Energy-aware path planning algorithm
Analysis and case study

Guirado Forment, Emma N.

Award date:
2022

Energy-aware path planning algorithm

Analysis and case study

TU/e & NXP Semiconductors
July 6, 2022

Emma N. Guirado (1529579)
e.n.guirado.forment@student.tue.nl

Klaas Brink
klaas.brink@nxp.com

Dip Goswami
d.goswami@tue.nl

Kees Moerman
kees.moerman@nxp.com

Abstract

Path planning allows the preparation of routes between a particular source and destination, which is incredibly important for autonomous driving. In the current literature, it is complicated to find a method that combines real-time battery measurements with existing trajectory data to determine a more optimal route. Additionally, many projects focus their analysis on a small-scale platform, but they do not generally report how the performance scales up to a bigger platform.

This project tries to tackle these two aspects.

For this purpose, a Q-learning path planner that uses real trajectory information and battery measurements is proposed (called Q-LEA). Its performance is evaluated –and compared against a method that finds the shortest route (named Q-LSP)– using simulated data; real data collected with a small-scale platform (i.e. mobile robot); and driving data gathered with an electric vehicle.

For each of these cases, a different scenario (with different properties) was assessed. In the case of simulated data, Q-LEA improved 15% of the routes proposed by Q-LSP, achieving a reduction of the energy consumption of up to 28.58%. With the small-scale platform and indoors setup, 22.22% of the routes were optimised by the energy-aware path planner and consumed up to 9.19% less energy than the shortest path. Finally, when considering real driving data collected in a region of Waalre, a total of 6% of the routes were improved with Q-LEA with a reduction of 1% of the energy consumption.

Even though different magnitudes of data were assessed and the energy reduction ranges from 1% to 28.58%, it is clear that, by using previous trajectory data, the resulting planned path can –in some occasions– be more energy-efficient than the shortest route. This first investigation opens a new line of study with several options suggested to further extend the proposed solution and analysis carried out.

Acknowledgements

In the past two years I have lived experiences that made me grow as a person and as a professional, and I would not be who I am today had I not taken the leap of moving abroad and starting this new chapter of my life.

I would like to thank my tutors and advisors, Dr. Dip Goswami, Klaas Brink and Kees Moerman, who provided me with great knowledge along this work. Moreover, I would like to extend my thanks to everyone involved in this project somehow, your help and support allowed me to continue this path.

Furthermore, I would like to take a moment to thank my family and friends who have been a big pillar throughout the way. Although we might have been far apart at times, your love reached all the way.

Additionally, I would like to thank TU/e for having such a strong community supporting students, making sure to organise so many activities that allowed us to have fun and stay somehow together through this strange period.

Moreover, I would like to give special thanks to the Students Sport Centre family, who kept me sane along the way.

And, last but certainly not least, I would like to thank you for making time to read this work.

Emma Núria Guirado Forment
June 27, 2022
Eindhoven, The Netherlands

"Get your hands on the changes you can make, because your possibilities are limitless." – Min Yoon-gi, 2020.

"You may find that any moment can be turned into an opportunity. Allow yourself to take it easy. Take it one step at a time." – Kim Seok-jin, 2020.

Contents

Notation

AI	Artificial Intelligence
BMS	Battery Management System
CNN	Convolutional Neural Network
DQN	Deep Q-Network
$DDQN$	Double Deep Q-Network
DRL	Deep Reinforcement Learning
ECU	Electronic Control Unit
EV	Electric Vehicle
HTC	High Tech Campus
IQL	Improved Q-Learning
MDP	Markov Decision Process
ML	Machine Learning
NN	Neural Network
OSM	Open Street Map
$Q-LEA$	Q-learning Energy-aware path planning algorithm
$Q-LSP$	Q-learning shortest path planning algorithm
$Q-value$	Proportion of false positives incurred when taking a particular action.
RL	Reinforcement Learning
ROS	Robotic Operating System
$SARSA$	State-Action-Reward-State-Action
SOC	State-of-Charge
UWB	Ultra-Wideband
$V2I$	Vehicle-to-Infrastructure
$V2P$	Vehicle-to-Pedestrian
$V2V$	Vehicle-to-Vehicle
$V2X$	Vehicle-to-Everything

1 Introduction

The shift towards Electric Vehicles (EV) is enabling a fast evolution in the automobile industry. Nowadays, vehicles are equipped with all sorts of sensors and sub-systems that provide great assistance to the driver. Good examples of the new features that appeared throughout the years are lane-keeping assistance systems [1], or parking assistance systems [2].

Another technology that has evolved alongside the connectivity of vehicles is the so-called Vehicle-to-Everything (V2X) [3], which allows to establish communication between vehicles and: infrastructures (V2I), pedestrians (V2P), vehicles (V2V), among others. When taken to the extreme, connected car solutions are the main enabler for autonomous driving and big data.

Autonomous vehicles can be defined as automotive systems that are capable of self-driving with no human intervention [3, 4]. As stated in [5], the autonomous level of a vehicle can range from level 0 –no automation whatsoever– to level 5 –full automation–, and it is expected that autonomous driving will replace human drivers in the coming years. Therefore, this is an active area of research nowadays.

Another greatly researched topic is related to the range of the EVs. Since the battery is the main source of energy of the vehicle, the Battery Management Systems (BMS) [6] –the real-time systems responsible for controlling the secure and safe operation of the vehicle battery pack(s) and their performance–, are more important than ever.

Due to the limited range of these vehicles, it is paramount to minimise as much as possible the energy consumed while driving. For this purpose, in the context of autonomous driving, there are mainly two types of approaches to lower the battery consumption: reducing the computation processes of the intelligent systems, as presented in e.g. [7], or by doing more efficient computation, or by optimising the motion of the vehicle with a careful planning of the path and the vehicle movements needed to reach the destination.

Even though the first approach can significantly reduce the energy consumed, this project focuses on the *path planning* problem, which if solved efficiently can also lower the energy consumption. Path planning can be described as finding a route, in a specific environment, to go from a starting point (A) to another position (B) avoiding any possible collisions in between [8, 9, 10]. As expected, path planning is an important task for the automotive industry and for autonomous driving in particular.

The most traditional approaches for path planning require a complete

model of the environment (e.g. a map of the streets of a city) in order to pre-plan the route from the start point to the end goal beforehand. An overview of these techniques most commonly used for path planning, such as Dijkstra [11] or A* [12], is presented in [13]. However, these methods do not work for unknown environments or unpredicted circumstances, such as a blocked road.

As an alternative, the usage of Deep Reinforcement Learning (DRL) in path planning algorithms has incremented in the recent years [14]. These methods present a more dynamic approach and do not require a complete understanding of the surrounding environment, which can become an advantage in terms of development and memory requirements. However, they do require a long training phase to ensure correct functionality, after which they should be adaptable to new scenarios (where they can keep on learning and adapting).

Additionally, the fact that vehicles are becoming heavily interconnected could be a game changer in terms of route planning. That is, trajectory data –including time to destination, waiting times, energy consumed, paths taken, etc.– could be recorded for a fleet of vehicles and used in order to determine an even more optimal route.

The aim of this project is to combine the real-time BMS measurements, such as State-of-Charge (SOC) [6], with trajectory data –emphasising on the energy consumed by the different routes– to implement an energy-aware DRL algorithm for path planning.

To achieve this purpose, driving data will be collected with a small-scale mobile robot to determine if there is a significant gain in terms of energy-efficiency when compared against a DRL shortest path planner. Moreover, a normal full-sized electric vehicle will also be used to collect relevant trajectory data with the intention of extrapolating the observations obtained in the small platform.

To the best of the author's knowledge, no previous work combines both trajectory data and real-time battery data in the implementation of the path planning algorithm. Furthermore, an extrapolation towards an EV of the obtained results on the small platform is also unexplored in the current literature.

The rest of the document is structured as follows. A brief introduction to Deep Reinforcement Learning (DRL) is presented in the following subsection. Section 2 summarises the state-of-the-art of path planning and energy-awareness. Sections 3 and 4 present the problem statement and the

proposed approach. Section 5 introduces the experimental platforms used in this project. Furthermore, Section 6 gathers the feasibility study together with the initial results. The proposed path planners (energy-aware and shortest route) are detailed in Section 7. Sections 8 and 9 gather the experiments conducted with the small-scale and EV platforms, respectively. The document finalises with some concluding remarks in Section 10.

1.1 Deep Reinforcement Learning

Artificial Intelligence (AI) is the discipline that creates systems that are able to learn from experience and, under certain conditions, take decisions –actions– according to the information received from the surrounding environment [15, 16]. Within this umbrella term, there are a lot of different disciplines [15, 17], where Machine Learning (ML) is the one that provides methods which enable automatic pattern detection in data.

ML algorithms can be divided into three big groups according to the goal to be achieved. To start, there is *supervised learning*, where by means of manually labelled data the machine is trained to be able to classify or detect similarities in data. Secondly, *unsupervised learning* is the term that refers to the algorithms capable of inferring properties from unlabelled data. In other words, the machine learns to find patters or hidden structures in the data.

Finally, *(Deep) Reinforcement Learning* comprises the tasks in which an agent has to learn how to act in a particular environment in order to maximise a numerical reward. In this case, the machine learns which actions are the best –that is, give the highest immediate or in the long run reward– to perform under certain circumstances by means of trial-and-error.

When applying DRL algorithms to solve a problem, two big groups of methods can be distinguished: the *tabular solutions* and *approximate solutions* [18].

For scenarios in which the state and action spaces are sufficiently small, the value functions can generally be represented in the shape of arrays or tables (hence the name, *tabular solutions*) and these methods usually find exact solutions to the problem. On the contrary, when the environment is much more complex, the table can quickly escalate due to bigger state and action spaces, making it impossible to use a tabular representation. In this case, the *approximate solutions* come to play, providing a feasible way to represent these combinations of states-actions, by means of –generally–

Neural Networks (NN) [19].

An RL algorithm can be formalised using a Markov Decision Process (MDP) [20], which allows to model decision-making scenarios. This way, a DRL algorithm can be represented with the following terminology: *agent*, *environment*, *action*, *reward* and *state*. Figure 1 depicts these components and their interactions between each other.

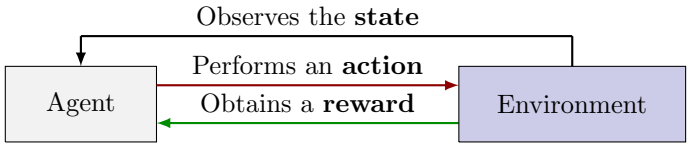


Figure 1: Block diagram of Reinforcement Learning and its components.

The *agent* represents the piece of software that interacts with the *environment*, which represents the scenario in which the agent needs to learn how to behave. The agent perceives or observes the current *state* of the environment and, according to it, decides to take an *action*, which will directly affect the state of the environment. Depending on which new state the environment ends up in, a *reward* –which can be positive or negative– is given to the agent.

The agent associates the states visited and actions performed with the obtained rewards, which shapes the knowledge it has of the environment. In other words, the agent learns by trial-and-error how to act depending on what situation it encounters.

In order to ensure that a complete analysis of the environment is performed and prevent the agent from taking only the actions that provide a high reward, a trade-off between exploration and exploitation has to be guaranteed.

2 Related work

Deep Reinforcement Learning (DRL) has been a very active area of research in the recent years due to its powerful characteristics and ability to learn increasingly complex policies in high dimensional environments [21].

Especially, DRL has been applied to a lot of the tasks related to automated driving, a brief summary of which is presented in [14]. However, the autonomous driving systems pipeline is quite extent and consists of various tasks, as presented in Figure 1 of [14]. As stated, the two main problems addressed by autonomous driving systems consist of *scene understanding* and *decision making and planning*. The focus of this project resides on the latter. To be precise, the targeted algorithm falls in the category of "plan and decide".

Therefore, this project scope is limited to determining the most energy-efficient path to go from the starting point A to the finishing point B. Nonetheless, the precise control of the vehicle –in terms of how the automobile should behave– is not included in this study. In other words, only the path to be followed to reach the target position is considered, also known as the *path planning* or *route planning* problem.

2.1 DRL in path planning

Works in [21, 22, 23] present an overview of the methods or technologies used in order to tackle different issues in the motion planning problem, as well as how to represent the states, actions and rewards of the DRL algorithm, respectively.

As briefly discussed above, DRL algorithms can be divided into two types of solutions: *tabular solutions* and *approximate solutions*. This distinction can also be made when talking about methods that solve the *path planning* issue.

2.1.1 Tabular solutions

Regarding tabular solutions, the published solutions usually involve applying Q-learning [24] (or improved variations of it). In this method, a table is used to keep track of all the possible combinations of states-actions. For each of these combinations, a value –the Q-value– is assigned, which determines

the best action to take from the current state while obtaining the biggest cumulative reward.

For instance, in [25], two algorithms are tested to solve the global path planning problem presented. Namely, Q-learning and its modification called State-Action-Reward-State-Action (SARSA). The former tries to find the most efficient path in terms of least movements possible; and the latter provides an extra level of safety by moving farther from the obstacles. In both cases, the goal is to cover the distance from the starting point to the finish point avoiding the obstacles on the way, both of these positions are fixed. The experiments revealed that the Q-learning requires less learning time in comparison to the Sarsa algorithm. However, the Sarsa method learned safer paths towards the goal. Therefore, both proved to be useful and could be applied in different circumstances, depending on the goals. However, the limitation this algorithm presents resides on the fact that it can only find the path between a fixed starting and finishing point. Thus, if these points change, the training process has to be done all over again.

The authors from [10] present an Improved Q-learning (IQL) algorithm that introduces a new exploration strategy. This method avoids finding the local optimum values, meaning that the Q-values are more accurate, and minimises the time needed for the Q-values to converge. The search heuristic used considers the current position of the device and the goal position in order to (randomly, when exploring the environment) select an action that gets the device closer to the goal. That is, if the goal is somewhere above and to the right of the current position, the algorithm chooses to go straight, go to the right diagonal or go to the right. Therefore, it does not consider the actions that will move the device farther from the goal. The proposed IQL proved to need less time for planning the path to follow in comparison to the classical Q-learning method or A*. Moreover, it was also shown through some experiments that this algorithm could be adapted to other environments (with different obstacles, distinct starting points and even moving impediments).

Another improvement of the Q-learning algorithm for path planning is introduced by the authors of [26]. In this case, only the best action is stored in the position of the Q-table corresponding to each state. This solution heavily reduces the dimensionality of the table and, as a consequence, the time required by the algorithm to converge onto the solution. Additionally, an effort for decreasing the 90^{o} turns is also done, which also reduces the energy consumption of the mobile robot. For this method to work, the algo-

rithm requires to know the distance from the current state to the next state and to the goal. Another limitation of this algorithm is the fact that the goal has to be fixed, since only the best action for each state is stored. This implies a new training process every time the goal position changes. All in all, this version of Q-learning outperforms the classical approach.

2.1.2 Approximate solutions

As aforementioned, when the state and action spaces increase in size, the convergence time dramatically increments as well –not to mention the memory usage– and it can even prevent the algorithm to find a solution in a feasible time. For this purpose, the usage of Neural Network (NN) [19] is introduced in order to provide an approximated solution. Furthermore, in some cases, since the agent needs to process raw input data, such as images, the usage of NN (CNNs [27] in particular for these cases) makes sense.

In [28] the goal of the presented algorithm is to autonomously move a mobile robot throughout a virtual environment and try to collect as many apples as possible (which are equivalent to 1 point), while avoiding the lemons (-1 points), in a limited amount of time. An agent composed by a Deep Q-Network (DQN) is trained to approximate the state-action value function by means of recognising an RGB input image as the current state of the mobile robot (i.e. a CNN). The network outputs the Q-value for all the possible actions from that state. Finally, the intelligent system selects which action to perform in order to move around the environment while maximising the points achieved. This method is done end-to-end inside the agent. After the network converges, the algorithm is tested a total of 30 times and the average obtained score by the agent is 10.03 (out of a maximum of 15 points), which proves that the algorithm indeed works.

The authors of [29] propose a Double DQN (DDQN) to address the path planning problem in an unknown environment. In this case, the robot uses a CNN to process the information received by the lidar sensors mounted on the device. The goal of the robot is to move from a starting point towards a global target position, but the sensors can only perceive information up to a certain range. This issue is solved by dynamically adjusting the target position to one that is inside the scope of the sensors. Hence, as the robot moves, the search space is expanded. Additionally, this algorithm is also proven to work with dynamic obstacles. The training procedure is done in different dynamic environments. Finally, the testing (and experimentation,

with Robotic Operating System or ROS) stage is performed in a new environment, showing that the agent is capable of reaching the target destination in an unknown environment.

