concave obstacle in the grid map, Dijkstra's Algorithm takes even more time but is guaranteed to find the shortest path, as shown in Fig. 3.3; on the opposite, Greedy Best-First-Search takes less time but the result is not optimal, as shown in Fig. 3.4. The problem is that Greedy algorithms only consider the heuristic value, but ignores the cost of the path so far.

To combine the best of both, A^* was developed in 1968, combining heuristic ap-

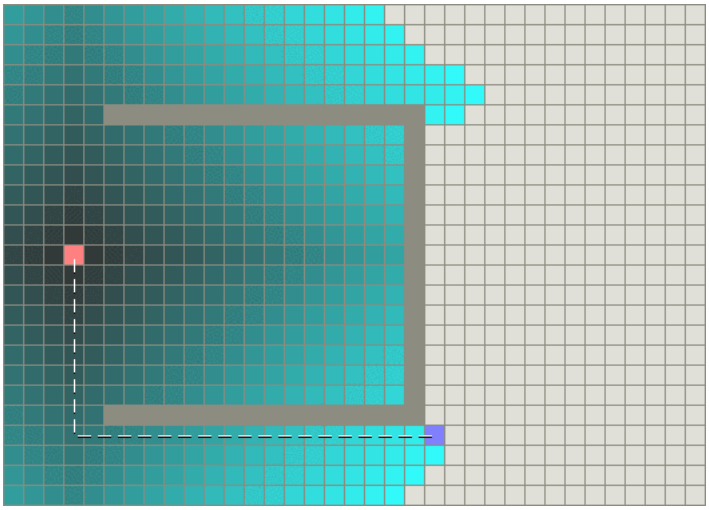

Figure 3.3: Dijkstra's algorithm with a concave obstacle [61]

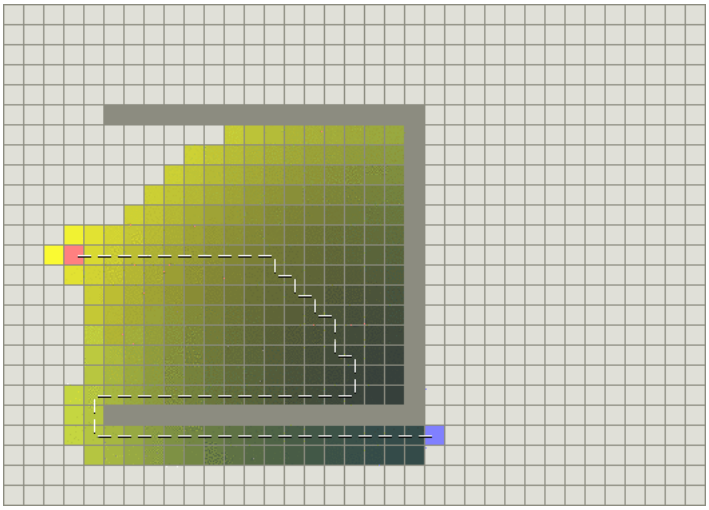

Figure 3.4: Best-first search algorithm with a concave obstacle [61]

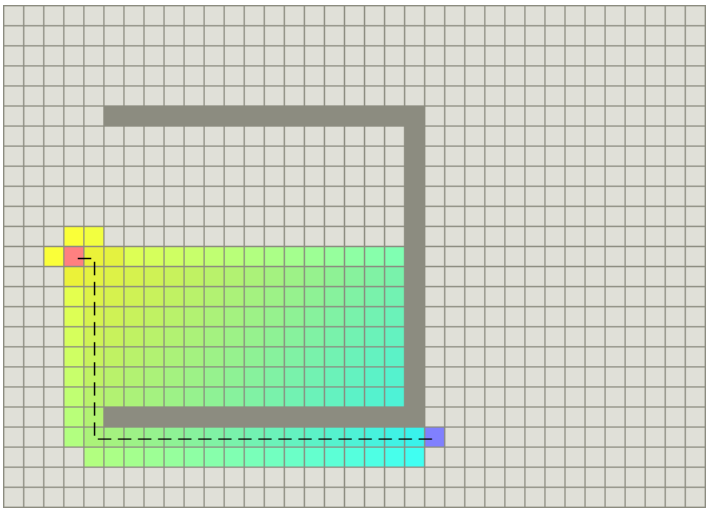

Figure 3.5: A* algorithm with a concave obstacle [61]

proaches like Greedy Best-First-Search, favoring grids that are close to the goal, and formal approaches like Dijkstra's Algorithm, favoring grids that are close to the start, which guarantees a shortest path and also a shorter time [61]. In the standard terminology used when talking about A^*, $g(n)$ represents the exact cost of the path from the starting point to any grid n, and $h(n)$ represents the heuristic estimated cost from grid n to the goal. In Fig. 3.5, grids with bigger h are further from the goal and more yellow, while grids with bigger g are further from the starting point and present more teal. A^* balances the two as it moves from the starting point to the goal. The priority of the queue is determined by the sum of these two parts. Each time through the main loop, it examines the node n that has the lowest $f(n) = g(n) + h(n)$.

From the lase examples we can see that the most important design aspect using A* Algorithm is the determination of the cost function, which defines the weights of the nodes. Even though $g(n)$ is easier to get, the designing of $h(n)$ significantly affects the efficiency of the algorithm. In the road damage avoidance case, the heuristic function should consider the different levels of severity of road damages and also different shapes and distributions of road damages, along with the traffic rules, which all makes the design of a general heuristic function a hard work.

3.2 Closed-loop rapidly-exploring random tree

Rapidly-exploring random tree is a sample-based algorithm. Imaging a two dimensional environment with obstacles, the RRT algorithm starts by choosing a random point within the environment, then creates a line between that new point and the start node

and place a new point at a distance of delta from the start along that line, as long as it does not result in any collisions with the obstacles. In each iteration of the algorithm we follow the same procedure to add new points and expand the tree. Each time a new point is created, an edge is created to the closest node within the tree. The tree continues to expand into the environment following the pattern. Occasionally instead of choosing a random point within the graph, the goal node is chosen in order to increase the chances of finding it. Once the tree reaches the goal node, we can return the path from start node to the goal node. The algorithm is shown in Fig. 3.6, the red nodes represent obstacles.

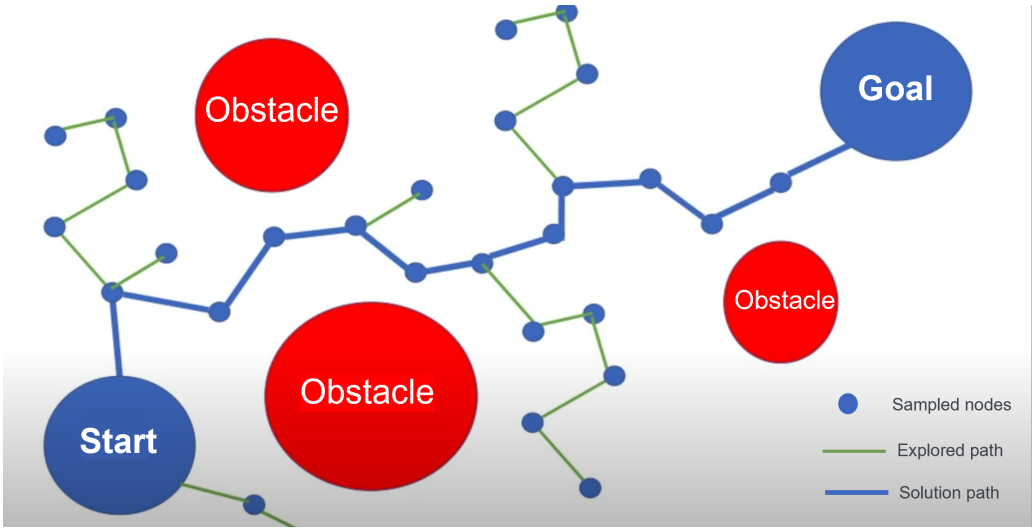

Figure 3.6: Rapidly-exploring random tree [62]

With the constraints existing in automated driving, CL-RRT is developed to consider the vehicles' feasibility. Closed-loop rapidly-exploring random tree(CL-RRT) is an extension of RRT that samples an input to a stable closed-loop system consisting of the vehicle and a controller. In the algorithm, an input to the controller is sampled, followed by the forward simulation using the vehicle model and the controller to compute the predicted trajectory. The process is shown in Fig. 3.7. In the demonstration, the red straight lines show reference paths as a result of rapidly-exploring random tree, as the paths in Fig. 3.6, while the green curves consider the constraints of vehicles based on the reference paths, so they are more feasible and look less stiff. The idea of planning with closed-loop prediction allows us to handle complex unstable dynamics and avoids the need to find computationally hard steering procedures. But the result of the sample-based algorithms depends on randomness, and when searching the entire solution space, it may lead to high computational costs.

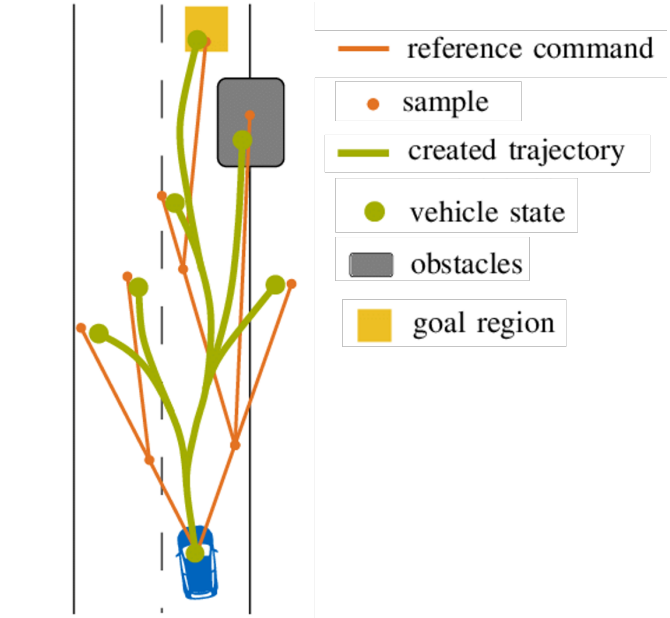

Figure 3.7: CL-RRT [63]

3.3 Potential Field

A potential field is any physical field that obeys Laplace's equation. Electrical, magnetic, and gravitational fields are all potential fields. A potential field algorithm uses the artificial potential field to regulate a robot in a certain space. The basic idea is to have the robot attracted to the goal and repelled from the obstacles. An artificial potential field can be generated using the potential field functions, creating attractive fields and repulsive fields, and the robot, which can be assumed as a ball, will simulate from the highest potential to the lowest potential as shown in Fig. 3.8. In the demonstration, the potential of each point is represented by colors and height. Red and high points have higher potential and construct the repulsive fields, while blue and low points have lower potential and construct the attractive fields.

The potential of each point is sum of the attraction force and the repulsion force at the point. The attraction force comes from the goal node and is inversely proportional to the distance from goal node. The repulsive forces come from boundaries and obstacles and keep the robot away from them.

