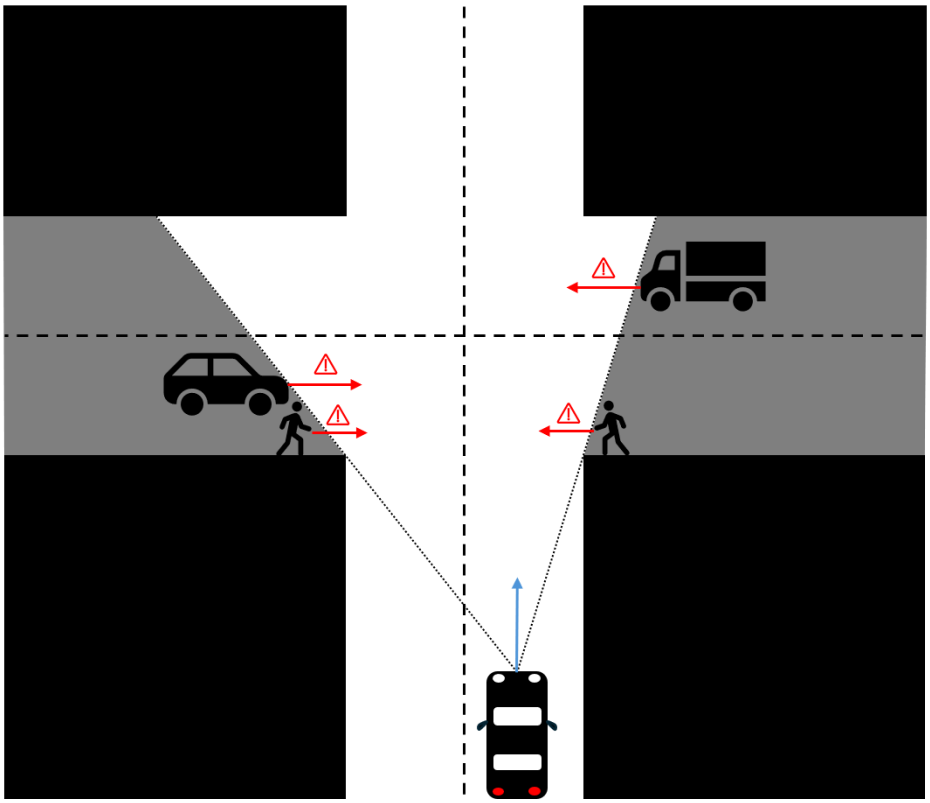
CHALMERS
UNIVERSITY OF TECHNOLOGY



Risk assessment for autonomous vehicles in occluded areas

Identifying and mapping potential risks originating from occluded areas

Master's thesis in Complex Adaptive Systems

Jonatan Andersson
Liam Hellring

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

MASTER'S THESIS IN COMPLEX ADAPTIVE SYSTEMS

Risk assessment for autonomous vehicles in occluded areas

Identifying and mapping potential risks originating from occluded areas

Jonatan Andersson
Liam Hellring

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2025

Risk assessment for autonomous vehicles in occluded areas
Identifying and mapping potential risks originating from occluded areas
Jonatan Andersson
Liam Hellring

Supervisor: Patrik Wallin, Aptiv
Examiner: Jonas Bärgman, Mechanics and Maritime Sciences

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:
Visualization of a vehicle approaching an occluded traffic scenario, where other road users could potentially emerge from occlusion and therefore pose a crash risk.

Department of Mechanics and Maritime Sciences
Göteborg, Sweden 2025

Risk assessment for autonomous vehicles in occluded areas
Master's thesis in Complex Adaptive Systems
Jonatan Andersson
Liam Hellring
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Chalmers University of Technology

Abstract

Similar to an experienced human driver, an automated vehicle needs to take the unknown into account while driving. This work proposes a way to perform risk assessment for autonomous vehicles when approaching areas that are occluded from the vehicle's sensor. The approach uses an occupancy grid and map data to describe risks associated with occluded areas. A risk assessment algorithm evaluates how occluded road users may pose a risk to safe driving, using the reachability of occluded road users through the use of phantom objects (e.g. objects that may be in an occluded area, with associated speeds, directions, etc.). The phantom object's predicted movement is modelled with an implementation of the A* algorithm, and this work incorporates a cost point in the A*'s heuristic function, to create realistic turning trajectories. The reachability is expressed in a grid as a probability of reach given a time horizon. The method may help automated vehicles evaluate the potential existence and movement of phantom objects to heighten their perception when navigating through an environment with occluded areas. Evaluation of the method was made on data collected from real traffic scenarios and self-created synthetic data depicting occluded scenarios in urban environments. Qualitative analysis of the results show how the method reliably finds occluded phantom objects and predicts reasonable trajectories of their movement. Successfully assigning the probability of a reach for each cell in the grid that a phantom object's predicted trajectory traverses.

Key words: Automated Vehicles, Occlusion, Risk Assessment, Phantom Objects, Occupancy Grid, Standard definition Map

Acknowledgements

We would like to thank Aptiv for the trust and support we have gotten during this project. A special thank you to our supervisors Patrik Wallin and Jonathan Jansson for their valuable guidance and support throughout this thesis. We would also like to thank our supervisor and examiner Jonas Bärgman for his helpful insights and feedback.

Göteborg 2025

Jonatan Andersson
Liam Hellring

Contents

List of Acronyms

AV	Automated vehicle
ADS	Automated driving system
Euro NCAP	The European New Car Assessment Programme
ORA	Occluded Risk Assessment
Radar	Radio detection and ranging
V2X	Vehicle-to-anything
BEV	Bird's eye view
LiDAR	Light detection and ranging
PO	Phantom object
SRQ	Simplified reachability quantification
UTM	Universal Transverse Mercator
HD	High definition
SD	Standard definition
EP	Emergence point
CDF	Cumulative distribution function
PDF	Probability distribution function

1 Introduction

Driving includes the continuous task of risk assessment. Human drivers adapt when they foresee the risk of hazardous events. However, other road-users' deviations from a driver's expectations can lead to confusion and errors [1]. According to [2], one cause of dangerous events between cars and pedestrians is occluded vision; when the driver does not see the road user and therefore cannot adjust to the situation. A common area where this problem presents itself is in urban environments, because they are densely filled with road users and buildings, and consequently, presenting more frequent occlusion and more complicated traffic scenarios [3]. One typical scenario where occlusion poses risk to road users is in intersections, where a large proportion of severe accidents in traffic occurs [4]. Due to the possibility of road users' paths crossing, all road users need to be cautious to avoid collision. The intuition of human drivers helps but human expectations and risk taking are still an issue. A potential solution to this is automated vehicles (AVs), which is backed up by data shown by [5]. Their analysis of pre-accident scenarios show that only 1.8% of AV accidents are due to inattention or poor driving behavior, compared to the 19.8% for human drivers.

In general AVs are, however, still facing unsolved problems and lack the robustness necessary for complete trust in its abilities [6]. An acknowledged problem the AV faces is occlusive and uncertain environments during decision making [7]. Similar to human drivers, AVs also need to consider potential threats that are hidden when sensing is occluded. An AV makes use of automated driving systems (ADS) [8] to control the vehicle. The ADS utilizes various sensors to depict its immediate surroundings and get an understanding of the environment. Algorithms then use the sensor's information to navigate traffic. The European New Car Assessment Programme (Euro NCAP) test the safety of new cars by putting them through a set of risk filled scenarios. These tests evaluate how the assessed cars perform in crash avoidance and crashes. When testing crash avoidance, several of the risk filled scenarios involve road users emerging from outside the vehicles vision, i.e. from occlusion. Currently, tests including occlusion are only tested for vehicles mainly controlled by a human driver. To perform well in these situations, the vehicle needs a fast reacting and smart reactive system that can manage to steer or brake in time to avoid crashing. If the vehicle was driven with an ADS instead of a human, and had the intuition to anticipate a dangerous event, it could take preventive measures to increase the likelihood of avoiding a crash or even getting into a conflict situation in the first place. However, ADS algorithms lack the intuition human drivers possess. This means that when placed in an occluded environment, the task of controlling the AV requires algorithms that can account for the occlusion and help the ADS get the information it needs to drive safely.

This work presents the development and assessment of the Occluded Risk Assessment (ORA) algorithm, meant to support ADS' decision-making in path planning by estimating risks present in occluded areas in the AV's environment.

1.1 Related works

To assess risk from occlusion there are several essential steps that need to be completed. First, an interpretation of the environment needs to be performed and where occlusion occurs needs to be established. From an interpretation of the environment, it is then necessary to analyze what significance the occluded areas have. Lastly, this information needs to be translated into an output that the ADS path planning algorithm can use, highlighting the risks stemming from the occlusion.

1.1.1 Interpreting the environment

To find where occlusion occurs, the AV's environment needs to be interpreted. An interpretation is made from sensor readings of the surrounding environment. There are multiple approaches for making an interpretation, using different methods and sensors for detecting what lies within the surrounding environment. In [9], five different object detection methods using radio detection and ranging (radar) point clouds are compared. The radar point clouds are used to detect where other road users such as vehicles and pedestrians are positioned in the AV's vicinity. The five methods consist of a combination of various methods such as clustering algorithms, point cloud filtering and machine learning. However, as these five methods do not detect elements in the environment such as infrastructure or where the drivable road is, it lacks important information to find where occluded road users can exist.

Another way of interpreting the environment is to make use of vehicle-to-anything (V2X) communication to find what is present in the AVs vicinity [10]. V2X communication systems typically makes use of a vehicle communication device, pedestrian communication device, roadside sensing unit, and an information processing unit [11] to find all other road users nearby. It does this by communicating with the systems on other road users such as vehicles and pedestrians, as well as infrastructure to get information about positions and movement. This method works given that all road users possess a viable communications system. If not, road users will go unnoticed and therefore unaccounted for, leading to potential dangers. A solution to this is merging a V2X system with the on board sensors with joint/integrated communication and sensing to improve the reliability of the system [11]. This could be realized by using the on board sensors to model the environment and compliment this model with the V2X communications to set a confidence to the measured information [12]. The on board sensors find an occluded area in the environment and the V2X communications either confirms that the area is occluded or denies it by, for example, utilizing the on board sensors of another vehicle to see the occluded area. However, V2X systems still face the issue of security. That is, since the vehicle is communicating with other devices, it is subject to potential malicious attacks and data privacy breaches [13]. This is an issue that would have to be solved before the implementation of such a system.

A different way to interpret the environment is to make a visual representation of the environment. To create a visual representation, sensor readings from on board sensors could be used to depict the environment with a bird's eye view (BEV) [14] by projecting a light detection and ranging (LiDAR) point cloud to a 2D overview. The BEV aims to classify the whole visible area of the environment with high resolution by making use semantic segmentation [15]. When projecting the LiDAR's sensor measurement into BEV, there will be gaps in the point cloud that grows larger with

the distance from the LiDAR. The gaps are caused by the lasers being placed with a fixed angle to each other, resulting in gaps that increase more with distance, and not by obstruction of the laser. [14] solves this through a method called inpainting, that with the help of neural networks fills in the missing information based on the context of the points that are classified previously. The result is a complete BEV representation of the visible environment and thus, a clear outline of what is visible and what is occluded.

Another solution for depicting the environment is to make use of an occupancy grid [16]. In [17] a dynamic occupancy grid is made from radar detections: cells in the grid are classified either as free, occupied by a static object, occupied by a dynamic object, or unknown. The classification of a cell is decided from a probability distribution between the classes, these probabilities are calculated by making use of the range rate of the radar measurement to estimate what the cells can contain. To help the classification algorithm, [18] make use of cells evaluated as occupied as reference points and analyzes the association probability between cells to section separate occupied sections into different objects. These objects are then given bounding boxes for tracking. To achieve a more robust dynamic occupancy grid, [19] combines the occupancy grid made from radar detections with laser measurements; using radar detections for object's velocity and the laser measurements for position and orientation. The detections help the environment interpretating algorithm find where there is free space by the area separating the sensor source and the detection. The cells behind occupied cells where no detections are made, get the unknown class and therefore tell where occlusion occurs. The use of an occupancy grid simplifies the environment compared to the aforementioned BEV, resulting in a less complex model of the environment to further analyze.

1.1.2 Analyzing occlusion

To know what is hidden from the AV's sensors, the occluded areas found from the interpreted environment need to be analyzed. The aim of the analysis is to find what significance the occluded areas have for finding potential risks affecting the task of driving the AV.

An example of finding what significance an occluded area have on the AV is presented in [20]. [20] make use of deep reinforcement learning to drive the AV and acknowledge where occlusion appears, teaching the AV to move in a way that emphasizes trying to spot potential vehicles in the occluded areas. By so, directly finding what significance occluded areas have. This method does not treat the task of finding risks stemming from occluded areas separate to the driving task but instead teaches the AV to move in a way that maximizes the AV's sensors' vision.

A different way to find what significance an occluded area could have on the AV, without driving the AV, is to place a traffic participant or "phantom object" [21][22][23] in the occluded area. Hereafter, a hypothetical occluded road user will be referred to as a phantom object (PO). To know which occluded areas to place POs within and where in these areas to place them, the cause of the occlusion and the scenario depicted by the modelled environment need to be further analyzed. For example if there is a connecting road or sidewalk being occluded, where in the

modeled environment could a PO possibly emerge, and how far into the path of the AV would it be able to get. [21] analyzes if an occluded area is deemed as hazardous by, for example, checking if the occluded area is behind a parked car. If an occluded area is deemed as hazardous, a PO is placed in the occluded area. A way to get a deeper understanding of the occluded areas when placing POs, is by making use of a precise road map. With a precise roadmap, it is possible to pinpoint where in the occlusion the road is, and with that, make a more accurate representation of what may lie within [22]. Using map data is also beneficial for understanding what occluded areas are relevant, because POs can be placed where they are most probable to exist. A PO classified as a vehicle would be placed on a road and specifically on the correct side of the road, and a pedestrian would not emerge behind a dynamic object in the middle of the road. While a vehicle driving on the wrong side of the road, and a pedestrian jumping out in front if dynamic objects is possible, it is very improbable. Therefore, by placing POs in areas where they are probable to exist, the POs' relevancy for the AV's risk assessment increases. Making the POs relevant are important to not make the output unrealistic and conservative. If the output of a risk assessment tool used by a path planner in an AV would be unrealistic and conservative, it would lead to conservative behavior by the AV which lessens the driving quality and can cause accidents by the AV driving unexpectedly [24].

To further understand what influence the occluded areas could have on the AV, predicted movements of the POs are modeled. A PO's movement is modelled differently depending on what type of PO it might be and what scenario it is placed in. For example, [21] only considers pedestrians as POs and assumes they move in a straight path across the road from the aforementioned hazardous areas, see *Figure 1*.

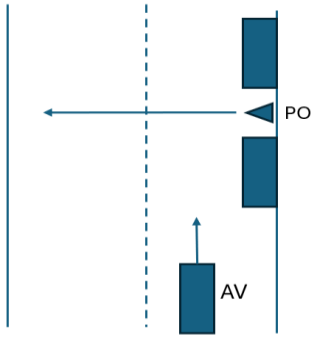


Figure 1. A PO moving with a straight path across the road

For this scenario, the area between the objects on the right side of the road is deemed as hazardous and a PO is then placed inside of this area and given a possible path of right across the road. Instead of considering distinct paths of the POs, [23] looks at the

possible interaction area between POs and the path planned by the path planner in the AV, see *Figure 2*.

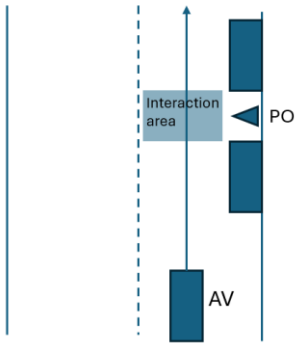


Figure 2. Interaction area between a PO and a given path for the AV

Another example of considering distinct paths is shown by [22], who only consider vehicles as POs and makes use of map data to model the paths given the geometry of the road network and their directions, see *Figure 3*.

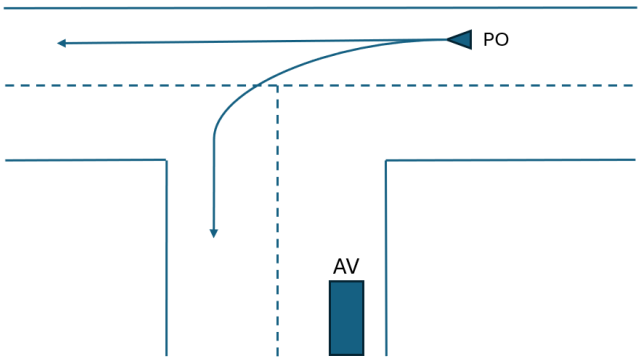


Figure 3. Potential paths given to a PO by following road network and directions

1.1.3 Risk assessment

When an understanding of what occluded areas are relevant and what could be hidden within these occluded areas, the risk which POs poses upon the AV need to be assessed. For example, [23] expresses the risk as potential collision risk between the AV and a PO. The collision risk is calculated by scaling the probability of a PO crossing the road with a risk distance coefficient. The probability of a PO crossing the road is calculated by assessing environmental factors such as the number of lanes and accessibility to a crosswalk. The environmental factors all carry a possibility of a pedestrian crossing the road, for example, “more than 90% of people would chose a crosswalk in road sections with signal control” [23] so if there is an occluded section between two cars on the side of the road but there is a cross walk next to it, the probability of a PO classified as a pedestrian crossing will be lower than if there was no crosswalk. The probability of a PO crossing the road is then scaled with a risk

distance coefficient. The risk distance coefficient is meant to symbolize the PO's ability to move or react if the risk of collision were to present itself and is calculated from the distance a PO has to the intersection area, the standard deviation, the degree of and other factors such as the attention of the PO, and the perception capabilities of the PO. It is assumed that a PO can move anywhere inside of the interaction area which gives a continuous risk value for the whole interaction area.

Instead of making use of an interaction area where a risk of collision is calculated, [25] gives a risk potential field to identified obstacles and makes use of a cost function to evaluate the risk the objects pose upon the AV. Since occlusion is caused by other obstacles, [25] gives a potential risk field to the obstacles and by so assuming that there is a risk of a PO being occluded close to these obstacles. A cost function is used to express a higher risk of collision when the AV is close to the risk fields caused by the obstacles, making the AV keep sufficient distance to the obstacles when driving. [21] also makes use of a cost function to control the AV but instead of depending on risk fields, the cost function depends on the potential harm to a PO. [21] only considers pedestrians as POs and are therefore calculating the potential harm to a pedestrian, given a collision with a pedestrian emerging from an identified occluded area and the velocity of the AV from a planned trajectory. The cost function makes sure that if the potential harm is too severe, the trajectory is discarded and for cases where the harm is under an acceptable threshold, the cost function helps the AV drive in a way meant to minimize potential harm.

A way to make the risk assessment algorithm more aware of potential movement from the POs is to make use of reachability [26][27][28]. The idea of reachability is that, given that a PO exists, where could it, spatially, reach. For example, a PO classified as a vehicle would move along the road and move with traffic. [26] places a particle on a known road inside an occluded area and propagates it forward in time with a set speed. A particle is spawned on one of the possible centroids of a vehicle in the occluded area considering a random displacement in the lane from a uniform distribution. By spawning a large enough set of particles and evaluating where they can end up given a time horizon, the reachability for a possible PO in an occluded area can be found. [27] extends on [26]'s method but instead of sampling particles, uses a simplified reachability quantification (SRQ) from the occluded areas. The SRQ makes a uniform distribution of possible velocities from zero to a set maximum and with a given maximum time horizon, the reachability from an occluded area can be expressed as the distribution of positions from the SRQ. Risk of a PO existing at a certain position after a certain time can then be assessed by looking at the SRQ.

1.2 Aims, objectives and scope

The aim of this work is to enable ADS path planning algorithms to account for risks stemming from occluded areas into account when planning a path. This objective is met by addressing the following objectives:

- Identify where phantom objects can emerge from occlusion.
- Predict the potential trajectories of these phantom objects over a defined time horizon.
- Assess the risk of a phantom object reaching each cell in their predicted trajectory.
- Evaluate if the ORA algorithm finds the expected phantom objects and successfully model their trajectories.

That is, in this work we develop the ORA algorithm, capable of identifying and mapping potential risks originating from occluded areas, for use in ADS path planning algorithms. Compared to other related works, this work does not include a path planner but a tool that presents information an ADS algorithm may use as part of its path planning algorithm. The ORA algorithm is made using a relatively course grid to be able to run in a real-time system. This work makes use of phantom objects (POs) with an initial position inside an occluded area. A potential trajectory will be assigned to each PO. The trajectory will be used to calculate which cells they can reach in a given time horizon. By doing so, the ORA algorithm can estimate a total risk of reach for each cell in a predicted trajectory from a potential PO emerging from occlusion.

2 Method

This section describes the different method steps used to create the Occluded Risk Assessment (ORA) algorithm, and provide the theory behind them. This includes a description of the data used for this work, how the input data was interpreted in the algorithm, how the occluded areas are analyzed, how POs are modeled, and the creation of the risk assessment algorithm.

2.1 Dataset

This project was mainly conducted using real data collected from real world traffic scenarios. Apart from the real data, a self-created synthetic dataset was made where all aspects of the data were controlled. Thus, presenting the ability to model relevant scenarios for testing. This work's focus was on urban environments as occlusion is most prominent in these areas and the real data was therefore collected in urban environments, and the synthetic data was made to represent scenarios which are common in urban environments.