In another example, the work in [30] proposes a Double Deep Q-Network with prioritised experience replay (DDQN with PER) for path planning in an unknown environment. In this case, when using PER [31], only more or less recent experiences are considered for the NN updates, which stabilises the learning procedure. The training procedure took place with six distinct scenarios: three of them had the same number of fixed obstacles and an increasing number of randomly placed target points; the other three had several fixed target points and different number of randomly placed obstacles. When comparing the behaviour of the DDQN with PER against a normal DQN, the new method proved to converge faster and its success rate was higher as well.

2.2 Energy awareness in path planning

As briefly introduced beforehand, the goal of this project is to implement an energy-aware path planning algorithm that takes into consideration the current battery state in order to determine which route to take. Energy-awareness in path planning is not a new field, since path planning itself usually tries to find an economical route (although energy is not usually explicitly taken into account). However, the shortest or fastest path is not always the less energy-consuming way. For instance, given the same distance to drive, the battery required to reach the summit of a mountain is much greater than just driving a straight line. Therefore, it is very important to consider which are the characteristics of the path the autonomous vehicle has to drive in.

It is obvious that the vehicle control (avoiding continuous rotations, or sharp turns, etc.) also has a saying in the energy efficiency of the path, as seen in for example [26], but for our case study, only the route planning will be considered.

Following the fact that energy consumption increases when more data needs to be processed, the project in [32] tries to avoid traversing nearby obstacles in the environment. Their proposed method maps energy areas on the environment, assigning more cost to the areas that are immediately surrounding the obstacles. Therefore, the agent learns to avoid moving too close to the obstacles, which implies the reduction of data sampling by the

sensors –since the distance to the object is still safe, there is no need to monitor the movements as frequently–, and this heavily reduces the data processing performed by the intelligent device. In order to test the idea, an experiment using the ROS framework was conducted, showing that the energy consumption was reduced approximately 52% in comparison to the conventional path planning method used by ROS.

A study on uneven and non-planar environments is presented in [33]. The authors introduce an energy-aware path planning algorithm that estimates the driving energy consumption by means of analysing the 3D geometry of the terrain the robot has to move through with a Deep Neural Network (DNN). For this, only the points that are in the trajectory of the wheels are analysed, since they are the only ones that will affect the driving energy. The DNN is capable of autonomously learning how to associate energy consumption to certain terrain types and estimate the total energy consumed over a planned trajectory, which improves the energy-efficiency of the planned path by choosing better-suited pathways. By means of simulation, using data collected from real scenarios, they establish that their method can predict routes that might reduce the energy consumption by a 5.5% in comparison to other path planning methods.

In the context of path coverage, [34] presents a solution that considers the windy conditions of the environment in order to plan the path of a drone (an Unmanned Aerial Vehicles, UAV). Their goal is to maximise the battery life of the UAV to increase the number of goals detected by the robot. In this case, using a simulated environment, they implement a variation of the Q-learning algorithm that considers 3-dimensional environments (using a Q-matrix). By integrating the wind currents on the planning algorithm, the presented method allows to detect from twice to four times, depending on the wind conditions, more goal objects than the normal coverage path planning algorithm.

Even though some research, such as [35], use a battery model derived from real measurements, to the author's knowledge, there is no work that considers real-time battery measurements in the context of path planning. This aspect will be tackled in this project.

2.3 Trajectory data inclusion in path planning

Another goal of this project is to take advantage of the trajectory data available on the cloud that has been generated and collected in other drives, either

by other vehicles and drivers or by the same driver/car.

Research, such as the works presented in [36] and [37], has proven that considering the paths taken by experienced taxi drivers generates a much more optimal route in terms of time, especially in peak hours, when compared against the result of traditional path planners.

This occurs because normal path planning algorithms do not consider the local road characteristics. That is, the information about traffic jams or traffic lights –and its waiting times– are not taken into consideration when planning the route to follow.

Therefore it would be interesting to take into consideration this data as well when planning a route between two points.

2.4 Unexplored areas

Even though there has been a lot of research in the topic of path planning, some questions still remain unanswered. For instance, no previous project seems to include the real-time data obtained from the Battery Management Systems (BMS) in order to determine if a path is feasible or not.

Similarly, the integration of both previous trajectory data and battery measurements into the path planning algorithm seems to still be unexplored. Most of the time, only one of the two is considered, but not both at the same time.

Furthermore, most of the projects centre the measurements and analysis only on a small-scale platform, but no analysis is performed afterwards in order to find out how the observations drawn from the experimentation scale up in a bigger platform.

This project aims to address these unexplored areas.

3 Problem statement

Currently, no path planning solution considers the real-time Battery Management Systems (BMS) measurements, such as the State-of-Charge (SOC), which determines the range of the electric device. In general, SOC is a very important indicator for Electric Vehicles (EV), but in the context of autonomous driving, it might determine if a planned path is feasible or not. For instance, if only the shortest path is considered, the method could determine that the best route to follow has to go across a highway, which typically implies a higher energy consumption due to higher speeds. However, a secondary road might be presented as a more energy-efficient alternative and could require a bit less energy to traverse, while only increasing the driving time a bit. In situations where the SOC is critical, taking one road or another could mean a remarkable difference in the final battery charge.

Considering that trajectory data is available from previous drives –either made public in the cloud or locally in the car– (i.e. GPS coordinates, speed, SOC consumed, time, etc.), it is one of the research questions to determine if, by combining the path planning with real battery measurements and trajectory data, the AI algorithm actually determines a more efficient way (in terms of energy consumption) to reach the target.

Due to safety reasons, and in order to limit the scope of the project, the practical experiments will be carried out with a small-scale platform: the iRobot Create 2 (see Section 5.1).

Furthermore, another aspect to analyse is how the measurements and observations obtained in the small-scale environment extrapolate to an EV. To assess this, the vehicle Citroën C-Zero, described in Section 5.2, will be used to see how the situation scales up and address this way the second research question.

These two research questions can be further divided into different milestones. Namely,

1. **Feasibility study**: the first step is to verify that all the hardware and platforms, as well as the technologies involved in the project are indeed suitable for investigating the presented problems.

2. **AI path planning algorithm**: implement an AI path planning algorithm that learns by means of trial-and-error (DRL) the correct movements needed to be performed in order to reach random target points in the simulated environment.

Initially, the algorithm will not consider the energy consumed in the movements and it will only learn the shortest paths between the points. In a second stage, an approximation of the energy consumed in the different routes –which will vary according to the terrain type and maximum speed allowed– will be estimated and used in order to determine which paths are actually more energy-efficient.

3. **Collect trajectory data on small-scale environment**: once the path planning algorithm works as expected, consumption data will be collected in a small environment. In this stage, an indoors environment will be prepared with different terrain types (e.g. steep pavement, not regular ground, etc.), in which the small-scale robot (see Section 5.1) will drive, by means of remote control. During this process, the trajectory data (i.e. SOC consumption, time, etc.) will be collected by means of ROS bagfiles and the NavQ board.

4. **Include trajectory data in the path planner**: after covering the indoors environment, the gathered data will be used in order to determine if the energy-aware path planner is capable of finding improved routes when comparing to the shortest path available. Thus, an analysis will be carried out to determine if there is any gain in terms of energy efficiency when considering both the battery usage and previous trajectory data.

5. **Collect driving data, extrapolate results and draw conclusions**: in parallel, the full-sized platform (Citroën C-Zero, see Section 5.2) will be used to collect information about various driving trajectories, including different terrains, to be able to extrapolate the observations drawn with the mobile robot. This way, the experiment results will be generalised for an EV. Finally, conclusions regarding the effectiveness of the path planning algorithm will be extracted from both experiments.

4 Proposed approach

As stated in Section 3, there are two main research questions. The first one is to analyse how the inclusion of previous trajectory data and real BMS measurements influence an AI path planning algorithm, and determine if there is any gain obtainable by considering this information. In other words, establish if, by having information about the routes –average estimated time, energy consumption, etc.–, the algorithm makes wiser (energy-efficient) decisions that might not be straightforward.

The second question refers to how the observations from the small-scale platform extrapolate to an electric vehicle.

To address the first issue, two Q-learning path planning algorithms will be implemented and their performance is going to be analysed. Additionally, a small-scale platform will be used to practically evaluate the behaviour of the path planners in a laboratory scale.

Regarding the second research question, an electric vehicle will be used to collect real driving data. This information will later be used to evaluate how the path planner algorithms scale up.

The proposed approach can be therefore divided into three big tasks. First, the implementation and simulation of the Q-learning path planners. Secondly, the small-scale experimentation. Lastly, the extrapolation of results in an EV.

4.1 Q-learning path planners

To solve these questions, two DRL path planning algorithms will be implemented. Throughout the literature study, numerous projects used this programming language for both the simulation of the environment and the algorithm implementation. Therefore, it is clear that this technology is useful for these purposes.

The chosen approach is based on a Q-learning method [24] that autonomously learns what actions (i.e. in which direction to move the robot) to perform from the current state in order to reach the desired destination (i.e. coordinate points). This is done by means of trial-and-error and awarding the algorithm a reward after certain states are reached.

Since the first goal is to determine if there is a gain by considering the trajectory data of the routes, one of the algorithms will learn the shortest path between any combination of source-destination (Q-LSP). On the contrary,

the second path planner –Q-learning energy-aware (Q-LEA)– will take into account the route that consumes less energy between two points. Comparing these two alternatives shall provide a response to the first research question.

Therefore, the desired behaviour of both path planning algorithms can be described as follows. Given the scenario in Figure 2, two paths can be taken to go from A to B (or vice versa), labelled as P1 and P2. P1 has the shortest distance to the destination (but highest SOC consumption), whereas P2 has a lower energy consumption (and longer distance travelled). The Q-LSP algorithm will learn that the best route –the shortest one– is P1. Nevertheless, the Q-LEA version will learn that P2 is the most energy-efficient one.

Figure 2: Example scenario with two possible routes between A and B that have different distances and energy consumption.

More in-depth details about these algorithms, including the pseudo-code, the state and action spaces, the rewards and the environment representation, are given in Section 7.

4.2 Practical evaluation: small-scale experiment

Once the implementation of both algorithms is done, the practical stage can begin. This stage can be further split into three sub-stages.

4.2.1 Small-scale environment creation

First and foremost, the testing environment has to be created in a room. Initially, Ultra-Wideband (UWB) technology (see Section B.1) was going to be used to locate the mobile robot for the autonomous movement. However, automation was discarded in the end due to communication issues. Nonetheless, ROS framework is used to collect relevant data from the mobile robot.

To allow the usage of ROS, the device has to be equipped with a NavQ board, which enables its remote control (via joystick) and data storage (for the bagfiles).

Additionally, the floor of the room has to be temporary modified in order to allow different terrain types. For instance, a carpet can be placed on the floor, which will modify the drag coefficient.

4.2.2 Trajectory data collection

Using the mobile robot, the idea is to collect information about battery usage and trajectory data in the small-scale environment. This data is later used together with the path planners to evaluate if they find the shortest and most energy-efficient routes.

4.2.3 Planning paths with Q-LSP and Q-LEA

Once the trajectory data is collected in the small-scale environment, the data will be processed and adapted so the path planners can make use of it. The algorithms will train in this new environment and will determine which routes are the shortest between any two points (Q-LSP) and which are more energy-efficient (Q-LEA).

At the end of this task, sufficient data should be available to determine if the energy consumption usage plays an important role in the path planning algorithms.

4.3 Extrapolation of results in an EV

Using the platform introduced in Section 5.2, the goal is to collect trajectory data –time, battery consumption, paths traversed, etc.– in different environments, such as asphalt roads and grind roads, as well as at different speeds, to be able to analyse which are the conditions that increment the battery consumption.

Moreover, a region of Waalre will be selected to be driven. Several paths will be established in this area and the information collected during the drives will be used to evaluate the performance of the path planners when using real driving data. This way, the literature gap that does not cover how the small-scale experiments would behave in a larger scale is expected to be covered.

4.4 Project restrictions

Initially, two options were considered for the small-scale experimentation. Nonetheless, only the iRobot Create 2 is currently available, which restricts the scope of the experimental setup to indoors environments.

Other restrictions derived from the usage of this robot are the battery capacity and the driving range. It is estimated that the iRobot can drive consecutively for more or less 45 minutes. This directly limits the distance and total time the robot can travel.

Regarding the EV, the range of the Citroën C-Zero is limited to 100 km. Even though both urban environments and high speed roads will be contemplated in the experiments, it is important to consider the vehicle range when designing the test drives to avoid running out of battery.

4.5 Project modifications

Initially the intention was to develop a proof-of-concept of the energy-aware algorithm with the iRobot Create 2 platform. However, due to unforeseen issues with the Wi-Fi connection, that led to package loss and discontinuous robot motion, the autonomous feature was unfortunately discarded.

Therefore, the problem statement had to be modified accordingly. In this case, the mobile robot was used and driven with a remote control, which still allowed the collection of data in the environment.

It is left as future work to implement the autonomous motion.

5 Experimental setups

As briefly introduced before, the project will include two experimental setups. Namely, a small-scale –the iRobot Create 2– and a full-sized vehicle (Citroën C-Zero). In this section, more information regarding these two environments is given as a basis for the project.

Figure 3 shows these two platforms.

(a) iRobot Create® 2 Programmable Robot.

(b) NXP's Citroën C-Zero vehicle.

Figure 3: Experimental platforms used in the project: small-scale setup (left); EV setup (right).

5.1 Small-scale platform: iRobot Create 2

The chosen platform for this small-scale experimentation, the iRobot Create 2[1], is a programmable roomba that can be armed with different devices through its peripheral ports, providing a lot of flexibility.

In particular, the device is equipped with a NavQ board which acts as the brain of the mobile robot, which in combination with Robot Operating System (ROS) framework, allows a complete and precise information reporting mechanism.

5.1.1 NavQ board

The 8MMNavQ or NavQ board[2] is a small Linux computer specifically built to satisfy the common needs of Mobile Robotics systems. Additionally, it

[1] `https://edu.irobot.com/what-we-offer/create-robot`
[2] `https://nxp.gitbook.io/8mmnavq/`

allows easy customisation.

The NavQ is suitable for projects that range from vision-processing systems to more general implementations. Moreover, it is equipped with ROS framework, OpenCV, TensorFlow and other libraries that enable common robotic applications. Since the board is running a Linux OS, the installation of additional packages is also possible.

Table 1 summarises the main features of the NavQ processor.

	CPU, GPU and DSP	**HMI and multimedia**	**GPU Libraries and Extensions**	**Interfaces**	**Memory Types**
i.MX 8M Mini	4 Cortex-A53, 1 Cortex-M4F and 1 GPU	1x MIPI DSI, 1x MIPI-CSI, **Video decode**: 1080p60 (h.265, VP9, h.264, VP8), **Video encode**: 1080p60 (h.264)	2.0 OpenGL® ES	1 PCIe, 1 Gigabit Ethernet	LPDDR4, DDR4, DDR3L

Table 1: NavQ i.MX 8M Mini processor features.

5.2 Electric vehicle: Citroën C-Zero

This platform, which has been thoroughly tested in the company before, consists of a Citroën C-Zero electric vehicle that has been modified with different components in order to enable testing and using different technologies developed by NXP.

The relevant components of this vehicle for this project are the on-board Linux computer, which is also equipped with the ROS framework and the GPS sensor that is mounted on the car. Additionally, the information received through the CAN bus of the car is also published in a ROS topic and its information is collected by the computer.

By combining these elements, the computer collects and generates bagfiles –by means of the ROS middleware– that contain all the relevant information about the test drive. Using this data, the whole drive can be reenacted. Furthermore, the car is also equipped with a camera that records the drive and the information on the dashboard, which can provide additional insights on the ride.

Section 5.3.2 provides more details about the ROS topics used to collect the information. Further information about the data processing is given in Section 6.2.1.

5.3 Robot Operating System (ROS)

ROS[3] is an open-source robotics middleware suite that enables robotics programming. Among others, it provides low-level device control and message-passing, which are paramount for the realisation of this project. Both platforms (iRobot and C-Zero) are compatible with this framework.

In particular, ROS can control the movement of the robot –for example, the direction, speed, position– and it allows, for instance, sniffing the CAN bus messages, which comprise relevant information about the device, such as the State-of-Charge (SOC) or remaining range.

ROS was designed in such a way that it allows a high-level of personalisation and adaptation of the software so it suits the final application better. Figure 4 shows an example of the network conformed by two ROS *nodes* (or processes), depicting the main relevant components of this framework.