Potential field algorithm is traditionally used in path planning and obstacles detection. It's quick and straight forward. A challenge for potential field is the local minima trap issue as shown in Fig. 3.9, which may lead to a local optimal result but not a global one. With the complexity of the environment rising, the potential field would be more complicated and reaction forces may cancel each other, which may creates more local

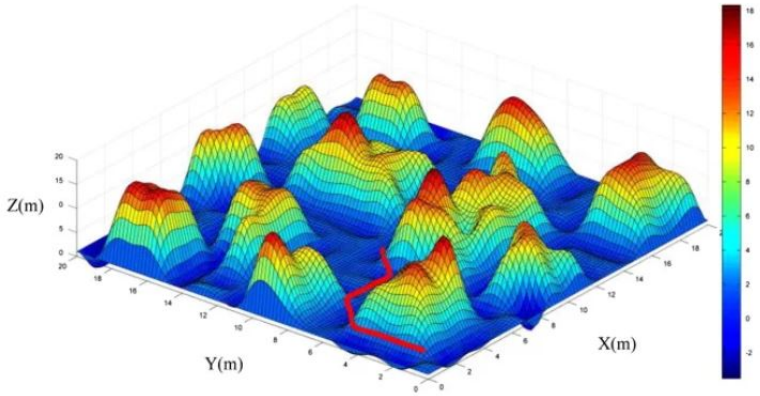

Figure 3.8: Potential field with obstacles [64]

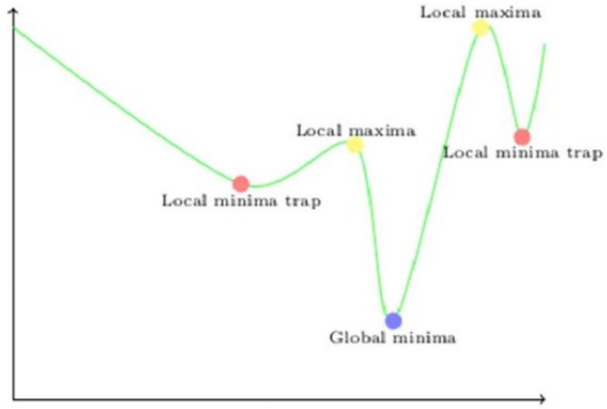

Figure 3.9: Local minima trap [64]

minima trap. Moreover in the scenario of autonomous driving, the vehicle feasibility is hard to be taken into consideration.

3.4 Convex Optimization

Convex optimization is a subfield of mathematical optimization that studies the problem of minimizing convex functions over convex sets. A function is called convex if the line segment connecting any two distinct points on the graph of the function lies above the graph between the two points. A set is convex, if given any two points in the set, the set contains the whole line segment connecting these two points. A function is convex if the set of points on or above the graph of the function is convex [65]. The abstract nature of the method enables the algorithm widely used in many problems. The key

idea of convex optimization is to formulate the problem into a convex optimization problem, as long as the problem is correctly built, the computation of the result will be simple. The standard form of convex optimization is shown below.

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \cdots, m \\ & h_i(x) = 0, \quad i = 1, \cdots, p \end{array}$$

x is the optimization variable; The objective function f is a convex function; The inequality constraint functions g_i are convex functions; The equality constraint functions h_i are affine transformations, that is, of the form: h_i (x)= $a_1 \cdot$ x- b_i ,where a_i is a vector and b_i is a scalar.

In the field of motion planning, convex optimization is usually used to smooth the trajectory, with the vehicle feasibility taken into account. In [66], given a reference trajectory, a convex set could be built as a collision-free elastic band, in which new waypoints can be set to create a smoother trajectory. The constraints for the placement of the new waypoints rely on a number of approximations to ensure convexity of the optimization problem. An application of the convex optimization in motion planning is shown in Fig. 3.10. The dashed line represents the reference path that goes through the map with obstacles without collision. It's stiff and not feasible for vehicles. The solid line is the smooth result of the convex optimization considering the kinematic constraints based on the reference path.

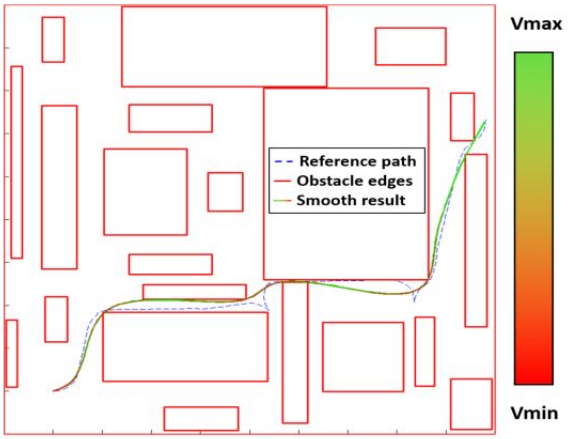

Figure 3.10: Using convex optimization for trajectory smoothing [66]

3.5 Benchmark

After having a theoretical understanding of the algorithms, the benchmarking focusing on our use case can be implemented. Five benchmarks are listed: completeness of the solution, optimality, kinematic feasibility, and computation speed.

Completness of the solution

The completeness of the solution depends on if the algorithm can solve the problem independently. Convex optimization and potential field are usually used based on a reference path. In our use case of road damage avoidance, the motion planning problem is hard to be written into a convex optimization problem or a potential field form. Therefore, if we need to use these two methods, another motion planner needs to calculate a solution as a reference first. As for A* search and reinforcement learning, the problem can be easily formulated. The different severity levels can be manifested in the heuristic function in A* search and reward function in reinforcement learning. In the case of CL-RRT, it is also used to optimize the vehicle dynamics in motion planning, but it also can find the trajectory in our scenario without a reference. While the road damage severity may be neglected.

Optimality

As convex optimization and potential field cannot find the solution independently, the optimality of the solution is locally optimal depending on the optimality of the reference path. As for the other three methods, global optimality can be ensured.

Kinematic feasibility

In A* search, the agent's actions can be defined under the constraints of vehicle kinematic feasibility. CL-RRT is an extension of the original RRT algorithm, but takes kinematic feasibility into account. It can be partially satisfied in convex optimization depending on which parts of the kinematics can be formulated into a convex problem. The potential field can hardly reach the benchmark. By introducing the vehicle model into the reinforcement learning environment, the kinematic feasibility can be satisfied.

Computation speed

In A* search method, each action in each frontier state needs to be explored and the cost and heuristic are meanwhile calculated. In our use case, the actions including acceleration and steering, the motion primitives are huge. In addition, the road damage severity level would complex the heuristic function. Therefore the computation speed of A* search in our use case would be very slow. CL-RRT benefits from sampling, the speed would be faster than A*, but considering the kinematic feasibility still slow down

Table 3.1: Benchmark of Motion Planning Algorithm.

Algorithm	Completeness	Optimality	Kinematic feasibility	Computation speed
A* Search	Solution complete	Global optimality	Yes	Very Slow
CL-RRT	Solution complete	Global Optimality	Yes	Slow, but it depends on randomness.
Potential Field	No	Local Optimality	No	Quick
Convex Optimization	No	Local Optimality	Partially	Quick, but a convex corridor must be provided by a global planner
RL	Solution complete	Global optimality	Yes	Quick

the computing. For convex optimization and potential field, as long as the problem is successfully formulated in the correct form, the computation would be fast. As for reinforcement learning, it takes time to train, while once the hypermeters are optimized, the model can be used in new scenarios and the computation speed would be fast.

To summarize the algorithms mentioned above and compare them with reinforcement learning approach in the road damage avoidance problem, the benchmark is shown in Table 3.1.

From the analysis above, it can be seen that due to the increased complexity of the obstacles(road damage) and the limitation of the vehicle's kinematic feasibility, reinforcement learning has advantages over conventional motion planning algorithms in our use case.

4 Reinforcement Learning Concept

In Chapter 2, the basic idea of Reinforcement Learning is briefly mentioned, i.e., learning how to map situations to actions, and how to maximize a numerical reward signal, and introduced its applications and relevant research. In this chapter, the fundamental elements of reinforcement learning, the mathematical essence, and the different algorithms will be described in detail.

4.1 Fundamental Principles

As a part of artificial intelligent learning methods, reinforcement learning focuses much more on goal-directed learning from interaction comparing with other learning methods. The intrinsic of reinforcement learning is learning from the interaction with the environment. The learner isn't been told which action to take, but instead must discover which actions yield the most reward by trying them. Furthermore, the action would not only lead to effects on the immediate reward but also all subsequent rewards. These two characteristics—trial-and-error search and delayed reward—are the two most important distinguishing features of reinforcement learning [55].

The major difference between Reinforcement Learning and Supervised Learning is that supervised learning relies on a labeled training set and then predicts the extrapolation of an unlabeled result, usually for classification problems, whereas interaction with the environment is difficult to actually sample and label, and the learner must learn from its own experience. RL and unsupervised learning have in common that they do not rely on labeled examples, but unsupervised learning focuses on finding hidden structure, while RL focuses on maximizing a reward signal. Therefore, we consider reinforcement learning to be a third machine learning paradigm, alongside supervised learning and unsupervised learning.

Components of RL system

The learner is also called the agent in a reinforcement learning system. The agent and the environment are the two main roles in a RL problem, and the agent must be able to sense the state of its environment to some extent and must be able to take actions that affect the state. The agent also must have a goal or goals relating to the state of the environment.

Beyond the agent and the environment, one can identify four main subelements of a reinforcement learning system: a policy, a reward signal, a value function, and, optionally, a model of the environment.

A policy π defines the learning agent's way of behaving at a given time. Roughly speaking, a policy $\pi : S- > A$ is a mapping from perceived states to actions the agent needs to take in those states. It's an imitation of the human being's stimulus-response rules or associations. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior. In general, policies can be deterministic:

$$\pi(s) = a \tag{4.1}$$

or stochastic, specifying probabilities for each action:

$$\pi(a|s) = Pr(a_t = a|s_t = s) \tag{4.2}$$

A reward signal defines the goal of a reinforcement learning problem. At each time step, the environment sends to the reinforcement learning agent a single number called the reward. The reward signal of the instant time step t can be marked as r_t. The agent's sole objective is to maximize the total reward it receives over the long run. The reward signal thus defines the good and bad events for the agent. In a biological system, we might think of rewards as analogous to the experiences of pleasure or pain. They are the immediate and defining features of the problem faced by the agent. The reward signal is the primary basis for altering the policy; if an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in that situation in the future. In general, reward signals may be stochastic functions of the state of the environment and the actions taken.

Differing from a reward signal indicating a good event in an immediate sense, a value function specifies what is good in the long run. The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. The value function needs to consider not only the immediate reward, but also the states that are likely to follow and the rewards available in those states. If a reward for a state is low, but the following states yield high rewards, the value of the state might still be high. It shows the long-term desirability of the state to the goal. Values are like a more refined and farsighted judgment in the long term of the situation humans are in, which means both immediate reward and future expectation should be taken into

account. A value function V^{π} can be constructed as an expected discounted sum of future rewards under a particular policy π:

$$V^{\pi}(s_t = s) = E_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | s_t = s] \tag{4.3}$$

Discount factor γ weighs immediate and future rewards. The value function can be used to quantify goodness/badness of states and actions, and decide how to act by comparing policies.

Even though that the value is an accumulation of rewards, it is still the primary indicator of making and evaluating decisions. Action choices are made based on value judgments. The process of reinforcement learning is actually seeking actions that bring about states of highest value. Unfortunately, it is much harder to determine values than to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and re-estimated from the sequences of observations an agent makes over its entire lifetime. In fact, the key point to solve a reinforcement learning problem is to efficiently estimate values.