2.1.1 Real data

The real data used in this work comprised three main parts: the occupancy grid, the vehicle data, and the map data. The occupancy grid and vehicle data were supplied by Aptiv [29], whereas the map data used was from OpenStreetMap [30].

The occupancy grid for the real data used in this project was made up of four classes: free, static occupied, moving occupied, and unknown. These classes and their corresponding color in the output grid are displayed in *Table 1*, and an example of an occupancy grid is visible in *Figure 4*.

Class	Label	Color
Free	0	
Static occupied	1	
Moving occupied	2	
Unknown	3	

Table 1. Labels and corresponding color for classes in occupancy grid

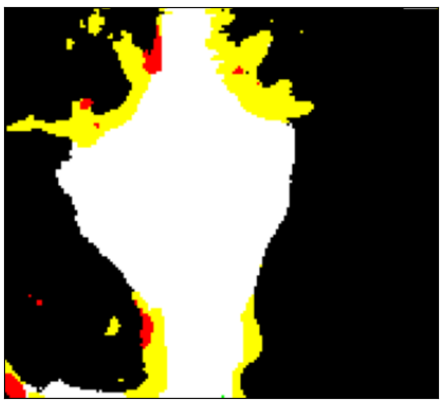


Figure 4. Example of an *occupancy grid* with 144x160 cells with a cell size of 0.5x0.5 m

The occupancy grid was based on sensor measurements from an AV and has a grid size of 72 m longitudinally (forward of the AV) and 40 m laterally (both right and left). A common speed limit in urban environments is 50km/h and with a longitudinal length of 72 m, 5 second time horizon predictions are possible. The reason to not have 72 m in both directions laterally was due this work's focus on urban environments which are commonly occluded laterally. In these occluded areas, there will most likely be an obstacle obstructing the sensors within 40 m laterally, meaning a larger grid would not give more relevant information. The grid does not extend to behind the AV, as it is not relevant for the future of the AV. Where the middle of the front of the AV was located in comparison to the grid was marked with a single green cell on the bottom row of the occupancy grid. The cell size chosen was 0.5 m as this was an appropriate size to fit the smallest object of interest, which was pedestrians. Even if smaller cell sizes also would work, by limiting the number of cells, less computation was needed, resulting in faster runtime.

Accompanying the occupancy grid was the vehicle data collected simultaneous as the sensor measurements that were used to create the occupancy grid. The AV's position data used in this project were provided in the Universal Transverse Mercator (UTM) and its heading angle was relative to east. UTM coordinates are expressed as meters northing and meters easting from the origin, if you view the world as a flat map. The origin is the same as for zero degrees latitude and zero degrees longitude in the WGS84 map projection. As UTM coordinates do not include height, it instead divides the earth into UTM zones to be able to more precisely express positioning. This data was gathered with high precision positioning, combining satellite system data, vehicle sensors and software to more precisely measure absolute position [31] in UTM coordinates.

Apart from the data collected when driving, map data was used to help the ORA algorithm better understand the environment of the AV. The map data was collected from OpenStreetMap using the python library OSMnx [32]. OpenStreetMap is a community created, open access map, containing information about road networks and infrastructure. The level of detail of some information in OpenStreetMap is comparable to that found in a High-definition (HD) map [33]. An HD map is constructed of nodes and edges with underlying information representing road networks. HD maps are defined by their ability to show in depth information such as crossroads, curbs, street signs etc. However, HD maps are costly and run the risk of being outdated due to their level of detail. Its counterpart, standard definition (SD) maps [34] are cheaper and more easily accessible but still share the most vital information for our task. That is, an SD map contains edges representing roads with their direction as well as the speed limit of the road, and nodes connecting the edges (see *Figure 5.*).

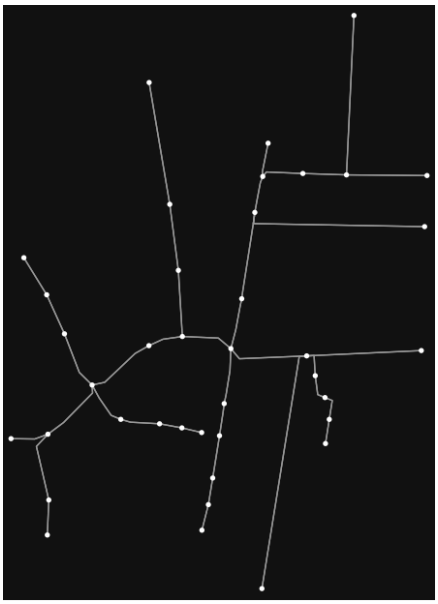


Figure 5. Example of a map gathered from OpenStreetMap and then simplified to represent an SD map. The dots are nodes, and the lines connecting them are edges.

Therefore, this work opts to use an SD map due to its accessibility and reliability. Since we did not have access to a real SD map, we needed to use OpenStreetMap. Therefore, this work modifies a map gathered from OpenStreetMap to resemble an SD map, containing a graph of edges for roads with direction and speed limits, and nodes for crossing edges.

2.1.2 Synthetic data

The synthetic data consists of three components. These components include an occupancy grid, a graph grid, and a speed grid. Synthetic scenarios were created to mimic different Euro NCAP scenarios and modeled in the form of occupancy grids. In order to simulate the scenarios, the scenes were created as extended grids of a continuous scenario, with the correct width a full-size occupancy grid but with extended height. The scene was then simulated by picking out frames from the extended grid along the height axis, where one frame has the same dimensions in both width and height as a full-size occupancy grid. By moving the frame one row at a time over the extended grid, it was possible to showcase a continuous scenario frame by frame in the form of grids, see *Figure 6.* The graph and speed grid were extended in the same manner.

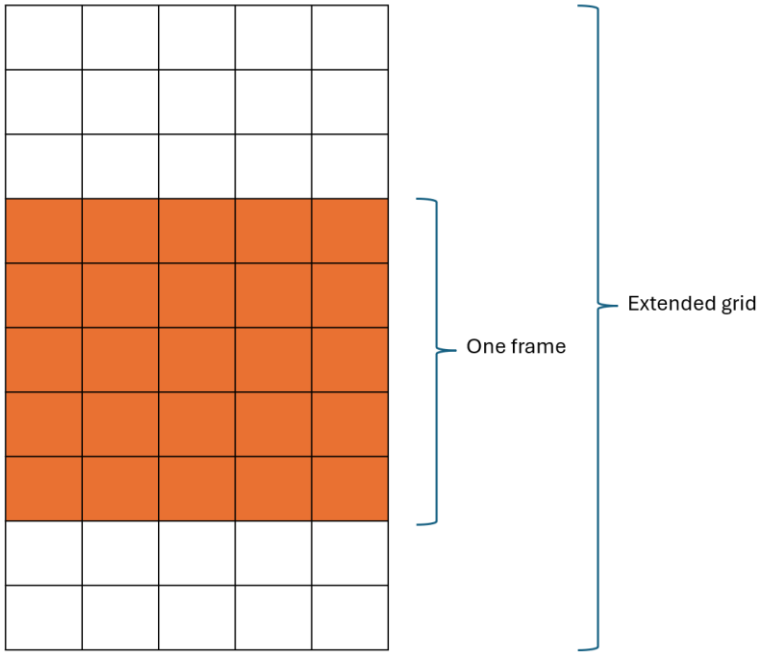


Figure 6. Extended grid with one selected frame for simulation with synthetic data.

A frame here refers to one full size occupancy, speed, and graph grid. The different scenarios were created in excel and then converted to a csv file as input to the algorithm for simulation.

The occupancy grid for synthetic data differs from the real data as the scenarios were created without defining where occlusion occurs, only where static objects were, where the free space was (the road), and an additional class named Occupied not blocking. In scenario S2 (see section 2.1.3.3.2.), the objects that were supposed to cast a shadow were the parked cars at the side of the road. The grey cells (occupied not blocking) in the scenario, were flat surfaces and they were not part of the road. The flat surfaces were not blocking the view of the intersecting road and should not cast a shadow. Therefore, the class Occupied not blocking (grey) was added to aid the construction of occlusion in the simulation, by defining it to not cast any shadows but not be classified as free.

An algorithm was created that simulates where occlusion should occur based on the occupied cells. Based on the position of the ego vehicle in the grid, the occupied cells would cast a shadow away from the ego vehicle and all cells making up this shadow would be marked as unknown. The class moving occupied was removed for the synthetic data due to how the scenarios were modeled, where the whole scenario was predefined and static (not to be confused with the class static occupied). All classes and their corresponding label and color for the synthetic occupancy grid are displayed in *Table 2.*

Class	Label	Color
Free	0	
Static occupied	1	
Unknown	3	
Occupied not blocking	4	

Table 2. Labels and corresponding color for classes in the synthetic occupancy grid

For the synthetic data, the following knowledge was assumed: where the roads were located, the speed of the roads, the center of the roads, and where the roads intersect. This information was stored in the graph and speed grid, they both have the same size as the occupancy grid to know exactly what cells correspond.

In the graph grid each cell was marked as 1 if they belong to the edge corresponding to the road that the ego vehicle was driving on. If they were on an edge for another road, they would instead be marked 2, and 5 if the cell were at an intersection point between two edges. The edges only occupied the center of the roads; all other cells were left empty. Therefore, the algorithm can find the closest intersection point and check if a cell was on an edge or not, including whether it was an intersecting edge. All edges (roads) in the synthetic data were assumed to be able to be two-way.

In the speed grid, all cells were marked with the speed limit of the road if the cell were on top of a road or as 0 if it was not on a road. This way the algorithm could deduce if a cell was on a road and what speed a phantom object could potentially have if it was placed in that cell. However, the direction of travel of the phantom object was still not specified.

2.1.3 Test cases

To test the performance of the ORA algorithm, scenarios with occluded areas were chosen as test cases. The test cases were occluded scenarios that are common to encounter in urban environments and where an understanding of what an expected result should look like could be established. For the real data, paused scenarios i.e. frames of such nature with corresponding occupancy grid, vehicle data, and map were chosen. The geographical location of the AV was known through the vehicle data, and beside the map data, satellite images of the location was also analyzed to get a further understanding of the location of the scenario. The information found from analyzing the satellite images was used to describe the scenarios. For the synthetic data, the occluded scenarios were inspired by Euro NCAP test scenarios.

2.1.3.1 Expected output labeling policy

By looking at the map and the traffic scenario depicted by the occupancy grid, possible POs and their trajectories were manually created by deciding where POs could emerge and where they were probable to move, for all scenarios to use for comparison with the output of the algorithm. The expected output was meant to represent what the output should include and not represent an exact ground truth. Meaning, it was created to be evaluate if with the expected POs and trajectories were found in the output.

No risk assessment was done on the expected output, it only represents the expected trajectory of a PO that moves with the same time horizon and speed as the POs in the actual output.

The map was used together with the occupancy grid to predict where POs were expected to emerge from occlusion. It is a rough prediction of where the POs were going to emerge and should not be interpreted as the exact location that the ORA algorithm should place them, but should be seen as a reference for the vicinity of an expected placement. To make it easier to match the map with the occupancy grid, the boundaries of the occupancy grid was drawn on the map. See *Figure D (b)* in section 2.1.3.2.1.

For expected vehicle POs, both a straight trajectory and a turning trajectory (if they were predicted to have one), were in the expected output. The expected output was not meant to represent the exact trajectory of a potential vehicle, that is, the expected output was meant to show which POs should exist and what type of trajectories they may have. Therefore, if a PO was expected to have a turning trajectory, the turning trajectory was modeled as a branch from the straight trajectory at a point where the branch's direction would, if continued, end up on the correct road at an expected end point. An example of POs with expected turning trajectories modeled as a branches from their straight trajectories is seen in *Figure 7.*

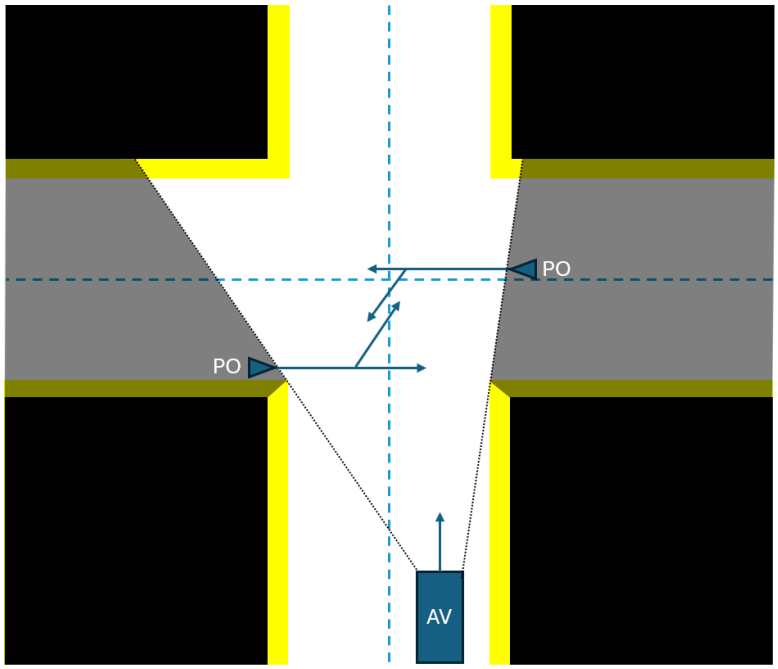


Figure 7. A four-way intersection with two expected POs. Both POs were expected to have a straight and a turning trajectory, where the turning trajectory looks like a branch from the straight trajectory.

Pedestrians were significantly less predictable in terms of where they might appear from since they do not follow traffic rules in the same manner as vehicles on the road. Therefore, not all potential pedestrians were included in the expected output. The pedestrians included in the expected output were those that came from behind

obstacles on the side of the road and those that came from intersecting roads with focus on occluded areas close to the AV.

2.1.3.2 Real scenarios

The real scenarios were saved frames of interest from the real data. These frames were selected to represent several real-life scenarios where parts of the environment were occluded. These scenarios were meant to showcase traffic scenarios where the output of this work could benefit an AV.

2.1.3.2.1 Scenario R1

The first real scenario (R1) was a four-way intersection where the AV stands still, with infrastructure and other static road users were causing occlusion, see *Figure 8.*

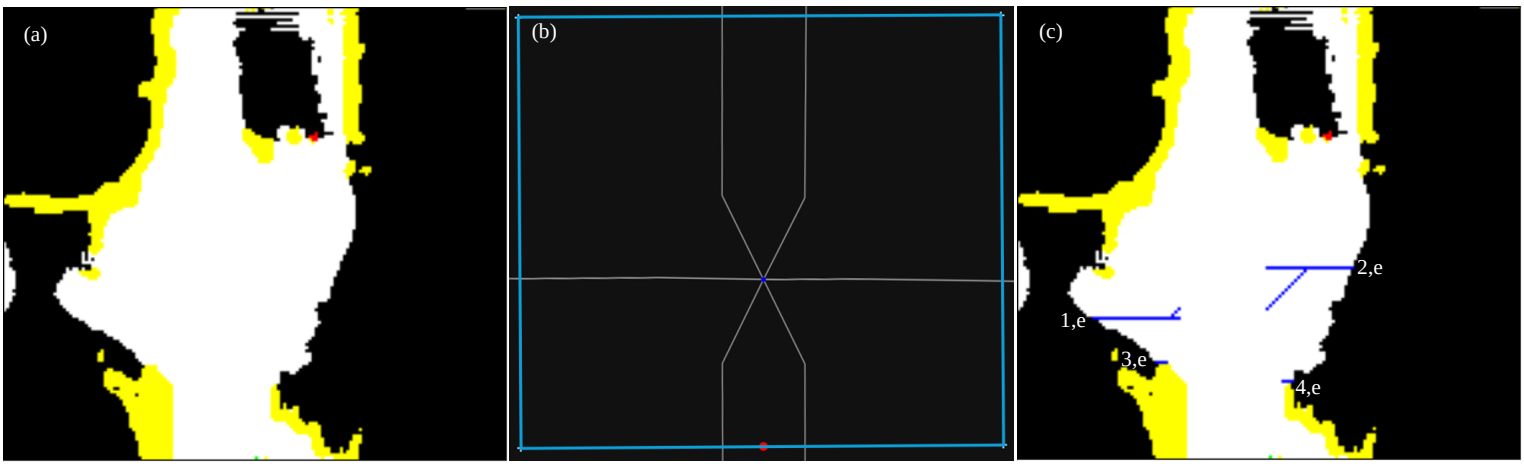


Figure 8. In scenario R1 the AV stands still, facing a four-way intersection. (a) input occupancy grid for scenario R1. (b) map of the scenario R1, the bounding box represent the borders of the occupancy grid, and the red dot represent the AV's position. (c) expected output of the scenario R1.

With the knowledge that the test scenario was a four-way intersection and by looking at the map of the depicted area (see *Figure 8. (b)*), two vehicle POs were expected (1,e and 2,e), one coming from each side of the intersecting road. Both POs should have a straight and a left turn trajectory. Aside from the two vehicles, there were expected to be at least two pedestrians (3,e and 4,e). There could be pedestrians at the other side of the intersection, however following the labeling policy for pedestrians, it was most vital that the algorithm detected the two closest pedestrians. Scenario R1 was chosen to test the algorithm on a four-way intersection when the map separates one of the intersecting roads into two edges.

2.1.3.2.2 Scenario R2

In scenario R2 (see *Figure 9.*) the AV was driving towards a four-way intersection, with all connecting roads being heavily occluded by infrastructure and other moving objects.

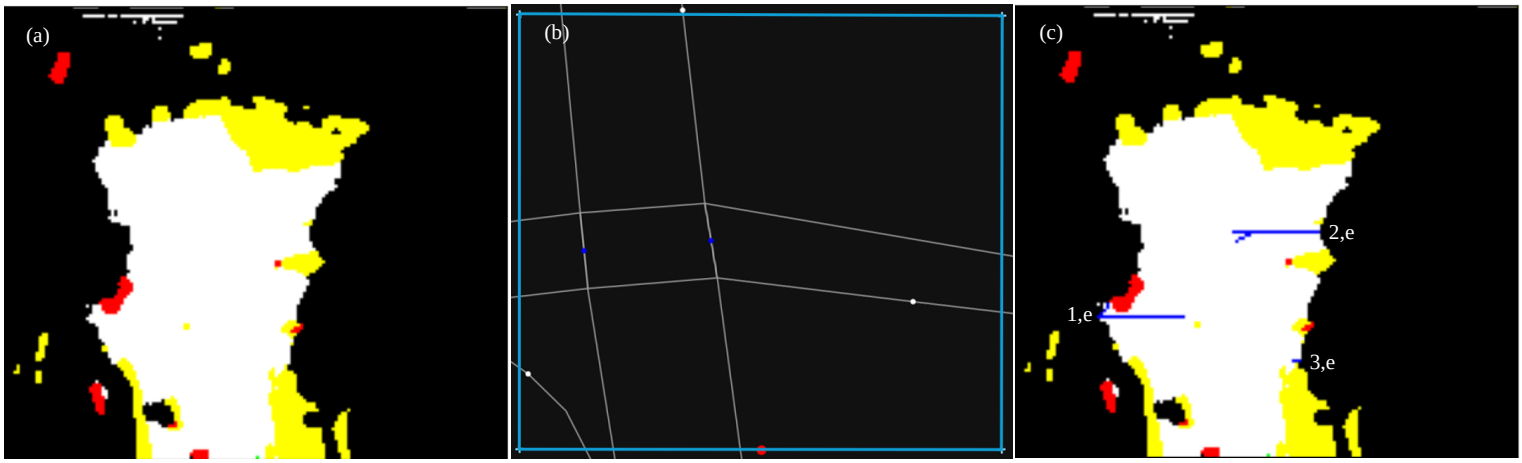


Figure 9. In scenario R2 the AV was driving towards a four-way intersection with infrastructure, and other moving objects causing occlusion in all intersecting roads. (a) input occupancy grid for scenario R2. (b) map of scenario R2, the bounding box represent the borders of the occupancy grid, and the red dot represent the AV's position. (c) expected output of the scenario R2.

Knowing that scenario R2 depicted a four-way intersection, there should be one PO emerging from each side of the intersecting road (1,e and 2,e), and these two POs should have a straight and a left turn trajectory. It was expected for a pedestrian PO (3,e) to emerge from the right of the AV, coming from the intersecting road and crossing in front of the AV. Scenario R2 was chosen to test how the algorithm worked when presented with a map that does not look as desired. The map was desired to have an edge in the direction of each intersecting road from a single node connecting an intersection, ideally looking like the map in Scenario R3 but also as in Scenario R1. In scenario R2, the simplification of the map resulted in two nodes making up the intersection and the edges representing the horizontally intersecting roads making abrupt "turns" into the nodes. Thus, the geometry of the edges in scenario R2 does not represent the actual geometry of the road network making up the intersection.

2.1.3.2.3 Scenario R3

In scenario R3 (see *Figure 10.*), the AV was once again driving towards a four-way intersection. No occlusion on the road straight across the intersection from the AV but the horizontally intersecting road was occluded on both sides by obstructing infrastructure.