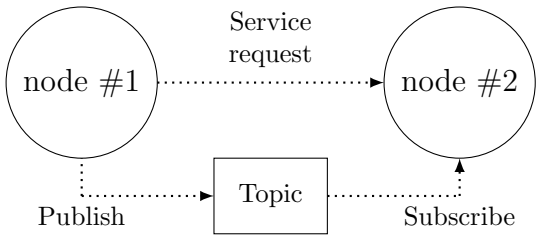


Figure 4: ROS network with two nodes and one topic.

Nodes can either perform actions and/or communicate with other processes. For instance, a process can request a service from another node, which will generally perform an action with a defined beginning and end and will provide a result (e.g. performing a calculation).

Additionally, nodes can also communicate by exchanging messages. *Messages* can contain practically anything from sensor data to control commands. These messages are transmitted by means of *Topics*, which are uniquely named buses that connect nodes. The processes can send a message to a

[3] `http://wiki.ros.org/`

topic by publishing to it; similarly, they can receive messages from that topic after subscribing to it.

All the exchanged messages between any two nodes can be stored – including the time stamp– in *bagfiles* for later processing.

5.3.1 Small-scale ROS topics

The ROS node present in the NavQ board allows data collection by means of the available topics[4]. Table 2 gathers the main topics used in this small-scale experiment, which are primarly related to the device battery usage.

ROS topic	Description
/battery/charge_ratio	Reports the SOC of the mobile robot
/battery/current	The current charge of the battery (Ah)
/battery/voltage	Voltage of the robot's battery (V)

Table 2: Relevant ROS topics to collect data with the mobile robot.

5.3.2 EV ROS topics

As aforementioned, the EV used in this project is equipped with ROS framework. In this platform, some topics are already specified, but the relevant ones for this project are summarised in Table 3.

ROS topic	Description
/received_messages	Contains the messages received via the CAN bus
/sensors/gps_extended	Contains localisation information (e.g. GPS coordinates)
/sensors/gps	Contains localisation information (e.g. GPS coordinates)

Table 3: Relevant ROS topics to collect data with the electric vehicle.

In this case, all the information related to the driving parameters, such as vehicle speed, State-of-Charge, battery voltage, odometer, etc. are received through the CAN bus, which is sniffed by the "/received_messages" topic.

Moreover, the GPS coordinates are collected by means of either the "/sensors/gps_extended" or "/sensors/gps" topics.

Controller Area Network (CAN bus) is a communication standard that allows connectivity between micro-controllers without relying on a host com-

[4] `https://github.com/AutonomyLab/create_robot`

puter. Its usage is widely spread in vehicles as it enables the communication with the electronic control unit(s) (ECU).

As expected, a lot of different messages are being sent using CAN bus, and only a few of them are relevant for the realisation of this project. Table 4 contains the messages [5] [6] that will be collected and analysed/used in this project.

Description	**Units**	**PID (hex)**	**Data**
Speed	km/h	215	(256*byte(0) + byte(1))/128.0
Battery volts	V	373	(byte(4)*256 + byte(5))/10.0
State of Charge 1	%	374	(byte(0) - 10.0)/2.0
Odometer	km	412	(byte(2)*256 + byte(3))*256 + byte(4)
Motor current	amps	696	((byte(2)*256)+byte(3) - 500)/20.0

Table 4: Relevant CAN messages from Citroën C-Zero.

[5] `https://github.com/BITPlan/can4eve` – "Triplet.json"

[6] `http://myimiev.com/forum/viewtopic.php?f=25&t=763&start=120`

6 Feasibility study and initial results

In this section, the experimental platforms that will be used in the project are thoroughly tested to assess if they can indeed be used in this context and no unexpected situations arise, avoiding as many hurdles as possible.

6.1 Small-scale experiment platform: iRobot

To assess the usability of the small-scale platform –iRobot Create 2–, two factors have to be addressed. Our goals require to be capable of remotely being able to move the mobile robot and using the ROS framework. Additionally, the platform has to be capable of collecting real-time navigation data (location, speed, time, etc.) as well as battery usage information.

Therefore, the feasibility analysis of this platform can be split in these two aspects.

6.1.1 Mobile robot movement and location

Initially, the idea was to be able to autonomously move the mobile robot and, for this, the intention was to use UWB technology (see Section B.1) to be able to locate the device indoors. For this stage, a remote control was used to allow remote movement of the robot. Figure 5 represents a schematic view of this setup.

For these purposes, an UWB anchor (antenna) is placed on each of the corners of a room (four antennas in total). Additionally, two other nodes are connected to the NavQ board – that is attached to the iRobot Create 2. On the other hand, a joystick is connected to a laptop, which will enable the remote control of the mobile robot.

To allow communication (via ROS topics) between the laptop and the robot, two ROS nodes are set: one in the laptop that will send the movement commands and the other on the NavQ board, which will process the commands and act accordingly so the roomba moves as expected. In this case, the NavQ board acts as the ROS master and the laptop as the ROS slave.

As the mobile robot moves across the room by means of the commands dispatched via the joystick, the node from the robot keeps exchanging messages with the anchors. The NavQ board collects the responses received from the anchors and, by means of a Python script, it calculates the distances to

the antennas. The location of the robot can also be plotted, as shown in Figure 6a, where the iRobot is represented with a coloured dot that corresponds to the area it is located in.

In order to obtain this location, the exact position of the anchors has to be known. Furthermore, a triangulation between the intersection of the anchors waves is performed in order to establish the precise position.

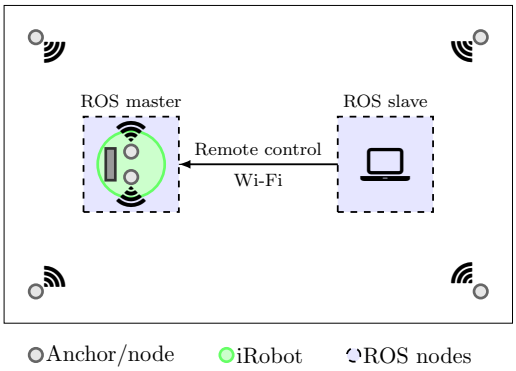


Figure 5: Room setup with four UWB anchors; the iRobot Create 2, equipped with the NavQ board and two UWB, which has a node acting as the ROS master; a laptop with a joystick attached acting as a ROS slave.

The idea was that by the end of the project, all the commands would be dispatched by the NavQ board –which executes the path planning scripts–, meaning that no additional components would be needed (e.g. a computer), achieving autonomous movement this way. Unfortunately, when trying to achieve this, communication errors derived from the Wi-Fi connection tampered the smooth flow and represented an issue for autonomous movement.

To be able to conduct the rest of experimentation with the mobile robot, the remote control and computer setup was used. Therefore, the UWB nodes and anchors were not really used.

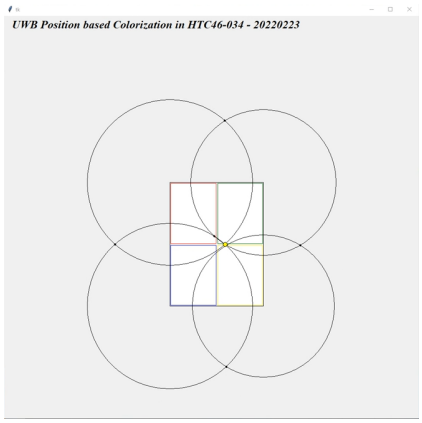
6.1.2 Data collection in the robot

Using the ROS node associated to the NavQ board, the data collected from the iRobot can be stored in bagfiles using the ROS topics as described in 5.3.1.

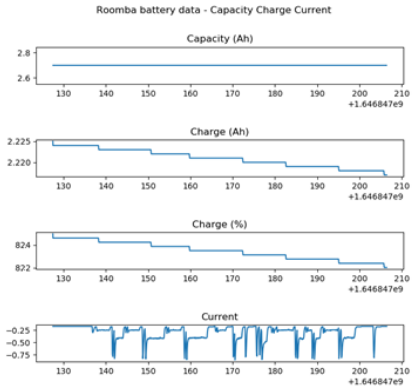
This way, the localisation data (Figure 6a), battery usage (Figure 6b) and other driving parameters can be stored for later processing and comparison.

This information can be sniffed by means of the ROS framework and stored in a bagfile.

Using a Python script, similar to the one that is used in Section 6.2.1, the raw data can be plotted as shown in Figure 6b, proving that battery data can be indeed accessed, stored and processed in this platform.



(a) Python visualisation of the mobile robot localisation test using UWB technology.



(b) iRobot Create 2 battery data collected in a test drive: capacity and charge in Ah (first and second plots), charge (third plot, in %) and current (last plot, in A).

Figure 6: iRobot Create 2 feasibility study results: plot of the localisation experiment (left) and battery data collection plots (right).

6.2 Data collection in the Citroën C0

For the second part of the project, it will be necessary to collect trajectory data with the EV (Citroën C-Zero or C0). This platform is fully equipped with components that allow the recording of GPS coordinates, CAN messages and other valuable information. This is done by means of an on-board Linux computer that controls multiple ROS nodes and topics, which are specified in Section 5.3.2.

6.2.1 Trajectory data processing

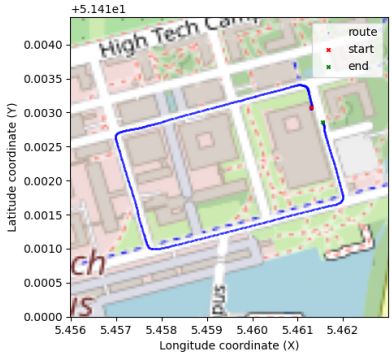
The data collected with the electric vehicle is stored in a `bagfile` document an uploaded to an internal NXP server. A Python script was implemented as

part of the feasibility study in order to allow the visualisation and processing of this data.

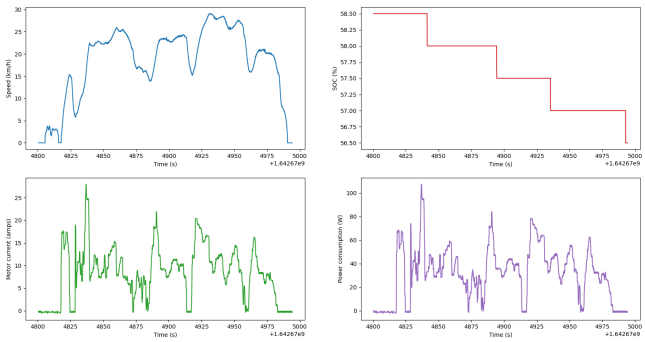
After a short test drive inside the High Tech Campus (HTC) of Eindhoven, the bagfile data is processed with the script in order to determine if it is usable or not. As a result, the GPS coordinates of the path followed by the vehicle are shown in Figure 7a. Furthermore, some of the driving parameters (i.e. speed, SOC, motor current and power consumption) are also read, or derived in the case of the power consumption ($W = V \cdot A$), and plotted in Figure 7b.

In general, the Python script and the data stored in the `bagfiles` provide a thorough insight on the driving tests, which will definitely help with the extrapolation of the data obtained in the smaller-scale experiment.

Furthermore, it is also possible to review the drive by means of all the data gathered during the test, including the camera images, which provides the possibility of double checking the reported data.



(a) GPS coordinates of the path followed in the drive.



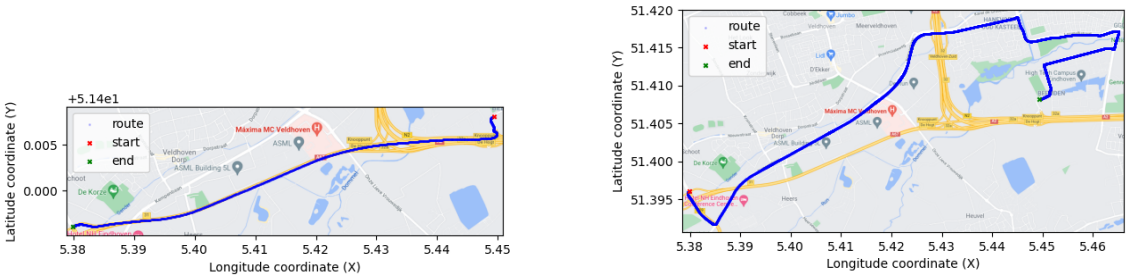
(b) Speed of the vehicle in km/h (top-left); State-of-Charge in % (top-right); Motor current in amps (bottom-left); and Power consumption in W (bottom-right).

Figure 7: Test drive trajectory data: GPS coordinates (left) and driving parameters –speed, SOC, motor current and power consumption– (right).

6.2.2 Citroën C0: initial results

To test the stability of the system, two more drives were performed. In this case, the first drive started in Eindhoven's HTC and finished in Oeienbosch, following the highway, as plotted in Figure 8a. The second route started

in Oeienbosch and finished in Eindhoven's HTC, taking urban roads in this case, following the path depicted in Figure 8b.



(a) Drive test 1: High Tech Campus to Oeienbosch through the highway.

(b) Drive test 2: Oeienbosch to High Tech Campus through urban roads.

Figure 8: Alternative roads between Oeienbosch and High Tech Campus: through the highway (left); or urban roads (right).

Table 5 contains some of the relevant data extracted from the trajectory recordings from Figure 8. Even though it is clear that the routes followed differ in a number of factors –distance, time, type of roads, etc.–, these drives are alternative options to reach the desired destination.

Origin	Destination	Type	Time	Distance	Battery cons.
HTC	Oeienbosch	Highway	0:04:37	5 km	10.5%
Oeienbosch	HTC	Urban	0:17:44	10 km	12%

Table 5: Relevant data collected in the different test drives with the Citroën C-Zero.

In these drives, the consumed state-of-charge (SOC) does not differ greatly, with a SOC reduction of 10.5% in the case of the highway drive (Figure 8a). On the contrary, the urban path (Figure 8b) required a 12% of the total SOC.

In the highway, the energy consumption escalates quickly due to the high speeds achieved and maintained, which presents a disadvantage in comparison to the other test drive. This small experiment suggests that it might be possible to find an alternative path to the highway route that consumes less energy and, if time to destination or distance travelled are not relevant, might present a better option in terms of energy-efficiency.

7 Energy-aware path planning algorithm

One of the purposes of this project was to develop an artificial intelligence path planning algorithm that, by means of available data from previous drives, was capable of determining an energy-efficient route between any combination of starting and finishing points (which for simplicity is referred to as Q-LEA path planning algorithm). In order to assess the value of including trajectory data, the results of this method had to be compared to another path planner. For the comparison, a path planner that determines the shortest route was chosen (called Q-LSP).

In this section, the details behind the virtual environment creation and simulation of different road types/speeds are given. Additionally, the algorithm characteristics of both Q-LSP and Q-LEA are also detailed. Later, the proposed routes by each path planner are analysed.

7.1 Virtual environment creation

Considering different road types and speeds plays an important role in the driving energy of the vehicle. Therefore, the virtual environment had to acknowledge these properties as well.

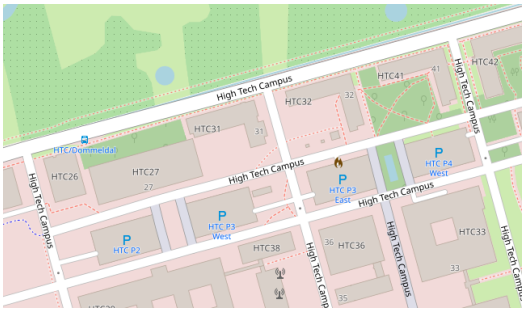
To create a more realistic scenario, the MATLAB tool Driving Scenario Designer[7] was used. Among other things, this tool can create simulating environments with real data from Open Street Map (OSM)[8].

Therefore, a small portion of the Eindhoven High Tech Campus was selected with the tool and the road coordinates were extracted with this application. Figure 9a depicts the OSM data from the chosen area. Similarly, Figure 9b shows the imported data in MATLAB. Later, this data was added in a Python script and processed in order to determine different starting and finishing points, as indicated in Figure 9c. In this case, the road intersections were used as viable points.

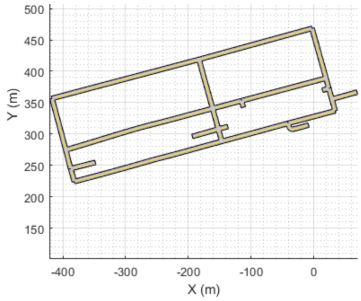
For the simulation of different road speeds, each segment can be assigned a speed value (with a maximum of 40 km/h since it is an urban environment), as shown in Figure 9d.