A model of the environment mimics the behaviour of the environment, represents how the world changes in response to agent's action. For example, given a state and action, the model might predict the resultant next state and next reward. The transition/dynamics model predicts the probability of the possible next agent state:

$$p(s_{t+1} = s' | s_t = s, a_t = a) \tag{4.4}$$

and reward model predicts immediate reward, where E means the Expectation of all possible rewards:

$$r(s_t = s, a_t = a) = E[r_t | s_t = s, a_t = a] \tag{4.5}$$

Models are used for planning, by which we decide a course of actions considering possible future situations before the agent actually experience them. Methods for solving reinforcement learning problems that use models and planning are called model-based methods, as opposed to simpler model-free methods that are explicitly trial-and-error learners.

As model-based and model-free methods distinguish different reinforcement agents, the existence or absence of value function and policy divided RL agents into more types, as shown in Fig.4.1, in which actor critic means, given a state, the value and the policy will be explicit.

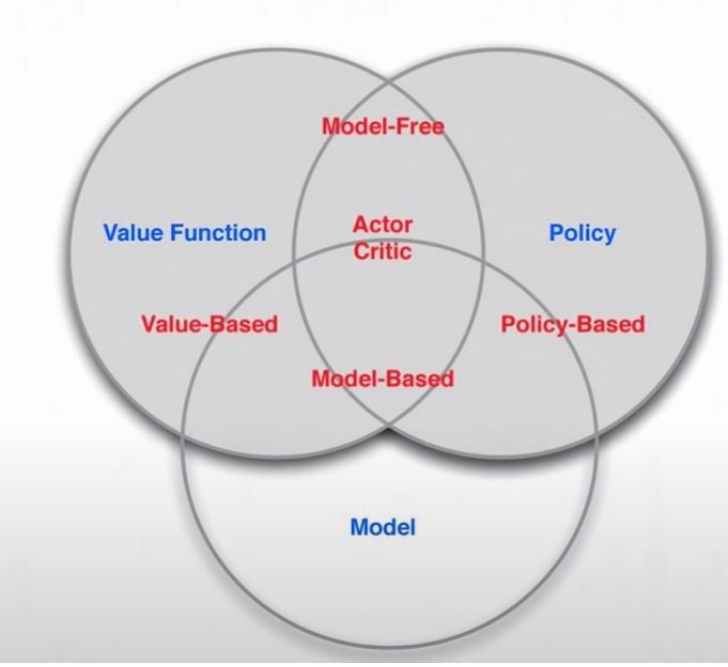

Figure 4.1: RL Agent classification [67]

Exploration and Exploitation

One of the challenges in reinforcement learning is the trade-off between exploration and exploitation. Since the agent learns from interaction with environment, it must take actions and then get to know the rewards. To get more rewards, the agent always faces to a dilemma: exploit what it has experienced, or explore other possibility in order to make better action selections in the future. None of them could be applied alone to succeed at the task. The agent must try a variety of actions and progressively favor those that appear to be the best.

4.2 Markov Decision Process

The process of reinforcement learning can be described as the agent learning to make good decisions. There are two types of sequential decision processes: Bandits and Markov Decision Process (MDP)/Partially observable Markov decision process(POMDP) [67]. In Bandits, actions have no influence on next observation/state and no delayed rewards, while in MDP/POMDP, actions will influence future observation/state, which corresponds to the feature of reinforcement learning.

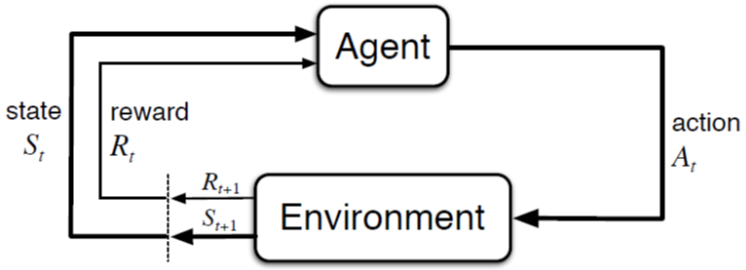

Figure 4.2: Interaction between environment and agent [55]

Markov decision process (MDP) is a discrete-time stochastic control process, an extension of the Markov chains, a stochastic model developed by the Russian mathematician Andrey Markov, describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

A framework for reinforcement learning problem can be summarized as a continual interaction between the agent and the environment, where the agent selects actions and the environment responds to these actions, presents new situations to the agent and gives rewards. The framework is shown in Fig. 4.2.

In detail, the process can be described as follows: the agent and the environment interact along discrete time steps, $t = 0, 1, 2, 3, ...$At each time step t, the agent receives a representation of the states of the environment S_t, based on the states an action or actions A_t are selected. One time step later as the consequence of this action, the agent receives a reward R_{t+1} , and at the same time the new states R_{t+1}. The whole process is based on a premise that the process is consistent with the Markov decision process.

As mentioned in Chapter 3, graph search methods are commonly used in motion planning also as a goal-directed approach. Then what are the advantages of MDP compared to search methods? We can compare the frameworks of both and better understand MDP. The major difference between search problems and MDP is that, search algorithms do not consider the uncertainty in the real world, which means, given a state and an action, the succeed state is deterministic. The uncertainty is taken into account in MDP as the transition probabilities $T(s, a, s')$, specifying the probability of ending up in state s' if taken action a in state s. For each state s and action a:

$$\sum_{s' \in States} T(s, a, s') = 1 \tag{4.6}$$

Along with the transition probabilities $T(s, a, s')$, $reward(s, a, s')$ is introduced in MDP

for the $transition(s, a, s')$ instead of a $cost(s, a)$ in search problems and the discount factor is also chosen for future rewards and calculating values. The discount factor shows how much we care about future. When the discount factor equals 0, MDP becomes a greedy approach.

The result for search problems is a path, in other words, a sequence of actions. And for MDP, the result is a policy, a mapping from each state to an action. The different feature of solutions enables MDP to deal with dynamic environment. The criteria to estimate the policy is utility, the discounted sum of the rewards on the path. But utility is a random quantity, therefore, value is introduced to represent the expected utility. The relationship between a state-action-value-function $Q_\pi(s, a)$, or so called Q-Value and value is described below:

$$V_\pi(s) = \begin{cases} 0 & if ls End(s) \\ Q_\pi(s, \pi(s)) & Otherwise \end{cases} \tag{4.7}$$

$$Q_\pi(s, a) = \sum_{i=1}^{n} T(s, a, s')[Reward(s, a, s') + \gamma V_\pi(s')] \tag{4.8}$$

The key idea to get an optimal policy is the iterative algorithm. The states start with arbitrary policy values and repeatedly apply recurrences to converge to true values. The policy evaluation algorithm is shown below:

Algorithm 2 policy evaluation

for all state s **do**
  $V_\pi^{(0)}(s) \leftarrow 0$
end for
for t in $1, \cdots, t_{PE}$ **do**
  for each s **do**
    $V_\pi^{(t)}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[Reward(s, \pi(s), s') + \gamma V_\pi^{t-1}(s')]$
  end for
end for

Reinforcement learning is based on MDP, but with unknown transition probabilities and unknown reward function.

4.3 Methods and Algorithms

There are two types of reinforcement learning: Tabular solution methods and approximate solution methods. Tabular solution methods describe almost all the core ideas of reinforcement learning algorithms in their simplest form: that in which the state and action space are small enough for the approximate value functions to be represented as arrays, or tables. In this case, the methods can often find exact solutions- the optimal value function and the optimal policy. Approximate solution methods are usually applied in larger problems, and can only find approximated solutions. It can be seen as a combination of reinforcement learning and an existing generalization method(function approximation) [55]. Supervised learning algorithms are applicable for approximating functions within reinforcement learning, as shown in Deep Reinforcement learning as example. Since approximate solution methods are extension of tabular solution, this section will start with the core methods in reinforcement learning and then explain some representative algorithms. According to the classification of reinforcement learning agent in Fig.4.1, this section will start with model-based and model-free methods, then introduce value-based, policy-based, and actor-critic methods. Finally, deep reinforcement learning is discussed as an example only for approximate solution.

4.3.1 Model-based and Model-free methods

If the method is based on model or not distinguishes the reinforcement learning methods into two classifications. This subsection will discuss a model-based method-dynamic programming, and two model-free methods-Monte-Carlo method and temporal-difference method.

Dynamic Programming

Dynamic Programming (DP) refers to a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a Markov decision process(MDP). Other reinforcement learning algorithms can be viewed as attempts to achieve the same effect as DP, only with less computation and without assuming a perfect model of the environment [55].

The key idea of DP, and of Reinforcement Learning generally, is the use of value function to organize and structure the search for good policies. A complete dynamic programming process consists of policy evaluation, policy improvement and policy iteration.

Policy evaluation is the process to compute the state-value function V_π for an arbitrary policy π. The initial approximation, V_0, is chosen arbitrarily, and each successive approximation is obtained by using the Bellman equation for V_π as an update rule[55]:

$$\begin{aligned} v_{k+1}(s) &\doteq E_\pi[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')] \end{aligned} \tag{4.9}$$

In the equation, k represents the iteration index of the value function. The calculated new value function is iterated again until it converges. Converging means that the difference between the new value and the last value is below a small threshold.

Policy evaluation is used for policy improvement, which means to find a better policy. Assume there are two policies π and π', when in every state s,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \tag{4.10}$$

then,

$$v_{\pi'}(s) \geq v_\pi(s) \tag{4.11}$$

In this situation, the new policy π' is as good as, or even better than the old policy π. The policy evaluation and policy improvement will repeat as the process of policy iteration, until the final policy is fund. The policy iteration process is shown below:

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdot \xrightarrow{I} \pi_* \xrightarrow{E} v_* \tag{4.12}$$

where $\xrightarrow{E}$ represents policy evaluation, and $\xrightarrow{I}$ represents policy improvement.

Monte-Carlo Method

DP is based on a perfect MDP model, which is not realistic in real-world reinforcement learning problems, because it's impossible to obtain the whole environment as a model. In this situation, Monte-Carlo method, which only learns from experience-sample sequences of states, actions, and rewards from actual or simulated interaction with an environment, is practical way.

Monte Carlo methods for RL are based on averaging sample returns. So to ensure that the well-defined returns are available, we assume experience is divided into episodes,

and that all episodes eventually terminate no matter what actions are selected. For a given policy π, within an episode a state s appears one or more times. The appearance of the state is called a visit. The value function $V_\pi(s)$ is tuned by averaging the returns G_t of all visits from the state s. When only the first visit from the s within an episode is used, the method is called first-visit-MC. If all visits from s are used, the method is then called every-visit-MC [55].

Due to the environment model is not complete, we use State-Action-Value-Function $q_{pi}(s, a)$ instead of State-Value-Function $V_\pi(s)$ in policy evaluation and policy improvement [55]:

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdot \xrightarrow{I} \pi_* \xrightarrow{E} q_* \tag{4.13}$$

The evaluation process is identical with it in DP, the policy improvement in MC will use a greedy approach in Q-value, which means in each state s, the action a with highest Q-value will be chosen:

$$\pi(s) \doteq arg\max_a q(s, a) \tag{4.14}$$

In the MC method, policy evaluation and policy improvement are performed only at the end of an episode. Accordingly, the method is an episode-by-episode method, not a step-by-step method. An every-visit MC can be represented by the following equation:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{4.15}$$

S_t : non-terminal states under policy π
V : Estimate of V_π
G_t the averaged return at time t
α : Constant for step sizes

Monte-Carlo methods can also be roughly divided into model-based MC and model-free MC. Model-based MC can estimate the transition function T and reward R in the frame of the MDP, while model-free MC tries to estimate Q_{opt} directly, as shown above.