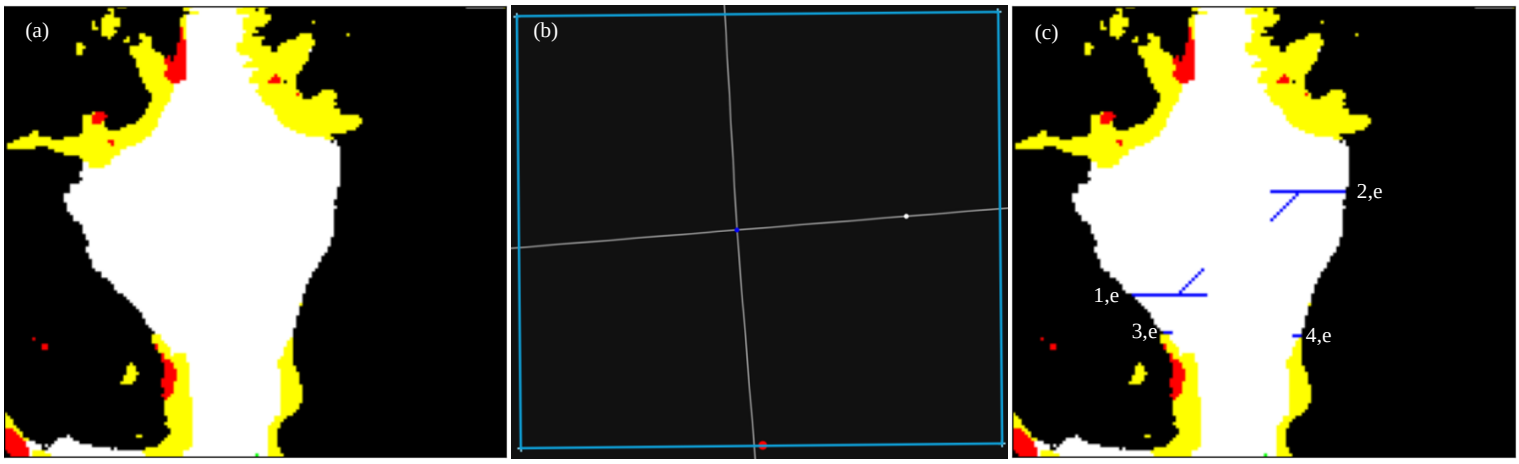


Figure 10. In scenario R3 the AV was driving towards a four-way intersection with infrastructure causing occlusion on both sides of the horizontally intersecting road. (a) input occupancy grid for scenario R3. (b) map of scenario R3, the bounding box represent the borders of the occupancy grid, and the red dot represent the AV's position. (c) expected output of the scenario R3.

The expected output for scenario R3 was that there should be a vehicle PO with a straight and left turn trajectory on each side of the intersecting road (1,e and 2,e). The expected trajectories represent. There should be two pedestrian POs (3,e and 4,e), one emerging from each side of the intersecting road, into the AV's road. Scenario R3 was chosen to test the algorithm when the map was simple and perfectly accurate to the scenario.

2.1.3.2.4 Scenario R4

In scenario R4 (see *Figure 11.*), the AV was driving on a straight road with several parked vehicles on the side of the road causing occlusion behind them.

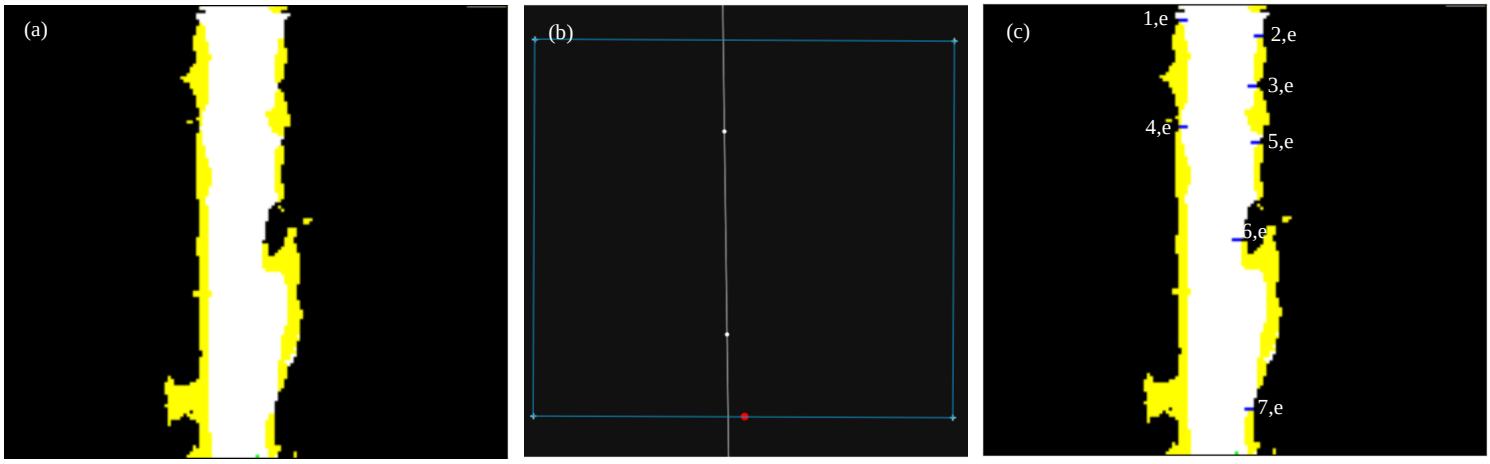


Figure 11. In scenario R4 the AV was driving on a straight road with vehicles parked on each side of the road. (a) input occupancy grid for scenario R4. (b) map of scenario R4, the bounding box represent the borders of the occupancy grid, and the red dot represent the AV's position. (c) expected output of the scenario R4.

The expected output for scenario R4 was that several pedestrian POs were emerging from each occluded area caused by parked vehicles. Scenario R4 was chosen to test the algorithm on a scenario with many smaller occluded parts where pedestrian could emerge, and where no vehicles were expected. All possible pedestrians emerging from occluded areas were included in the expected output for scenario R4 since the focus of this scenario is pedestrians.

2.1.3.3 Synthetic scenarios

Two synthetic scenarios were modeled with inspiration from Euro NCAP test scenarios. The Euro NCAP test scenarios include pedestrians and vehicles occluded by other parked vehicles.

2.1.3.3.1 Scenario S1

Scenario S1 is a combination between CPNCO 1.3.1 and CPNCO 1.3.2 [35]. Both of these scenarios present the case of two parked vehicles next to the road on the right hand side with the ego vehicle (AV in our case) driving past. CPNCO 1.3.1 show a pedestrian entering the road from behind both cars, CPNCO 1.3.2 show a pedestrian emerging from between the two cars. The modeled scenario combining the two show the AV driving straight ahead with two vehicles parked on the side of the road, introducing the risk of a pedestrian emerging from behind either of the two vehicles and between the two vehicles. This scenario is visualized in *Figure 12.*

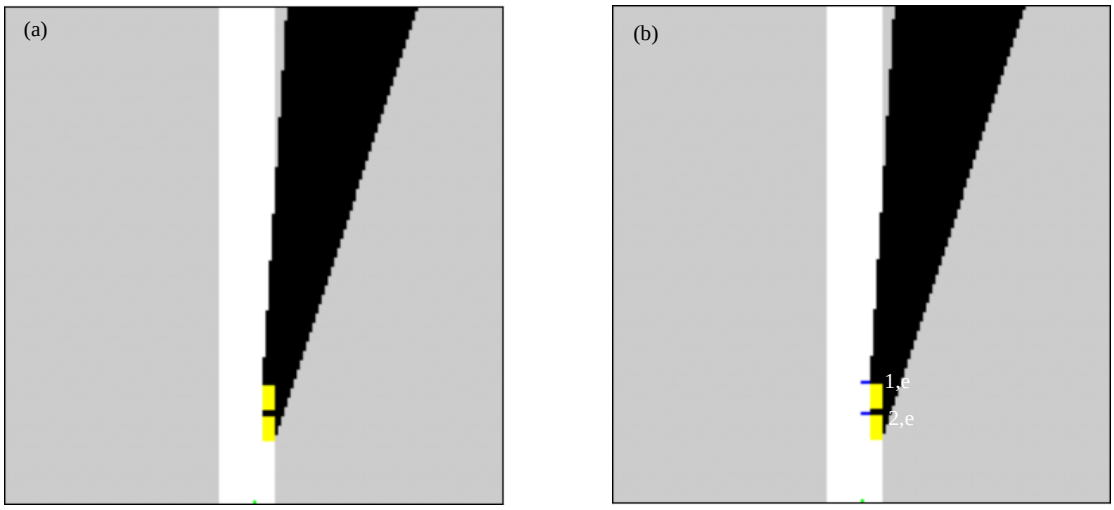


Figure 12. The first scenario, with two parked vehicles. Inspired by Euro NCAP scenarios "CPNCO 1.3.1" and "CPNCO 1.3.2". (a) input occupancy grid for scenario S1. (b) expected output of the scenario S1.

2.1.3.3.2 Scenario S2

The second scenario is based on "3.1.3.1 Car-to-Car Crossing" [36]. Here three cars were parked on each side of the road, ahead of an intersection. The AV was driving straight ahead, and a vehicle may emerge from each side of the intersecting road (1,e and 2,e). Pedestrians are expected to exist between and behind the parked vehicles (5,e – 10,e), similar to scenario S1, there is also expectations for two pedestrians coming from each side of the intersecting road (3,e and 4,e). The second scenario is visualized in *Figure 13.*

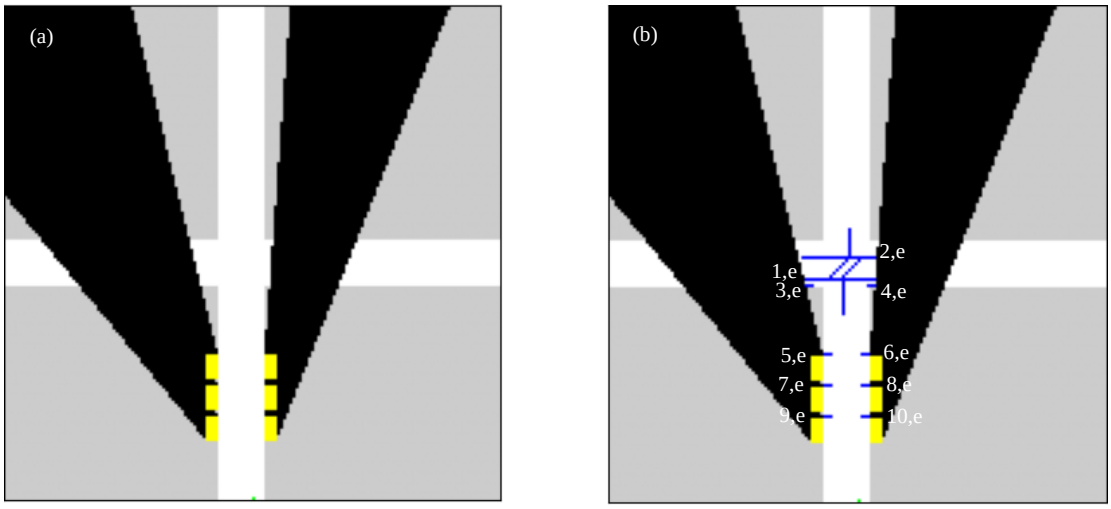


Figure 13. The second scenario, with six parked vehicles and an intersection. Inspired by Euro NCAP scenario 3.1.3.1 Car-to-Car Crossing. (a) input occupancy grid for scenario S2. (b) expected output of the scenario S2.

2.2 The ORA algorithm

The ORA algorithm followed a pipeline of rule based operations, see *Figure 14*. Beginning with interpreting the occlusion through outlining the area of free cells connected to the AV, this area was called the free area. The outline was then sectioned into emergence intervals. An emergence interval is defined as a set of connected unknown cells bordering the free area, and it was from emergence intervals that POs could appear from. The emergence intervals were further investigated, and an emergence point (EP) was placed at each cell a PO was probable to emerge from in the emergence interval. With EPs identified, POs were placed at the EPs and the POs' movement was modeled. The movement was modeled by finding where possible end points were located, meaning points where the POs were possibly traveling towards. Paths were then generated from the EPs to the end points by using a graph search algorithm. The paths were turned into trajectories, by the introduction of a time and speed dependency. Lastly risk assessment was done by using the POs' trajectories and a probability function that gave an output based on the probability for a PO to reach a cell within a certain time horizon.

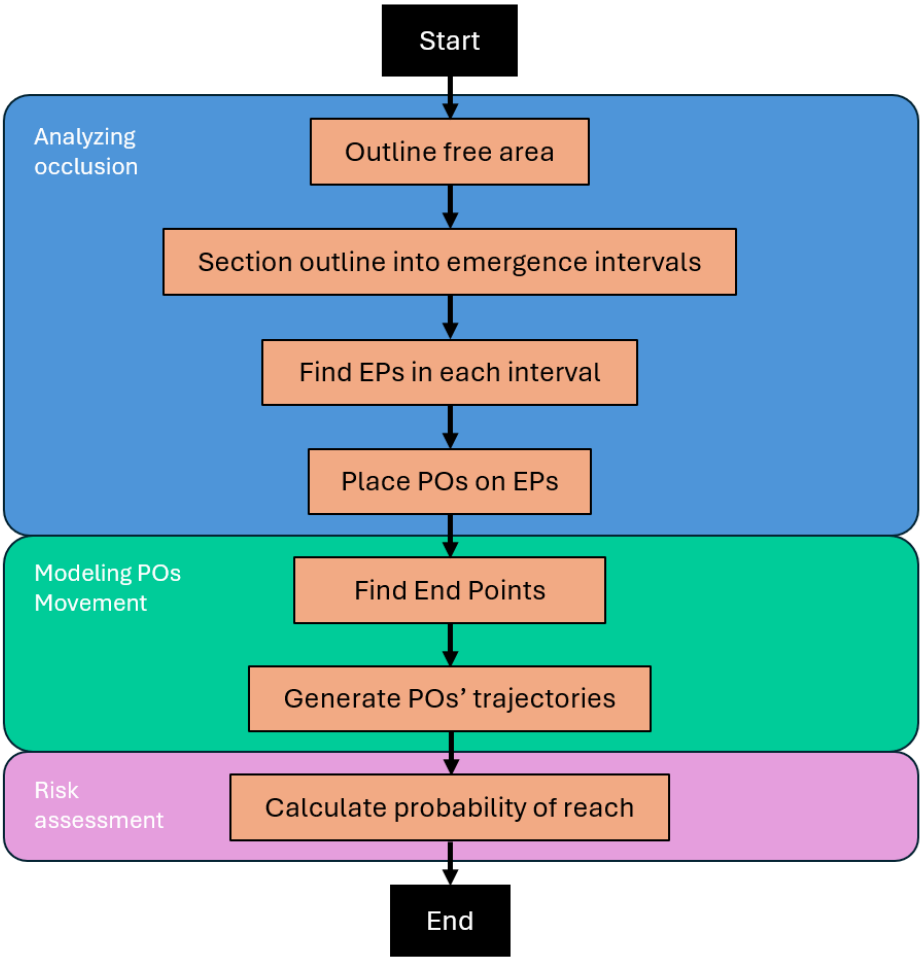


Figure 14. Block diagram, showcasing an overview of the pipeline for the ORA algorithm.

2.2.1 Analyzing occlusion

The first step of the ORA algorithm was to analyze the occlusion in the AV's environment, which was represented by the unknown occupancy grid class. When a cell has been given the unknown class, it was assumed that the view of the cell was obstructed by an object.

2.2.1.1 Identifying free outline

For a PO to pose a risk to the AV, it needs to enter the free area, the area encompassed by free cells that were connected to the AV. Because the free area was considered as space that the AV can drive on, if an object were to enter this area it would therefore be considered a risk. The first part of the algorithm was to outline all the cells that border the free area. The grid was explored with the help of a breadth-first search (see Algorithm 1). The search starts at the AV's position and navigates through the grid by recursively investigating all neighbors that were free cells, if a cell that was not of the free class appears it was labeled as part of the outline in a grid called *outline_grid*, that was a grid with the same dimensions as the occupancy grid. The before and after is shown in *Figure 15.* in section 2.2.1.2.

ALGORITHM 1: FREE OUTLINE

Output: Changes class variable *outline_grid*

```
1:  function free_outline( ):
2:      que ← [ego_position]  // position of ego vehicle in grid
3:      visited ← [array of zeros same shape as grid]
4:      while que:
5:          (x, y) ← que.pop(0)
6:          visited[x][y] ← 1
7:          if grid[x][y] ≠ free:
9:              outline_grid[x][y] ← outline label
10         else:
11             neighbors ← [array of all neighbouring cells]
12             for (i, j) in neighbors:
13                 if (i, j) in grid:
14                     if not(visited[i][j]) and (i, j) not in que:
15                         que.append((i, j))
```

2.2.1.2 Identifying emergence intervals

After identifying the cells that outlines the free area, the next step was to extract emergence intervals from the outline, the process is displayed in Algorithm 2. An emergence interval is a set of connected unknown cells bordering the free area, and it is from emergence intervals that POs can appear from occlusion. Because emergence intervals consist of cells bordering occlusion, they were part of the free outline that was created in the last part of the algorithm. Therefore, all unknown cells within the outline were extracted, and will be referred to as emergence cells. The emergence cells were grouped into intervals, this was done by extracting one emergence cell at a time, and recursively searching its neighbors for other emergence cells with the help

of a que. Any emergence cell explored in the recursion was marked as visited and added to a set representing the current emergence interval. Once the que was empty, the current interval was added to a set of intervals, containing all emergence intervals. The process was then repeated until all emergence cells have been visited. The before and after is shown in *Figure 15*.

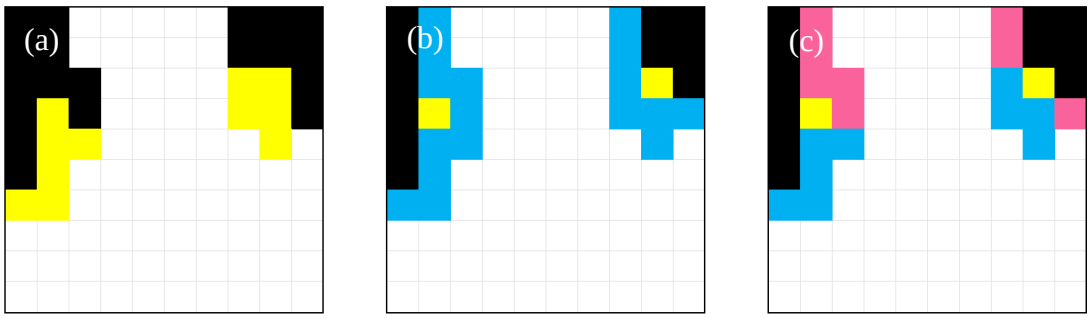


Figure 15. (a) an original occupancy grid, (b) occupancy grid after free outline identification, and (c) after emergence interval identification. The blue cells were part of the free outline, and the pink cells were part of some emergence interval.

ALGORITHM 2: EMERGENCE INTERVALS

Output: Changes class variable *intervals*

```
1:  function emergence_intervals( ):
2:      emergence_cells ← [where(cell is unkown, and in outline)]
3:      visited ← [array of zeros same shape as grid]
4:      while emergence_cells:
5:          que ← [emergence_cells.pop(0)]
6:          interval ← [ ]
7:          if not(visited[que]):
8:              while que:
9:                  (x, y) ← que.pop(0)
10:                 visited[x][y] ← 1
11:                 interval.append([x, y])
12:                 neighbors ← [array of all neighbouring cells]
13:                 for (i, j) in neighbors:
14:                     if (i, j) in grid
15:                         if not(visited[i][j]) and (i, j) not in que
                                                  and (i, j) in emergence_cells:
16:                             que.append((i, j))
17:         if interval:
18:             intervals.append(interval)
```

2.2.1.3 Identifying EPs

The identification of EPs starts by analyzing what kind of road users could emerge from occlusion in the current scenario. When faced with an intersection point there is a possibility that a vehicle can appear from occlusion. Therefore, the algorithm looks for the closest intersection point to the AV within the grid. If one exists, the emergence intervals were further investigated. Cells in emergence intervals adjacent to moving objects were not relevant, because anything occupying them would collide with the moving object. Therefore, if the interval was connected to an occupied moving object, it was adjusted by removing the four closest cells from the interval. After the adjustment was made, there was a check for if the number of remaining cells in the interval was greater than four. With 0.5 m in cell size and a car's width being around two meters, an interval of length greater than four cells was needed for a car to pass. If this was true, the relevancy of the interval was evaluated. Because of potential errors in the real data, sometimes there were spots of occlusion without anything blocking vision, meaning it should be visible. Therefore, the interval was regarded as irrelevant and filtered out if the first and last cells were neighbors. Because the interval was not connected with an occupied cell, the vision should not be obscured.