[7] `https://nl.mathworks.com/help/driving/ref/drivingscenariodesigner-app.html`

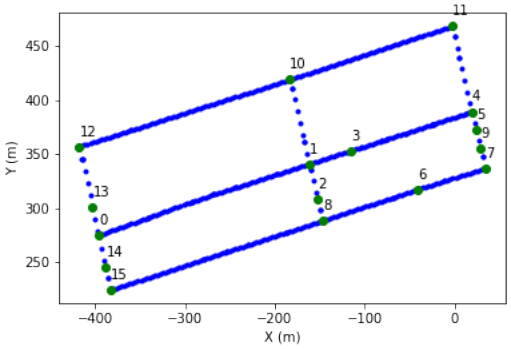
[8] `https://www.openstreetmap.org/`



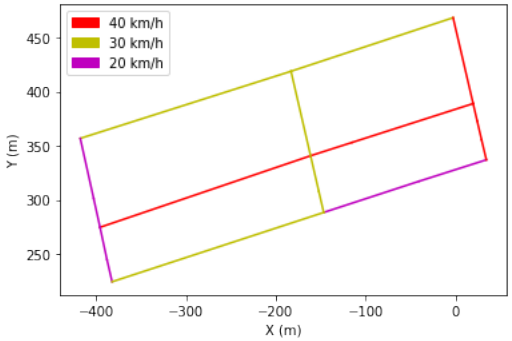
(a) Eindhoven High Tech Campus (HTC) selected area.



(b) HTC OSM data imported with Driving Scenario Designer.



(c) HTC OSM data processed with Python.



(d) Maximum road speeds of the environment.

Figure 9: High Tech Campus area selected (top-left) and its road data imported in MATLAB (top-right). Python processed road segments (bottom-left) and the segment speeds (bottom-right).

Furthermore, diverse road friction coefficients can be contemplated to simulate different terrain types. Table 6 summarises the road types considered in this simulation, which values were calculated in past research [38] at a driving speed of 40 km/h.

Road surface type	Road friction coefficient
Asphalt or concrete	0.8
Gravel	0.7
Unsurfaced road	0.6

Table 6: Friction coefficients for different road types, based on the study presented in [38], which were determined at a driving speed of 40 km/h.

If these parameters –road friction and speed– are added to the road data extracted from the environment, the final road properties are as shown in Table 15 from Appendix A.

For ease of explanation, some of these roads are summarised in Table 7. The energy consumption estimation ('Cons.' column) is detailed in the next subsection.

Source	Destination	Drag coeff.	Speed (km/h)	Dist. (m)	Cons. (J)
0	1	0.8	40	243.30	13094.14
...					
12	10	0.7	30	242.42	9948.73
12	13	0.6	20	57.51	1635.60
13	0	0.6	20	27.60	785.04
0	14	0.7	20	30.74	885.32
14	15	0.7	20	21.37	615.59
15	8	0.7	30	243.88	10008.99

Table 7: Details of some of the environment roads.

In this table only one way of the road is represented. Nonetheless, roads can be driven both ways. That is, from intersection 0 to intersection 1, as well as from intersection 1 to 0. For simplicity, the consumption is the same in both directions since the terrain elevation is not considered (due to minimal change).

7.2 Simulating energy consumption

The last component needed for our simulation is the energy consumption of the vehicle when driving the environment roads (Figure 9). For simplification purposes, the roads are driven at constant speed, meaning that the acceleration component does not have to be considered. Moreover, the vehicle always drives at the maximum allowed speed (as specified in Figure 9d), which determines the maximum energy consumption expected in that road segment. Lastly, the battery regeneration produced by the brakes is also ignored.

Therefore, a simple approach to compute the electrical power (in J/s, or W) consumed could be the following

$$\text{Electrical power} = \beta v + \gamma v^2 + k \text{ J/s}$$

Where β represents the road type drag coefficient, as shown in Table 6. Additionally, γ is the air drag coefficient, which is specific for the car model

and can be calculated with the formula specified in [39]. For the Mitsubishi i-MiEV (very similar to the Citroën C-Zero) it is 0.35. Finally, k represents the constant energy consumed by the vehicle when it is on, which for our EV is estimated to be 6 Joules. If these values are substituted, the formula can be further simplified as

$$\text{Electrical power} = \beta v + 0.35v^2 + 6 \text{ J/s}.$$

When multiplying the electrical power by the time needed to cover the road segment in question, the electrical energy is derived for this road (in J).

Taking as an example the roads that go from intersection 0 to 1 (R0-1) and the one from intersection 12 to 10 (R12-10), which have more or less the same distance, but different speeds and drag coefficients (see Table 7), the power needed to go over them can be estimated as follows:

$$\text{Electrical power(R0-1)} = 0.8 \cdot 40 + 0.35 \cdot (40^2) + 6 = 598 \text{ J/s}.$$

$$\text{Electrical power(R12-10)} = 0.7 \cdot 30 + 0.35 \cdot (30^2) + 6 = 342 \text{ J/s}.$$

Considering the distance and speed of these roads, the time ($T = dist/speed$) to go from one intersection to the other is 21.8966 and 29.0898 seconds for R0-1 and R12-10, respectively. Thus, their energy consumption is

$$\text{Electrical energy(R0-1)} = \text{Electrical power(R0-1)} \cdot T = 598 \text{ J/s} \cdot 21.8966 \text{ s} = 13094.14 \text{ J}.$$

$$\text{Electrical energy(R12-10)} = \text{Electrical power(R12-10)} \cdot T = 342 \text{ J/s} \cdot 29.0898 \text{ s} = 9948.73 \text{ J}.$$

Which seem to correlate with the road properties of each segment. The energy required for each road is also specified in Table 15.

7.3 Algorithm properties

As aforementioned, the implemented path planning algorithms are based on the Q-learning technique. This decision was prompted due to the fact that we wanted to limit the human intervention in the decision making as much as possible (i.e. avoid supervised learning). Moreover, the purpose was to have algorithms that could be easily adapted to other environments (model-free). Finally, the usage of approximate solutions was ruled out in an attempt to look for a method which functionality was easier to understand – essential for debugging purposes.

Both algorithms (Q-LSP and Q-LEA) follow the pseudo-code shown in Algorithm 1 –which is explained in detail in the sections to follow– and have

the same training process. Their main difference resides in how the agent is awarded after taking the best action, whether it is shorter distance or less energy consuming, respectively (line 11, *getReward(...)*).

Algorithm 1 Q-LSP and Q-LEA path planners pseudo-code.

```
 1: qTable ← initialiseTable(0)
 2: training_done ← false
 3: while ! training_done do
 4:     A ← startPoint
 5:     B ← endPoint
 6:     episode_end ← false
 7:     while ! episode_end do
 8:         currentState ← getState(A, B)
 9:         action ← getAction(currentState, qTable)
10:         newState, A ← performAction(currentState, action)
11:         reward ← getReward(currentState, action, newState)
12:         qTable ← updateQTable(currentState, action, newState, reward)
13:         episode_end ← isEpisodeFinished()
14:     end while
15:     updateEpsilon()
16:     training_done ← getTrainingDone()
17: end while
```

7.3.1 State and action spaces

The environment is represented by a number of interconnected roads. These connections, known as intersections (enumerated from 0 to 15 in Figure 9c), mark the start and end of a road segment. The particularity of these segments is that they are traversed from beginning to end (i.e. the vehicle cannot change its direction in the middle of the road).

Therefore, these intersection points can be used to determine the start position and the destination in the simulations. Since there are a total of 16 intersections –and start and goal positions cannot be the same–, the state space consists of 240 states, as computed below.

$$\text{Num. of states} = \#\text{ start points} \cdot \#\text{ finish points} = 16 \cdot 15 = 240 \text{ states}$$

Thus, the current state can be derived as follows,

$$\text{Curr. state} = (\text{start point} \cdot \#\text{ intersection points}) + \text{finish point}$$

Regarding the action space, there are a total of four actions possible from every intersection point: (0) going straight, (1) going to the left, (2) going to the right or (3) going back.

7.3.2 Q-learning implementation details

In the Q-learning method, the Q-table is used to keep track of the quality (or chances of success) of each combination of state-action. Initially, this table is filled with zeros (Alg. 1, line 1, *initialiseTable(0)*), as shown in Figure 8.

State	**Action**			
	0: Straight	**1: Left**	**2: Right**	**3: Back**
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
...				
239	0.0	0.0	0.0	0.0

Table 8: Q-table initialised to 0.

Then, the agent starts the training process, in which the values of the Q-table are going to be modified every time the agent takes a step (line 12, *updateQTable(...)*). This training takes place during a number of episodes –which finish every time a terminating state (collision or the target is attained) is reached–, particularly the Q-LSP and Q-LEA algorithms train for 100000 episodes. In each episode, the start and finish positions (they cannot be the same) are randomly defined (lines 4 and 5). While the episode lasts, the current state is defined (line 8, *getState(...)*) with the current and goal positions and an action is chosen (line 9, *getAction(...)*–following the ϵ-greedy policy, see Section 7.3.4). When the action is performed (line 10, *performAction(...)*), the state is modified with the new device location, which updates the current position A, and the reward the agent receives is determined (line 11, *getReward(...)*, see Section 7.3.3). Lastly, the Q-value is updated (line 12) using the Bellman equation below,

$$Q'(s,a) = (1-\alpha) \cdot Q(s,a) + \alpha \cdot (R + \gamma \cdot max(Q(s')))$$

and it is decided if the episode is over or not (line 13, *isEpisodeFinished()*). After the episode finishes, the value of ϵ is updated (line 15, *updateEpsilon()*), as specified in Section 7.3.4, and it is determined if the maximum number of episodes have been completed (line 16, *getTrainingDone()*).

In the formula, $Q(s, a)$ represents the old (or current) Q-value for the current s state and chosen a action to be taken. Similarly, $Q'(s, a)$ represents the new value of this combination of (s, a). Hence, s' represents the new state. Thus, $max(Q(s'))$ is the maximum Q-value from the next state s'.

This formula also considers the reward (R) received by the agent, the learning rate α (determines to what extent the new Q-value modifies the previous value) and γ (importance given to future rewards), which shape the rate of the learning process. The learning rate can range from $0 < \alpha \leq 1$. Similarly, $0 \leq \gamma \leq 1$. For both algorithms $\alpha = 0.1$ and $\gamma = 0.875$. These values were determined using a Python script that tested all the amounts between 0.1 and 0.9 for both variables, with a precision of 0.1, and then manually adjusted to obtain the best results possible.

7.3.3 Reward functions

The agent is rewarded every time the goal position is reached or the device collides (i.e. the device has gone out of route), as determined in Figure 10.

$$R(Q{-}LSP) = \begin{cases} -1 & \text{collision} \\ 0.5 & \text{goal reached, more distance travelled} \\ 1 & \text{first time goal reached} \\ 2 & \text{goal reached, less distance travelled} \end{cases} \qquad R(Q{-}LEA) = \begin{cases} -1 & \text{collision} \\ 0.5 & \text{goal reached, more energy consumed} \\ 1 & \text{first time goal reached} \\ 2 & \text{goal reached, less energy consumed} \end{cases}$$

Figure 10: Reward function of Q-LSP (left) and Q-LEA (right) algorithms.

In both cases, the agents are penalised if the device collides –goes out of the route– with -1 point. When the target is reached for the first time (from a particular starting position), the agents are awarded 1 point. However, if the route between start-finish has been covered before, the reward will depend according to the distance/energy consumed. If more distance is travelled (or energy needed), the reward will be 0.5. On the contrary, if the new path is more efficient (less distance or energy), the reward will be 2 points. This way, the agents will receive bigger rewards for better paths and avoid collisions.

7.3.4 ϵ-greedy policy: exploitation and exploration trade-off

This policy is introduced to ensure that the agent balances random actions (exploring) and exploiting the already known paths. Initially, the value of ϵ is 1, and it is decreased 0.01 every 700 episodes, until it reaches 0.1.

This causes an impact in the $getAction(...)$ function, where a random number between 0 and 1 is generated, if this value is smaller than ϵ, then a random action from the current state is selected. Otherwise, the best action (the action with the biggest Q-value) is chosen.

Therefore, as the algorithm learns, the randomness will decrease and since ϵ does not reach 0, it guarantees that random actions will be considered as long as the training lasts.

7.4 Path planning results

Once it is guaranteed that both algorithms generate a free-of-collisions path between any two points of the environment, the performance of the planners can be analysed and compared.

A remark on the algorithm limitations is given in Section A.3 in Appendix A. In particular, the scalability of the solution is addressed in Section A.3.1. The efficiency is also discussed in Section A.3.2.

7.4.1 Q-LSP: performance analysis

This path planner finds the shortest routes between any two points in the environment and it is used as a guideline to compare against the energy-aware path planner. Thus, it is important to determine that the shortest paths are correctly generated.

For this purpose, a brute-force algorithm was elaborated to calculate the length of shortest route between any combination of start-finish points. These would be used as a ground truth to determine if the Q-LSP behaves as expected and finds the correct routes.

When verifying the routes generated by the Q-LSP against this ground truth, the path planner successfully finds the shortest path for up to 92% of the routes. Listing 1 from Appendix A contains an example of the routes that differ from the expected shortest path (which may vary after every training instance), together with the computation of the algorithm efficiency.

The error produced by the path planner is attributed to the inherent randomness of the AI algorithm. It is expected that with infinite training process and enough data samples for all the possible combinations of states-actions, the efficiency would increase. Developing a method that improves the learning process and performance of the Q-LSP is considered as part of the future work.

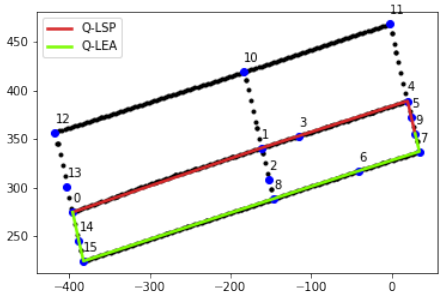
7.4.2 Q-LEA: improving over Q-LSP routes

When comparing the routes generated by both path planners, 15-20% of the Q-LEA routes differ from Q-LSP paths, out of which 87.5-100% reduce the energy consumption of the shortest paths (see Listing 3, Appendix A).

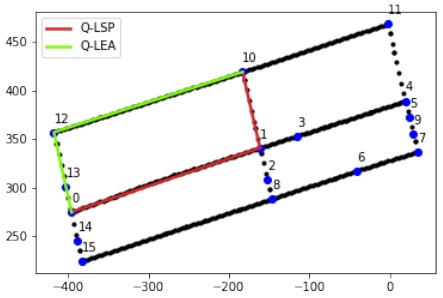
Even though these numbers suggest that the Q-LEA efficiency is not as high as the Q-LSP (as detailed in Listing 2 from Appendix A), it was determined, after some testing, that the performance upper boundary was reached for this path planner. A note on the efficiency (and the tests performed to improve it) of Q-LEA is given in Section A.3.2 from Appendix A. Although this performance is good enough for our goals, improving it is proposed as a line of future work.

7.4.3 Comparison of chosen paths

Figure 11 depicts a couple of examples of the routes considered by both algorithms to go from a start location to a particular finish point.



(a) Routes taken by Q-LSP (red) and Q-LEA (green) starting at intersection 9 and finishing at 0.



(b) Routes taken by Q-LSP (red) and Q-LEA (green) starting at intersection 0 and finishing at 10.

Figure 11: Comparison of the routes taken by both Q-LSP and Q-LEA path planners between intersections 9 and 0 (left); and 0 and 10 (right).

Table 9 gathers the trajectory data of the routes planned between intersections 9 and 0, and 0 and 10 by Q-LSP and Q-LEA.

Path planner	Route	Dist. (m)	Cons. (J)	Reduction (%)
Q-LSP	**9** $\rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 1 \rightarrow$ **0**	465.08	25030.64	-
Q-LEA	**9** $\rightarrow 7 \rightarrow 6 \rightarrow 8 \rightarrow 15 \rightarrow 14 \rightarrow$ **0**	502.63	17877.02	28.58
Q-LSP	**0** $\rightarrow 1 \rightarrow$ **10**	324.39	16422.34	-
Q-LEA	**0** $\rightarrow 13 \rightarrow 12 \rightarrow$ **10**	327.53	12369.36	24.68

Table 9: Alternative routes determined by the path planners between intersections 9 and 0 (first two rows); and intersections 0 and 10 (last two rows).