Temporal-Difference Learning

As DP and MC methods both have their own advantages and disadvantages, the temporal-difference learning can combine the pros of both. TD learns the policy from experience, like in MC, but like in DP, it doesn't need to wait till the end of an episode.

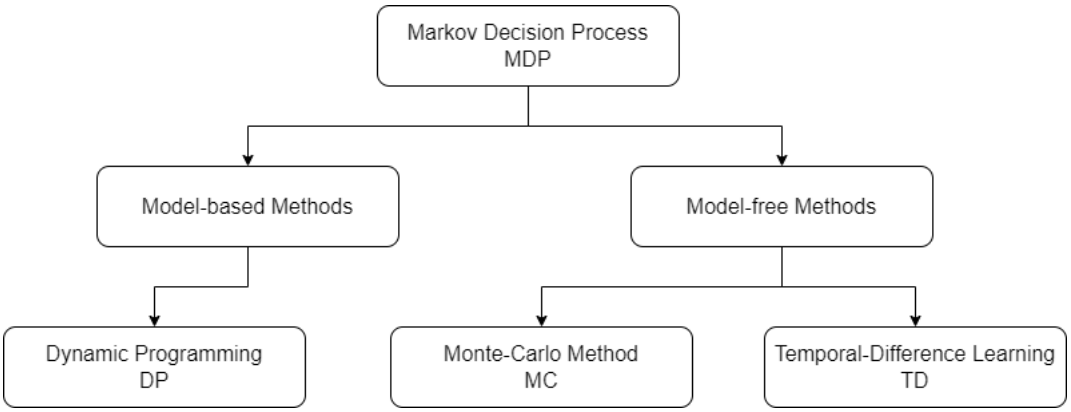

Figure 4.3: Relationship between DP,MC and TD methods

This is realized by performing the so-called bootstrap method. That is, compared to Eq. (4.15), the TD method only has to wait until time $t+1$ to update the estimate $V(S_t)$:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{4.16}$$

Where the components in square brackets $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called TD-Error δ_t. Comparing with Eq. (4.15), it can be seen that the update target of the value function is changed from G_t to $R_{t+1} + \gamma V(S_{t+1})$. I.e., no longer waiting until the end of an episode for an update. TD-Error denotes the extent to which the estimate $V(S_t$ deviates from the expected value $R_{t+1} + \gamma V(S_{t+1})$.

Since the procedure considers only the first future time point, it is also called TD (0). If the procedure considers n time points (n>1), the procedure is called n-step TD, Eq. (4.16) is then extended accordingly:

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha[G_{t:t+n} - V_{t+n-1}(S_t)], 0 \leq t < T \tag{4.17}$$

The Fig. 4.3 shows the relationship between DP, MC and TP.

4.3.2 Value-based methods

Even though the ultimate goal of reinforcement learning is to find the optimal policy, the value function or Q-value function play an important role. As the methods mentioned above, the key point of the problem is to find the optimal value function, and the policy derived from the value function. These type of methods are called value-based

methods, which can also be divided into two sub-types: on-policy methods and off-policy methods.

In reinforcement learning, a policy with randomness is needed to explore the environment, and collect samples. Randomness is needed because until the optimal policy $\pi*$ is found, the current policy π is still not the best. The action specified by it is also not optimal. Therefore, the algorithm must also try with the actions that are not optimal from the point of view of current policy. This is called exploration. A deterministic policy, as represented by Eq. (4.18), is a "greedy" method on Q-value. A stochastic policy can be realized by adding probability parameter ϵ:

$$V_\epsilon(s) \doteq \begin{cases} arg\max_a q(s,a), & \text{with probability 1-}\epsilon \\ \text{random action,} & \text{with probability}\epsilon \end{cases} \tag{4.18}$$

Off-policy methods use two policies, the behavior policy b and the target policy π, to separate sample collection from actual learning process. The behavior policy is specifically responsible for collecting data, with some randomness there is always some probability of selecting the potentially optimal action. The target policy improves steadily with the help of the samples collected by the behavior policy and the policy improvement strategy, eventually becoming the optimal policy.

A representative off-policy method is Q-Learning, its algorithm is shown below:

Algorithm 3 Q-Learning

Initialize $Q(s,a)$ arbitrarily
for each episode **do**
  Initialize s
  for each step of episode **do**
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_\alpha Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$
  end for
end for

In comparison with off-policy procedures, there is only one policy in the on-policy procedure, this policy is used as behavior and target policy at the same time. SARSA is a successful example of on-policy methods. The name SARSA comes from the elements considered in one iteration, s, a, r, s', a'. The algorithm is shown below:

Algorithm 4 SARSA

Initialize $Q(s, a)$ arbitrarily
for each episode **do**
  Initialize s
  Choose a from s using policy derived from Q
  for each step of episode **do**
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_\alpha Q(s', a') - Q(s,a)]$
    $s \leftarrow s'; a \leftarrow a'$
  end for
end for

4.3.3 Policy-based methods

Beside the value-based methods, where the policy is derived from Q-value, there are also policy-based methods, where the agent learn a parameterized policy directly through the interaction with the environment. In policy-based methods, the parameters θ of policy are introduced, along with a value-function, to be used to learn θ. The probability that action a is performed at time t under state s with parameter θ can be described by the following equation:

$$\pi(a|s, \theta) = Pr(a_t = a|s_t = s, \theta_t = \theta) \tag{4.19}$$

To learn the parameters θ, a scalar performance measure $J(\theta)$ is used. The goal of a policy-based method is to maximize the performance measure. This can be achieved by different approaches, for example, the policy gradient (PG) method, where the gradient ascent is used to adjust the $J(\theta)$:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)} \tag{4.20}$$

$\widehat{\nabla J(\theta_t)}$: a stochastic estimate whose expectation approximates the gradient of the performance measure with respect to its argument θ_t
α: Learning rate

The definition of $J(\theta)$ is different, depending on whether it is episodic task or continuous task. For episodic task, $J(\theta)$ is defined like this:

$$J(\theta) \doteq v_{\pi_\theta}(s_0) \tag{4.21}$$

Then the policy gradient theorem is like:

$$\nabla J(\theta) \propto \sum_{s} \mu(s) \sum_{a} q_\pi(s,a) \nabla \pi(a|s,\theta) \tag{4.22}$$

μ:	On-Policy distribution
π:	Policy with parameter θ
$\propto$:	proportional

It can be seen that the gradient of $J(\theta)$ depends on partial derivatives with respect to the components θ. On-policy distribution μ indicates the occurrence frequency of a state. If the agent follows policy π, this distribution can be replaced by an expected value:

$$\nabla J(\theta) = E_\pi[\sum_{a} q_\pi(s,a) \nabla \pi(a|s_t,\theta)] \tag{4.23}$$

In the equation above, the part $\pi(a|s_t,\theta)$ can be further reformed:

$$\begin{aligned} \nabla J(\theta) &= E_\pi[(\sum_{a} \pi(a|S_t,\theta) q_\pi(s,a) \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}] \\ &= E_\pi[(\sum_{a} q_\pi(s_t,a_t) \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}] \\ &= E_\pi[G_t \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}] \end{aligned} \tag{4.24}$$

Then we replace the $\nabla J(\theta)$ in Eq. 4.20 with Eq. 4.24:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \tag{4.25}$$

The equation shows, that each update of θ_t is proportional to the G_t and the vector $\frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$, which is the ratio between the gradients of the probability of selecting actions A_t and the probability itself. The ratio points out a direction for the parameters' update. The step size of the update $\theta_{t+1} - \theta_t$is proportional to G_t, meaning that one can get multiple rewards in this direction. Step size is inversely proportional to the probability $\pi(A_t|S_t,\theta)$ to avoid repeating non-optimal actions. This algorithm is called

REINFORCE. It is a Monte-Carlo method because the reward G_t is only available after the end of an episode.

An obvious drawback of the REINFORCE algorithm is that the variety of rewards sum Gt is too big. If we assume that there are 2 states and 6 actions in the environment, then there will be 6 rewards corresponding 6 state-action pairs. That means in each time step there will be 6 possibilities of reward. When the number of time steps grows, the range of the possibilities will further grow, which makes the calculation of G_t very slow.

There are two solutions to avoid a huge variety. The first one uses the baseline $b(s)$ as the basis of value function. The Eq. 4.22 then can be reformed into:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla \pi(a|s,\theta) \tag{4.26}$$

If we choose the average value of the rewards in one state, then the average value of $(q - b)$ will be zero, then the variety will be reduced.

The second solution to avoid the accumulation of the rewards is to consider the reward of only the next step or the rewards of a few next steps. This method uses the concept of TD, and it's called Actor-Critic(AC).

4.3.4 Actor-Critic methods

The key idea as we mentioned in section 4.1 is the combination learning of policy and state-value-function. Although the REINFORCE algorithm with Baseline learns the policy and at the same time a state-value-function $b(s)$, the value function isn't really engaged in the Bootstrap process, so it's not an actor-critic method.

The actor-critic method can use value-function with Bootstrap(Critic) and TD methods, then the update of the parameters can be reformed as below:

$$\begin{aligned}
\theta_{t+1} &= \theta_t + \alpha(G_{t:t+1} - \hat{v}(S_t, w)) \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \\
&= \theta_t + \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1}, w)) \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \\
&= \theta_t + \alpha \delta_t \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}
\end{aligned} \tag{4.27}$$

where a 'critic' learns the value-function to lead the 'actor'. The 'actor' improves the policy. Depending on whether the behavior and target policies are the same or different, the actor-critic procedure can be on- or off-policy.

4.3.5 Deep Reinforcement Learning

Deep reinforcement learning is a representative approximate solution for bigger problems. It combines the deep neural network to approximate the environment. As we explained above, a reinforcement learning method can be used with value-based or policy based principles. The successful DRL approaches with Value-based principle are e.g. Deep Q Network DQN and Double Q Network DDQN, with Policy-based principle are e.g. Proximal Policy Optimization Algorithms PPO and Trust Region Policy Optimization TRPO. The approaches that use both ideas (Actor-Critic) are e.g. Deep Deterministic Policy Gradient DDPG and Soft Actor-Critic SAC.

5 Technical realization

5.1 Scenario Design

Although the environmental factors are more complex in urban streets, the driving speed is lower and the driver has enough time to flexibility deal with road damage. However, on a highway the driver does not have enough time to react in a safe and reasonable manner. Therefore, this work regards the application scenario on the highway.

Considering the possibility that the self-driving vehicle is driving in the rightmost lane, and it cannot avoid road damage without lane changing, a second lane adjacent to the rightmost lane is designed in the scenario too. Also, to make the model more general, some curvature is added to the lanes.

By checking the Austrian highway road standards, we acknowledged that the rightmost lane is usually 3.75m wide. In a two-lane highway, the other lane is also 3.75m wide; in a three-lane freeway, the other two lanes are both 3.5m wide. Since we introduce the second lane only to satisfy the possibility of lane change, and do not consider the existence of the third lane, we put both lanes in the scenario set to 3.75m wide. Meanwhile, a monitoring distance ahead of the vehicle is set to 100m, allowing the vehicle to make speed and direction adjustments. Since this thesis is to show the superiority of reinforcement learning compared with conventional algorithms in complex problems, we increase the number of various types of road damages to more than 6, distributed over a 50m stretch of road. A final section of about 25m is reserved for cars to return to the middle of the rightmost lane to drive. Therefore we set the total length of the test track from the starting point to the destination at about 175m.