After the interval has potentially been adjusted and has passed the relevancy check, the algorithm looks for potential emergence points within the interval. The algorithm takes out the distance between all cells in the interval and their respective closest edge. These cells were treated as potential EPs. All potential EPs with a distance larger than 3.5 m were filtered out, because their distance was larger than one lane from the middle of the road. Then the algorithm checks if the EP's closest edge was the same as the AV's. One case where the EP and the AV share the same closest edge was when occlusion was caused by other traffic participant on the same road as the AV, for example a car being hidden behind a truck. However, this was a case outside of this project's scope. Another case was if the potential EP was coming from occlusion where no road was present. Implying that it was not an EP for a vehicle as there were no connecting edge that it can arrive from. Therefore, an EP was discarded if it shared the same closest edge as the AV. Otherwise, a check was made for if the edge was approaching the intersection, because only the EPs with a heading towards the intersection were relevant. Then, if the EP was approaching from the left, the closest edge within 3.5 m (average lane size) was kept and all others that share the same edge was discarded. If the EP was not approaching from the left, then the algorithm will look for the EP with the least distance to their closest edge and keeping only the closest. The difference in handling EPs for vehicles approaching from the left and right, comes from the assumption of right way traffic. Therefore, the nearest point of the vehicles was different. The Pseudocode for this process is displayed in *Algorithm 3.*

ALGORITHM 3: GET POTENTIAL EPS

Input:

Output: *indices:* the indices for the potential EPs within the interval, *edges:* the

```
function get_potential_eps(...)
    ep_dict ← { }
    edge_ids, edge_distances ← nearest edges(intervall_UTM)
    closest ← indices of where edge_distances < 3.5 m
    for index in closest:
        closest_edge ← edge_ids[index]
        if closest_edge is not ego_edge:
            // checking if the edge is heading toward intersection:
            if approaching from left: // see Equation 1
                if closest_edge not in ep_dict:
                    ep_dict[closest_edge] ← index
            else if edge_distances[index] < 2:
                if closest_edge not in ep_dict:
                    ep_dict[closest_edge] ← index
                else if edge_distances[index] <
                        edge_distances[ep_dict[closest_edge]] :
                    ep_dict[closest_edge] ← index
    return ep_dict values, ep_dict keys
```

The equation $approaching\_from\_left$ calculates if the EP is to the left of the intersection, from the perspective of the AV, by making use of the properties of a cross product.

$$approaching\_from\_left = (\|\boldsymbol{C_I} - \boldsymbol{C_{AV}}\| \times \|\boldsymbol{C_{EP}} - \boldsymbol{C_{AV}}\| > 0) \qquad (1)$$

Where **c** is the cell coordinates within the grid, expressed as vectors. I refer to the intersection.

Regardless of the presence of intersection points within the grid, the algorithm looks for EPs for pedestrians. All emergence intervals were considered to possibly contain an EP for pedestrians. What limited which emergence intervals EPs were placed in, was the evaluation of its relevancy in relation to pedestrians. This was done by checking if the first cell in the interval was neighboring a static occupied cell. If it does, then the first cell was considered an EP for a pedestrian. A pedestrian could potentially appear somewhere else in the interval. However, the most dangerous pedestrian was the one appearing closest to the AV which was why the first cell was chosen. It was also more realistic that a pedestrian would walk on the side of a road which most likely will be at the start of an emergence interval.

If there were no static occupied cells among the neighbors, this meant that it was a dynamic occupied cell causing the occlusion. In this case, it was concluded that placing an EP for a pedestrian in this emergence interval was irrelevant as there would

not be hiding any pedestrian walking into the road behind moving cars. This would also filter out the emergence intervals caused by errors, as mentioned in the first paragraph of 2.3.1.3.

2.2.1.4 Creating POs from EPs

At each established EP, a PO will be created. A PO's properties can be defined as:

$$PO = [(x, y),\ U,\ v,\ t,\ C] \quad (2)$$

Where (*x*,*y*) is the position of the PO in the grid, *U* is the classification of the PO, currently "Pedestrian" or "Vehicle", *v* is the speed of the PO, *t* is the time horizon which the POs movement will be predicted, and *C* is the closest intersection point.

The position of the PO was the same as the EP and the type of road user was set based on the type of EP. The speed of the PO was gathered by looking at the speed limit on the edge representing the road it was on. All road edges on OpenStreetMap does not have a speed limit specified, for these edges, a speed of 50km/h was given to vehicles as it is a common speed limit within urban areas. All pedestrians were given a speed of 5km/h which is the average walking speed for people aged 0-79 rounded up (from ~4.6km/h) [37]. The time horizon was specified when running the algorithm and the closest intersection point was found when identifying the emergence point.

2.2.2 Modelling PO's movement

When the PO was defined, its movement was modelled based on potential end points. End points are where a PO's path would end, i.e. the goal they were traveling toward. POs classified as "vehicle" got their end points from the closest intersection point. Where the outgoing edges represent the roads which a vehicle could exit the intersection through. The direction of each of the considered edges were made into vectors and a point seven meters (two lane widths) from the intersection point in the edge's direction was selected as an end point for the PO. For this project, only the end points resulting in a left turn or straight drive by the POs will be considered as right turns from POs were unlikely to cross paths with the AV. To check if the end point represents a straight path, the angle between the vectors from the EP to the intersection point and to the end point was compared with a max angle. The max angle was the angle from the PO, between the intersection point and a point 3.5 meters (one lane width) away perpendicular to the vector from the PO to the intersection point (see *Figure 16.*).

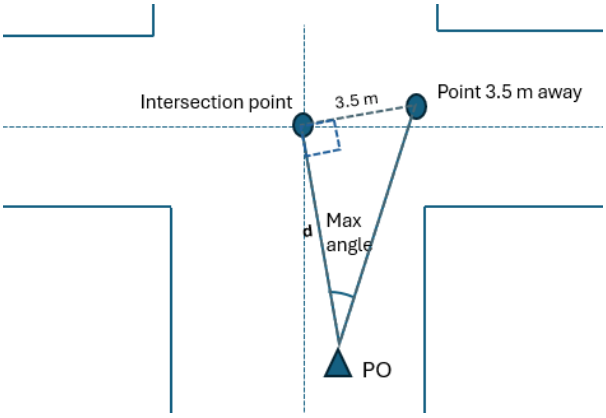


Figure 16. Visualization of how the max angle was found

For an end point to be considered straight, its angle to the intersection point from the PO must be less than the max angle. The max angle constructing two vectors: $\vec{\boldsymbol{A}} = \begin{bmatrix} d \\ 0 \end{bmatrix}, \vec{\boldsymbol{B}} = \begin{bmatrix} d \\ 3.5 \end{bmatrix}$, where d is the Euclidian distance between the EP and intersection point and 3.5 meters being the width of a lane.

$$max\_angle = \cos^{-1}\left(\frac{\vec{\boldsymbol{A}} \cdot \vec{\boldsymbol{B}}}{\|\vec{\boldsymbol{A}}\|\|\vec{\boldsymbol{B}}\|}\right) \tag{3}$$

If the angle to the end point was not less than the max angle, it meant that the path would be a turn. In order to only keep the left turns, a check was made for if the end point was to the left or the right side compared to the intersection point. This was done using the cross product of the vectors from the EP to the intersection point and from the EP to the end point. If the vector to the end point was to the left of the vector to the intersection point, the resulting cross product is positive.

$$is\_left = (\|\boldsymbol{C_I} - \boldsymbol{C_P}\| \times \|\boldsymbol{C_E} - \boldsymbol{C_P}\| > 0) \tag{4}$$

For each of the end points, a path was made with a search algorithm based on the A-star (A*) algorithm [38]. The modified version of the A* algorithm is displayed in Algorithm 4, with corresponding heuristic displayed in Equation 3.

ALGORITHM 4: A*

Input: *end:* indices of end cell, *start:* indices of start cell, *cost_point:* indices of

Output: *path:* [*list containing from start to end*]

```
function A*(end, start, cost_point):
    open_dict ← {start: infinity} // dictionary with cells and their f_score
    g_score ← [array with elements set to infinity]
    g_score [start] ← 0
    paths ← {start: [ ]} // dictionary with cells and their current best path
    while open_dict is not empty:
        (x, y) ← cell in open_dict with lowest f_score
        if (x, y) = end
            return paths[end]
        open_dict.pop((x, y))
        for (i, j) in neighbours:
            if (i, j) in grid:
                new_g_score ← g_score[(x, y)] + euclidian((x, y), (i, j))
                if new g_score < g_score[(i, j)]:
                    g_score[(i, j)] ← new_g_score
                    f_score ← new_g_score + heuristic(end, (i, j), cost_point)
                    paths[(i, j)] ← paths[(x, y)] + [(x, y)]
                    open_dict[(i.j)] ← f_score
```

Algorithm 4: describes modified A algorithm.*

$$heuristic = \|\boldsymbol{C_E} - \boldsymbol{C_{current}}\| - w_{cost\_point} \cdot \|\boldsymbol{C_{cost\_point}} - \boldsymbol{C_{current}}\| + \begin{cases} w_{static}, & if\ current = static\ occupied \\ -w_{free}, & if\ cell\ i = free \end{cases} \quad (5)$$

With chosen weights: $w_{cost\_point} = 0.8$, $w_{static} = 100$, and $w_{free} = 10$.

The heuristic used was a weighted sum of the Euclidean distance to the end point, the Euclidean distance to a "cost point" and the classification of the cell which it traverses. The cost point was a point that the vehicle should avoid and was placed at a point calculated by rotating the vector between the PO and the end point $\theta = 45°$ counterclockwise with the rotation matrix displayed in equation 2.

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (6)$$

This vector was then scaled with a factor of $\cos(45°) = \frac{\sqrt{2}}{2}$ to make the connecting lines between the PO, end point, and cost point form a right triangle, see *Figure 17*.

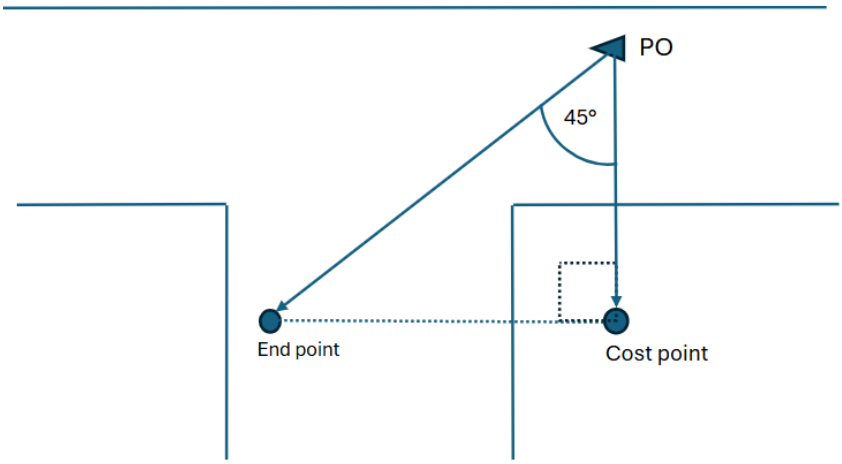


Figure 17. Visualization of cost point placement

The role of the cost point was to reward traversed cells more, the further they were from the cost point. By rewarding cells further away, trajectories avoid their respective cost point. The path does not necessarily become curved, but the turning becomes delayed. Instead of taking the shortest possible path, being the diagonal path, the paths would start by traveling straight or almost straight until it was more rewarding to approach the end point than avoiding the cost point further, see *Figure 18*.

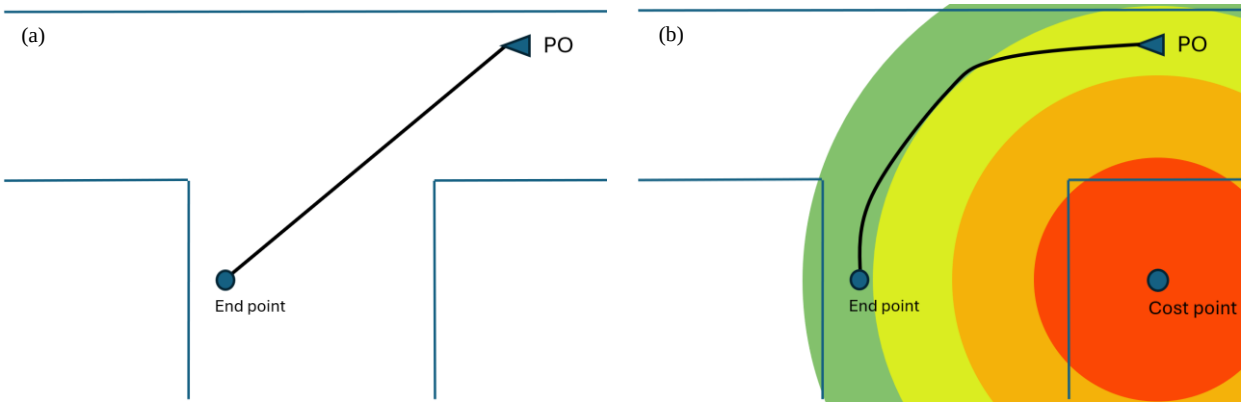


Figure 18. (a) Path without cost point. (b) Path with cost point and a visualization of how the reward increases with distance from the cost point. Green being high reward and red low reward.

The avoidance of the cost point in combination with the desire to find the shortest path to the end point creates a more realistic driving path as the PO will move within its lane, and not turn into an opposing lane too early. The cost point was only used when creating turning paths, but the heuristic also punishes traversing on static cells while it gives a reward for traversing free cells for all paths. The punishment and rewards of what type of cell it traverses, makes the A* prioritize traversing free cells, and if possible, avoid static cells. Not traversing static cells was also a measure made to make paths more realistic as they would not traverse permanent static objects such as traffic islands. However, there were cases where the algorithm must traverse a static cell to find the end point because it was possible for the end point to be placed on a static cell, and if the algorithm cannot traverse static cells, the end point cannot be found. The reason that an end point was allowed to be placed on a static cell, was

because the possibility for cells to be mislabeled in the occupancy grid, due to imperfections in the data.

All parts of the heuristic have individual weights, which is a scaling factor of how much each part should matter when making the path. The baseline of the heuristic, without any weighted components was the Euclidean distance to the end point. This means that for the weight of the punishment for traversing static cells, which was set to 100, is equal to traveling 100 meters away from the end point. This is a large weight that made sure that a static cell was not traversed unless absolutely necessary (when the end point was a static cell). The weight of the reward of traversing free cells was set to ten, which is not as large as the weight for the punishment of traversing static cells but large enough so that the A* algorithm would opt to traverse cells that were free as long as traversing another cell (except static) would not save 10 meters. What the reward of traversing free cells in the heuristic assures, was that a path would not be a "shortcut" through unknown areas outside of the free area (road), but due to the weight not being as high as the static punishment weight, the path was still able to go through patches of moving or unknown cells surrounded by the free area. From these two parts of the heuristic, the final path would be within the free area but they would not affect how the path look like, i.e. if the shape of the path would be realistic. Straight paths would look realistic due to the A* algorithm finding the shortest path, and the shortest path to an end point on a road straight ahead would be a realistic straight path. However, the shortest path to an end point for a turning path would not resemble a realistic path. Therefore, to make the turning paths more realistic, the distance to a cost point was added to the heuristic. How the cost point part of the heuristic was weighted determines how the final shape of the path would look like, making it a more sensitive weight than the other two. Therefore, a parameter search for finding the weight which created the most realistic paths was done, and can be seen in section 3.3. What defined a realistic path were the criteria presented in *Table 3.*

	Realistic turning path criteria and explanations	
	Criteria	Explanation
1	Not turning too early	It was important that a turning path does not make the turn too early and by so, crosses into an opposing lane too early. A law abiding, realistic driver would cross the opposing lane when it is time to make the turn, not drive in the opposing lane beforehand.
2	Turn radius not being too large	If the turn radius of a turning path becomes too large, the path will start of by turning in the opposite direction of the end point. This kind of behavior is not realistic and should therefore be avoided.
3	Promoting curvature	A realistic driver will take turns smoothly, giving the resulting turning path a curvature.

Table 3. Table showing criteria for realistic turning paths, and the explanation behind them.

To model the movement of POs classified as "Pedestrian", the algorithm looks at the position of the PO and its neighbors to find the cells with the least distance to an edge, and continues until it finds a cell that was within 0.5 m of an edge. This cell was then set as the end point of the PO. By using the A* algorithm to find the shortest path to this end point, the resulting path will depict the pedestrian PO crossing straight over the road (see *Figure 1* for an example).

2.2.3 Risk assessment

The output of the ORA algorithm is a probabilistic simplified reachability expressed in a grid. What position a PO can reach within a given time horizon, in other words what space a PO could reach if it existed. This was done by marking all the cells along the calculated path of the PO, stopping at the distance that the PO would reach given the speed of the road and the time horizon that the ORA algorithm was done with. By adjusting the POs paths through the introduction of time dependency and speed, the paths can be considered trajectories. In this work, POs do not have their own width and length, but instead were only seen as the size of one cell. Therefore, a PO's reach was expressed in the width and length of one cell, this cell was meant to symbolize the closest corner of a PO to the AV.

A PO could potentially be speeding or driving slower because it was approaching an intersection with limited visibility. Therefore, the reachability was calculated with the probability function:

$$F(x) = P(X > x) \tag{7}$$

Based on a normal distribution, F is the probability that a random variable X takes a higher value than a given variable x. The function F was created using a cumulative distribution function (CDF) [39], [40]. If X is a discrete random variable and the probability that it obtains the value x is $P(x)$, the CDF can be expressed as:

$$CDF(x) = P(X \leq x) = \sum_{t \leq x} P(t) \tag{8}$$

The function F is the opposite of the CDF, and it can be calculated through 1 minus the CDF:

$$F(x) = P(X > x) = 1 - P(X \leq x) = 1 - \sum_{t \leq x} P(t) \tag{9}$$

Algorithm 5 shows the creation of the probability function F. The first step was to sample a normal distribution N times. The normal distribution in this project was based on data found by [41] that measured the distribution of the speed of drivers on multiple roads with the same speed limit. By rearranging the data into a histogram, and dividing the count for each bin by N, a probability distribution function (PDF) was created, representing the probability for each bin. The CDF was then calculated using a cumulative sum on the PDF, and finally F was created by subtracting all values in the CDF from 1.

ALGORITHM 5: PROBABILITY FUNCTION

Input: *d:* distance

Output: *F(d):* set of probabilities, *bins:* set of distances

function F(d):
    $S \leftarrow Norm(d, \sigma, N)$
    $count,\ bins \leftarrow histogram(S)$
    $PDF \leftarrow \frac{count}{Sum(count)}$
    $CDF \leftarrow cumulative\ sum(PDF)$
    $F \leftarrow 1 - CDF$
return $F, bins$

The resolution, meaning how exact a value was in F, depends on the number of samples N and the number of bins. *Figure 19.* shows a graph created through the probability function F.

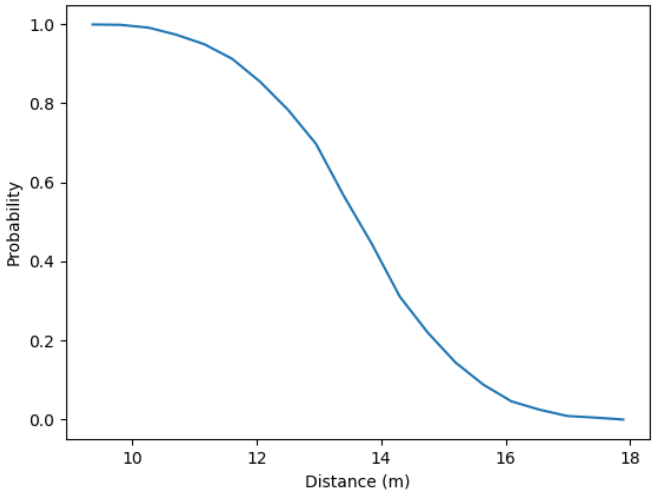


Figure 19. a graph created with the distance 13.89 (the distance an object travels in one second at 50 km/h) as input to the probability function F

Although the graph does not begin at zero distance, it was assumed that the probability was 100 % before the first point because of the possibility of a PO existing further within the occlusion than at the emergence point. The maximum length that a PO could travel from an occluded area was achieved by a PO existing exactly at the emergence point at the beginning of the time horizon, but where a PO exists within the occlusion was impossible to know. Therefore, the possibility of a PO's starting position being further from the emergence point within the occlusion was accounted for by assuming that if the worst-case PO (the one starting on the emergence point) was able to reach a cell with 100% certainty, then for all previous cells in the path there was another PO which could reach it with 100% certainty. This concept is visualized in *Figure 20.*

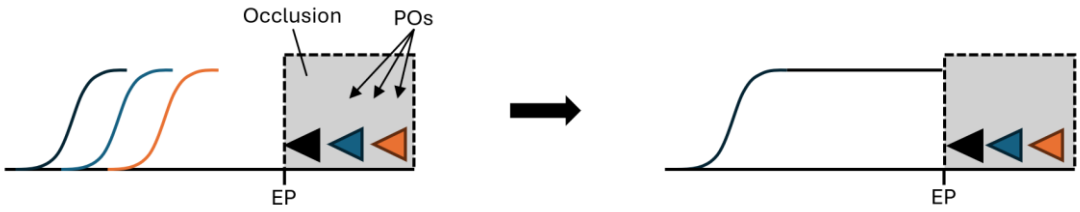


Figure 20. A visualization of how several POs can exist in occlusion and how the POs behind the worst-case PO (black) extends the graph of the worst-case PO.

2.2.4 Differences in implementation for the synthetic data

The synthetic implementation of the algorithm does not make use of a map, instead a graph grid and a speed grid were used. Consequently, some parts of the main algorithm were implemented differently.

Due to not using a map, finding the closest intersection point differs a bit. All intersection points were marked in the graph grid. Therefore, the algorithm checks for any intersection points within the current graph frame, and then selects the closest point, if it exists.

An emergence point for a PO classified as vehicle was placed if the interval crosses an edge in the graph grid. Similar to the real data case, the placement differs depending on which side of the intersection that the EP was located at. In the right-side case, the point was simply placed on top of the edge. However, in the left-side case, an adjustment was made, shifting the placement 7 cells or 3.5 m towards the beginning of the interval, if it was possible, to model the corner of the PO which was closest to the AV. The placement of the EPs for vehicles are visualized in *Figure 21*.