As shown in the examples above, in this simulated environment, the shortest route is not always the most energy-efficient option and considering an alternative way with lower speeds (see Figure 9d) can indeed reduce the energy consumption of the established path up to a 28.58%.

In the following sections, this algorithm is tested using real data to determine if the gain obtained by considering trajectory data in the path planning is still present in a real-case scenario.

8 Small-scale experimentation

To practically evaluate the implemented algorithms, the goal was to equip the iRobot Create 2 mobile robot with a NavQ board (see Section 5.1) to achieve movement automation. Thus, the Q-learning path planners were to be executed in the board and the device, autonomously, would choose which movements to make in order to reach the target destination from a specific starting point.

Unfortunately, due to unforeseen issues with the Wi-Fi communication, resulting in packages getting lost and intermittent motion, autonomous movement was finally discarded. Nevertheless, trajectory data was collected with the NavQ board and motion was controlled remotely using the existing setup from the feasibility study (see Section 6.1). Additionally, some modifications were made in the environment to test different terrain types.

In this section, the environment creation, data collection and processing and the resulting planned paths are explained and analysed.

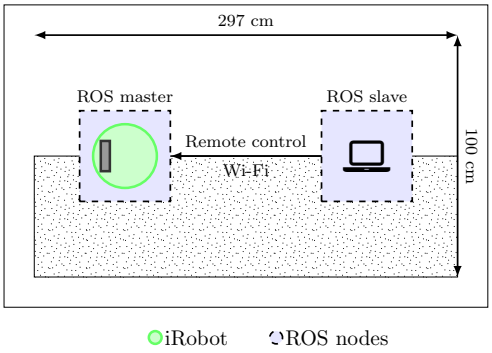
8.1 Environment setup

Given that the platform chosen to conduct the experimentation is the iRobot Create 2, the only suitable environment is indoors. This reduces the scope of the problem and simplifies the use-cases.

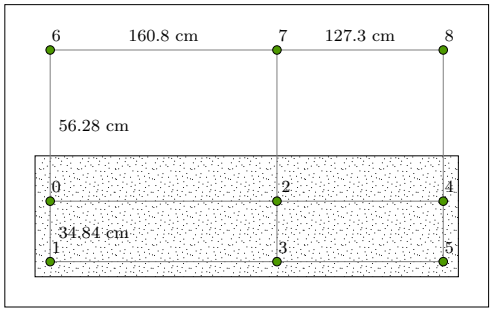
For our practical evaluation, it was necessary to have a room with different floor materials. In particular, the chosen scenario is shown in Figure 12.

Following a similar structure to the one shown in Section 6.1, in this case half of the room is covered with carpet, whereas the other half consists in plastic-tiled floor.

The idea was to scale down the environment extracted from HTC (see Figure 9) so it could fit in the room available to carry out the experiments. Moreover, some simplifications were done and the road symmetry was enforced (in reality the roads are not exactly the same length, although pretty similar). The resulting scenario is shown in Figure 12b. Since UWB nodes and anchors were no longer used, they could be removed from the scenario.



(a) Room where half the floor is carpet and the other half is plastic-tiled floor.



(b) Small-scale environment layout.

Figure 12: Small-scale environment: left figure shows the room and its components; right figure depicts the different intersections and the connecting edges.

8.2 Data collection

In order to remotely move the mobile robot, a joystick was connected to the laptop acting as ROS slave, from which the commands to allow motion would then be sent to the mobile robot.

With this setup, current, voltage and power consumption measurements were collected when the mobile robot was spinning non-stop on the carpet and plastic surfaces (see Figure 17 and 18 in Appendix B); as well as when it was constantly moving back and forth on said floor types (see Figure 19 and 20 in Appendix B). In both cases, the maximum speed was used: 0.5 m/s when moving back and forth and 4.25 m/s when spinning.

Surface	Avg. spin (A)	Avg. linear (A)	Average (A)
Carpet	0.3807	0.4143	0.3975
Tiles	0.3400	0.3689	0.3544

Table 10: Amperage consumption of the mobile robot derived from the tests.

8.3 Environment creation

Using the collected data, the environment can be represented by the combination of sources and destinations, and the edges between them, as shown in Table 11 (see Table 16 for the full road properties). In this case, since the

energy consumption (J) can be derived from the collected data, there is no need to estimate it with the drag coefficient factor and road speed.

Source	Destination	Distance (cm)	Consumption (J)
6	7	160.80	182.39
...			
8	4	56.28	65.31
4	8	56.28	65.31
4	5	34.84	44.32
5	4	34.84	44.32

Table 11: Road properties summary of the scaled-down HTC environment.

This environment can be processed by both path planners (Q-LSP and Q-LEA), from which the resulting paths are analysed in the following section. In this case, no other modifications are needed in the algorithms.

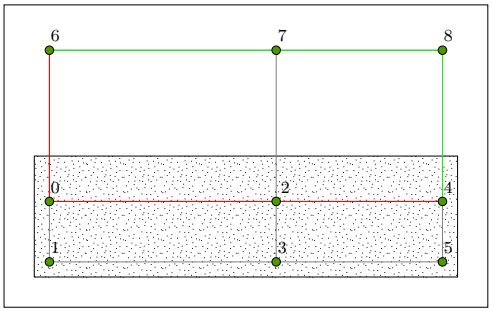
8.4 Path planning results

When comparing the generated routes with the expected shortest and most energy-efficient paths (Q-LSP and Q-LEA, respectively) both path planners achieve a success rate of up to 100%, meaning that in almost all the occasions (can vary between 98-100% depending on the learning process) the planned route is indeed the best one.

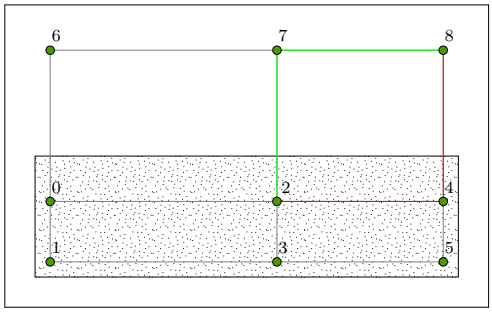
Listing 4 contains the paths that Q-LEA improves in comparison to Q-LSP. In total, up to 22.22% of the routes improve the shortest path by reducing up to an 9.19% of the energy consumed.

Granted, depending on the randomness of the algorithms and due to the fact that the paths are symmetric, the Q-LSP sometimes chooses the most efficient path as the shortest one, thus the number of improved routes can decrease depending on the training process.

As computed in Appendix B, the roads defined by Q-LEA can improve the energy consumption between 7.71-9.19% of the shortest path consumption. Figure 13a presents the case where the energy consumption is reduced the most (9.19%), whereas Figure 13b depicts the scenario in which the reduction achieved is 7.71%.



(a) Paths chosen by Q-LSP (red) and Q-LEA (green) to go from 4 to 6 (or vice versa).



(b) Paths chosen by Q-LSP (red) and Q-LEA (green) to go from 8 to 2 (or vice versa).

Figure 13: Path that reduces 9.19% of the energy consumed (left); route that improves the consumption on a 7.71% (right).

These routes are also summarised in Table 12, together with their reduction in energy consumption.

Path planner	Route	Dist. (cm)	Cons. (J)	Reduction (%)
Q-LSP	**4** $\rightarrow 2 \rightarrow 0 \rightarrow$ **6**	344.38	431.77	-
Q-LEA	**4** $\rightarrow 8 \rightarrow 7 \rightarrow$ **6**	344.38	392.09	9.19
Q-LSP	**8** $\rightarrow 4 \rightarrow$ **2**	183.58	227.23	-
Q-LEA	**8** $\rightarrow 7 \rightarrow$ **2**	183.58	209.70	7.71

Table 12: Alternative routes determined by the path planners between intersections 4 and 6 (first two rows); and intersections 8 and 2 (last two rows).

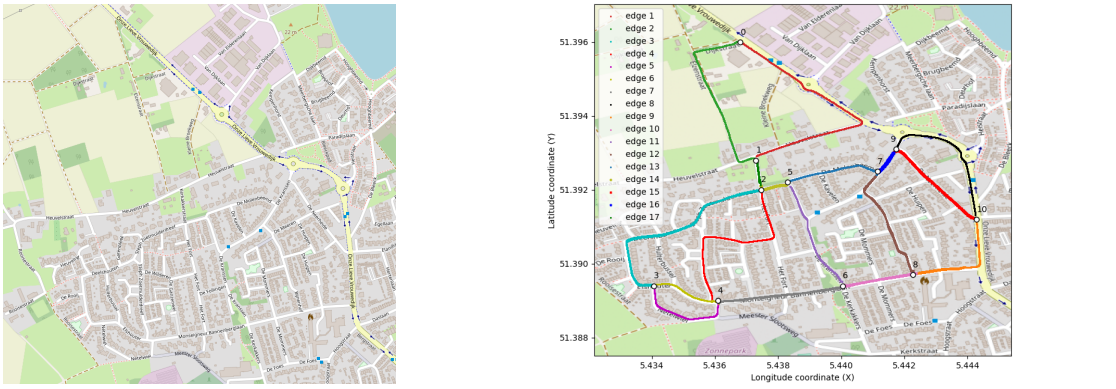
9 Electric vehicle data collection

To check how the proposed algorithms behave with real data, a region of Waalre is selected to perform driving tests, in which trajectory data was collected by means of ROS bagfiles (introduced in Section 5.3). The purpose is to process this data, create an environment with it, and use the implemented path planners to find the most optimal routes in terms of shortest path (with Q-LSP) and most energy-efficient route (using Q-LEA).

9.1 Creating the environment from real data

Looking at the initial results presented in Section 6.2.2, it is clear that certain drive factors affect the total energy consumption. For instance, high speeds contribute to a bigger energy drainage. Similarly, traffic congestion also plays an important role in this, which is why all the tests were performed in comparable conditions (little to none traffic congestion).

To ensure diversity of terrains and speeds, the chosen roads have a mix of high speeds (indicated in yellow in Figure 14a), unpaved roads (represented with brown dashed lines), and paved roads. Figure 14b shows how the driven roads are split in intersections (enumerated with 0-10) and segments (or edges, marked with 1-17).



(a) Waalre area chosen for conducting the driving experiments.

(b) Processed trajectory data divided in segments (edges) and intersections (enumerated from 0 to 10).

Figure 14: Region of Waalre where the collection of trajectory data is carried out (left); Waalre environment divided in 17 segments and 11 intersections (right).

Figure 21 in Appendix C contains the different circuits defined in the

region of Waalre (five in total). For each of these circuits, five instances were performed so the average consumption could be calculated.

Using Python, the relevant data such as the electrical energy consumption (in Joules) could be derived from the voltage and current measurements received through the CAN bus. Additionally, using the reported GPS data, the consumption can be divided per areas, as defined in Figure 14b.

Table 17 from Appendix C gathers the trajectory data, collected with the drive tests, relevant for the path planner algorithms proposed in this project. Moreover, Table 13 shows a summary of some of these roads.

Source	Destination	Distance (m)	Consumption (J)
0	1	639	566021.09
...			
9	7	79	15727.20
1	2	99	49888.02
2	1	99	49888.02

Table 13: Details of some of the roads of the Waalre environment.

In this case, the energy consumption is the average computed with multiple instances of gathered data (for both ways), so there is no need to derive it from the drag coefficient factor and road speed. The magnitude of these values is addressed in Section C.3, which confirms the validity of the energy consumption costs. Finally, the distance is taken from the OSM data.

This data can be used as a new environment in which the path planners have to train. Nonetheless, no other modifications are needed in the algorithms.

9.2 Path planning results

To assess the performance of the algorithms when processing real data derived from the electric vehicle trajectory data, the generated paths are once again compared to the shortest path and the most energy-efficient paths obtained with a brute-force method (same as in previous sections).

Appendix C contains Listings 5, 6 and 7, which present the planned paths that differ from the shortest route; the routes different from the most energy-efficient way; and the comparison between both algorithms, respectively.

The success rate of both algorithms surpasses 92%, meaning that, in almost every occasion, the algorithms are capable of finding the most efficient (either in terms of distance or energy consumption) route between two points.

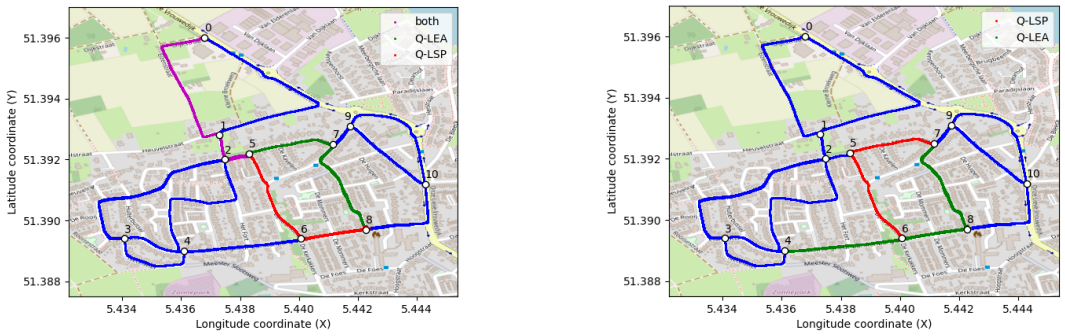
When both algorithms manage to find the expected optimal routes between any combination of two points, Q-LEA path planner improves a 6% of the total number of routes (see Listing 8). In these cases, the routes defined by Q-LEA reduce 1% the energy consumption of the shortest path.

Table 14 contains two examples of planned routes between points 0 and 8 (first two rows); and 7 and 4 (last two rows). In the first case, the routes are optimal for both algorithms (Q-LSP and Q-LEA). In the other case, the route defined by Q-LSP is not optimal, which indeed increases the reduction achieved by Q-LEA up to 16.79%.

Path planner	Route	Dist. (m)	Cons. (J)	Reduction (%)
Q-LSP	**0** $\rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow$ **8**	1158	836449.79	-
Q-LEA	**0** $\rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow$ **8**	1237	828590.32	1 %
Q-LSP	**7** $\rightarrow 5 \rightarrow 6 \rightarrow$ **4**	845	624877.82	-
Q-LEA	**7** $\rightarrow 8 \rightarrow 6 \rightarrow$ **4**	786	535029.40	16.79 %

Table 14: Alternative routes, in Waalre, determined by the path planners between intersections 0 and 8 (first two rows); and intersections 7 and 4 (last two rows).

Figure 15 offers a visual representation of these paths.



(a) Paths planned in Waalre between points 0-8. **(b)** Paths planned in Waalre between points 7-4.

Figure 15: Planned paths by Q-LSP (red line) and Q-LEA (green line); left figure shows the routes between points 0 and 8; right figure depicts the proposed routes between intersections 7 and 4.

Certainly, an improvement of 1% is not much, but it is left as possible future work to analyse longer and more complex routes in order to establish if this reduction can increase.

10 Final results

This project addressed the inclusion of trajectory data from previous drives (containing energy consumption, GPS localisation, etc.), as well as battery usage data, in a path planner algorithm. The main difference between this method and the ones available in the current literature is the usage of real data, which can rarely be found open-source.

For this purpose, two Q-learning path planners were developed. One of which learnt the shortest path between any two points in the environment (Q-LSP). The other one, learnt the most energy-efficient way (Q-LEA).

Additionally, it was also in our interest to investigate how the algorithm with trajectory data would scale up in the case of being used in an electric vehicle.

To solve this question, an electric vehicle –Citroën C-Zero– was used to collect real trajectory data that would be used in the path planners to analyse which decisions were taken and what was the gain obtained from choosing the energy-efficient path, instead of the shortest route, to reach the desired destination.

10.1 Conclusions

The implemented algorithms used a simple representation of the environment by means of intersections –used to determine the starting and finishing points– and roads between them. These path planners proved to be adaptable to different scenarios with no modifications required, which was targeted with this DRL approach, requiring less human effort between iterations.

When using simulated data, it was clear that the Q-LEA path planner could provide more energy-efficient routes (up to 15% of different paths) by reducing up to 28.58% of the energy consumed when driving certain routes. When using the small-scale platform, Q-LEA optimised 22.22% of the routes, with a reduction of up to 9.19% of the energy consumed by the shortest path.

With real trajectory data derived from the driving experiments performed with the electric vehicle, the reduction in the energy consumption dropped to 1% of the total (in up to 6% of the pathways), which does not represent a big quantity. However, these driving tests were only performed using one type of battery, one vehicle and one driver. It would be interesting to analyse if broadening the scope of analysis –including more cars, routes, battery types and drivers– would represent a big change in the results.