The maximum speed limitation Austrian motorway is 130km/h [68], and to make sure the deceleration will not disturb the traffic, set the minimum speed to 70km/h.

CommonRoad has an existing scenario dataset, comprising scenarios from different countries and with different static and dynamic obstacles for researchers to test their models. However, we need to generate scenarios for training purposes, and the road damages explored in this thesis require consideration of different severity levels, which are not considered in CommonRoad, so we cannot use the dateset directly. Fortunately, they also provide a graphical user interface (GUI) in a toolbox called CommonRoad Scenario Designer[69] that can be used to generate simple scenarios into XML files.

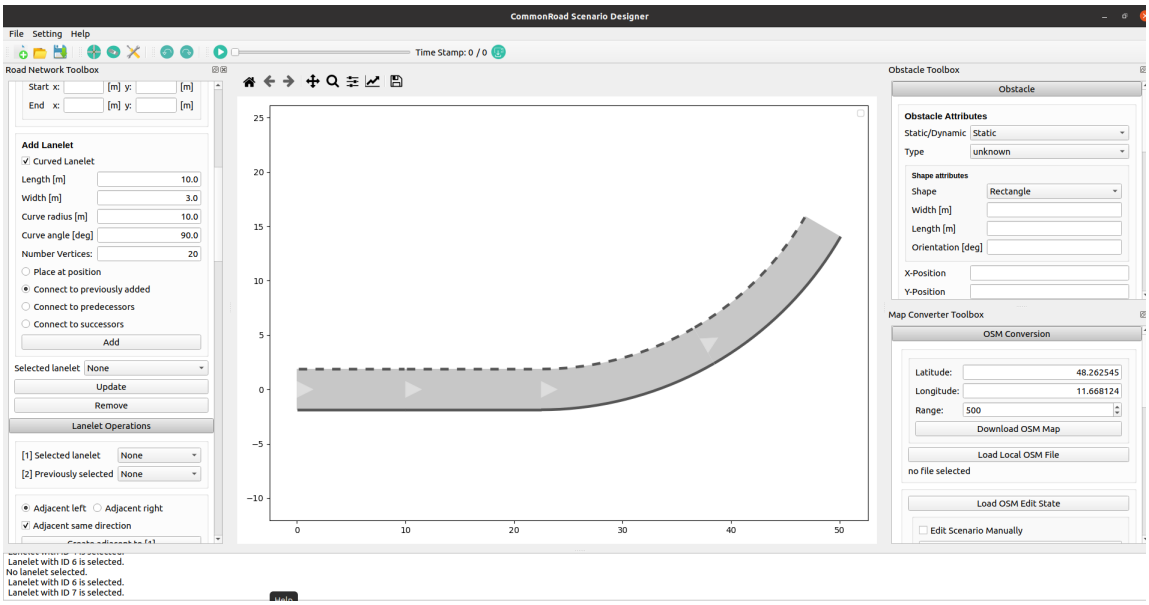

Figure 5.1: CommonRoad Scenario Designer GUI [69]

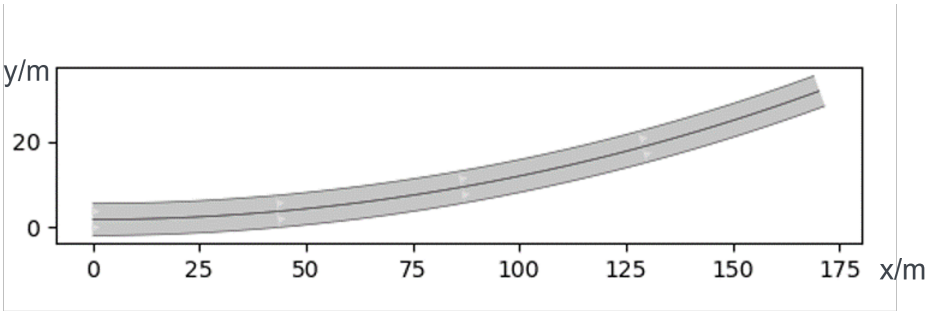

Figure 5.2: Two-lane highway scenario

The CommonRoad Scenario Designer can convert maps from the Lanelet/Lanelet2, OpenDRIVE, OpenStreetMap (OSM), and SUMO formats to the CommonRoad format. Additionally, they provide conversions from the CommonRoad map format to the SUMO and Lanelet format. The GUI is shown in Fig.5.1. Through this, we can simply add lanes, set their length, width and curve radius. Obstacles, and traffic signs can also be added on the interface, but road damage is not considered in it. We will add road damage in the scenario through coding in the next section. The scenario we designed is shown in Fig.5.2.

Planning Problem

A planning problem in CommonRoad is defined by a planning problem ID, an initial state and a goal region. Different planning problems can be inserted and respectively

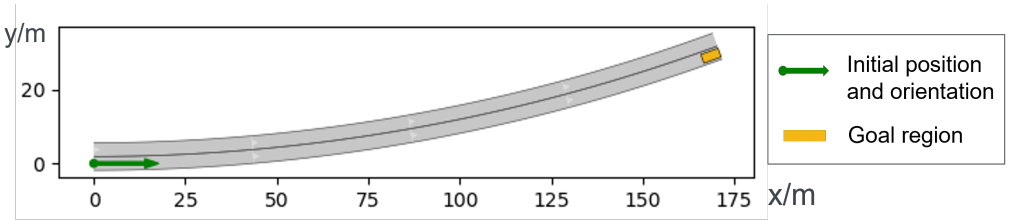

Figure 5.3: Planning problem

called by different planning problem IDs. The scenario generated in Fig.5.2 can be seen as the basic environment, based on which different road damage distributions can be generated to create different scenarios. To control the variables, in all the scenarios the vehicle shares the same start point and destination, therefore, only one planning problem needs to be defined in the basic environment and will be called in all scenarios.

As we can see from the generated scenario in Fig.5.2, the road has curvature, but its coordinate system is still a right angle coordinate system, which makes it difficult to locate the relative coordinates of the vehicle on the road, so we need to create a curvilinear coordinate system along the road. CommonRoad_io API provides tools to help with the coordinates conversion.

In the planning problem, the vehicle will start from the middle of the rightmost lane from the beginning of the road with the coordinate [0,0] and reaches to the middle point of the rightmost lane at the end of the road [175,0]. In the initial state, several parameters need to be defined, including position in a global coordinate system, orientation, velocity in the longitudinal direction, acceleration, yaw rate and slip angle. The vehicle has to obey the speed limitation on highway, so the initial velocity is set as 100km/h, and other parameters to zero value. The goal region is set to be a rectangular area, with a width of 3.0m and a length of 5.0m, whose right side located at the end of the road. The position of the goal region center is calculated through the curve radius and curve angle we used in the scenario designer. During the demonstration API from CommonRoad, the planning problem is shown in Fig.5.3.

Curvilinear Coordinate System

To set up a curvilinear coordinate system along the road, we need a reference path along the middle line of the rightmost lane. In this simple case, we can directly use the route planner API in CommonRoad. It is competent to generate a route from the starting point to the destination along the lanes. Another planning problem needs to be defined to provide this information. The generated planning problem can be slightly

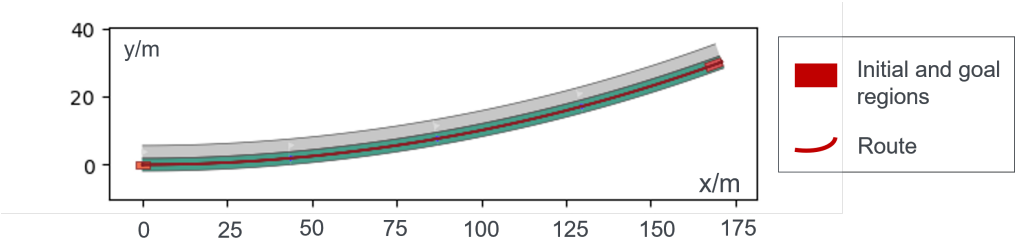

Figure 5.4: Reference path from route planner

modified and used here. By modifying the goal region to a 0.1m*0.1m square centering on the end of the middle line, the route can be accurately set to the middle line of the rightmost lane. Calling the route planner to solve the planning problem, the reference path is generated and shown in Fig.5.4

Based on the trajectory, the curvilinear coordinate system can be built. Using the Commonroad Drivability Checker API we can convert the coordinates from the right angle coordinate system to the curvilinear coordinate system and vice versa. And this function will be frequently used in road damage generation to ensure their position whthin the lanes.

5.2 Road Damage Generation

There are different classification methods for road damage types and severity levels. To better integrate into the project ESRIUM, we use the classification from the project, in which 17 types of road damages are defined with different shapes, sizes and severity levels. The 17th road damage type is rut, which usually occupies a wide range but with a minor severity, so this road damage type will be ruled out from the generation process. The 16 types of road damage are listed in Table 5.1.

To generalize the road damage shapes, convex polygons are used to represent these road damages, and using different distributions of vertices to distinguish them from each other. These polygons can be overlapped and form non-convex shapes.

First of all, a road damage type function is defined, where 16 road damage types are numbered, so that they can be easily called later. There is no existing road damage class in CommonRoad, for better compatibility with CommonRoad, the road damage is defined as Static Obstacle class, and its information including obstacle ID, damage type, position and severity level will be stored in a YAML file through generating. In future training, the severity information can be accessed by calling the obstacle ID.

Table 5.1: Road damage classification in ESRIUM [70]

Damage ID	Damage Type	Severity class
01	Lack of mortar in asphalt	1
02	Grain breakout in asphalt	1
03	Chip breakout in asphalt	2
04	Pothole in asphalt	2
05	Mending area in asphalt	1
06	Binder emission in asphalt	1
07	Grain breakout in concrete	2
08	Chip breakout in concrete	2
09	Detachment in concrete	2
10	Edge defects in concrete	2
11	Single cracks < 2 mm and mended cracks in asphalt	1
	Open cracks >2 mm and <10 mm	2
	Open cracks >10 mm or parallel cracks	3
12	Seam cracks <2 mm and mended cracks in asphalt	1
	Seam cracks >2 mm	2
13	Unmended network cracks in asphalt	2
	Unmended network cracks with polygon breakouts	3
14	Unmended cracks and corner defects in concrete	3
15	Mending area in concrete, mended by asphalt	2
16	Damaged mending area in concrete, mended by concrete	2

The Static Obstacle can be defined as Polygon by providing vertices. So the next step is to configure the polygons. We will use the convex polygon functions to construct the convex polygons, where multiple vertices are given and the outermost vertices are connected together to form a convex polygon. We want to sample the vertices according to Gaussian distribution, but different types of road damage have different sizes. So another function is built to specify the standard deviation for different types according to the ESRIUM road damage description in Table 5.1 and to control their size and shape. The larger the standard deviation in a given direction is, the wider the distribution of vertices in this direction becomes. Since only the outermost points will be taken as the vertices, the final shape will also have a greater length in this direction. For example, if the road damage is a crack, the standard deviation in the x direction can be set much greater than it in the y direction. And for a hole, the greater the standard deviation is, the bigger the size of the hole becomes. According to the ESRIUM road damage classification, under the main types, there are subtypes with different sizes and severity. So in the function, we use random sampling with a fixed proportion for the subtypes in each main type. Then according to the subtype information, we set different standard deviations for x and y directions and randomly sample the severity level. The position of these polygons will be defined by the random sampling of the mean of the Gaussian distribution in the road stretch of [100,y]-[150,y] in the rightmost lane. During this process, each time when we call a road damage type, the information of its position, standard deviation in x and y directions and severity level will be returned.