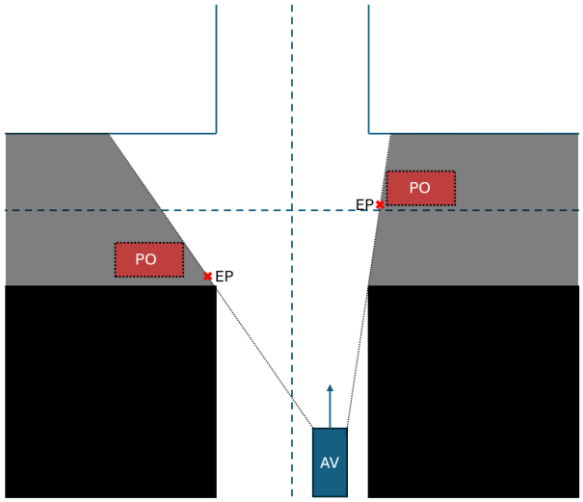


Figure 21. Visualization of how EPs were placed in relation to POs, depending on which side that the POs were on in the intersection.

In the synthetic scenarios, the environment was created in a simple manner: the roads were going straight, and intersecting roads were orthogonal to the current road. Therefore, the straight trajectories of POs, both vehicle and pedestrian, were assumed to be orthogonal to the AV's heading.

For POs classified as vehicles, both straight trajectories and the left turn trajectories were considered. In the synthetic implementation, instead of using a function that returns all possible end points, two separate functions were created. The first function returns the end point for the straight trajectory and second function returns the end point for the left turn trajectory. Before calculating a straight trajectory, a check was made for if there was an outgoing edge in the direction that the PO was going. This was done by looking at the cell next to the intersection point in the forementioned direction. If there was an outgoing edge, the end point was calculated through multiplying the direction with PO's direction with a calculation of the PO's distance travelled within a certain time horizon:

$$endpoint = \vec{\boldsymbol{D}} \cdot \frac{v \cdot T}{cell\_size} \tag{10}$$

Where $\vec{\boldsymbol{D}}$ is the direction**,** v is speed on the road, and T is the time horizon.

The end points for left turns were calculated under the assumptions that the intersecting roads would be orthogonal to each other, with a lane width of 3.5 m. Depending on the side of the road the PO was approaching the intersection from, the trajectory was calculated to 3.5 meters to the left of the intersecting point.

Left turn trajectories were not calculated if the PO was further than 3.5 m plus the distance it would traverse within the time horizon to where the PO's road and the AV's road intersect. Because the left turn trajectory would not get to the point that it needs to turn to get to the end point, i.e. it would not extend into the intersection, it was not relevant to calculate it.

3 Results and analysis

The results and analysis section presents the results from the real and synthetic data, analyzing it through a qualitative point of view. A scenario depicted by the input occupancy grid is shown along with the expected output where potential paths from POs were marked in blue, and the actual output. In the actual output, the trajectories were given a color representing the probability of reach, equal to that shown in the corresponding color bar. For the real data, the map is shown as well, both the map before POs have been identified and after. All outputs were simulated with a one second time horizon, and a cost point weight of 0.8. The reason for the choice of 0.8 is described in section 3.3.

3.1 Real data

The results from the real data are presented alongside the input occupancy grid, the expected result, and the map.

The map, which is made up of nodes and edges, show regular nodes marked as white dots and intersection nodes marked as blue dots. Intersection nodes were nodes connected to at least three edges. The edges were represented by white lines connecting the nodes. The AV's position was marked with a red circle and the initial position of the POs from the output were marked with orange circles. Only POs of the type "vehicle" were included in the map, meaning POs of the "pedestrian" type was not included.

3.1.1 Scenario R1

In the first scenario (see *Figure 22*), the AV was standing still in a four way intersection, where the left and right connecting roads were occluded and the outgoing lane in front of the AV was occluded.

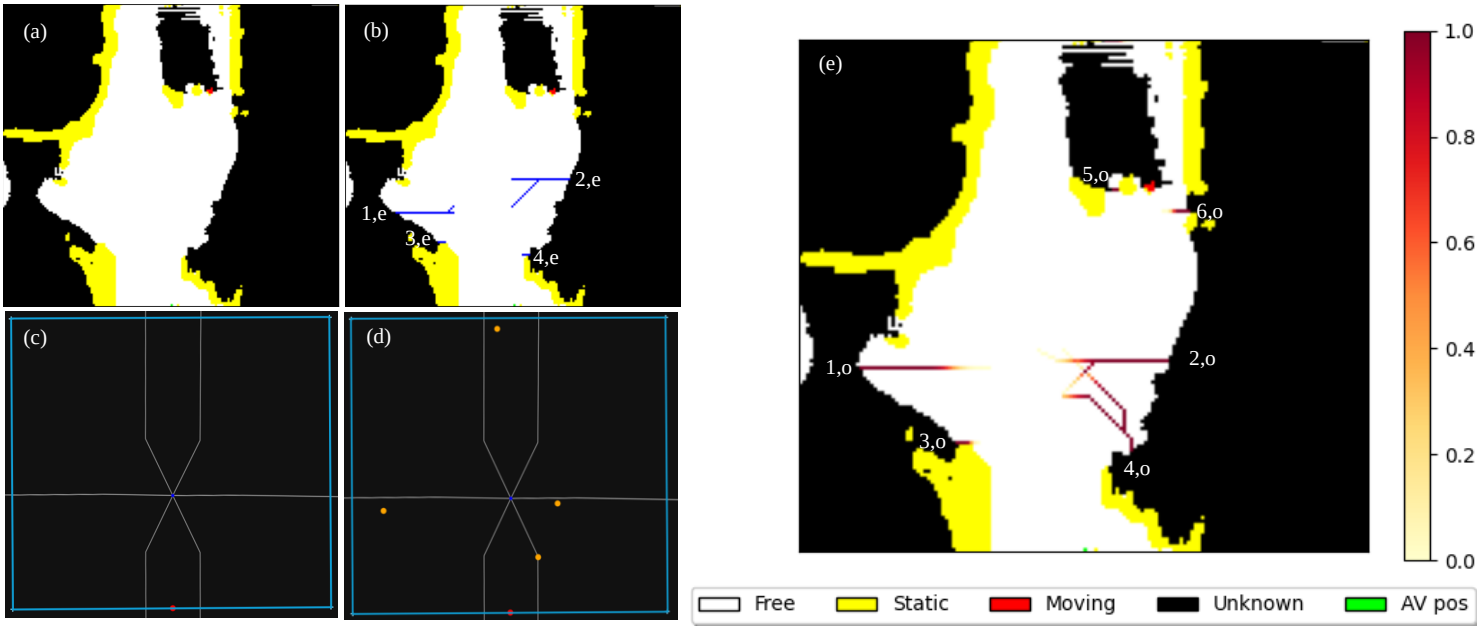


Figure 22. An intersection without moving objects. (a): input, (b): expected output, (c): map showing only the AV, (d) map showing POs' starting position and the AV, and (e): output of the ORA algorithm on top of the input occupancy grid. The color of the POs' trajectories indicates the risk, red being most probable and light yellow being the least probable.

In scenario 1, the expected output was that there should be vehicle POs to the left and the right (1,e and 2,e), both coming from the intersecting road. Both POs should have a straight and a left turn trajectory. The output includes both POs, and they both have a straight and a turning trajectory. However, the turning trajectory of the PO on the left side (1,o) was not visible because the time horizon was too short for it having started to turn, the turning starts further into the intersection than what the time horizon permits. The output has found a third vehicle PO (4,o), that was on the same road as the AV. This PO has two left turn trajectories. This PO was not expected because the algorithm does not put POs on the same edge as the AV. The reason it has been placed was that the AV's UTM coordinates were closer to the left edge compared to the right edge (where the PO has been placed). This seems like a fault from the algorithm at first, but the AV was in fact standing in a turning lane which was closer to the center of the side of the road going in the opposite direction compared to the center of its own side of the road, making the AV closer to the left edge. With the AV's closest edge being the left, the algorithm thinks that it stands on the left edge, meaning a PO could be placed on the right edge. The cause of this issue is the disparity between how the map looked like in this scenario, and the assumption that the closest edge to the AV would always be the road that is drives on.

Even though the PO on the same side of the road was not expected to be there, given that it has been placed, it should have a straight and a turning trajectory. However, the

PO was missing a straight trajectory, due to the angle between the vector from the PO to the intersection point and from the PO to the end point was too large and exceeded the max angle which makes the algorithm identify that trajectory as a right turn, this issue is further discussed in section 4.2. The reason for two left turns was that the algorithm accounts for all possible left turn trajectories of the PO, including U-turns. Three POs classified as pedestrians (3,o, 5,o and 6,o) were found by the algorithm, 3,o crossing the road close to the AV from the left and 5,o and 6,o crossing the road on the road across the intersections from the AV. The algorithm does not show a possible pedestrian crossing the road close to the AV from the right (4,e in the expected output), due to how the pedestrians were modeled. The pedestrians were assumed to want to walk towards the nearest “middle” of a road, i.e. edge, and was therefore walking into the occlusion due to how the map looks.

3.1.2 Scenario R2

In the second scenario (see *Figure 23.*) the AV was driving towards a four way intersection, with all connecting roads being heavily occluded.

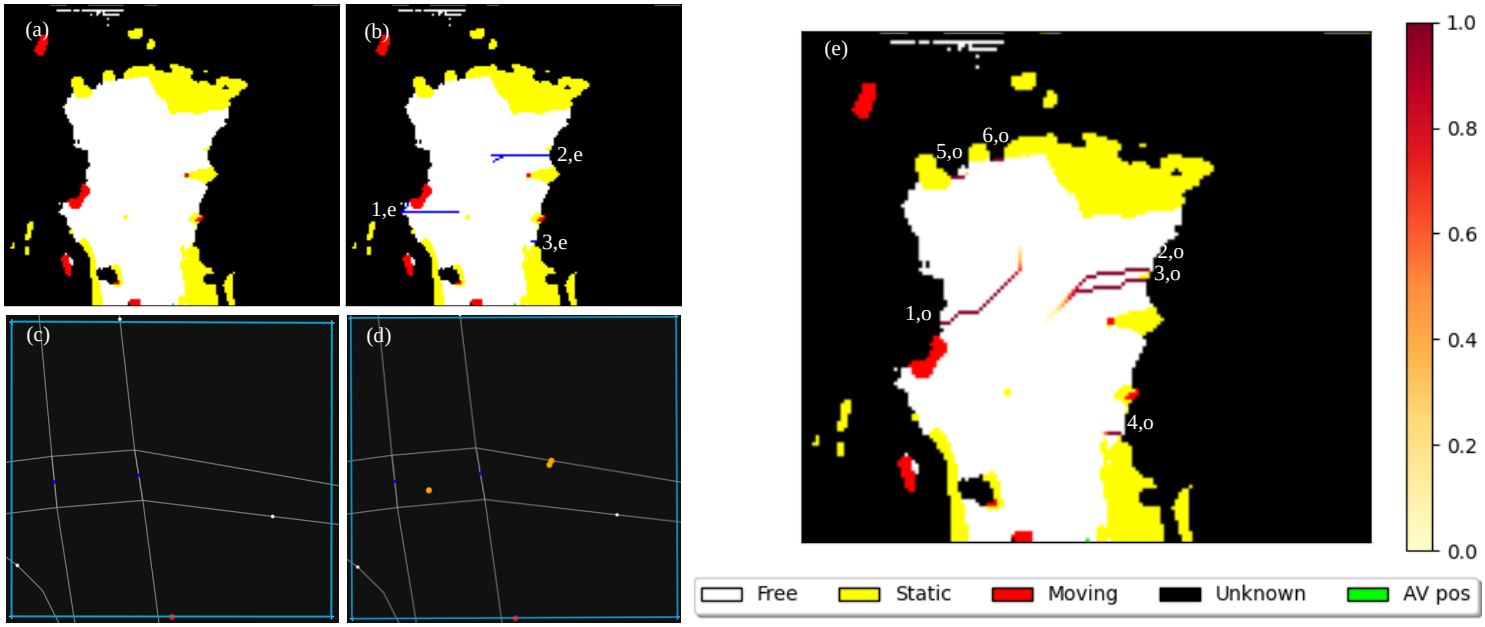


Figure 23. An intersection including moving objects. (a): input, (b): expected output, (c): map showing only the AV, (d) map showing POs' starting position and the AV, and (e): output of the ORA algorithm on top of the input occupancy grid. The color of the POs' trajectories indicates the risk, red being most probable and light yellow being the least probable.

Scenario 2 show worse results, because of issues in the map data. The map data was expected to look like scenario R1 and R3 where all outgoing edges were connected to an intersection node in a way which closer resemblance the geographical geometry of the road. In scenario R2, we can see in *Figure 23. (c)* that the two roads intersecting horizontally were not connected with the intersection nodes horizontally. This leads to the three POs classified as vehicles (1,o, 2,o and 3,o), were all missing straight trajectories due to the trajectories being classified as right turns. The connecting road to the right gets two POs (2,o and 3,o) due to there being a small spot of static separating what would have been one emergence interval into two emergence intervals, resulting in two emergence points. The vehicle PO to the left (1,o) emerges from the "top" interval whereas the manually marked PO (1,e) was expected to emerge from the bottom interval on the left and drive through the occlusion with its left turn. The disparity between the expected and actual was due to the bottom interval being closer to the edge representing the road on the opposite side of the AV and not the edge representing the crossing road and therefore was assumed to be driving away from the intersection. A PO emerging from the top interval was not manually marked because it was assumed this interval was closer to the top edge of the two intersecting edges and therefore representing an edge with a direction out of the free area. Scenario 2 also shows a PO classified as a pedestrian crossing from the right (4,o) and two POs classified as pedestrians coming from occluded areas on the road straight ahead (5,o and 6,o) trying to cross towards the side on the right side of the AV. The

pedestrian PO coming from the right close to the AV (4,o and corresponding 3,e) was the only one we manually recognized as a probable PO as it was the most relevant due to it emerging from what would be a sidewalk in real life, whereas the two others were in the middle of a road. Recognizing the POs on the road (5,o and 6,o) was still important as it was possible that it was pedestrians who were currently crossing the road but these POs would most likely disappear as the AV would approach further since the sensors would see more across the road.

3.1.3 Scenario R3

In scenario 3 (see *Figure 24.*), the AV was once again driving towards a four way intersection. However, this time with perfect map data, and no occlusion in the road straight across the intersection from the AV.

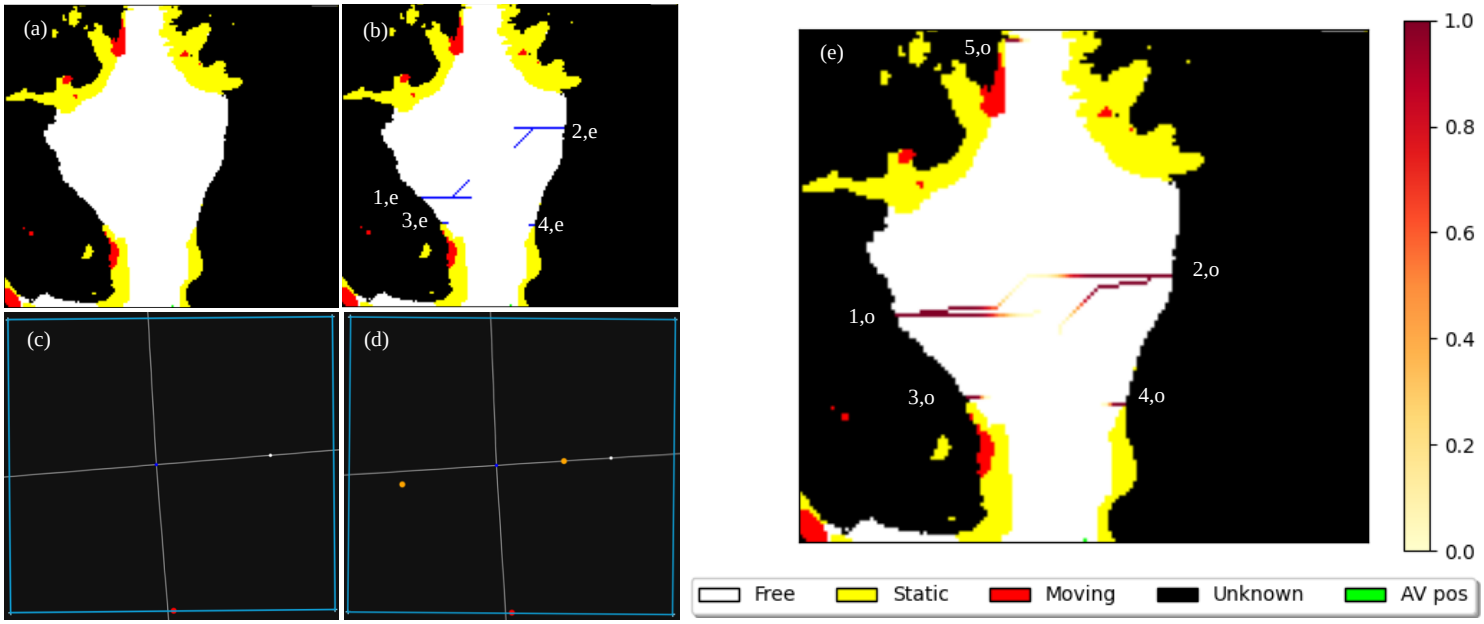


Figure 24. An intersection. (a): input, (b): expected output, (c): map showing only the AV, (d) map showing POs' starting position and the AV, and (e): output of the ORA algorithm on top of the input occupancy grid. The color of the POs' trajectories indicates the risk, red being most probable and light yellow being the least probable.

Because of the simple map representation of the scenario, the output looks as expected. Meaning, that all expected POs and their trajectories were found. Both potential vehicle POs (1,o and 2,o) have a left turn and a straight trajectory, and potential pedestrian POs (3,o and 4,o) can be seen on both side of the road from the AV as well as one (5,o) crossing from a small occluded patch on the road straight across from the AV.

3.1.4 Scenario R4

In scenario 4 (see *Figure 25.*), the AV was driving on a straight road with several parked vehicles on both sides of the road causing occlusion behind them.

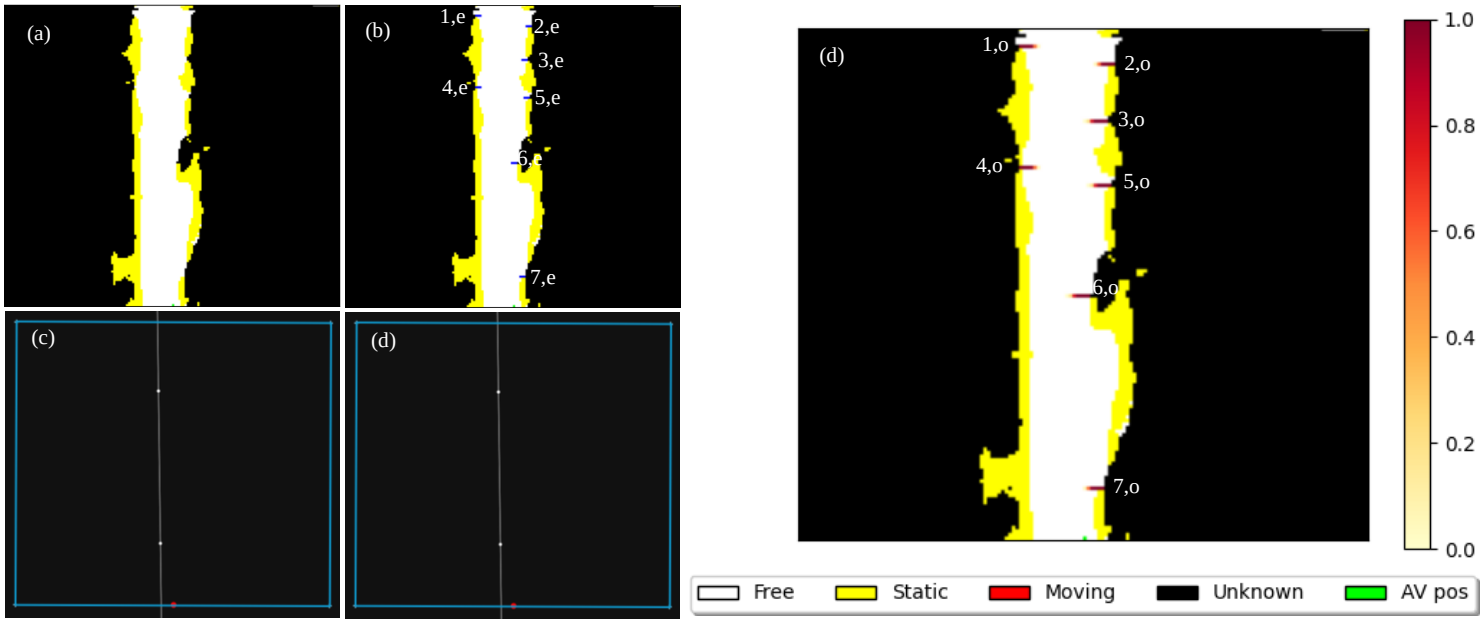


Figure 25. A straight road with parked vehicles on both sides of the road (a): input, (b): expected output, (c): map showing only the AV, (d) map showing POs' starting position and the AV, and (e): output of the ORA algorithm on top of the input occupancy grid. The color of the POs' trajectories indicates the risk, red being most probable and light yellow being the least probable.

As can be seen in the input, there were several places where pedestrians could appear, and in the output, we see that the algorithm catch them successfully. This shows that the ORA algorithm handles parked vehicles successfully.