The difference in energy reduction achieved through the different testing scenarios resides in the fact that different data (with different magnitude) was being handled in each case. Nevertheless, in all scenarios there was an improvement achieved with the energy-aware path planner.

Furthermore, the scalability of the solution was addressed in Section A.3.1. Since Q-learning methods do not scale well when the environment complexity heavily increases, an alternative solution is raised in Future work.

In general, both research questions were answered and it is clear that including trajectory data in the path planning mechanisms can reduce the EV energy consumption.

10.2 Future work

As discussed throughout the project, in some occasions the performance of the algorithm was not as good as expected (Q-LEA in particular). The issue seemed to be caused by a performance upper boundary caused by the simulated data. Thus, one of the aspects to be further analysed is the performance improvement of the path planners.

Another element to consider would be the expansion of the algorithm, which eventually would lead to the usage of a Neural Network (NN) agent –an approximate solution– to tackle the scalability issue introduced in Section A.3.1. Additionally, with the problem complexity expansion, other parameters could be taken into account to plan better routes, such as adding the third dimension (height or terrain elevation) –which was not considered for simplification purposes (although there are no big changes in the analysed areas)–, battery temperature, regeneration capabilities of the battery, weather parameters, battery type, live traffic information, etc. Along these lines, it would also be interesting to analyse the algorithm efficiency with this method.

Furthermore, in an attempt to have more data available for the NN method, it could be interesting to evaluate the usage of a Digital Twin [40], which would allow the possibility of working with a simulated model of the device(s). However, this option was discarded as the goals of the current project were to use real data and reduce the human effort in the method used.

Regarding the small-scale proof-of-concept, one of the goals was to achieve autonomous movement with the mobile robot. Unfortunately, this could not be fulfilled due to unforeseen difficulties derived from the Wi-Fi connection.

Thus, it is left as future work to implement the autonomous motion.

In relation to the deployment of the algorithm itself, and given the observed trend towards distributed systems, it would be interesting to study an implementation that moved the heavy computation from the embedded board to the cloud (that would enable more power and better performance), which also would facilitate the usage of trajectory data available in the cloud.

Finally, when looking at the results obtained with the full-sized car, it is clear that a 1% of energy reduction is not a big improvement. However, it would be interesting to further analyse more scenarios that include different (and longer) routes, more driving profiles, different cars and batteries and other parameters that could influence the driving consumption.

References

[1] S. Yenikaya, G. Yenikaya, and E. Düven, "Keeping the vehicle on the road: A survey on on-road lane detection systems," *acm Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–43, 2013.

[2] M. Wada, K. S. Yoon, and H. Hashimoto, "Development of advanced parking assistance system," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 1, pp. 4–17, 2003.

[3] "V2x: What is vehicle to everything?." `https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/industries/automotive/use-cases/v2x`, June 2021. Accessed on 2021-12-03.

[4] "Autonomous driving: An overview." `https://www.zf.com/mobile/en/technologies/domains/autonomous_driving/autonomous_driving.html`. Accessed on 2021-12-03.

[5] "In five steps to a self-driving car." `https://www.zf.com/mobile/en/stories_2497.html`. Accessed on 2021-12-03.

[6] A. Vezzini, "15 - lithium-ion battery management," in *Lithium-Ion Batteries* (G. Pistoia, ed.), pp. 345–360, Amsterdam: Elsevier, 2014.

[7] T. Abukhalil, H. Almahafzah, M. Alksasbeh, and B. A. Y. Alqaralleh, "Power optimization in mobile robots using a real-time heuristic," *Journal of Robotics*, 2020.

[8] Y. Gigras and K. Gupta, "Artificial intelligence in robot path planning," *International Journal of soft computing and Engineering*, vol. 2, no. 2, pp. 471–474, 2012.

[9] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Motion and operation planning of robotic systems*, pp. 3–27, 2015.

[10] S. Li, X. Xu, and L. Zuo, "Dynamic path planning of a mobile robot with improved q-learning algorithm," in *2015 IEEE international conference on information and automation*, pp. 409–414, IEEE, 2015.

[11] J.-C. Chen, "Dijkstra's shortest path algorithm," *Journal of formalized mathematics*, vol. 15, no. 9, pp. 237–247, 2003.

[12] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[13] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[14] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[15] "Artificial intelligence." `https://www.encyclopedia.com/science-and-technology/computers-and-electrical-engineering/computers-and-computing/artificial-intelligence`. Accessed on 2021-12-05.

[16] "What is artificial intelligence?." `https://www.auraquantic.com/what-is-artificial-intelligence/`. Accessed on 2021-12-05.

[17] "Artificial intelligence technologies and their categories." `https://www.auraquantic.com/artificial-intelligence-technologies-and-their-categories/`. Accessed on 2021-12-05.

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

[19] C. M. Bishop, "Neural networks and their applications," *Review of scientific instruments*, vol. 65, no. 6, pp. 1803–1832, 1994.

[20] N. Buduma and N. Locascio, *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms.* " O'Reilly Media, Inc.", 2017.

[21] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[22] F. Ye, S. Zhang, P. Wang, and C.-Y. Chan, "A survey of deep reinforcement learning algorithms for motion planning and control of autonomous vehicles," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1073–1080, IEEE, 2021.

[23] E. Leurent, "A survey of state-action representations for autonomous driving," 2018.

[24] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[25] V. N. Sichkar, "Reinforcement learning algorithms in global path planning for mobile robot," in *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pp. 1–5, IEEE, 2019.

[26] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved q-learning for path planning of a mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1141–1153, 2013.

[27] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, pp. 1–6, Ieee, 2017.

[28] J. Xin, H. Zhao, D. Liu, and M. Li, "Application of deep reinforcement learning in mobile robot path planning," in *2017 Chinese Automation Congress (CAC)*, pp. 7112–7116, IEEE, 2017.

[29] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *Journal of Robotics*, vol. 2018, 2018.

[30] Y. Wang, Y. Fang, P. Lou, J. Yan, and N. Liu, "Deep reinforcement learning based path planning for mobile robot in unknown environment," in *Journal of Physics: Conference Series*, vol. 1576, p. 012009, IOP Publishing, 2020.

[31] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015.

[32] R. Maidana, R. Granada, D. Jurak, M. Magnaguagno, F. Meneguzzi, and A. Amory, "Energy-aware path planning for autonomous mobile robot navigation," in *The Thirty-Third International Flairs Conference*, 2020.

[33] M. Visca, A. Bouton, R. Powell, Y. Gao, and S. Fallah, "Conv1d energy-aware path planner for mobile robots in unstructured environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2279–2285, 2021.

[34] A. Niaraki, J. Roghair, and A. Jannesari, "Visual exploration and energy-aware path planning via reinforcement learning," *arXiv preprint arXiv:1909.12217*, 2019.

[35] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of uavs," in *2015 IEEE international conference on autonomous robot systems and competitions*, pp. 111–117, IEEE, 2015.

[36] Q. Li, Z. Zeng, T. Zhang, J. Li, and Z. Wu, "Path-finding through flexible hierarchical road networks: An experiential approach using taxi trajectory data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 13, no. 1, pp. 110–119, 2011.

[37] J. Zhang, W. Meng, Q. Liu, H. Jiang, Y. Feng, and G. Wang, "Efficient vehicles path planning algorithm based on taxi gps big data," *Optik*, vol. 127, no. 5, pp. 2579–2585, 2016.

[38] A. Novikov, I. Novikov, and A. Shevtsova, "Study of the impact of type and condition of the road surface on parameters of signalized intersection," *Transportation Research Procedia*, vol. 36, pp. 548–555, 01 2018.

[39] "Vehicle Coefficient of Drag List." `https://ecomodder.com/wiki/Vehicle_Coefficient_of_Drag_List`, 2018. [Online; accessed 30-March-2022].

[40] F. Dembski, U. Wössner, and C. Yamu, "Digital twin," in *Virtual Reality and Space Syntax: Civic Engagement and Decision Support for Smart,*

Sustainable Cities: Proceedings of the 12th International Space Syntax Conference, Beijing, China, pp. 8–13, 2019.

[41] E. Karapistoli, F.-N. Pavlidou, I. Gragopoulos, and I. Tsetsinas, "An overview of the ieee 802.15.4a standard," *IEEE Communications Magazine*, vol. 48, no. 1, pp. 47–53, 2010.

[42] "Citroen C-Zero." `https://ev-database.org/car/1094/Citroen-C-Zero`. [Online; accessed 27-June-2022].

A Path planning algorithm simulations

In this section, additional information about the Q-learning path planners implemented in this project is presented.

A.1 Environment road properties

Figure 9 depicts the simulation environment considered in this project. More detailed data about each road segment is provided in Table 15. For each road, the intersection source and destination, the drag coefficient of the terrain, the maximum speed, distance and estimated energy consumption are given.

In this table, only one way of the segment is given, however since the height dimension is not currently considered in the algorithm, the columns 'Source' and 'Destination' can be interchanged in any present road segment.

Source	Destination	Drag coeff.	Speed (km/h)	Dist. (m)	Consumption (J)
0	1	0.8	40	243.30	13094.14
1	2	0.7	30	34.11	1400.06
3	4	0.8	40	139.44	7504.92
1	3	0.8	40	47.64	2564.16
4	5	0.8	40	16.75	901.69
6	7	0.6	20	77.49	2203.89
8	6	0.6	20	109.80	3122.83
2	8	0.7	30	20.31	833.32
9	7	0.8	40	19.33	1040.40
5	9	0.8	40	17.94	965.73
10	1	0.7	30	81.10	3328.21
10	11	0.7	30	186.94	7671.82
11	4	0.8	40	82.49	4439.46
12	10	0.7	30	242.42	9948.73
12	13	0.6	20	57.51	1635.60
13	0	0.6	20	27.60	785.04
0	14	0.7	20	30.74	885.32
14	15	0.7	20	21.37	615.59
15	8	0.7	30	243.88	10008.99

Table 15: HTC simulated environment road properties.

A.2 Path planners performance

To determine the efficiency of the developed methods, the paths planned between the different starting-finishing points of the simulated environment have to be compared to some ground truth or other planned routes.

A.2.1 Q-LSP algorithm results

In the case of Q-LSP (Q-learning Shortest Path), the generated routes are compared with the paths planned by a brute-force method that determines the shortest way between any two points of the map.

This comparison determined that 90-93% of the paths generated by the Q-LSP path planner are indeed the shortest anticipated route. This loss in efficiency is expected due to the presence of randomness in the Q-learning training process. For instance, some of the paths that are different, which might change with every new training process, are shown in Listing 1. Even though the paths might differ between training processes, the efficiency percentage is always around those boundaries.

"`path_dist`" is the value estimated with Q-LSP and "`dist_short_path`" the shortest distance between the same start and finish points (determined by the brute-force algorithm).

Considering that the algorithm works with a 90%+ success rate, this behaviour was deemed as good enough for the project. Nevertheless, improving this method is left as a possible future line of work.

```
[1, 7] path_dist: 241.7165 m - dist_short_path: 241.1168 m
[4, 8] path_dist: 241.5077 m - dist_short_path: 241.3256 m
[4, 12] path_dist: 511.8377 m - dist_short_path: 510.6001 m
[5, 12] path_dist: 528.5916 m - dist_short_path: 527.3540 m
[7, 1] path_dist: 241.7165 m - dist_short_path: 241.1168 m
[7, 10] path_dist: 322.8131 m - dist_short_path: 322.2135 m
[7, 12] path_dist: 565.2285 m - dist_short_path: 564.6288 m
[8, 4] path_dist: 241.5077 m - dist_short_path: 241.3256 m
[9, 12] path_dist: 546.5353 m - dist_short_path: 545.2977 m
[10, 5] path_dist: 286.1762 m - dist_short_path: 284.9386 m
[10, 7] path_dist: 323.4511 m - dist_short_path: 322.2135 m
[10, 9] path_dist: 304.1200 m - dist_short_path: 302.8823 m
[12, 5] path_dist: 528.5916 m - dist_short_path: 527.3540 m
[12, 7] path_dist: 565.8664 m - dist_short_path: 564.6288 m
[12, 9] path_dist: 546.5353 m - dist_short_path: 545.2977 m
[13, 10] path_dist: 351.9950 m - dist_short_path: 299.9258 m
[15, 3] path_dist: 345.9467 m - dist_short_path: 343.0531 m
[15, 4] path_dist: 485.3915 m - dist_short_path: 482.4980 m
[15, 11] path_dist: 566.3353 m - dist_short_path: 563.4417 m
num. not optimal paths: 19/240 - algorithm efficiency: 92.0833 %
```

Listing 1: Q-LSP paths that differ from shortest route and algorithm efficiency.

A.2.2 Q-LEA algorithm results

While the other algorithm learns to find the shortest route between two points, this path planner determines the least energy-consuming way between all the origin-destination combinations. Similarly to the previous test, the paths generated by Q-LEA (Q-learning Energy-Aware) are also compared to the expected minimal energy consumption between two points computed by a brute-force method.

In this case, the Q-LEA path planner successfully finds up to 80% of the predicted most energy-efficient routes. Listing 2 contains an example of the paths that might differ, where "`path_ene_cons`" is the value estimated with Q-LEA and "`eff_path`" the expected minimum consumption between the same start and finish points.

Several tests were performed, as detailed in Section A.3.2. However, it was determined that with the data being treated, the current results were the best achievable ones.

```
[0, 2] path_ene_cons: 14494.19167 J - eff_path: 12343.2240 J
[0, 4] path_ene_cons: 23163.2181 J - eff_path: 19744.4465 J
[0, 5] path_ene_cons: 24064.9107 J - eff_path: 18842.7540 J
[0, 11] path_ene_cons: 24094.1633 J - eff_path: 20041.1851 J
[1, 5] path_ene_cons: 10970.7751 J - eff_path: 9566.2357 J
[1, 14] path_ene_cons: 13979.4529 J - eff_path: 12857.9628 J
[2, 4] path_ene_cons: 11469.1386 J - eff_path: 9067.8722 J
[2, 13] path_ene_cons: 15279.2329 J - eff_path: 13128.2652 J
[3, 7] path_ene_cons: 10412.7481 J - eff_path: 10124.2628 J
[3, 14] path_ene_cons: 16543.6123 J - eff_path: 15422.1222 J
[4, 0] path_ene_cons: 23163.2181 J - eff_path: 19744.4465 J
[4, 2] path_ene_cons: 11469.1386 J - eff_path: 9067.8722 J
[4, 13] path_ene_cons: 23948.2594 J - eff_path: 20529.4878 J
[4, 14] path_ene_cons: 24048.5354 J - eff_path: 18859.1292 J
[4, 15] path_ene_cons: 22311.4545 J - eff_path: 18243.5384 J
[5, 0] path_ene_cons: 24064.9107 J - eff_path: 18842.7539 J
[5, 1] path_ene_cons: 10970.7751 J - eff_path: 9566.2357 J
[5, 10] path_ene_cons: 13012.9721 J - eff_path: 12894.4419 J
[5, 12] path_ene_cons: 22961.6983 J - eff_path: 21263.3912 J
[5, 13] path_ene_cons: 24849.9520 J - eff_path: 19627.7952 J
[6, 12] path_ene_cons: 18633.1473 J - eff_path: 17053.3704 J
[7, 3] path_ene_cons: 10412.7481 J - eff_path: 10124.2628 J
[7, 12] path_ene_cons: 20837.0357 J - eff_path: 19257.2588 J
[8, 11] path_ene_cons: 13233.4087 J - eff_path: 12674.0053 J
[8, 12] path_ene_cons: 15510.3133 J - eff_path: 13930.5364 J
[9, 10] path_ene_cons: 13978.7032 J - eff_path: 11928.7108 J
[9, 12] path_ene_cons: 23927.4294 J - eff_path: 20297.6601 J
[9, 13] path_ene_cons: 25815.6831 J - eff_path: 18662.0641 J
[10, 5] path_ene_cons: 13012.9721 J - eff_path: 12894.4419 J
[10, 9] path_ene_cons: 13978.7032 J - eff_path: 11928.7108 J
[10, 14] path_ene_cons: 17307.6590 J - eff_path: 13254.6808 J
[10, 15] path_ene_cons: 15570.5781 J - eff_path: 13870.2716 J
[11, 8] path_ene_cons: 13233.4087 J - eff_path: 12674.0053 J
```

```
[11, 14] path_ene_cons: 24979.4806 J - eff_path: 20926.5024 J
[11, 15] path_ene_cons: 23242.3997 J - eff_path: 21542.0932 J
[12, 5] path_ene_cons: 22961.6983 J - eff_path: 21263.3912 J
[12, 6] path_ene_cons: 18633.1473 J - eff_path: 17053.3704 J
[12, 7] path_ene_cons: 20837.0357 J - eff_path: 19257.2588 J
[12, 9] path_ene_cons: 23927.4294 J - eff_path: 20297.6601 J
[13, 2] path_ene_cons: 15279.2329 J - eff_path: 13128.2652 J
[13, 4] path_ene_cons: 23695.6018 J - eff_path: 20529.4878 J
[13, 5] path_ene_cons: 24597.2943 J - eff_path: 19627.7952 J
[14, 1] path_ene_cons: 13979.4529 J - eff_path: 12857.9628 J
[14, 3] path_ene_cons: 16543.6123 J - eff_path: 15422.1222 J
[14, 4] path_ene_cons: 24048.5354 J - eff_path: 18859.1292 J
[14, 11] path_ene_cons: 24979.4806 J - eff_path: 20926.5024 J
[15, 10] path_ene_cons: 15570.5781 J - eff_path: 13870.2716 J
[15, 11] path_ene_cons: 23242.3997 J - eff_path: 21542.0932 J
num. not optimal paths: 48/240 - algorithm efficiency: 80.0 %
```

Listing 2: Q-LEA paths that differ from most energy-efficient route and algorithm efficiency.