Then the points from Gaussian distribution will be input into the convex polygon function to get the real vertices. Then we can build the road damage generation function. This function allows you to create polygon road damage based on the road damage type, and add it to the original scenario.

Since we need to create multiple road damages in a scenario, another function is built. Giving the function the number of road damages, it will randomly choose a given number of the road damages types, then calls the road damage generation function, and finally visualize road damages with different severity in different colors, as well as saving the scenario along with the picture of the scenario in different folders. The pictures are saved for visual checking, whether the scenario is successfully generated and whether the solution is reasonable. A scenario with 6 road damages with different severity levels are shown in Fig.5.5. The details are extracted and shown in Fig.5.6

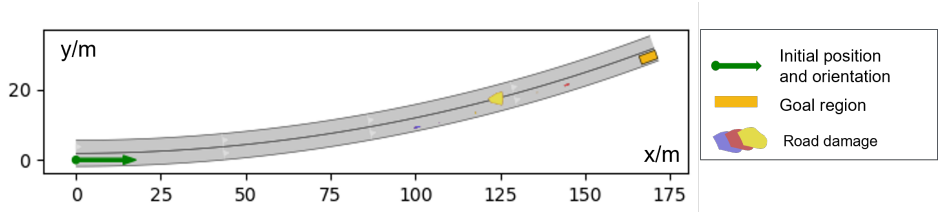

Figure 5.5: Scenario with road damages

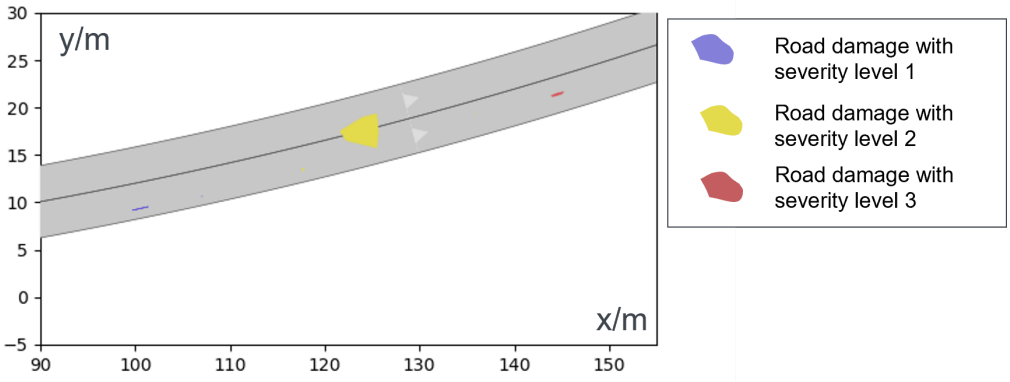

Figure 5.6: Road damage with different severity levels

5.3 Data preprocessing

To train the reinforcement learning, a dataset with multiple scenarios is required. This can be realized by another function repeating the road damage generation process to build the dataset. In this work, 200 scenarios are built and saved in XML files, along with 200 scenario pictures and a YAML file storing the road damage information. Here we will create 200 scenarios.

Even though the XML files are saved under different names, these scenarios are generated from one original scenario, the 200 scenarios share the same scenario ID, by which different scenarios are identified, and it will cause a problem in later training. So we have to execute a batch modification of the scenario is in these XML files to number and distinguish them.

To check if the generated XML files comply with the CommonRoad scenario format, CommonRoad provides a tool to validate the XML files agiant the CommonRoad .xsd specification. Passing the validation means the XML files can be used in the CommonRoad Reinforcement Learning framework.

Since a reinforcement learning training/testing session involves tens of thousands of iterations and accesses to the scenarios, it is a good idea to convert the XML files to PICKLE format so that they will be loaded more efficiently during training and testing. CommonRoad-RL provides tools to achieve the procedure. Then the 200 Pickle data can be split into two folders with the ratio 7:3, separately for training and testing.

5.4 Problem formulation with Reinforcement Learning concept

The fundamental components of Reinforcement learning are explained in section 4.1. This section will define the details of states, actions and rewards to formulate the road damage avoidance problem into a reinforcement learning frame.

States

For reinforcement learning, states mean what the agent needs to observe from the environment, also named as observations. Observation space in the use case is continuous. Various signals need to be observed to define the state of the agent. The observation space includes ego-vehicle-related signals, goal-related signals, surrounding-related signals and termination-related signals.

Before defining the observation space, it's necessary to determine the vehicle model as the agent of the reinforcement learning problem and create the state space of the vehicle. A kinematic single-track model is sufficient for our use case. The model is demonstrated in Fig. 5.7, in which, ICR means Instantaneous Center of Rotation, v_h is longitudinal velocity, δ represents the steering angle and ψ represents the heading, the parameter l describes the wheelbase.

In the model, two wheels are connected by a rigid link, differential constraints are considered, while tire slip is disregarded. The differential equations of the model are:

$$\begin{aligned} \dot{x} &= v_h \cos(\psi) \\ \dot{y} &= v_h \sin(\psi) \\ \dot{\psi} &= \frac{v_h}{l} \tan(\delta) \\ \dot{v}_h &= a_{\text{long}} \\ \dot{\delta} &= v_\delta \end{aligned} \tag{5.1}$$

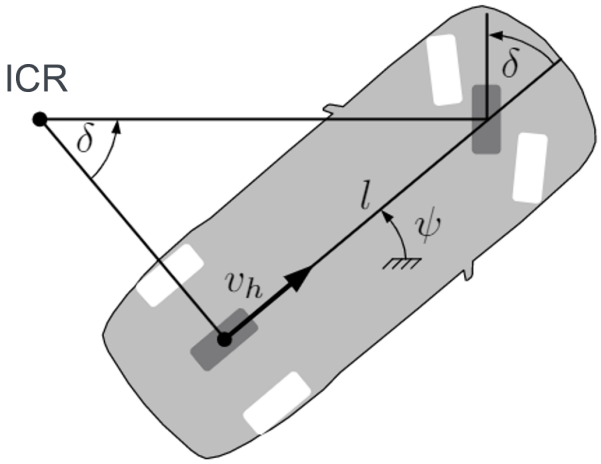

Figure 5.7: Kinematic single-track model [71]

To write the kinematic single-track model in state-space form of $\dot{x} = Ax + Bu$, the following state variables are introduced:

$$x_1 = x, \quad x_2 = y, \quad x_3 = \delta, \quad x_4 = v_h, \quad x_5 = \psi \tag{5.2}$$

The input variables are

$$u_1 = v_\delta, \quad u_2 = a_{\text{long}} \, . \tag{5.3}$$

Inserting the state and input variables into (6.1) results in

$$\begin{aligned} \dot{x}_1 &= x_4 \cos(x_5) \, , \\ \dot{x}_2 &= x_4 \sin(x_5) \, , \\ \dot{x}_3 &= f_{\text{steer}} \ (x_3, u_1) \, , \\ \dot{x}_4 &= f_{acc}(x_4, u_2) \, , \\ \dot{x}_5 &= \frac{x_4}{l} \tan(x_3) \, . \end{aligned} \tag{5.4}$$

The parameters are provided of CommonRoad according to vehicle type. In the use case, the Ford Escort is chosen as the agent. The constraints on steering, velocity, and acceleration are also defined in the function $f_{\text{steer}} \ (x_3, u_1)$ and function $f_{acc}(x_4, u_2)$ in the documentation [71].

Table 5.2: Ego-related Observation Space [60]

variable name	variable type	variable description
v_ego	float	absolute velocity of ego vehicle
a_ego	float	absolute acceleration of ego vehicle
steering_angle	float	steering angle of ego vehicle
heading	float	ego vehicle orientation
global_turn_rate	float	global turn rate of ego vehicle
left_marker_distance	float	lateral distance from ego vehicle center to left marker of ego lanelet
right_marker_distance	float	lateral distance from ego vehicle center to right marker of ego lanelet
left_road_edge_distance	float	lateral distance from ego vehicle center to left road network bound
right_road_edge_distance	float	lateral distance from ego vehicle center to right road network bound
lat_offset	float	from ego vehicle center to ego lanelet center line

Motion planning based on the kinematic models can ensure the kinematic feasibility of the solution. Setting velocity constraints can ensure the vehicle obeys the speed limit. Through the single-track model, and combining the parameters of the distance between tires, the tire tracks can be easily calculated. To determine if the ego vehicle drives in the lane, the distance between the vehicle and the lane marker, as well as the distance between the vehicle and the road network bound is also required. The lateral offset from the center line of the lane is also included in the observation space. Table 5.2 lists the ego-related signals in the observation space.

Besides the ego vehicle states, its relative positions with goal and surroundings are included in the observation space. In the goal-related observation space, not only the longitudinal and lateral distances from the ego vehicle to the goal region, but also the statically and dynamically extrapolated longitudinal distances are comprised. In the surrounding-related observation space, the longitudinal and lateral distances between the ego-vehicle and road damage, along with the damage severity information are observed. There are also two boolean signals observed for further rewards design: if the vehicle changed the lane and if there is a collision with damage happened. The goal-related observation space and surrounding-related observation space are shown in Table 5.3 and Table 5.4.

Table 5.3: Goal-related Observation Space[60]

variable name	variable type	variable description
distance_goal_long	float	relative longitudinal distance (euclidean) from ego vehicle to goal
distance_goal_lat	float	relative lateral distance (euclidean) from ego vehicle to goal
distance_goal_long_ extrapolated_static	list[float]	relative longitudinal distances (euclidean) from statically extrapolated ego vehicle positions to goal
distance_goal_long_ extrapolated_dynamic	list[float]	relative longitudinal distances (euclidean) from dynamically extrapolated ego vehicle positions to goal

Table 5.4: Surrounding-related Observation Space

variable name	variable type	variable description
distance_damage_long	float	relative longitudinal distance (euclidean) from ego vehicle to next road damage
distance_damage_lat	float	relative lateral distance (euclidean) from ego vehicle to next road damage
damage_severity	float	severity level of next road damage
is_damage_collision	boolean	if ego vehicle tire moves over road damage
is_lane_changing	boolean	if ego vehicle changed the lane

Table 5.5: Termination-related Observation Space[60]

variable name	variable type	variable description
remaining_steps	int	number of time steps left for current episode
is_goal_reached	boolean	identifier to determine if ego vehicle reaches goal region
is_off_road	boolean	identifier to determine if ego vehicle is off road
is_collision	boolean	identifier to determine if ego vehicle collides with road damage of severity level 3
is_time_out	boolean	identifier to determine if maximum episode length is met
is_friction_violation	boolean	identifier to determine if ego vehicle violates the friction constraints

Table 5.6: Action space[60]

No	variable type	description
0	float	ego vehicle acceleration
1	float	ego vehicle steering angle rate

In the end, the termination flags are considered. They indicate when the episode should be determined. The termination-related observation space comprises, the remaining steps, if the goal is reached, if the vehicle is off the road, if there is a collision with road damage of severity level 3, if there is a time-out, if there is friction violation. They are listed in Table 5.5

Actions

In our use case, the anticipated behaviors of the agent are acceleration/deceleration and steering. According to the vehicle model, the differential constraints are also considered. Therefore, steering angle rate rather than steering angle is more suitable for the use case to maintain the continuity of steering action. The action space consisting of ego vehicle acceleration and ego vehicle steering angle rate is shown in Table 5.6.