3.2 Synthetic data

The result from the synthetic data is presented by showing the input occupancy grid, the expected output and the actual output. The expected output in the synthetic data comes from the Euro NCAP scenarios and was further explained in section 2.1.3.

3.2.1 Scenario S1

The first scenario (see *Figure 26.*) of the synthetic data was created with inspiration from the Euro NCAP scenarios CPNCO 1.3.1 and CPNCO 1.3.2, depicting the AV driving on a straight road with two vehicles parked next to the road on the right hand side causing occlusion behind them.

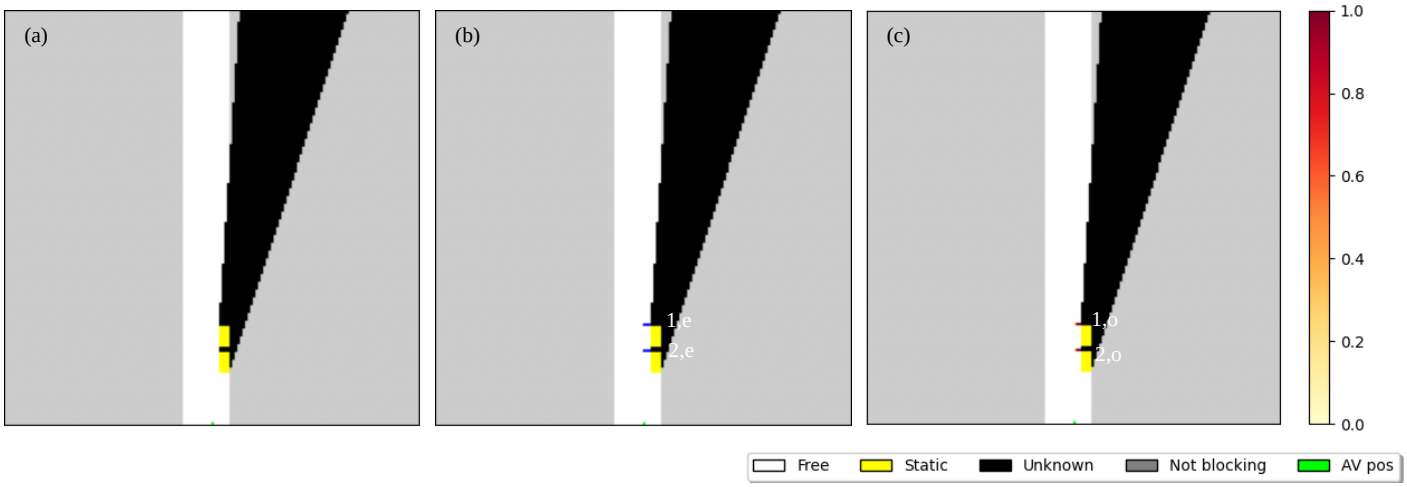


Figure 26. Euro NCAP scenarios "CPNCO 1.3.1" and "CPNCO 1.3.2". (a): input, (b): expected output, (c): output of the ORA algorithm on top of the input occupancy grid. The color of the POs' trajectories indicates the risk, red being most probable and light yellow being the least probable.

The expected output for the first scenario of the synthetic data was that a pedestrian could appear behind each of two vehicles. We can see that the output manages to find the expected POs and predict their path, see *Figure R.* for a zoomed in version of the expected and actual output in *Figure 27. (b),(c).*

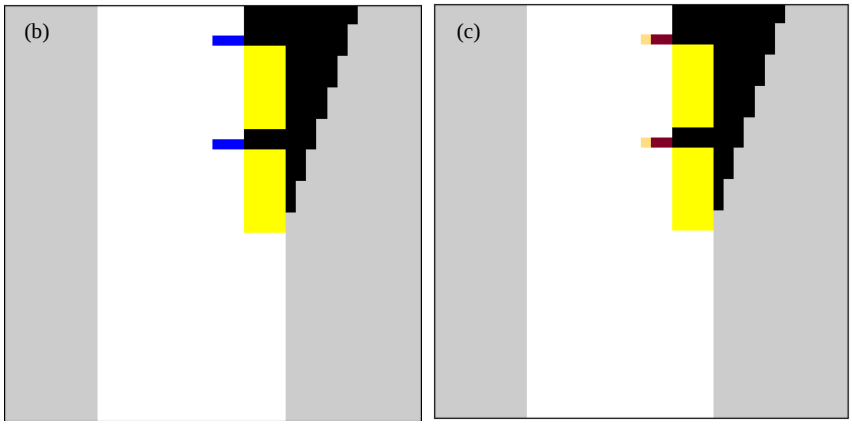


Figure 27. (b) and (c) from Figure 26 zoomed in.

3.2.2 Scenario S2

The second scenario (see *Figure 28.*) of the synthetic data shows the AV driving towards a four-way intersection with three cars on each side of the road causing occlusion, inspired by the Euro NCAP scenario 3.1.3.1 Car-to-Car Crossing.

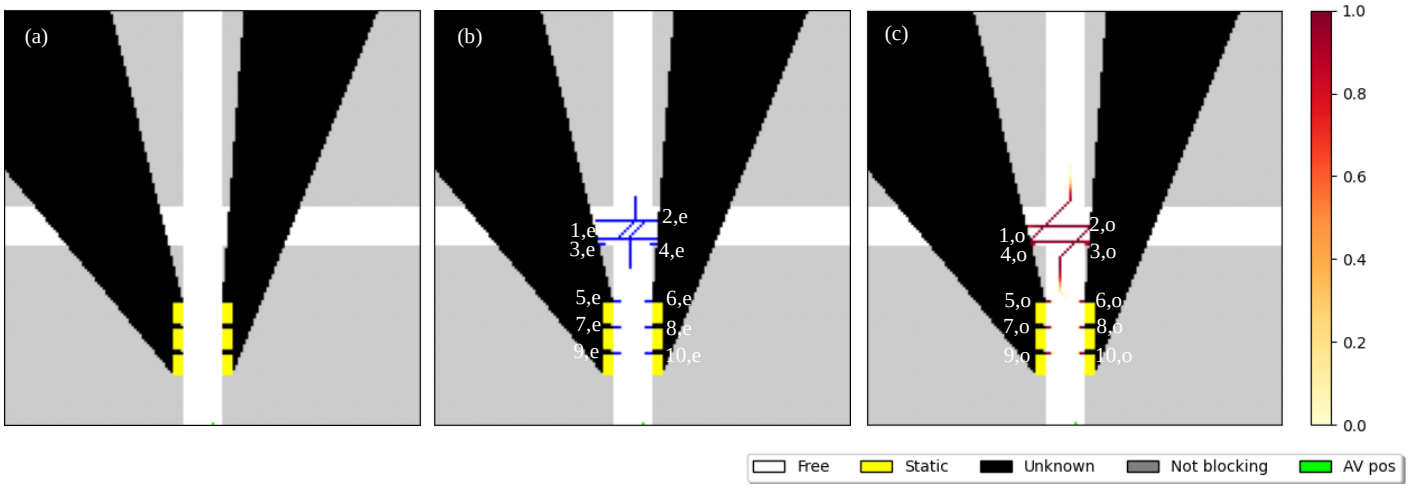


Figure 28. Euro NCAP scenario "3.1.3.1 Car-to-Car Crossing". (a): input, (b): expected output, (c): output of the ORA algorithm on top of the input occupancy grid. The color of the POs' trajectories indicates the risk, red being most probable and light yellow being the least probable.

The cars in the second synthetic scenario were placed mainly to obscure the intersecting roads, creating an occluded intersection. However, in this work, it doubles as a test for the possibility of pedestrians appearing from behind and between vehicles, as shown in *Figure 28. (b).* In the intersection, two vehicles were predicted to appear (1,e and 2,e), both with a straight trajectory and a left turn trajectory. The expected output also includes the possibility of two pedestrians coming from occlusion at the edge of the intersecting roads (3,e and 4,e). Although the emergence points and shape of the trajectories differ slightly in the output from the expected output, the output successfully identifies the two vehicle POs (1,o and 2,o), their trajectories, and all expected Pedestrian POs.

3.3 Finding the best cost point weight

The cost point used to model turning trajectories was an important part of the movement prediction for making the turns depict a realistic turning trajectory. The impact the cost point has on the trajectory was scaled with a weight see section 2.2.2 for a description of a weight's role. What the best weight for this work could be was decided by a parameter search where the result for different values of the weight parameter was compared and assessed based on the criteria listed in *Table 3*.

The first search was done for a weight of 0.0, 0.5, 1,0, and 1,5 and the resulting trajectories with these weights are displayed in *Figure 29,30,31,32* respectively.

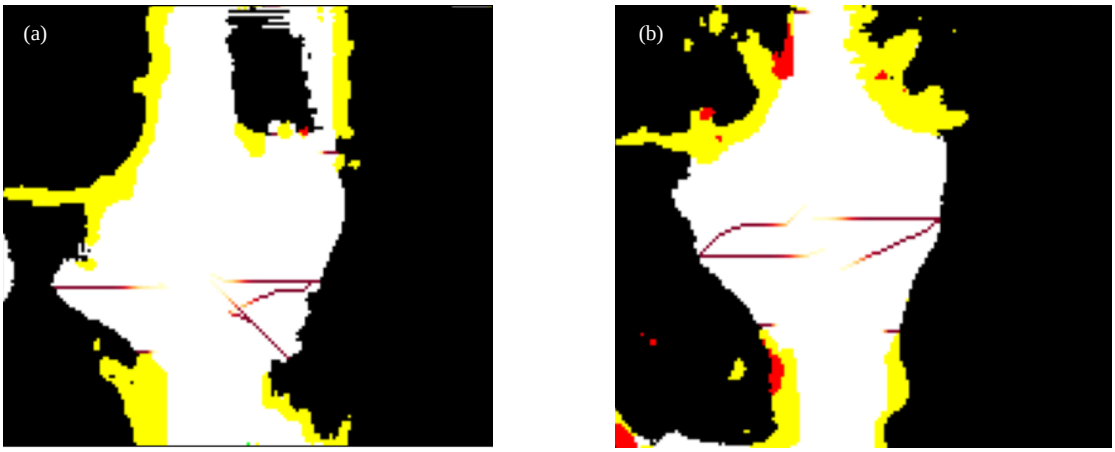


Figure 29. Trajectories with cost point weight = 0.0, (a) Scenario R1 (b) Scenario R3

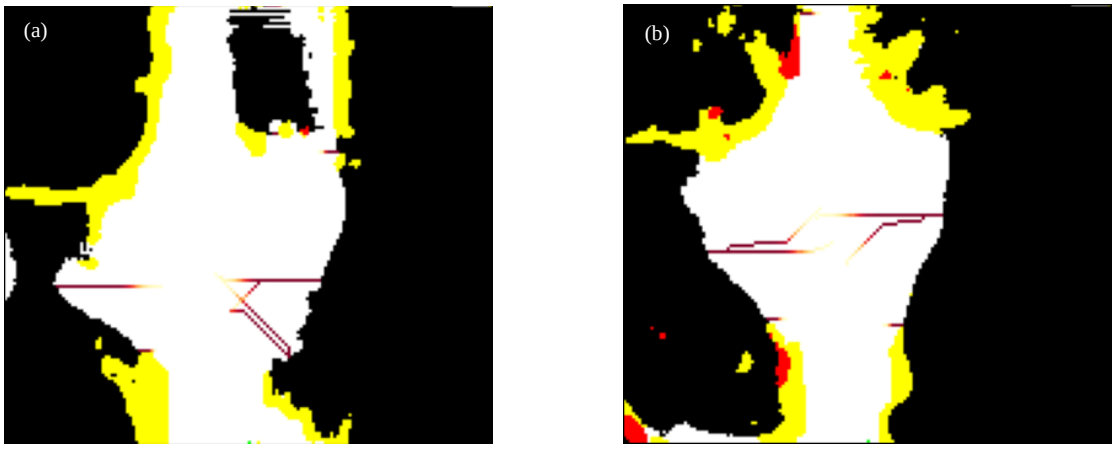


Figure 30. Trajectories with cost point weight = 0.5, (a) Scenario R1 (b) Scenario R3

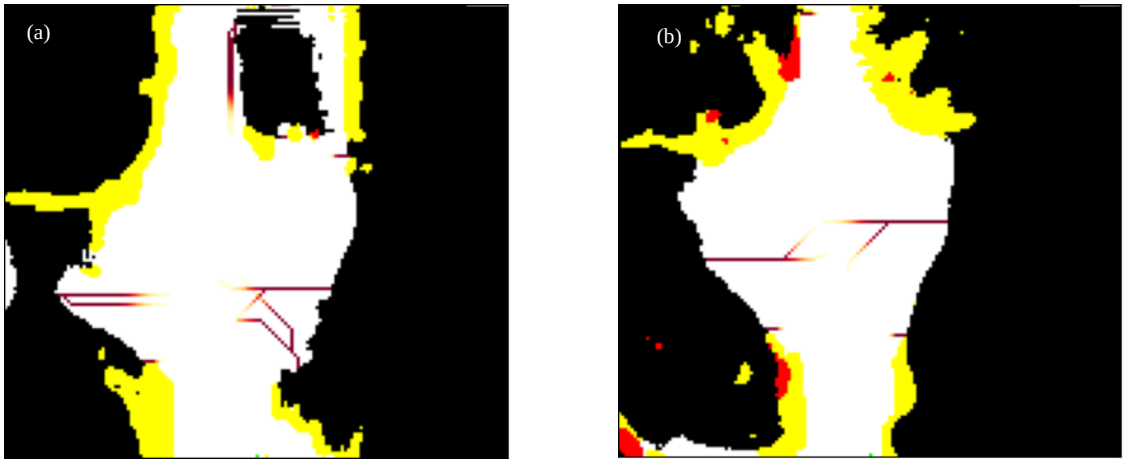


Figure 31. Trajectories with cost point weight = 1.0, (a) Scenario R1 (b) Scenario R3

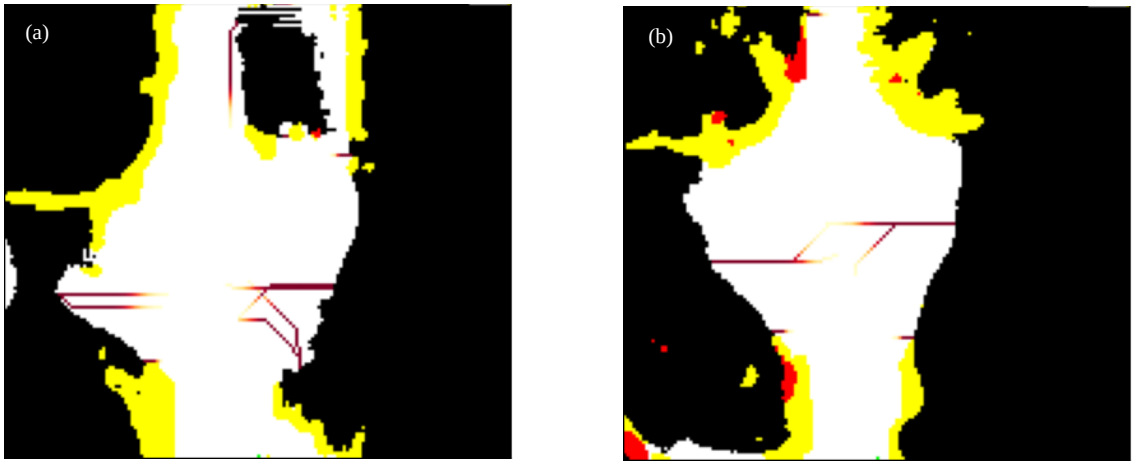


Figure 32. Trajectories with cost point weight = 1.5, (a) Scenario R1 (b) Scenario R3

For a weight of 0.0, i.e. no cost point, we see clearly in *Figure 29. (b)* how the PO to the left makes a left turn too early and drives in the opposing lane to the endpoint violating criteria 1, and resulting in an unrealistic turning trajectory. Using a weight of 0.5, see *Figure 30*, shows more realistic turning trajectories, implying that making use of a cost point was beneficial for realistically modeling turning trajectories. However, as seen in *Figure 30. (a)* the turning trajectories were lacking any form of curvature and makes the turn too early (violating criteria 1 and 2), resulting in an unrealistic early narrow turn. Setting the weight to 1.0, see *Figure 31.* gives the turning trajectories more curvature but introduces a new issue. That is, the turning trajectories, as seen from the leftmost PO in *Figure 31. (a)*, get too much radius from the cost point, resulting in left turn trajectories turning right first, and therefore violates criteria 3. Setting the weight to 1.5 was also tested to see if the issue of some POs turning right first due to the radius being too high persisted, which it did. The first search implies that the best weight should lie within the interval of values between 0.5 to 1.0. Hence, weights of 0.6, 0.7, 0.8, and 0.9 were tested, see *Figure 33,34,35,36*.

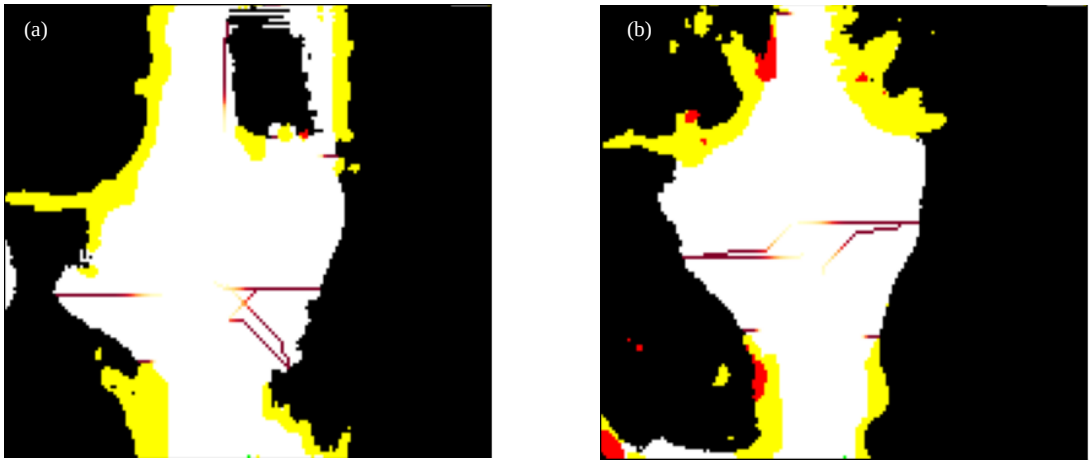


Figure 33. Trajectories with cost point weight = 0.6, (a) Scenario R1 (b) Scenario R3

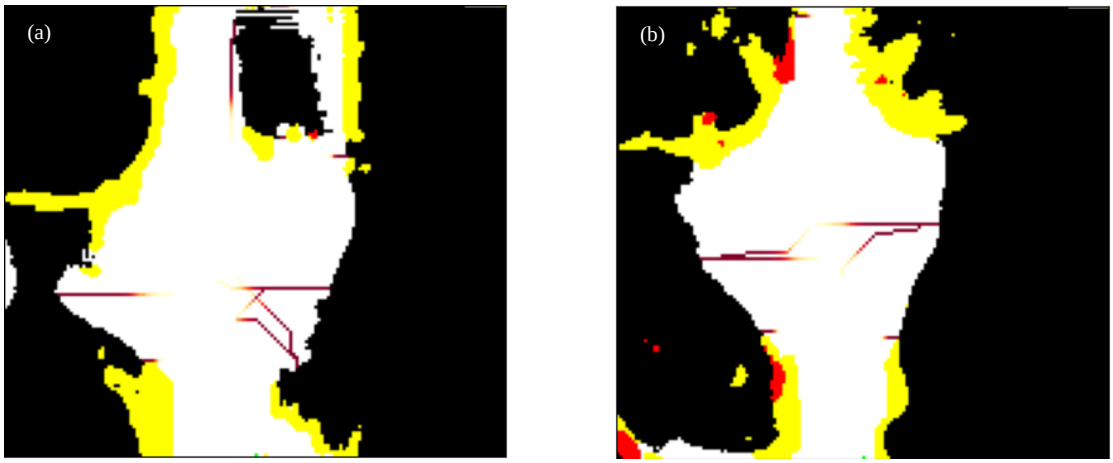


Figure 34. Trajectories with cost point weight = 0.7, (a) Scenario R1 (b) Scenario R3

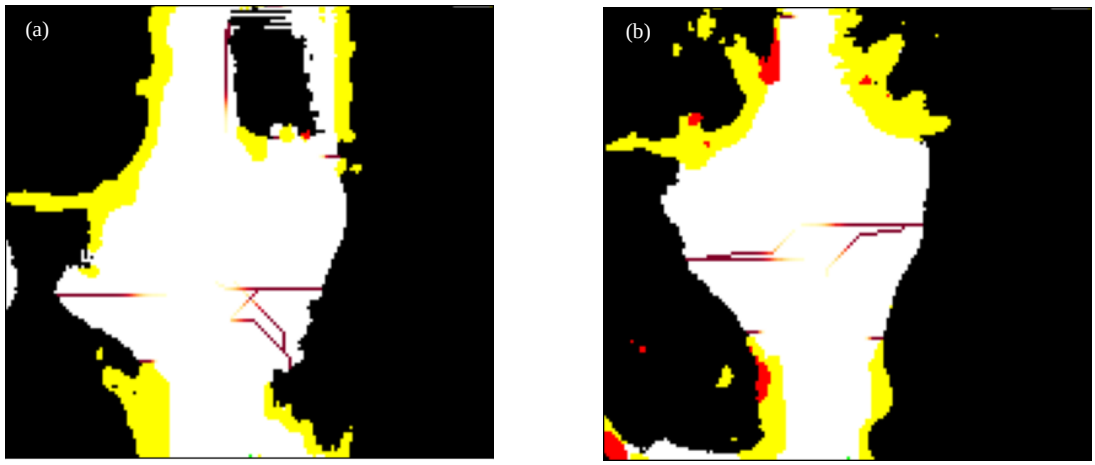


Figure 35. Trajectories with cost point weight = 0.8, (a) Scenario R1 (b) Scenario R3

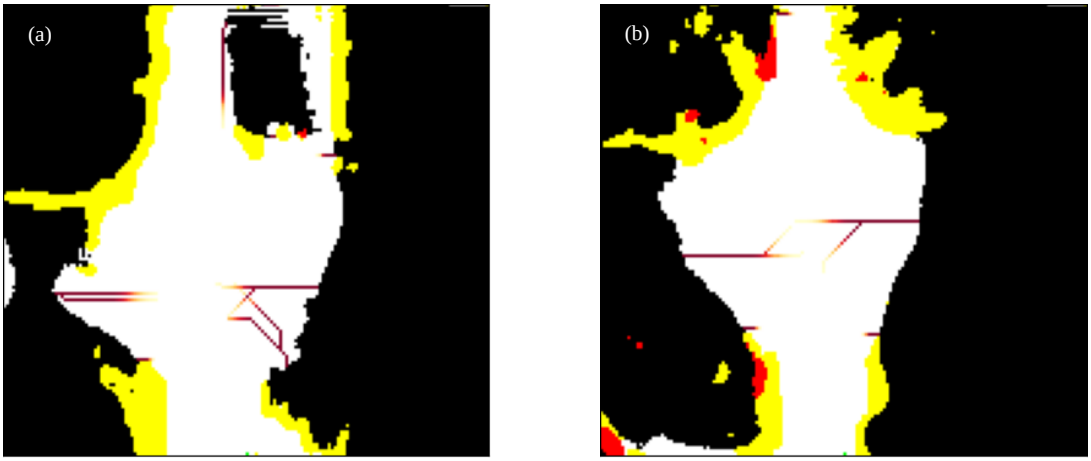


Figure 36. Trajectories with cost point weight = 0.9, (a) Scenario R1 (b) Scenario R3

As seen in *Figure 33 (a)*, using a weight of 0.6 was not enough to achieve the more delayed turns seen in *Figure 34 (a)* and *Figure 35. (a)*. A weight of 0.9, was too high since we get the same result as for a weight ≥ 1, where the turn radius was too large and the left turn trajectory starts with a slight right turn, violating criteria 3. Therefore, a weight of 0.7 or 0.8 proves to be the best. For the tested scenarios, there was no visual difference between a weight of 0.7 and 0.8. However, the result show that one of the main changes to the path that a higher weighted cost point makes, was that the turn was taken later, and this being one of the criteria for a realistic path, 0.8 was chosen as the best weight. This was the weight that was used in all previous analyses.