A.2.3 Q-LEA paths compared with Q-LSP

If the Q-LEA paths are compared directly to the routes generated with Q-LSP, Q-LEA finds a total of approximately 15% different routes (36/240), out of which up to 87-100% improve (reduce) the energy consumption of the shortest path. In other words, most of the paths (if not all) found by Q-LEA –that are different from the shortest path–, have a reduced energy consumption.

Listing 3 contains the routes that differ in consumption between both algorithms Q-LSP and Q-LEA (note that these might vary with every training process), where "`path_ene_cons`" is the value estimated with Q-LEA and "`path_short_cons`" the estimated consumption of the route defined by Q-LSP between the same start and finish points. Additionally, the difference between these routes is specified next to "`diff`".

```
[0, 6] path_ene_cons: 14632.7332 J - path_short_cons: 18450.3506 J - diff: 3817.6174 J
[0, 9] path_ene_cons: 17877.0228 J - path_short_cons: 25030.6418 J - diff: 7153.6190 J
[0, 10] path_ene_cons: 12369.3635 J - path_short_cons: 16422.3417 J - diff: 4052.9782 J
[1, 9] path_ene_cons: 8600.5046 J - path_short_cons: 11936.5062 J - diff: 3336.0016 J
[1, 15] path_ene_cons: 12242.3720 J - path_short_cons: 14595.0437 J - diff: 2352.6718 J
[2, 0] path_ene_cons: 12343.2240 J - path_short_cons: 14494.1917 J - diff: 2150.9678 J
[2, 5] path_ene_cons: 8166.17965 J - path_short_cons: 12370.8312 J - diff: 4204.6516 J
[3, 15] path_ene_cons: 14806.5313 J - path_short_cons: 17159.2031 J - diff: 2352.6718 J
[4, 8] path_ene_cons: 8234.5474 J - path_short_cons: 12302.4635 J - diff: 4067.9161 J
[4, 10] path_ene_cons: 12111.2795 J - path_short_cons: 13397.2887 J - diff: 1286.0092 J
[4, 15] path_ene_cons: 22311.4545 J - path_short_cons: 24664.1263 J - diff: 2352.6718 J
[5, 2] path_ene_cons: 8166.1797 J - path_short_cons: 12370.8312 J - diff: 4204.6516 J
[5, 10] path_ene_cons: 13012.9721 J - path_short_cons: 14298.9813 J - diff: 1286.0092 J
[5, 14] path_ene_cons: 17957.4366 J - path_short_cons: 24950.2280 J - diff: 6992.7914 J
```

```
[6, 11] path_ene_cons: 9551.1713 J - path_short_cons: 16356.2427 J - diff: 6805.0715 J
[8, 4] path_ene_cons: 8234.5474 J - path_short_cons: 12302.4635 J - diff: 4067.9161 J
[9, 0] path_ene_cons: 17877.0228 J - path_short_cons: 25030.6418 J - diff: 7153.6190 J
[9, 1] path_ene_cons: 8600.5046 J - path_short_cons: 11936.5062 J - diff: 3336.0016 J
[9, 10] path_ene_cons: 13978.7032 J - path_short_cons: 15264.7124 J - diff: 1286.0092 J
[10, 0] path_ene_cons: 12369.3635 J - path_short_cons: 16422.3417 J - diff: 4052.9782 J
[10, 4] path_ene_cons: 12111.2795 J - path_short_cons: 13397.2887 J - diff: 1286.0092 J
[10, 15] path_ene_cons: 15570.5781 J - path_short_cons: 17923.2498 J - diff: 2352.6718 J
[11, 0] path_ene_cons: 20041.1851 J - path_short_cons: 24094.1633 J - diff: 4052.9782 J
[11, 15] path_ene_cons: 23242.3997 J - path_short_cons: 25595.0714 J - diff: 2352.6718 J
[12, 4] path_ene_cons: 22060.0057 J - path_short_cons: 23346.0149 J - diff: 1286.0092 J
[12, 8] path_ene_cons: 13930.5364 J - path_short_cons: 15510.3133 J - diff: 1579.7769 J
[13, 4] path_ene_cons: 23695.6018 J - path_short_cons: 23948.2594 J - diff: 252.6576 J
[13, 5] path_ene_cons: 24597.2943 J - path_short_cons: 24849.9520 J - diff: 252.6576 J
[13, 6] path_ene_cons: 15417.7744 J - path_short_cons: 19235.3918 J - diff: 3817.6174 J
[13, 9] path_ene_cons: 18662.0641 J - path_short_cons: 25815.6831 J - diff: 7153.6190 J
[14, 5] path_ene_cons: 17957.4366 J - path_short_cons: 24950.2280 J - diff: 6992.7914 J
[14, 10] path_ene_cons: 13254.6808 J - path_short_cons: 17307.6590 J - diff: 4052.9782 J
[15, 1] path_ene_cons: 12242.3719 J - path_short_cons: 14595.0437 J - diff: 2352.6718 J
[15, 4] path_ene_cons: 18243.5384 J - path_short_cons: 22311.4545 J - diff: 4067.9161 J
[15, 10] path_ene_cons: 15570.5781 J - path_short_cons: 17923.2498 J - diff: 2352.6718 J
[15, 11] path_ene_cons: 23242.3997 J - path_short_cons: 25595.07144 J - diff: 2352.6718 J
total different paths: 36/240 - algorithm efficiency: 100.0 %
lower value than shorter path found: 36
greater value than shorter path found: 0
```

Listing 3: Q-LEA paths that improve the energy consumption of the routes determined by Q-LSP.

A.3 Solution limitations

As of now, only the '`x`' and '`y`' coordinates are considered. However, the '`z`' component is also relevant, since the consumption is affected by the terrain inclination road data. Nonetheless, as a first approach it was deemed as not necessary. Moreover, in the area selected, there is not much difference in the height component. Extending the path planners so they consider the terrain elevation is contemplated as an aspect to investigate in the future work.

Nonetheless, the usage of Q-learning can present some restrictions due to the table used to keep track of the state-action combinations. This is detailed in Section A.3.1.

Furthermore, the performance of the Q-LEA path planner is not as good as expected. Section A.3.2 presents the different tests carried out in order to improve the performance of this algorithm, as well as the reached conclusions.

A.3.1 Note on scalability

Given that the current scope only considers the trajectory information and the real-time measurements of a particular battery, together with the simulated environment at hand (only using a few coordinate points), this Q-learning approach is enough to obtain a proof-of-concept.

Nonetheless, if a much more complex environment was considered, or other additional information/input parameters, such as weather conditions, battery type, etc., were taken into account, the Q-table would grow exponentially (specifically the state space), making this method infeasible to solve the problem. In such case, it would make more sense to use an approximate solution (i.e. a Neural Network). With an expansion of the scope, other variables could be included like live traffic information, etc. These aspects are considered as a possible future work.

All in all, the current solution is scalable up to a certain point. From that point onward, the convergence of the Q-learning algorithm is no longer guaranteed.

A.3.2 Note on efficiency

Due to the unexpectedly "low" success rate of the Q-LEA algorithm, several tests were conducted to determine the cause of this issue. For starters, long-run tests were conducted initially to determine the values of α and γ. This first attempt already revealed that the efficiency of the Q-LEA algorithm was not ideal.

Therefore, in another pursuit to determine if the low accuracy rate came from the fact that the algorithm was not converging, the state space was reduced. That is, the number of intersections (starting and finishing points) in the environment were reduced. Nevertheless, after the training process, the performance of the path planner was similar to one obtained in the original environment. Additionally, the training process was extended as well –to make sure there was enough time for the algorithm to converge–, which ended up in the same result. Hence, the possibility of not converging was discarded.

Ultimately, it was concluded that, considering the environment data and reward function used, the performance upper boundary was reached. A more in-depth analysis in an attempt to solve this issue is left as possible future work.

B iRobot Create 2 additional information

In this section, a brief introduction on Ultra-Wideband (UWB) is given. This technology was used in the feasibility study and it is meant to be used in future work to allow autonomous movement.

Additionally, more information about the data collected is also provided. The environment information is also summarised and the obtained results with the path planners algorithms are presented as well.

B.1 Ultra-Wideband (UWB) Technology

NXP's Ultra-Wideband (UWB)[9] –standardised as IEEE 802.15.4a [41]– is a wireless technology that enables precise device location. This technology is proven to be very robust due to the band it uses (around 500 MHz) and it provides an accuracy of around 10 centimetres. Moreover, it is very energy-efficient and fast.

It can be used in a variety of situations, however the use-case targeted in this project is indoors navigation and tracking.

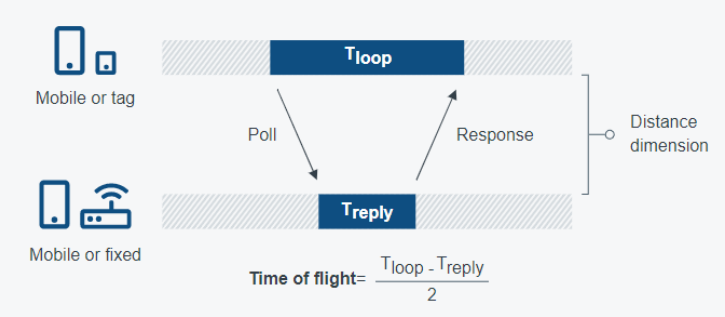


Figure 16: Determining the relative position of a device and the time of flight with Ultra-Wideband[9].

In our scenario, different antennas (or anchors) are placed around a room and a couple of nodes are mounted on the mobile robot. By means of exchanging messages and applying the formula depicted in Figure 16, the device is capable of accurately estimating the distance to the anchors (and its orientation) and act accordingly.

[9] `https://www.nxp.com/applications/enabling-technologies/connectivity/ultra-wideband-uwb:UWB`

An example of UWB usage for this project is presented in Section 6.1.1.

B.2 Data collection

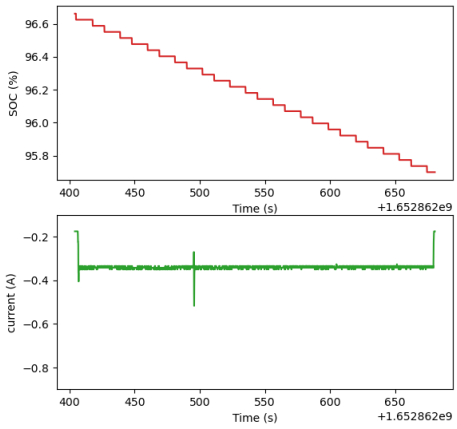
Using the reduced version of the HTC as shown in Figure 12b, several tests were carried out in order to capture the State-of-Charge, current, voltage and power consumption of the mobile robot.

Figures 17 and 18 show the SOC, current; voltage and power consumed, respectively, when the mobile robot is constantly spinning for a long time (more than 4 minutes) on top of the carpeted surface or the plastic-tiled floor.

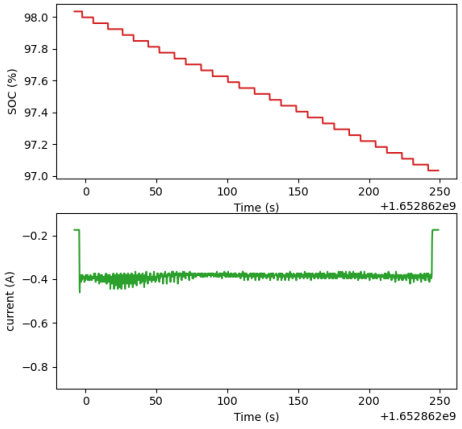
Similarly, Figures 19 and 20 present the same parameters when the mobile robot is moving back and forth on these terrains.

As seen in the average value of the current consumption (see Table 10), it is clear that the iRobot Create 2 employs more current when moving through the carpet. Thus, this surface is less energy-efficient than the normal plastic-tiled floor.

Using this data and computing an average of all the scenarios, the road properties can be derived as shown in Table 16.

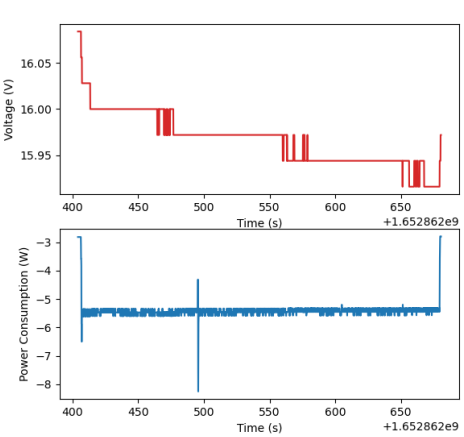


(a) SOC percentage (top) and current in A (bottom) of the mobile robot when spinning on the plastic-tiled surface.

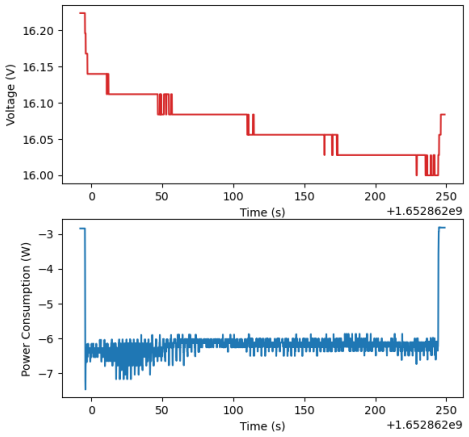


(b) SOC percentage (top) and current in A (bottom) of the mobile robot when spinning on the carpet surface.

Figure 17: State-of-Charge (%) and battery amperage (A) consumed by the robot when constantly spinning on a plastic-tiled surface (left) and carpet floor (right).

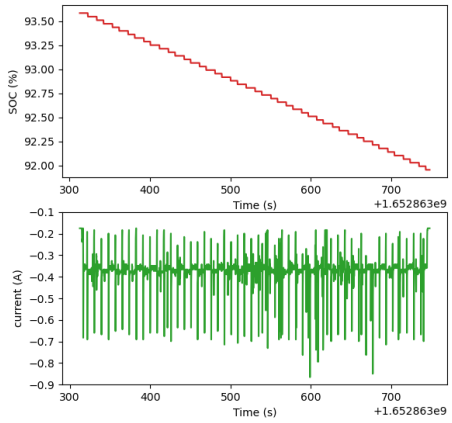


(a) Battery voltage (top) and power consumption (in W, bottom) of the mobile robot when spinning on the plastic-tiled surface.

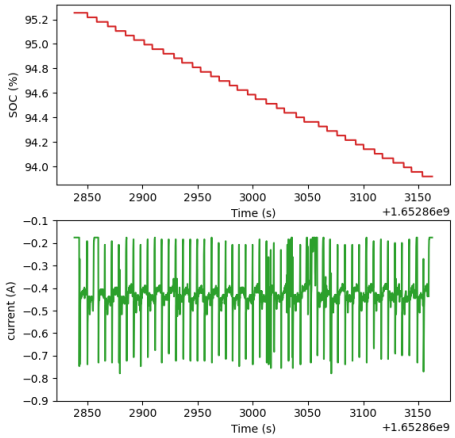


(b) Battery voltage (top) and power consumption (in W, bottom) of the mobile robot when spinning on the carpeted surface.

Figure 18: Battery voltage (V) and power consumption (W) used by the robot when constantly spinning on a plastic-tiled surface (left) and carpet floor (right).

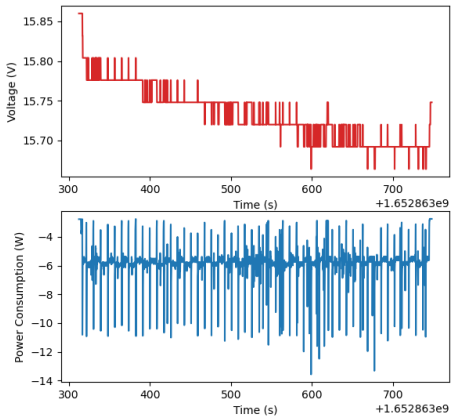


(a) SOC percentage (top) and current in A (bottom) of the mobile robot when moving back and forth on the plastic-tiled surface.

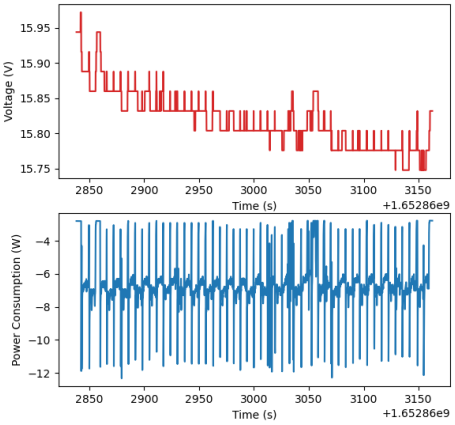


(b) SOC percentage (top) and current in A (bottom) of the mobile robot when moving back and forth on the carpet surface.

Figure 19: State-of-Charge (%) and battery amperage (A) consumed by the robot when constantly moving back and forth on a plastic-tiled surface (left) and carpet floor (right).



(a) Battery voltage (top) and power consumption (in W, bottom) of the mobile robot when moving back and forth on the plastic-tiled surface.



(b) Battery voltage (top) and power consumption (in W, bottom) of the mobile robot when moving back and forth on the carpeted surface.

Figure 20: Battery voltage (V) and power consumption (W) used by the robot when constantly moving back and forth on a plastic-tiled surface (left) and carpet floor (right).

B.3 Scaled-down environment road properties

The complete road description of the scaled-down environment used in the small-scale experiment can be found in Table 16.

Source	Destination	Distance (cm)	Consumption (J)
6	7	160.80	182.39
7	6	160.80	182.39
7	8	127.30	144.39
8	7	127.30	144.39
0	2	160.80	204.53
2	0	160.80	204.53
2	4	127.30	161.92
4	2	127.30	161.92
1	3	160.80	204.53
3	1	160.80	204.53
3	5	127.30	161.92
5	3	127.30	161.92
6	0	56.28	65.31
0	6	56.28	65.31
0	1	34.84	44.32
1	0	34.84	44.32
7	2	56.28	65.31
2	7	56.28	65.31
2	3	34.84	44.32
3	2	34.84	44.32
8	4	56.28	65.31
4	8	56.28	65.31
4	5	34.84	44.32
5	4	34.84	44.32

Table 16: Scaled-down HTC environment road properties.

B.4 Q-LEA paths compared against Q-LSP

Both algorithms showed a great success rate (up to 100% in most cases), which is why only the comparison between both path planners is shown in this section.

As gathered in Listing 4, 16 paths (22.22% of the total) can be improved by Q-LEA. The difference in energy consumption can range from 7.71% to 9.19% of the total consumed by the shortest path (defined by Q-LSP). Similar to the previous case, "`path_ene_cons`" indicates the value estimated by Q-LEA, and "`path_short_cons`" the energy consumption computed by Q-LSP.

"`diff`" indicates the difference in energy consumption between the paths defined by both algorithms.

```
[0, 7] path_ene_cons: 247.6973 J - path_short_cons: 269.8461 J - diff: 22.1489 J
[0, 8] path_ene_cons: 392.0859 J - path_short_cons: 431.7693 J - diff: 39.6834 J
[1, 7] path_ene_cons: 292.0130 J - path_short_cons: 314.1619 J - diff: 22.1489 J
[1, 8] path_ene_cons: 436.4018 J - path_short_cons: 476.0851 J - diff: 39.6834 J
[2, 6] path_ene_cons: 247.6973 J - path_short_cons: 269.8461 J - diff: 22.1489 J
[2, 8] path_ene_cons: 209.7002 J - path_short_cons: 227.2348 J - diff: 17.5345 J
[3, 6] path_ene_cons: 292.0131 J - path_short_cons: 314.1619 J - diff: 22.1489 J
[3, 8] path_ene_cons: 254.0161 J - path_short_cons: 271.5506 J - diff: 17.5345 J
[4, 6] path_ene_cons: 392.0859 J - path_short_cons: 431.7693 J - diff: 39.6834 J
[4, 7] path_ene_cons: 209.7002 J - path_short_cons: 227.2348 J - diff: 17.5345 J
[5, 6] path_ene_cons: 436.4018 J - path_short_cons: 476.0851 J - diff: 39.6834 J
[5, 7] path_ene_cons: 254.0161 J - path_short_cons: 271.5506 J - diff: 17.5345 J
[6, 3] path_ene_cons: 292.0131 J - path_short_cons: 314.1619 J - diff: 22.1489 J
[7, 1] path_ene_cons: 292.0131 J - path_short_cons: 314.1619 J - diff: 22.1489 J
[8, 1] path_ene_cons: 436.4018 J - path_short_cons: 476.0851 J - diff: 39.6834 J
[8, 3] path_ene_cons: 254.0161 J - path_short_cons: 271.5506 J - diff: 17.5345 J
total different paths: 16/72 - algorithm efficiency: 100.0 %
```

Listing 4: Scaled-down HTC Q-LEA paths that improve the energy consumption of the routes determined by Q-LSP.

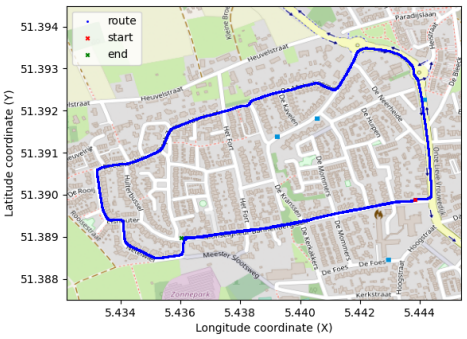
C Data collected with the Citroën C-Zero

In this section, the data collected with the electric vehicle is summarised together with the information related to the environment, which is derived from the real trajectory data collected in the area of Waalre (see Figure 14a).

As depicted in Figure 14b, some of the roads in Waalre were selected to be driven several times. For this purpose, the region was divided in multiple sub-regions and the data was collected for each of these.

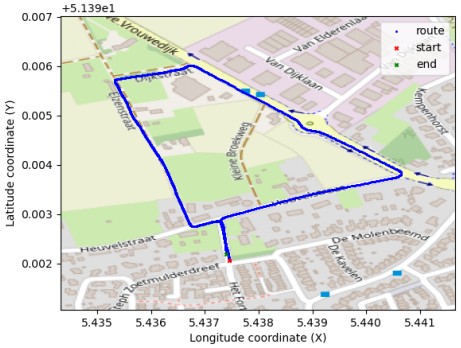
Figure 21 shows the five circuits in which the area of Waalre was divided. For ease of data collection, these areas are driven a total of five times, separately, and the data is collected independently.

Currently, the terrain height is not taken into account in the path planner algorithms. Therefore, the consumption of a road is the same in both directions (it is the average of both ways).



(a) First test circuit.

(b) Second test circuit.



(c) Third test circuit.

(d) Fourth test circuit.

(e) Fifth test circuit.

Figure 21: Waalre area divided in five circuits where trajectory data is collected.

C.1 Waalre environment road properties

Table 17 collects the relevant data extracted from the trajectory bagfiles. The Waalre environment is built using this information (distance of the road segments, consumption, etc.). It is worth to mention that distance and consumption are the computed average.

Source	Destination	Distance (m)	Consumption (J)
0	1	639	566021.09
1	0	639	566021.09
0	1	500	390094.51
1	0	500	390094.51
2	3	525	380481.03
3	2	525	380481.03
2	4	522	390900.88
4	2	522	390900.88
3	4	250	154636.56
4	3	250	154636.56
3	4	157	83433.88
4	3	157	83433.88
4	6	278	210882.83
6	4	278	210882.83
9	10	386	360967.89
10	9	386	360967.89
8	10	304	183768.07
10	8	304	183768.07
6	8	159	123991.95
8	6	159	123991.95
6	5	339	249008.56
5	6	339	249008.56
7	8	349	200154.62
8	7	349	200154.62
5	7	228	164986.43
7	5	228	164986.43
2	5	61	23466.75
5	2	61	23466.75
9	10	272	243524.52
10	9	272	243524.52
7	9	79	15727.20
9	7	79	15727.20
1	2	99	49888.02
2	1	99	49888.02

Table 17: Waalre environment road properties.

C.2 Path planners performance

Similar to the assessments carried out with the other environments, to quantify how well the algorithms perform it is important to compare the results with a ground truth. In this section, this analysis is presented.

C.2.1 Q-LSP algorithm results

Once again, the paths generated by Q-LSP are compared against the result of the brute-force algorithm. Similar to the previous case, the comparison determined that in 90-93 % of the times, the path planner successfully finds the shortest route between any two points.

The paths that differ from the expected minimal distance appear in Listing 5. However, this values change with every learning instance. This is associated with the random quality of the Q-learning method. In the listing, "`path_dist`" is the value estimated with Q-LSP and "`dist_short_paths`" the shortest distance between the same start and finish points (determined by the brute-force algorithm).

```
[0, 10] path_dist: 1378.0 m - dist_short_paths: 1239.0 m
[5, 4] path_dist: 617.0 m - dist_short_paths: 583.0 m
[7, 4] path_dist: 845.0 m - dist_short_paths: 786.0 m
[8, 0] path_dist: 1237.0 m - dist_short_paths: 1158.0 m
[8, 2] path_dist: 638.0 m - dist_short_paths: 559.0 m
[9, 4] path_dist: 924.0 m - dist_short_paths: 865.0 m
[10, 1] path_dist: 962.0 m - dist_short_paths: 739.0 m
[10, 2] path_dist: 754.0 m - dist_short_paths: 640.0 m
num. not optimal paths: 8/110 - algorithm efficiency: 92.73 %
```

Listing 5: Waalre Q-LSP paths that differ from the shortest path and algorithm efficiency.

C.2.2 Q-LEA algorithm results

In the case of the energy-aware path planner, when the resulting paths are correlated with the expected most energy-efficient paths obtained with the brute-force method, the success rate is also around 92-94%.

Listing 6 contains an example of the routes that might differ (which can change with every execution), where "`path_ene_cons`" is the energy consumption estimated with Q-LEA and "`eff_path`" the expected minimum energy consumption (obtained with the brute-force method) between the same start and finish points.

```
[4, 2] path_ene_cons: 463914.9058 J - eff_path: 390900.8803 J
[5, 4] path_ene_cons: 459891.3869 J - eff_path: 414367.6276 J
[6, 1] path_ene_cons: 651671.7233 J - eff_path: 322363.3264 J
[8, 0] path_ene_cons: 836449.7870 J - eff_path: 828590.3218 J
[10, 0] path_ene_cons: 1020217.8598 J - eff_path: 887687.4205 J
[10, 2] path_ene_cons: 572375.8685 J - eff_path: 447704.8944 J
num. not optimal paths: 6/110 - algorithm efficiency: 94.55 %
```

Listing 6: Waalre Q-LEA paths that differ from most energy-efficient route and algorithm efficiency.

C.2.3 Q-LEA paths compared with Q-LSP

Listing 7 presents the energy consumption of the paths that differ between Q-LSP and Q-LEA. Using real data, the difference between the routes is not much. This might be due to the similarity between the routes. Nonetheless, in some cases the difference can be of up to 8 kJ (1%), which might increase in longer distances.

It is worth to mention that some routes appear to have a huge difference, but if Listings 5 and 6 are considered, some of these routes are not the optimal.

As aforementioned, "`path_ene_cons`" is the energy consumption estimated with Q-LEA and "`path_short_cons`" the estimated value of the route defined by Q-LSP between the same start and finish points. Additionally, the difference between these routes is specified next to "`diff`".

```
[0, 8] path_ene_cons: 828590.3218 J - path_short_cons: 836449.78698 J - diff: 7859.4652 J
[0, 10] path_ene_cons: 887687.4205 J - path_short_cons: 1063614.00053 J - diff: 175926.5800 J
[1, 8] path_ene_cons: 438495.8133 J - path_short_cons: 446355.27848 J - diff: 7859.4652 J
[2, 8] path_ene_cons: 388607.7957 J - path_short_cons: 396467.26085 J - diff: 7859.4652 J
[4, 2] path_ene_cons: 463914.9058 J - path_short_cons: 390900.8803 J - diff: -73014.0255 J
[5, 8] path_ene_cons: 365141.0484 J - path_short_cons: 373000.5136 J - diff: 7859.4652 J
[6, 1] path_ene_cons: 651671.7233 J - path_short_cons: 322363.3264 J - diff: -329308.3970 J
[7, 4] path_ene_cons: 535029.3967 J - path_short_cons: 624877.8161 J - diff: 89848.4194 J
[8, 0] path_ene_cons: 836449.7870 J - path_short_cons: 828590.3218 J - diff: -7859.4652 J
[8, 1] path_ene_cons: 438495.8133 J - path_short_cons: 446355.2785 J - diff: 7859.4652 J
[8, 5] path_ene_cons: 365141.0484 J - path_short_cons: 373000.5136 J - diff: 7859.4652 J
[9, 4] path_ene_cons: 550756.5995 J - path_short_cons: 640605.01885 J - diff: 89848.4194 J
[10, 0] path_ene_cons: 1020217.8598 J - path_short_cons: 887687.4205 J - diff: -132530.4392 J
[10, 1] path_ene_cons: 497592.9120 J - path_short_cons: 630123.35128 J - diff: 132530.4392 J
[10, 2] path_ene_cons: 572375.8685 J - path_short_cons: 565148.2707 J - diff: -7227.5977 J
total different paths: 15/110 - algorithm efficiency: 66.67 %
lower value than shorter path found: 10
greater value than shorter path found: 5
```

Listing 7: Waalre Q-LEA paths whose energy consumption differ from the routes determined by Q-LSP.

All in all, if only the optimal (and correct) routes are considered, the paths that differ between both algorithms are as shown in Listing 8. In such case, the Q-LEA manages to improve 6 % of the total routes.

```
[0, 8] path_ene_cons: 828590.3218 J - path_short_cons: 836449.7870 J - diff: 7859.4652 J
[1, 8] path_ene_cons: 438495.8133 J - path_short_cons: 446355.2785 J - diff: 7859.4652 J
[2, 8] path_ene_cons: 388607.7957 J - path_short_cons: 396467.2609 J - diff: 7859.4652 J
[5, 8] path_ene_cons: 365141.0484 J - path_short_cons: 373000.5136 J - diff: 7859.4652 J
[8, 1] path_ene_cons: 438495.8133 J - path_short_cons: 446355.2785 J - diff: 7859.4652 J
[8, 5] path_ene_cons: 365141.0484 J - path_short_cons: 373000.5136 J - diff: 7859.4652 J
num. improving paths: 6/110 - algorithm efficiency: 6.00 %
```

Listing 8: Waalre Q-LEA paths that improve the energy consumption of the routes determined by Q-LSP.

C.3 EV consumption consistency check

To make sure that the consumption values obtained make sense, a brief consistency check is provided in this section.

The battery of the Citroën C-Zero has a capacity of 16 kWh [42], additionally the energy consumption in mild weather conditions and combined roads (city and highways) is in average 145 Wh/km (ranges between 107-242 Wh/km), which is equivalent to 522 kJ/km (between 385.2-871.2 kJ/km). If this value is further divided by two, then the consumption per 500 m is expected to be around 261 kJ/500 m (ranging between 192.6-435.6 kJ/500m).

Now, considering the values reported in Table 17, the road segment between 0 and 1 has a length of approximately 500 m and a consumption of 390 kJ, which is a bit of an increase (on a factor of 1.5) from the expected average of 261 kJ, but is still inside the range of anticipated values computed above.

This increment might be due to the fact that NXP's Citroën C-Zero is equipped with extra components and devices, which makes it heavier (and therefore more energy-consuming) than the original vehicle.

Nonetheless, all the values gathered with the rosbags are reasonable considering the expected values.