Rewards

In the field of reinforcement learning, rewards are numerical signals that guide the agent in learning desirable actions based on the optimal policy. In the automotive industry, the so-called Key Performance Indicators (KPIs) are used to objectively evaluate the quality of the application. We can also use the KPI method to define the reward. Some

KIPs can be proposed for this problem: If the acceleration and steering is above the people's comfort zone, if there is a collision, if there is a lane changing, if the vehicle reaches the goal region. But only with these sparse rewards, the RL agent can barely learn and may just take no action.

In reinforcement learning (RL), it's typically assumed that the agent only observes the sequence of instantaneous rewards that correspond to the state-action trajectory. A sparse reward refers to a reward function that is zero in most of its domain, and only gives positive values to very few state, action, and future state transitions. For an RL agent, if the environment has a sparse reward function, it means that the agent won't get any feedback about whether the instantaneous actions that it takes are good or bad. A dense reward refers to a reward function that gives value to most of the transitions, thus the agent gets feedback at almost every time step. To mark the importance of the events like reaching the goal or colliding the severe road damage, in the problem, the hybrid reward function is adopted.

The sparse part of the reward function is a sum of the rewards for the following indicators:

- If the agent reaches the goal?
- If there is a collision with road damage with severity level 3?
- If the agent drives off the road?
- If there is a time-out?
- If there is friction violation?
- If there is a lane-changing?
- Tire rolls over the road damage with severity level 1 and level 2.

These events correspond to huge positive or negative rewards, depending on if it contributes to the goal of the use case. For example, the agent should be encouraged to reach the goal, so the reward for a goal-reaching should be a huge positive number. While a collision with very severe road damage should be avoided, so the reward for the collision is a huge negative number. For the last indicator, a reward coefficient is introduced to multiply with damage severity levels, to differentiate the impact from different severity levels.

The dense part of the reward is divided into a goal-related part and a comfort-related part. The goal-related part is a multiplication of a coefficient and the normalized distance to the goal. The comfort-related part considers the impact of steering angle rate

and acceleration on passengers' comfort. Two coefficients are designed to separately multiply steering angle rate and acceleration.

The hybrid reward function can be formed like below:

$$\begin{aligned}\text{Hybrid reward} = {} & \text{reward\_goal\_reached} + \text{reward\_collision} + \text{reward\_off\_road} \\ & + \text{reward\_time\_out} + \text{reward\_friction\_violation} \\ & + \text{reward\_lane\_changing} \\ & + \text{reward\_tire\_collision\_coefficient} * \text{damage\_severity} \\ & + \text{reward\_goal\_distance\_coefficient} * \text{normalized\_distance\_to\_goal} \\ & + \text{reward\_steering\_angle\_rate\_coefficient} * \text{steering\_angle\_rate} \\ & + \text{reward\_acceleration\_coefficient} * \text{a\_ego}\end{aligned} \tag{5.5}$$

The reward function can be modified through the training process according to the performance to optimize the solution.

To summarize the reinforcement learning concept in road damage avoidance problem, the interaction between the agent and the environment can be described in Fig. 5.8.

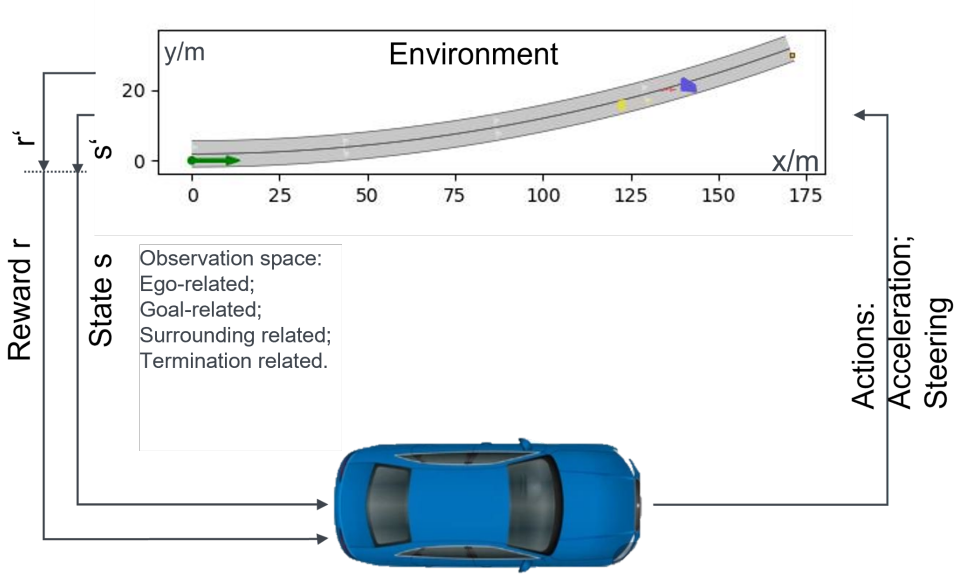

Figure 5.8: Road Damage Avoidance in RL frame

At this point, the training dataset is prepared and the reinforcement learning concept

has been built. The implementation can be executed in Commonroad-RL or by building the custom environment using Stable Baseline3.

6 Conclusion

In this chapter, the results of the work are summarized once again. The outlook is also described.

6.1 Summary

The thesis first introduces the road damage problem that further research hasn't been pursued. The ESRIUM project, which is also the grounding point of this thesis, is also briefly introduced in this chapter.

Although this thesis only considers motion planning based on known road damage information, the road damage avoidance problem is not a separate topic and it needs the support of many other techniques. So the technical background including general autonomous driving system, computer vision in detecting road damages and I2V techniques are then explained and relevant literature is reviewed. For the motion planning problem itself, the recent research on motion planning algorithms and reinforcement learning is also summarized in this chapter.

And then conventional motion planning algorithms are explained and compared to reinforcement learning algorithms. From the benchmark, we can see the advantages of the RL methods in the road damage avoidance problem. The fourth chapter explains the details of the reinforcement learning concept to provide the theoretical basis for the technical realization.

Finally, the process of scenario design and training dataset preparation is described. The abstract problem formulation of the road damage problem is also accomplished in the reinforcement learning frame.

To summarize, the thesis explains the necessity of research in road damage, review the literature on motion planning algorithms and explore the possibility of using reinforcement learning methods in road damage avoidance problem. Since there is no standard system of Reinforcement learning theory yet, the elaboration of reinforcement learning in this thesis is summarized and sorted out from several sources.

Although this paper implements the formulation of the road damage avoidance problem into a reinforcement learning problem and set up the entire training framework, only a specific training implementation and test results can determine whether reinforcement learning methods are superior for this problem. To achieve the goal, besides the RL

training, implementation of A* search algorithm, CL-RRT algorithm, potential field and convex optimization should also be executed. Due to the specialty of the use case, a standard frame cannot be directly used, specified algorithm frames should be built. For example, in the A* search algorithm, we cannot directly use the obstacle avoidance frame in our problem, considering different severity levels, the heuristic function needs to be modified to fit our problem. The above implementation needs a lot of work, so this thesis only compares them from a theoretical aspect. But in the road damage avoidance problem itself, these implementations are worth executing.

6.2 Prospect

Road damage, with the existing technical support like computer vision and I2V, will become a more worthy and researchable topic in the future. The massive data from the Road Damage Detection Challenge (GRDDC) can be valuable learning resources, which allow other machine learning algorithms demanding massive data to be used in the problem. Hybrid methods can also be researched to compensate for each other's shortcomings.

The interplay of reinforcement learning with other engineering and science is exciting. Of all forms of machine learning, reinforcement learning is the closest to the kind of learning done by humans and other animals, and many of the core algorithms of reinforcement learning were originally inspired by biological learning systems. And reinforcement learning has also provided influential models for psychology and neuroscience. Reinforcement learning is part of a decades-long trend toward greater integration of artificial intelligence and machine learning with statistics, optimization, and other mathematical disciplines. For example, the ability of some reinforcement learning methods to use parametric approximators to learn solves the classic "curse of dimensionality" problem in operations research and control theory. We can expect to apply reinforcement learning to more interdisciplinary research, and even, in the future, it may become a tool to mimic human thinking and be commonly used on human tasks.

Reinforcement learning is also part of the larger trend of AI returning to simple general principles. Modern AI has a vast array of special-purpose tricks, procedures, and heuristics to achieve intelligence by feeding in enough relevant facts to make the machine think. But approaches based on simple general principles will no longer rely on huge scale datasets and thus can be applied more to simply different problems.

Another problem is the robustness of reinforcement learning algorithms. Supervised learning and deep learning have been iterated over decades and have more robust

algorithms, which allow the training process to yield more reliable results without relying on a lot of experience. Nowadays, reinforcement learning tools are still relatively crude, and researchers still need to rely on a lot of practical experience and professional intuition to get the desired results. Robust reinforcement learning algorithms can lower the threshold for using reinforcement learning, making it a more pervasive tool for greater use.

Bibliography

[1] T. R. Miller and E. Zaloshnja. “On a crash course: The dangers and health costs of deficient roadways”. In: *Pacific Institute for Research and Evaluation (PIRE)* (2009),

[2] M. G. Fields and S. Purnell. “23rd Annual Highway Report on the Performance of State Highway Systems”. In: *Reason Foundation Policy Study* 457 (2018),

[3] Department of Transport and Main Roads. *Cost-benefit analysis manual road projects*. 2011, pp. 27–54,

[4] D. M. Newbery. “Road Damage Externalities and Road User Charges”. In: *Econometrica* 56.2 (1988), pp. 295–316. ISSN: 00129682, 14680262,

[5] Sangwin Group of Companies. *What Are The Main Causes Of Road Damage?* URL: https://www.sangwin.co.uk/news/what-are-the-main-causes-of-road-damage, on 12.01.2023 at 13:00.

[6] O. Mavrouli, J. Corominas, I. Ibarbia, N. Alonso, I. Jugo, J. Ruiz, S. Luzuriaga, and J. A. Navarro. “Integrated risk assessment due to slope instabilities in the roadway network of Gipuzkoa, Basque Country”. In: *Natural Hazards and Earth System Sciences* 19.2 (2019), pp. 399–419,

[7] ESRIUM. *Safe and efficient roads for a smarter, safer, greener mobility*. 2020. URL: https://esrium.eu/, on 12.01.2023 at 13:12.