4 Discussion

This chapter will provide reflections about design choices from the creation of the Occluded Risk Assessment (ORA) algorithm with potential improvements, a discussion about the results, and ethical considerations in the design and usage of the algorithm.

4.1 Solution

The introduction introduces various related works that handles issues related to occlusion in different ways. This section provides reasoning behind the chosen approaches for the ORA algorithm and discusses potential improvements of said approaches.

A foundation for the project is the interpretation of the environment. By establishing a way to depict the environment, the output of the ORA algorithm is restricted to fit the established depiction. The use of an occupancy grid and BEV is two similar methods in regard to how they make use of onboard sensor measurements of the environment to create a 2D model of the environment. An advantage of using an occupancy grid is that, depending on the cell size, the number of computations is substantially reduced compared to using the BEV approach. The reason for this is that the occupancy grid is a less complex state representation compared to the BEV. Although some information such as more detailed shapes of obstacles, is lost when using a less complex model, knowing what cells were occupied, free, and unknown gives a sufficient understanding of the environment for the purpose of identifying and mapping potential risks originating from occluded areas. Therefore, the level of detail in the information of an occupancy grid is sufficient for this work. Another advantage of an occupancy grid is that the grid structure enables the use of efficient graph search algorithms. For a high resolution model such as a semantic segmentation BEV, representing the state as a graph and using a graph searching algorithm might be to complex to run in a typical automotive system.

Making use of POs to model risk from occluded areas is a common method in related work [21][22][23], and by being able to model the POs without being dependent on a planned path for the AV makes the use of POs fitting for this work. For assessing the risks a PO poses upon the AV, this work makes use of simplified reachability (see [42] for a description of traditional reachability). Instead of calculating the whole reachable set in the free space, which is what is typically done, here reachability was used to calculate how far a PO can travel along a given path, given some assumptions and the given time-horizon. The path for POs classified as vehicles was made from analyzing map data and finding the legal possible end points (e.g. turning into the correct lane during a left turn), and POs classified as pedestrians were assumed to cross the road with the shortest possible path to cross the road. These paths were assumed to be the most relevant for the POs, as vehicles were most probable to drive in the legal lane and follow the road directions, and pedestrians were most probable to cross a road straight across.

A possible issue with using map data was that the roads in the map data could differ from reality when there, for example, is construction on a road. However, major changes to the road networks due to construction are rare, and if the map data is updated regularly then unexpected changes to the road network will not be a common

issue. If the maps would not have been updated, a major change to the road network would cause the output of the ORA algorithm to be inaccurate. The output could predict risks that currently does not exist or fail to predict risks from temporary road solutions. However, the ORA algorithm is not a decision making algorithm, and it is the job of a path planner to make descisions. A solution that could solve this problem for the path planner is to equip the AV with an algorithm that is able to spot construction sites on the road [43] and with that, be able to adapt to scenarios with construction by knowing that the map could be inaccurate.

When predicting the movement of the POs, right turns for vehicles were dismissed. Dismissing right turns was a simplification meant to reduce conservative behavior and limit the output to the most relevant risks. The most relevant risks refer to the risks most relevant to account for when planning a path for an AV. Because a PO making a left turn or who moves straight would overlap with the AV's path first, the risk these paths pose on the AV is deemed to be more relevant than the right turn. There are still cases where a right turn from a PO is relevant and it is when the PO is making a right turn into the road which the AV is driving on. For example, in a four-way intersection with POs coming from intersecting roads and making right turns, only one of them overlap with the AVs path and that is the PO who is making a right turn into the same road that the AV is traveling to. However, making right turns for all POs would require more computations while only one of the right turns would be relevant. With the current implementation, possible risks could go unnoticed and cause safety hazards. To counteract this, a future implementation that had access to the planned path of the AV could use this planned path to know which road the AV is traveling towards and calculate the right turns going into that road.

Making an ADS safe enough but not too conservative is an act of balancing, and a way to make the output of this work less conservative would be to take traffic rules such as exit rules and traffic lights into account when predicting trajectories. Since it would be illegal for a vehicle to drive into the AVs path from a connecting road in scenarios with traffic lights, predicting a trajectory which poses risk of collision with the AV would be conservative. More scenarios where other vehicles should not cross path with the AV are for example when following priority rules. For scenarios with traffic lights, where technology is telling a driver when it is okay to drive it could be a good idea for reducing conservative behavior to detect traffic lights [44] and adjust the output to lower the risk for POs classified as vehicles. It is also possible to incorporate more information, such as traffic lights, into the map. The process of including more information in a map is demonstrated by [45], who incorporates more detailed data into an SD map using open access data such as OpenStreetMap. The risk for pedestrians crossing should not be changed as pedestrians are likely to cross illegally [46] and lowering the predicted risk of a pedestrian crossing could therefore be a safety hazard. For scenarios with priority rules that other drivers need to interpret themselves could pose a safety hazard if treated as a less risky scenario as it is possible for other drivers to misinterpret the situation and drive when they should not. The issue of ethicality also presents itself when assuming the risk of collision differently in different situations, this matter is further discussed in Section 4.5.

To measure what risk the PO's reachability poses, this work looks at the probability of a PO being able to reach a cell in the free area, given that a PO exists. Since this work did not include path planning for an AV, it did not have access to a reference path of the AV, which was why the risk was defined by how a PO could affect the

environment in which an AV wishes to traverse. When planning the path, the output of this work can then be used to understand how occluded areas of the environment can pose risk upon the AV.

The risk assessment of the POs trajectories currently does not take width of the POs into account. All POs were seen as being the size of one cell and therefore only occupies one cell while moving. For pedestrians, one cell is enough but for vehicles, one cell is too small. This work chooses the occupied cell by a vehicle to be its closest corner to the AV, and what could be done is to widen the reach of this cell with the size of a normal car given we know where its front corner is. However, vehicles come in many varying sizes, especially considering trucks and motorcycles so the most relevant point of reach is still the closest corner because the closest corner will be the first to reach a cell. What could be done to demonstrate the possibility to reach different parts of the lane laterally is to give the whole lane the probability of reach but scaling it with a uniform distribution representing lane placement. This approach is used by [26].

The placement of EPs were done under the assumptions of the worst case scenarios for POs emerging from an emergence interval. Meaning, if a PO emerges from an emergence interval and approach the intersection from the left side. Then the EP will be placed further down from the middle of the road. This placement is done under the assumption that a vehicle's closet point (corner) would be closer to the opposite side of the lane compared to the center of the road. One issue with this assumption is that it works well for a car or a truck but when it comes to smaller vehicles like motorcycles, placing the EP in the middle of the lane might be more realistic. Another issue is that for left turns, a vehicle would be more probable to be closer to the left side of the lane. For POs approaching from the right, the worst case placement is by the midline of the road, that is, to the left in the lane of the road approaching from the right. For POs approaching from the left, the placement of the EP is on the right side of the approaching lane (see *Figure 21*). Therefore, the left turn trajectories of POs approaching from the left are resembling wider turns than what is probable. Similar to what was discussed in the previous paragraph, another approach to placement of EPs would be to consider the whole lane instead of just the closest corner of the worst case PO. The EPs would then be placed in the middle of their respective lanes.

4.2 The ORA algorithm

In this section, various design choices and potential improvements for the creation of the ORA algorithm are discussed.

The first step in the algorithm is interpreting occlusion. What is meant by interpreting occlusion is that with the knowledge of where occlusion occurs through the unknown class, how can it be used to predict where POs could appear. The problem is narrowed down by assuming that threats come from occlusion and enters the free area connected to the AV. Under this assumption, the relevant occlusion for placing a PO was at the border of the area created by free cells in connection to the AV. Therefore, the first part of the algorithm was the outlining of the free area and the extraction of emergence intervals from the outline. These intervals were made up of unknown cells bordering the free area.

With the search limited to the emergence intervals (defined as a set of connected unknown cells bordering the free area, and it is from emergence intervals that POs can appear from occlusion), the next step was to look for and classify EPs with the help of map data. In this work, only POs of the type vehicles and pedestrians were considered. While it was described as looking for and then classifying the EPs, the algorithm looks for EPs of the two different types separately. The algorithm looks for EPs where POs of the vehicle type could be placed, and then EPs for where POs of the pedestrian type could be placed. The order was done this way because there were a set of conditions created from a set of assumptions made of where different types of POs were probable to exist. In the case of vehicles, the first assumption made was that an occluded vehicle would be located on a road. Therefore, only cells within the emergence interval that had a certain distance to an edge in the map data were considered for the placement of EPs. With the assumption of right side traffic, the distance to an edge from an EP differs depending on what side the emergence interval is located at relative to the intersection, from the perspective of the AV. If the emergence interval located on a road that was approaching from the left, if possible, the vehicle would be placed further to the right in its lane compared to if it was approaching from the right. This was done so that the output trajectory represented a PO's closest corner to the AV. See *Figure 21*, in section 2.2.4.

The pedestrians had other conditions when choosing an EP, such as they would not be placed next to a moving object, because it is unlikely that an object would be moving if there is a pedestrian in front of it. This logic was a bit flawed, because the cells in the occupancy grid classified as moving (representing a visible moving object) could be braking to not crash into a pedestrian crossing the road. A common scenario where pedestrians are crossing the road, and could be occluded by a moving object is at crosswalks, and a future solution to this type of scenario should be implemented. However, a solution requires knowledge of the moving object braking, which was not available in this work. Currently a pedestrian is placed in the beginning of an interval, if the closest cell only has neighboring static occupied cell and no moving occupied cells.

When a PO has been placed at an EP, the next step was to model the PO's movement using the A* algorithm. The A* algorithm is an efficient graph search algorithm that can be applied to grids, and the reason it fits well in this work is because it makes use of a heuristic function. The heuristic can be specified to punish or reward cells traversed by the A* graph search algorithm, allowing for punishments and rewards tailored to the graph which is searched. In this work, the heuristic was made up of the reward a traversed cell gets from avoiding the cost point, the reward for traversing cells classified as free, and the punishment for traversing cells classified as static (see section 2.2.2. for further explanation). How much the heuristic rewards or punishes a condition depends on its corresponding weight. By manipulating the different weights, it is possible to influence how much each part of the heuristic is affecting the total heuristic. In this work a wide parameter search was not conducted for the weight of the reward for traversing free cells and the punishment for traversing static cell. However, the chosen weights were chosen to be large enough to make sure the A* algorithm prioritize traversing free cells. The weight for the cost point on the other hand, was further tested to find a weight yielding realistic trajectories. What defines a realistic trajectory is described in section 2.2.2. It should be noted that the PO's trajectories do not need to be perfect, they are ultimately only to help the path planner

to take additional care in the planning. This will typically not require exact positioning of other road users.

For the A* star algorithm to work, an end point is needed. In this work the end points were found using the outgoing edges of the common intersection point of the PO and the AV. The algorithm for finding end points requires the PO and the AV to have a common closest intersection point. To ensure that the PO and the AV share the same closest intersection point, the map used was simplified using a built-in function within the OSMnx library, called consolidate intersections, which merged multiple nodes that were close together into one node. Some roads were expressed with just one edge in the map and those were labeled as two-way roads, meaning they include at least one lane in each driving direction. Other roads were represented with two parallel one-way edges. In intersections that includes roads consisting of parallel edges, there were multiple intersection nodes, which complicates the search for end points. Therefore, by merging the multiple nodes, it is enough to explore the outgoing edges of the one intersection point.

The method of using a merged intersection points turned out to have a varied degree of success. The issue with this method was that the shape of the edges, close to the consolidated nodes was not preserved, see *Figure 22.* (c) Sometimes the function failed to consolidate all the nodes in the intersection, in *Figure 23.* (c) the original four nodes was turned into two. The consequence of manipulating the map with the consolidate function was that some end points would not be found. End points not being found were mainly for straight trajectories that were seen as right turns and discarded because the new shape of the edges creates a greater angle between the intersection point and the end point, making it similar to a right turn, see scenario R1 in section 3.1.1. Using the “consolidate_intersections” function made it simple to find end points but made the algorithm less robust when the map was changed in a way that changes the relationship between the nodes and edges, causing straight trajectories to be discarded. The issue of consolidating intersections affecting the output was left for future work, mainly because the use of the OSMnx library is suboptimal in the first place. If anyone wanted to replicate this work for use in their own algorithm, it is recommended to find a map better suited for the task. In this work, using the public library OSMnx with public data from OpenStreetMap was convenient, because it does not require any special license but since it is community made, the quality varies, and is therefore not suitable for a safety system. Another approach to consolidating intersections would be to clean up the graph with the OSMnx built-in function called “simplify_graph”, this function removes all nodes that are not intersections or dead-ends. Then a clustering approach could be used to put a group of intersection nodes together if they are connected and within a certain distance to each other. Then similar to the case with just one intersection node, the end points would be located on the outgoing edges among all edges connected to the nodes within the cluster, this is visualized in *Figure 37.*

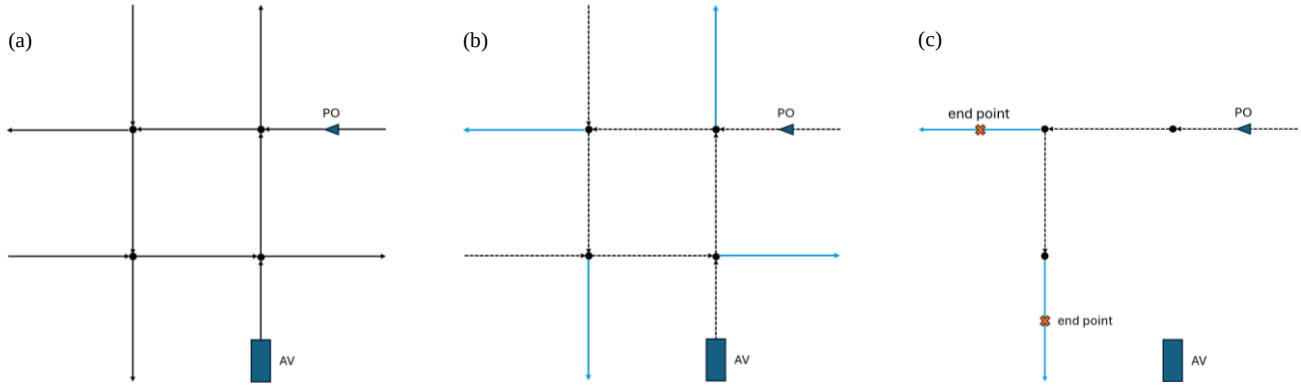


Figure 37. Visualization of the steps for getting end points from a cluster of intersection nodes. (a) shows the input graph with the AV and PO position displayed. (b) highlights all outgoing edges from the cluster. (c) show how end points are placed at the relevant outgoing edges, representing end points for straight and left turn trajectories.

In the current implementation, the shape of the PO's turning trajectories were based on the PO's end point and cost point. The use of the cost point was meant to make the trajectories more realistic, however by using a waypoint the trajectories could be improved. The issue with using only the POs' end point as a reference point for the trajectory is that if presented with a narrow turn like in *Figure 38. (a)*, because the cost point is based on the PO and the end point, the resulting turning trajectory becomes a long curve. The desired trajectory would be to have a curved trajectory from the PO to the beginning of the road that the PO is traveling to, and once the PO enters the new road its trajectory is straight towards the end point. This could be accomplished by using a waypoint. If the waypoint is placed at the beginning of the new road and the cost point is calculated using the waypoint, there would be a curved trajectory to the waypoint. The full trajectory could be calculated using A* twice. First from the PO to the waypoint with a cost point, then from the waypoint to the end point without a cost point.

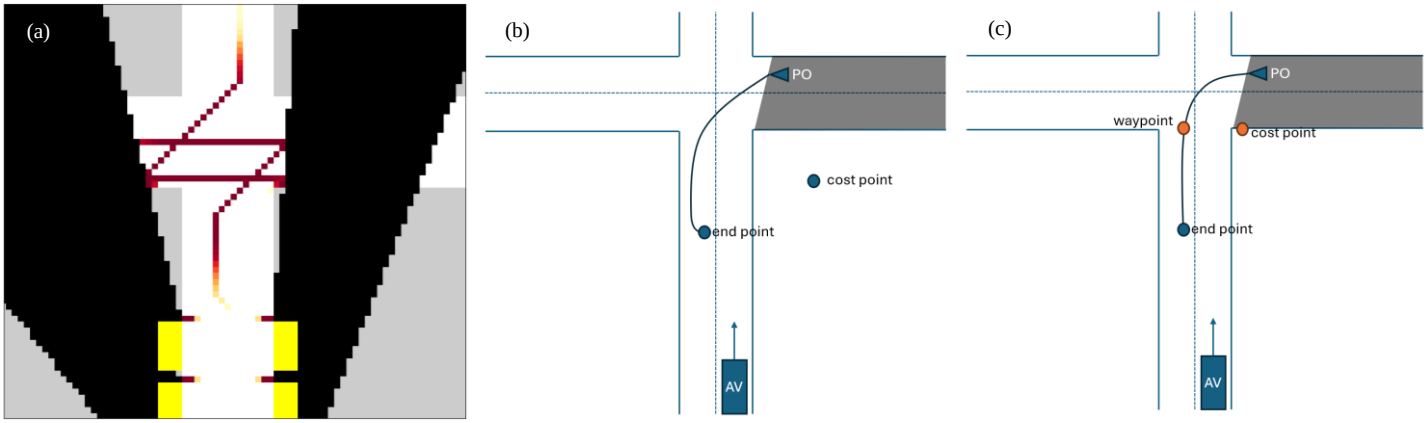


Figure 38. Visualizing how narrow trajectories can be better modeled with the help of a waypoint. (a) Figure T. (c) zoomed in. (b) visualization of trajectory without waypoint. (c) visualization of trajectory with waypoint.

Instead of using the A* algorithm for making the paths which the trajectories are made from, the trajectories could be calculated using a movement model, such as a Constant Speed Changing Rate and Constant Turn Rate (CSCRCTR) [47] model. In the current implantation, the POs were assumed to have the same speed when turning as when driving straight, i.e. the speed limit of the road. Drivers usually slow down when turning, but the current algorithm does not model the POs slowing down when turning. In scenario S2 the POs were taking the turns in 50 km/h, but a real driver would take it slower in such a narrow 90-degree turn. The consequences of not modeling a realistic speed for the turns are that the trajectories become longer, and more cells get a high probability of reach. These longer trajectories can in turn lead to conservative behavior by an ADS path planner if it was to use this work's output when planning a path. Using a movement model such as CSCRCTR, the trajectories could be calculated to fit vehicle dynamics instead of using a heuristic function with a cost point when modeling turning.