[8] F. Hofbauer, M. Walch, W. Schildorfer, and M. Neubauer. “Towards Requirements Related to Future CCAM Services for Road Usage Optimization”. In: *Human Interaction, Emerging Technologies and Future Systems V*. Ed. by T. Ahram and R. Taiar. Vol. 319. Lecture Notes in Networks and Systems. Cham: Springer International Publishing, 2022, pp. 1294–1301. ISBN: 978-3-030-85539-0,

[9] M. Rudigier, G. Nestlinger, K. Tong, and S. Solmaz. “Development and Verification of Infrastructure-Assisted Automated Driving Functions”. In: *Electronics* 10.17 (2021). ISSN: 2079-9292,

[10] M. Rudigier, S. Solmaz, G. Nestlinger, and K. Tong. “Development, Verification and KPI Analysis of Infrastructure-Assisted Trajectory Planners”. In: *2022 International Conference on Connected Vehicle and Expo (ICCVE)*. 2022, pp. 1–6,

[11] L. Masello, G. Castignani, B. Sheehan, F. Murphy, and K. McDonnell. “On the road safety benefits of advanced driver assistance systems in different driving contexts”. In: *Transportation Research Interdisciplinary Perspectives* 15 (2022), p. 100670. ISSN: 2590-1982,

[12] C. R. Storck and F. Duarte-Figueiredo. “A Survey of 5G Technology Evolution, Standards, and Infrastructure Associated With Vehicle-to-Everything Communications by Internet of Vehicles”. In: *IEEE Access* 8 (2020), pp. 117593–117614,

[13] World Health Organization. *Global status report on road safety 2018*. 2018. ISBN: 9789241565684,

[14] National Highway Traffic Safety Administration. *2016 Fatal Motor Vehicle Crashes: Overview*. 2016,

[15] SAE. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_202104*. 2021. URL: https://www.sae.org/standards/content/j3016_202104/, on 12.01.2023 at 13:13.

[16] D. Arya, H. Maeda, S. Kumar Ghosh, D. Toshniwal, H. Omata, T. Kashiyama, and Y. Sekimoto. “Global Road Damage Detection: State-of-the-art Solutions”. In: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 5533–5539. ISBN: 978-1-7281-6251-5,

[17] W. Zhan, C. Sun, M. Wang, J. She, Y. Zhang, Z. Zhang, and Y. Sun. “An improved Yolov5 real-time detection method for small objects captured by UAV”. In: *Soft Computing* 26 (2022), pp. 361–373,

[18] V. Hegde, D. Trivedi, A. Alfarrarjeh, A. Deepak, S. H. Kim, and C. Shahabi. “Yet another deep learning approach for road damage detection using ensemble learning”. In: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, pp. 5553–5558,

[19] R. Miucic. *Connected vehicles: Intelligent transportation systems*. Springer, 2018,

[20] Y. Li, H. Wang, W. Wang, S. Liu, and Y. Xiang. “Reducing the risk of rear-end collisions with infrastructure-to-vehicle (I2V) integration of variable speed limit control and adaptive cruise control system”. In: *Traffic Injury Prevention* 17.6 (2016). PMID: 26761633, pp. 597–603,

[21] O. Grembek, A. Kurzhanskiy, A. Medury, P. Varaiya, and M. Yu. “Making intersections safer with I2V communication”. In: *Transportation Research Part C: Emerging Technologies* 102 (2019), pp. 396–410. ISSN: 0968-090X,

[22] M. A. García-Garrido, M. Ocaña, D. F. Llorca, E. Arroyo, J. Pozuelo, and M. Gavilán. “Complete Vision-Based Traffic Sign Recognition Supported by an I2V Communication System”. In: *Sensors* 12.2 (2012), pp. 1148–1169. ISSN: 1424-8220,

[23] H. He, Y. Wang, R. Han, M. Han, Y. Bai, and Q. Liu. “An improved MPC-based energy management strategy for hybrid vehicles using V2V and V2I communications”. In: *Energy* 225 (2021), p. 120273. ISSN: 0360-5442,

[24] M. N. Tahir, P. Leviäkangas, and M. Katz. “Connected Vehicles: V2V and V2I Road Weather and Traffic Communication Using Cellular Technologies”. In: *Sensors* 22.3 (2022). ISSN: 1424-8220,

[25] ASFiNAG. “Getting Ready for Infrastructure Supported Automated Driving”. In: *Proceedings of ESRIUM Workshop1, 24.09.2021*,

[26] ASFiNAG. “C-ITS - Road infrastructure for connected cooperative automated driving”. In: *Proceedings of ESRIUM Workshop1, 24.09.2021*,

[27] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser. “A Review of Motion Planning for Highway Autonomous Driving”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.5 (2020), pp. 1826–1848,

[28] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo. “Development of Autonomous Car—Part II: A Case Study on the Implementation of an Autonomous Driving System Based on Distributed Architecture”. In: *IEEE Transactions on Industrial Electronics* 62.8 (2015), pp. 5119–5132,

[29] V. Kunchev, L. Jain, V. Ivancevic, and A. Finn. “Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review”. In: *Knowledge-Based Intelligent Information and Engineering Systems*. Ed. by B. Gabrys, R. J. Howlett, and L. C. Jain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 537–544,

[30] O. Sharma, N. Sahoo, and N. Puhan. “Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey”. In: *Engineering Applications of Artificial Intelligence* 101 (2021), p. 104211. ISSN: 0952-1976,

[31] M. Elbanhawi and M. Simic. “Sampling-Based Robot Motion Planning: A Review”. In: *IEEE Access* 2 (2014), pp. 56–77,

[32] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza. “Self-driving cars: A survey”. In: *Expert Systems with Applications* 165 (2021), p. 113816. ISSN: 0957-4174,

[33] D. González, J. Pérez, V. Milanés, and F. Nashashibi. “A Review of Motion Planning Techniques for Automated Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (2016), pp. 1135–1145,

[34] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. “A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles”. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 33–55,

[35] R. Kala and K. Warwick. “Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization”. In: *Journal of Intelligent & Robotic Systems* 72 (2013), pp. 559–590,

[36] A. Stentz. “Optimal and efficient path planning for partially-known environments”. In: *Proceedings of the 1994 IEEE international conference on robotics and automation*. IEEE. 1994, pp. 3310–3317,

[37] D. Ferguson and A. Stentz. “Using interpolation to improve path planning: The Field D* algorithm”. In: *Journal of Field Robotics* 23.2 (2006), pp. 79–101,

[38] K. Daniel, A. Nash, S. Koenig, and A. Felner. “Theta*: Any-angle path planning on grids”. In: *Journal of Artificial Intelligence Research* 39 (2010), pp. 533–579,

[39] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. “Anytime search in dynamic graphs”. In: *Artificial Intelligence* 172.14 (2008), pp. 1613–1643,

[40] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. “Junior: The stanford entry in the urban challenge”. In: *Journal of field Robotics* 25.9 (2008), pp. 569–597,

[41] D. Ferguson, T. M. Howard, and M. Likhachev. “Motion planning in urban environments”. In: *Journal of Field Robotics* 25.11-12 (2008), pp. 939–960,

[42] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee. “Motion planning for autonomous driving with a conformal spatiotemporal lattice”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 4889–4895,

[43] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. “Real-time motion planning with applications to autonomous urban driving”. In: *IEEE Transactions on control systems technology* 17.5 (2009), pp. 1105–1118,

[44] S. Karaman and E. Frazzoli. “Optimal kinodynamic motion planning using incremental sampling-based methods”. In: *49th IEEE conference on decision and control (CDC)*. IEEE. 2010, pp. 7681–7687,

[45] J. Reeds and L. Shepp. “Optimal paths for a car that goes both forwards and backwards”. In: *Pacific journal of mathematics* 145.2 (1990), pp. 367–393,

[46] T. Fraichard and A. Scheuer. “From Reeds and Shepp’s to continuous-curvature paths”. In: *IEEE Transactions on Robotics* 20.6 (2004), pp. 1025–1035,

[47] P. Petrov and F. Nashashibi. “Modeling and nonlinear adaptive control for autonomous vehicle overtaking”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.4 (2014), pp. 1643–1656,

[48] J. P. Rastelli, R. Lattarulo, and F. Nashashibi. “Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles”. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE. 2014, pp. 510–515,

[49] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. “Stanley: The robot that won the DARPA Grand Challenge”. In: *Journal of field Robotics* 23.9 (2006), pp. 661–692,

[50] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. “Path planning for autonomous vehicles in unknown semi-structured environments”. In: *The international journal of robotics research* 29.5 (2010), pp. 485–501,

[51] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, et al. “Making bertha drive—an autonomous journey on a historic route”. In: *IEEE Intelligent transportation systems magazine* 6.2 (2014), pp. 8–20,

[52] C. Chen, L. Zanotti Fragonara, and A. Tsourdos. “Go wider: An efficient neural network for point cloud analysis via group convolutions”. In: *Applied Sciences* 10.7 (2020), p. 2391,

[53] S. Zhu and B. Aksun-Guvenc. “Trajectory planning of autonomous vehicles based on parameterized control optimization in dynamic on-road environments”. In: *Journal of Intelligent & Robotic Systems* 100 (2020), pp. 1055–1067,

[54] H. Du, S. Leng, F. Wu, X. Chen, and S. Mao. “A new vehicular fog computing architecture for cooperative sensing of autonomous driving”. In: *IEEE Access* 8 (2020), pp. 10997–11006,

[55] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018,

[56] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. “Learning to drive in a day”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8248–8254,

[57] X. Wang, C. Pillmayer, and M. Althoff. “Learning to Obey Traffic Rules using Constrained Policy Optimization”. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. 2022, pp. 2415–2421,

[58] H. Krasowski, X. Wang, and M. Althoff. “Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–7,

[59] X. Wang, H. Krasowski, and M. Althoff. “CommonRoad-RL: A Configurable Reinforcement Learning Environment for Motion Planning of Autonomous Vehicles”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 466–472,

[60] X. Wang, H. Krasowski, and M. Althoff. “CommonRoad-RL: A configurable reinforcement learning environment for motion planning of autonomous vehicles”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 466–472,

[61] Amit. *Introduction to A**. 2023. URL: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html, on 12.01.2023 at 13:55.

[62] MIT. *Motion Planning Algorithms (RRT, RRT*, PRM) - [MIT 6.881 Final Project]*. 2020. URL: https://www.youtube.com/watch?v=gP6MRe_IHFo&t=116s, on 12.01.2023 at 13:55.

[63] Y. Lin, S. Maierhofer, and M. Althoff. “Sampling-Based Trajectory Repairing for Autonomous Vehicles”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 572–579,

[64] R. Siddiqui. *Path Planning Using Potential Field Algorithm*. 2018. URL: https://medium.com/@rymshasiddiqui, on 12.01.2023 at 13:25.

[65] Ahmadi. *Theory of convex functions*. 2015. URL: https://www.princeton.edu/~aaa/Public/Teaching/ORF523/S16/ORF523_S16_Lec7_gh.pdf, on 12.01.2023 at 14:55.

[66] Z. Zhu, E. Schmerling, and M. Pavone. “A convex optimization approach to smooth trajectories for motion planning with car-like robots”. In: *2015 54th IEEE conference on decision and control (CDC)*. IEEE. 2015, pp. 835–842,

[67] Stanford. *CS234: Reinforcement Learning Winter 2023*. 2023. URL: http://web.stanford.edu/class/cs234/index.html, on 12.01.2023 at 13:35.

[68] Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology. *Speed limits*. 2023. URL: https://www.oesterreich.gv.at/en/themen/freizeit_und_strassenverkehr/kfz/10/Seite.063300.html, on 12.01.2023 at 13:53.

[69] S. Maierhofer, M. Klischat, and M. Althoff. “Commonroad scenario designer: An open-source toolbox for map conversion and scenario creation for autonomous vehicles”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 3176–3182,

[70] ESRIUM. *Road Damage Types Considered in ESRIUM*. 2022. URL: https://confluence.nng.com/display/ESRIUM/Road+Damage+Types+Considered+in+ESRIUM, on 12.01.2023 at 13:35.

[71] M. Althoff, M. Koschi, and S. Manzinger. “CommonRoad: Composable benchmarks for motion planning on roads”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 719–726,