To model the reachability, this work looked at the predicted trajectories for all POs and assessed the probability of a PO reaching a certain cell in that trajectory. The assessment was based on a probability distribution of measured speed of drivers driving on different roads with the same speed limit. This distribution therefore shows the distribution between speeding drivers, drivers who follow the speed limit, and drivers driving slower than the speed limit. This distribution acts as the base for the probability function explained in section 2.2.3 which gave the probability of a PO reaching a cell in its trajectory. A different way of measuring probability of reach would be to use a Monte Carlo sampling method [48] to sample a set of POs in an occluded area and with the same speed distribution used for this work's risk assessment method, measure what percentage of the sample set manages to reach each cell of the trajectory. To sample several POs in an occluded area, a method for finding which positions in an occluded area that is relevant to place a PO in need to be made. A method for finding relevant positions is plausible given access to map data but would require more data processing in each step and would therefore likely increase runtime.

An unaddressed part of the risk assessment in this work was the case where two or more cells with reach probability from different POs overlap. That is, if two or more POs had a probability of reaching the same cell. Currently, this work handles overlapping probabilities by to choosing the highest probability of the ones overlapping as the final probability but only choosing the highest does not address how the different probabilities could influence each other. A future implementation would be to evaluate what a final probability would be for overlapping probabilities by making use of statistical models.

4.2.1 Implications of the choice of time horizon and cell size

Another key area of exploration is how the selection of parameters, particularly cell size and time horizon, affect the results. When choosing the cell size in the occupancy grid, two main parts need to be considered. The first is what resolution the grid would have, meaning how big are the objects that can be portrayed in the grid. The chosen cell size was 0.5 m, because a human has similar dimensions. If the cell is too large, then a cell containing a human might not be classified as occupied, because only a small fraction of the cell is occupied. The second part when choosing a cell size is the

performance of the algorithms using the occupancy grid. When reducing the cell size to half the size, the number of cells in the grid quadruples, and by doubling the cell size the number of cells is reduced by four times. In this work, the grid used was 80x72 m, with a 0.5 m cell size there were 23 040 cells. Meaning, the grid would have 5760 cells if the cell size was 1 m, and 92 160 cells if the cell size was 0.25 m. The number of cells directly corelate with the number of computations needed in different steps of the algorithm. Therefore, a larger cell size would positively increase performance. In conclusion the choice of cell size is a trade-off between resolution and performance.

Changing the time horizon is something that could tailor the output to usage. If the output of this work would be used in constant frequency while the AV drives, choosing a shorter time horizon could be fitting for maintaining an understanding of what risks lie within the occlusion, if a high update frequency is possible. With a high update frequency, this work's system could be run frequently and the AV could adapat accodingly. Wheras choosing a long time horizon could be fitting if this project's output was to be used to get an understanding of what may lie witihin occluded areas before entering them. The AV could detect when it is approaching a possible occluded area, by for example knowing that it is approaching an intersection in an utban environment, and use this knowledge to trigger this work's system. The output of this work could then be used to better understand what risks that could be occluded, and for example inform the AV that it should be cautious.

The results only show predicted trajectories with a 1 s time horizon, however the size of the grid allows for 5 s time horizons. The grid size was chosen so that predictions of 5 s time horizons are possible in common urban enviornments, and was chosen so that a path planner could be able to make predicted paths with help of the output of the ORA algorithm with a 5 s time horizon. While it is possible to predict 5 s trajectories for the POs too, the trajectories would be able to reach the whole grid which would not be relevant. With long trajectories the probability of reach would be high for a long distance from the EP. For a path planner that takes the output of the ORA into considerations, this could potentially cause conservative behavior, or alternatively make the path planer value the ORA output lower. Therefore, a lower time horizon is more fitting when making use of the ORA algorithm, because the predicted trajectories should be used as an indication of what risks could be occluded in different parts of the environment. By using a lower time horizon, the probability of reach represent more imediate risks compared to with a higher time horizon.

4.3 Discussion of the results

When analyzing the results of the ORA algorithm, the main point of interest is if the algorithm is able to find the expected POs and model their movement. That is, if it is able to identify and map potential risks stemming from occlusion. Since there exists no ground truth of what POs should exist and where they should move, there is no possibility to compare the results of the ORA algorithm with a ground truth from a quantified point of view. That is, for example, with what accuracy the ORA algorithm finds all existing POs. Instead, we chose to make an expected output based on what POs are probable to exist, and where they are able to move, given how the scenario is depicted and the same set of assumptions made for creating the algorithm. A good

result, from a qualitative point of view, is then proposed as the ability to find the expected POs and identify how they could move.

The results show how the ORA algorithm reliably found the expected POs but had trouble identifying all movements. For some of the scenarios (R1 and R2), not all expected trajectories were found. The missing trajectories largely depends on the use of a map that had been simplified by consolidating intersections. Thus, as discussed in section 4.2, another approach for finding end points that does not need consolidated intersections would be preferable. To further analyze the results, the algorithm could be tested on more scenarios to further test the reliability of the algorithm for finding all expected POs. Testing on more scenarios is also a good way to analyze if new issues present themselves, and how these issues could be solved to improve the algorithm.

The current way of analyzing the result of the ORA algorithm is not interpretable to how the algorithm could potentially make a path planner safer. To test this, for future work, the ORA algorithm should be evaluated by implementing it together with a path planner and analyzing how the ORA algorithm improves safety by comparing with the same path planner when not using the ORA algorithm. The two versions could then be tested on test scenarios such as scenario S1 and S2 to see if collision could be avoided. If with the help of the ORA algorithm, a path planner manages to avoid collision, further comparisons could be made by comparing the ORA algorithm to other related works and analyze how they perform in safety and comfort, as done by [23]. Measuring comfort together with safety is most of all a test for evaluating the path planner but it is also a factor which could indicate how conservative the risks found by the compared risk assessment methods are. Too conservative risks could for example lead to unnecessary braking, causing discomfort.

4.4 Ethical considerations

The creation of a system made to enhance safety for an AV presents several ethical considerations that need to be thought through before implementation. Even though this work aims at creating a product made to increase safety, since assumptions were made along the way to create it, there could be risks unaccounted for which could pose a safety hazard. One prominent example of such an assumption is that POs were given a finite number of possible trajectories. The POs classified as vehicles were assumed to move according to traffic laws, but it is possible that an occluded vehicle drives illegally, introducing a risk filled trajectory unaccounted for. Would a collision with this vehicle be the fault of a system not accounting for the risk or would it be the driver who drives illegally. Most would agree that it is the illegal driver who is at fault, but is it ethically correct to make it a possibility to collide with an illegal driver and by so, putting passengers in the AV at risk. Another part of this work that needs to be addressed is the risk assessment for right turns by POs, which this work chooses to dismiss. For a future implementation, right turns would need to be accounted for to not to pose a safety hazard, but should be done so in a fitting way, as described in section 4.1.

Another consideration is how to ethically make use of the output from this work. If a path planner was to use this work's output to plan a path for an AV, at what probability of reach would it view a path over potential trajectories as risk filled enough to adjust the path. Adjusting for all probabilities of reach would not be an option since the AV would drive in an unsafe way itself or hardly manage to drive due to the number of occluded risks often present in urban environments. Assuming a path planner uses some form of assessment model for the output of this work that lets the AV drive safely and unconservative while guaranteeing collision avoidance with 90% of POs, is it ethically correct to using this path planner even though a 100% guarantee of collision avoidance cannot be achieved. We argue that for all aforementioned ethical dilemmas, making use of systems which can increase safety and give an AV information it was previously without, is always more ethical than not using it.

4.5 Conclusion

This work proposed the ORA algorithm, capable of identifying and mapping potential risks originating from occluded areas, for use in ADS path planning algorithms. The ORA algorithm made use of an occupancy grid, as well as an SD map to make an interpretation of the surrounding environment. From the interpretation, the occluded parts of the environment were analyzed to find where potential POs could emerge from. POs, represented road users occluded from sensor measurements that could pose a risk to the AV. The PO's potential paths were modeled in the occupancy grid, and from these paths, potential trajectories with a corresponding risk were given to each PO by making use of simplified reachability. The risk was expressed as the probability of a PO to reach each cell in the PO's trajectory.

From the results, it was possible to conclude that the ORA algorithm could reliably find POs in the occluded areas of the AV's environment and model relevant trajectories and assess the risk of reach from the corresponding PO of each cell in the trajectory. However, the implementation of the ORA algorithm could be further improved. The result show that trajectories could lack realistic shapes, and some expected trajectories were missing. Potential future implementations such as making use of waypoints, and changing approach when analyzing the map were proposed to address the flaws in the algorithm.

5 References

[1] G. J. Alexander and H. Lunenfeld, “Driver expectancy in highway design and traffic operations,” United States. Federal Highway Administration. Office of Traffic Operations, Washington, DC, USA, Tech. Rep. FHWA-TO-86-1, 1986. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/890, Accessed: 2025-01-17.

[2] A. Habibovic, E. Tivesten, N. Uchida, J. Bärgman, and M. Ljung Aust, “Driver behavior in car-to-pedestrian incidents: An application of the driving reliability and error analysis method (dream),” *Accident Analysis Prevention*, vol. 50, pp. 554–565, Jan. 2013, doi: 10.1016/j.aap.2012.05.034.

[3] M. Pechinger, T. Niels and K. Bogenberger, “Threshold analysis of static and dynamic occlusion in urban areas: a connected automated vehicle perspective”, *Transportation Research Record: Journal of the Transportation Research Board*, vol 2678, no. 9, pp. 823-836, Feb. 2024, doi: 10.13140/RG.2.2.26345.26729.

[4] M.C. Simon, T. Hermitte and Y. Page, “Intersection Road Accident Causation: A European View”, *International Technical Conference on the Enhanced Safety of Vehicles (ESV)*, Stuttgart, Germany, 2009, pp. 1-10. [Online]. Available: https://www-nrd.nhtsa.dot.gov/pdf/ESV/Proceedings/21/09-0370.pdf, Accessed 2025-04-28.

[5] M. Abdel-Aty and S. Ding, “A matched case-control analysis of autonomous vs human-driven vehicle accidents”, *Nature Communications*, vol. 15, no. 4931, Jun. 2024, doi: 10.21203/rs.3.rs-3401212/v1.

[6] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, “A Survey of Autonomous Driving: Common Practices and Emerging Technologies”, *IEEE Access*, vol. 8, pp. 58443-58469, Mar. 2020, doi: 10.1109/ACCESS.2020.2983149.

[7] R. Hussain and S. Zeadally, “Autonomous Cars: Research Results, Issues, and Future Challenges”, *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275-1313, Sep. 2018, doi: 10.1109/COMST.2018.2869360.

[8] A. Eichberger, Z. Szalay, M. Fellendorf and H. Liu, “Advances in Automated Driving Systems”, *Energies*, vol.15, no. 3476, 2022, doi: 10.3390/en15103476.

[9] N. Scheiner, F. Kraus, N. Appenrodt, J. Dickmann, and B. Sick, “Object detection for automotive radar point clouds – a comparison,” *AI Perspectives*, vol. 3, no. 6, Nov. 2021, doi: 10.1186/s42467-021-00012-z.

[10] J. Müller, J. Strohbeck, M. Herrmann and M. Buchholz, “Motion Planning for Connected Automated Vehicles at Occluded Intersections With Infrastructure Sensors”, *IEEE Transactions of Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17479 – 17490, Oct. 2022, doi: 10.1109/TITS.2022.3152628.

[11] S. A. Yusuf, A. Khan and R. Souissi, “Vehicle-to-everything (V2X) in the autonomous vehicles domain – A technical review of communication, sensor, and AI technologies for road user safety”, *Transportation Research Interdisciplinary Perspectives*, vol. 23, Jan. 2024, doi: 10.1016/j.trip.2023.100980.

[12] C. Zhang, F. Steinhauser, G. Hinz and A. Knoll, “Occlusion-Aware Planning for Autonomous Driving With Vehicle-to-Everything Communication”, *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 1229-1242, Jan. 2024, doi: 10.1109/TIV.2023.3308098.

[13] V. Rishiwal, U. Agarwal, A. Alotaibi, S. Tanwar, P. Yadav and M. Yadav, "Exploring Secure V2X Communication Networks for Human-Centric Security and Privacy in Smart Cities”, *IEEE Access*, vol. 12, pp. 138763-138788, Sep. 2024, doi: 10.1109/ACCESS.2024.3467002.

[14] Y. Han, J. Banfi, and M. Campbell, “Planning paths through unknown space by imagining what lies therein,” in *4th Conference on Robot Learning (RoCL)*, Cambridge MA, USA, 2020. [Online]. Available: https://arxiv.org/abs/2011.07316, Accessed: 2025-02-05.

[15] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation.”, in *Computer Vision – ECCV 2018*, Munich, Germany, 2018, pp. 833-851. [Online], Availbale: https://doi.org/10.48550/arXiv.1802.02611, Accessed 2025-05-08.

[16] A. Elfes, “Using occupancy grids for mobile robot perception and navigation”, *Computer*, vol. 22, no. 6, pp. 46-57, 1989, doi: 10.1109/2.30720.

[17] M. P. Ronecker, M. Schratter, L. Kuschnig, and D. Watzenig, “Dynamic occupancy grids for object detection: A radar-centric approach,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, 2024. [Online]. Available: https://arxiv.org/abs/2402.01488, Accessed: 2025-02-05.

[18] S. Steyer, G. Tanzmeister, and D. Wollherr, “Object tracking based on evidential dynamic occupancy grids in urban environments”, in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, USA, 2017, pp. 1064-1070. [Online], Available: https://doi.org/10.1109/IVS.2017.7995855, Accessed 2025-05-09.

[19] M. E. Bouzouraa, U. Hofmann, “Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors”, in *2010 IEEE Intelligent Vehicles Symposium*, La Jolla, CA, USA, 2010, pp. 294-300. [Online], Available: https://doi.org/10.1109/IVS.2010.5548106, Accessed 2025-05-09.

[20] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian and K. Fujimura, "Navigating Occluded Intersections with Autonomous Vehicles Using Deep Reinforcement Learning", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, 2018, pp. 2034-2039. [Online]. Available: https://ieeexplore.ieee.org/document/8461233, Accessed 2025-04-29.

[21] R. Trauth, K. Moller, and J. Betz, “Toward safer autonomous vehicles: Occlusion-aware trajectory planning to minimize risky behavior,” *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 929–942, 2023, doi: 10.1109/ojits.2023.3336464.

[22] M. Lee, M. Sunwoo, and K. Jo, “Collision risk assessment of occluded vehicle based on the motion predictions using the precise road map,” *Robotics and Autonomous Systems*, vol. 106, no. 12, pp. 179–191, Aug 2018, doi: 10.1016/j.robot.2018.05.005.

[23] D. Wang, W. Fu, Q. Song, and J. Zhou, “Potential risk assessment for safe driving of autonomous vehicles under occluded vision,” *scientific reports*, vol. 12, no. 4981, 2022, doi: 10.1038/s41598-022-08810-z.

[24] W. Zhan, C. Liu, C.-Y. Chan, and M. Tomizuka, “A non-conservatively defensive strategy for urban autonomous driving”, in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, 2016, pp. 459-464. [Online], Available: https://doi.org/10.1109/ITSC.2016.7795595, Accessed 2025-05-12.

[25] Y. Qian, C. Deng, J. Xu, X. Qu, and Z. Song, “Occlusion-aware collision avoidance trajectory planning with potential collision risk assessment for autonomous vehicle,” *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 72, no. 4, p. e149819, Aug. 2024, doi: 10.24425/bpasts.2024.149819.

[26] M. -Y. Yu, R. Vasudevan and M. Johnson-Roberson, "Occlusion-Aware Risk Assessment for Autonomous Driving in Urban Environments", *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2235-2241, 2019, doi: 10.1109/LRA.2019.2900453.

[27] H. Park, J. Choi, H. Chin, S.-H. Lee, and D. Baek, “Occlusion-aware risk assessment and driving strategy for autonomous vehicles using simplified reachability quantification,” *IEEE Robotics and Automation Letters. Preprint Version*, vol. 8, no. 12, pp. 8486–9493, Dec 2023, doi: 10.1109/LRA.2023.3329627.

[28] C. van der Ploeg, T. Nyberg, J. M. G. Sánchez, E. Silvas, and N. V. D. Wouw, “Overcoming Fear of the Unknown: Occlusion-Aware Model-Predictive Planning for Automated Vehicles Using Risk Fields”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no.9, pp. 12591-12604, 2024, doi: 10.1109/TITS.2024.3382507.

[29] Aptiv, “Aptiv”, 2025. [Online], Available: https://www.aptiv.com/ (accessed on: 2025-05-26).

[30] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[31] M.G. Pletnev, P.I. Pospelov, A.A. Akulov, D.S. Taldykin and A.G. Zaitsev "Determination of Traffic Flow Parameters Using High-Precision Positioning System to Improve Traffic Efficiency and Safety" in *2024 Intelligent Technologies and Electronic Devices in Vehicle and Road Transport Complex (TIRVED)*, Moscow, Russian Federation, 2024, pp. 1-8. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10769862, Accessed: 2025-04-21.

[32] Boeing, G. 2025. "Modeling and Analyzing Urban Networks and Amenities with OSMnx." *Geographical Analysis*, published online ahead of print. doi:10.1111/gean.70009.

[33] X. Tang et al. "High-Definition Maps Construction Based on Visual Sensor: A Comprehensive Survey", *IEEE Transactions on Intelligent Vehicles*, vol 99, pp.1-23, Jan. 2023, doi: 10.1109/TIV.2023.3336940.

[34] J. Li, P. Jia, J. Chen, J. Liu and L. He, "Local map Construction Methods with SD map: A Novel Survey" *Journal of LATEX Class Files*, vol. 18, no. 9, Sep. 2020, doi: 10.48550/arXiv.2409.02415.

[35] Euro NCAP, Leuven, Belgium, *"CA 002 - Verification Conditions for Robustness Layers v1.0"*, Accessed: 2025-05-06. [Online]. Available: https://www.euroncap.com/en/for-engineers/protocols/2026-protocols/.

[36] Euro NCAP, Leuven, Belgium, *"Euro NCAP Protocol - Crash Avoidance - Frontal Collisions v1.0"*, Accessed: 2025-05-06. [Online]. Available: https://www.euroncap.com/en/for-engineers/protocols/2026-protocols/.

[37] R. W. Bohannon and A. W. Andrews, "Normal walking speed: a descriptive meta-analysis", *Physiotherapy*, vol. 97, no. 3, pp. 182-189, Sep. 2011, doi: 10.1016/j.physio.2010.12.004.

[38] Felix, W. Yodianto, H. L. H. S. Warnars, L. L. H. S. Warnars, A. Ramadhan and T. Siswanto, "Searching Routing using A-Star (A*) Search Algorithm", in *2024 3rd International Conference on Creative Communication and Innovative Technology (ICCIT)*, Tangerang, Indonesia, 2024, pp. 1-7. [Online]. Available: https://ieeexplore.ieee.org/document/10701177, Accessed: 2025-04-21.

[39] K. I. Park, "Random Variables", in *Fundamentals of Probability and Stochastic Processes with Applications to Communications*, Cham, Switzerland: Springer Cham, 2017, ch. 4, pp. 76-88, [Online] Available: https://doi.org/10.1007/978-3-319-68075-0, Accessed: 2025-05-06.

[40] L. Råde, B. Westergren, "Probability theory", in *Mathematics Handbook: for Science and Engineering*, 6th edition, F. Wikström, Ed. Lund, Sweden: Studentliteratur AB, 2019, ch. 17, pp. 440.

[41] R. V. Clarke, "The Distribution of Deviance and Exceeding the Speed Limit", *The British Journal of Criminology*, vol. 36, no. 2, pp.169-181, 1996, doi: 10.1093/oxfordjournals.bjc.a014080.

[42] F. Lercher and M. Althoff, "Specification-Compliant Reachability Analysis for Autonomous Vehicles Using On-the-Fly Model Checking," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, Jeju Island, Korea, Republic of, 2024, pp. 1484-1491. [Online], Available: https://doi.org/10.1109/IV55156.2024.10588549.

[43] P. Kunz and M. Schreier, "Automated Detection of Construction Sites on Motorways", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, USA, 2017, pp. 1378-1385. [Online], Available: https://doi.org/10.1109/IVS.2017.7995903, Accessed 2025-05-14.

[44] E. S. Dawam and X. Feng, "Traffic Light Detection and Recognition in Autonomous Vehicles (AVs)," in *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, Falerna, Italy, 2022, pp. 1-6. [Online], Available: https://doi.org/10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927813, Accessed 2025-05-15.

[45] H. Diwanji, J. Liao, A. Tumu, H. I. Christensen, M. Vazquez-Chanlatte and C. Tsuchiya, “SD++: Enhancing Standard Definition Maps by Incorporating Road Knowledge using LLMs” To be published. Accessed: 2025-04-23, doi: 10.48550/arXiv.2502.02773.

[46] D. Zhu, N. N. Sze, L. Bai, “Roles of personal and environmental factors in the red light running propensity of pedestrian: Case study at the urban crosswalks”, *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 76, pp.47-58, 2021, doi: 10.1016/j.trf.2020.11.001.

[47] G. Zhai, H. Meng and X. Wang, “A Constant Speed Changing Rate and Constant Turn Rate Model for Maneuvering Target Tracking”, *Sensors*, vol. 14, no. 3, pp. 5239-5253, Mar. 2014, doi:10.3390/s140305239.

[48] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, New York, NY, USA: Springer, 1996. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4757-2553-7.

DEPARTMENT OF MECHANICS AND
MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 20xx
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY