Semantic Explainable Navigation in Structured Environments (SENSE)

Hobma, Rinse

Award date:
2022

Department of Mechanical Engineering
Control Systems Technology

Semantic Explainable Navigation in Structured Environments (SENSE)

Master thesis

Author	R. Hobma (1022328)
Academic supervisor	dr. MSc. C.A. López Martínez
Mentor	dr. ir. M.J.G. van de Molengraft
CST report number	CST2022.064

Graduation committee

December 5, 2022

Semantic Explainable Navigation in Structured Environments (SENSE)

R. Hobma C.A. Lopez Martinez M.J.G. van de Molengraft

***Abstract*—When a mobile robot needs to autonomously move from one place to another, a navigation strategy is required. One of the most common strategies for mobile robot navigation is defining this task as a numerical optimization problem with constraints. Such methods come with drawbacks like the robot getting stuck in local minima, and not being able to give a specific reasoning for the actions of the robot. In this thesis a different approach is explored, which determines the control input using semantic information about environment. The structured environment is provided as a semantic map to the robot, consisting of connected polygons indicating hallways and junctions. As this thesis focuses on a dynamic environment which the robot shares with humans some simple traffic rules behavior is desired like driving on the right-hand side of hallways as much as possible, and not blocking junctions, areas around doors and emergency exits. To represent this behavior polygons with certain semantic behavioral labels such as 'do not enter this area' and 'do not stop in this area' attached to them can be placed in the environment at a specific location. The semantic and geometrical information of these polygons are used in combination with predictions to determine a locally sufficient velocity and steering angle to make progress towards the goal, whilst complying with the semantic behavior described by the polygons. Deriving the control input based on the local geometry and semantics allows to provide clear explanations of the robot actions. Furthermore, the behavior can be tailored to specific situations, as moving through a hallway can require different behavior than crossing a junction. Simulations and experiments demonstrate that this navigation strategy approach is capable of determining control input that complies with the desired semantic behavior without using standard optimization methods, resulting in explainable actions of the robot.**

***Index Terms*—Mobile, robotics, navigation, semantic, explainable**

I. INTRODUCTION

Mobile robots are robots which can move around in a certain environment. Over the years robots have evolved from static robots like robotic arms on assembly lines, to mobile robots that can drive around autonomously. Where traditionally mobile robots operate in controlled environments, sharing an environment with humans forms new challenges to overcome. Over the years there has been an increase in demand for mobile robotics in environments like warehouses, hospitals, nursing homes, etc. The layout of these buildings is known, but people can walk around and place, move, and remove objects, resulting in a semi-structured environment. For example, the growing e-commerce sector demands the use of mobile robots in a warehouse environment for fast and efficient order picking[1] alongside human employees. The Covid pandemic provides an interesting perspective where mobile robots could be used to perform logistical tasks in healthcare facilities to reduce the risk for medical personnel who would need to suit up in protective gear and risk getting infected. This event has also shown that personnel shortage is a large issue for upscaling the intensive care in the Netherlands. [2] As the population grows older this also causes an increase in demand for healthcare personnel [3]. This goes to show that these industries could profit from using mobile robotics for automatable logistical tasks such that the time of the available personnel can be spent as effectively as possible and reduce their physical effort.

A. Environment

A semi-structured environment with dynamic obstacles like warehouses or nursing homes brings many challenges for the navigation strategy. For example when it shares the environment with humans who tend to follow basic traffic rules in indoor environments the robot also needs to follow these rules to disrupt the natural flow in the environment as little as possible. The physical environment in which the robot operates thus consists of: (i) known obstacles, e.g. the layout of the walls, (ii) unknown static obstacles, e.g. things that have been placed in the environment which are not on the provided map like chairs or plant pots, (iii) unknown dynamic obstacles, e.g. people walking around. The physical measurable environment is however not the only thing that describes an environment. Often these physical things or specific areas have semantic information attached to them that can further help describe the desired behavior of a mobile robot. For example, a door has not only the property of 'being a door', but has additional meaning attached to it. People can can for example come through the door without seeing what is on the other side, so the area around the door must not be blocked. Being able to use this kind of semantic information about the environment can help specify the desired robot behavior.

B. Navigation

An autonomous mobile robot is often equipped with a set of sensors to obtain information about its environment and deduce its pose with respect to that environment. The robot can often be given an assignment in the form of 'move from location A to location B' which introduces a big challenge: navigation. Given an assignment the robot needs to deduce what its control input must be to complete the assignment. The environment described in I-A results in a challenge for the navigation. The navigation must be able to interpret a representation of this semantic information and corresponding desired behavior, and provide a control input which satisfies the specified behavior while executing the assignment.

C. *Explainability*

This project will focus on *explainable* navigation of an autonomous robot transporting a cart through a semi structured environment from area A to area B. The term *explainable* can have several meanings depending on whom it concerns. For people who work alongside the robot in a warehouse for example explainability can result from the robot following basic traffic rules such that a person working in the same environment can make a broad estimate of what the behavior of the robot will be in a certain situation. From a engineering point of view explainability can be that every action of the robot can be traced back to a specific function or reasoning within the software. In this project the focus will be on the latter.

D. *Background*

Motion planning for mobile robots has been around for quite a while and many different strategies have been explored. This section will go through some existing methods.

Many navigation strategies rely on a finding a global path. This global path can be defined as a series of waypoints which the navigation strategy then uses to determine the control input. To plan this path the robot needs a map, and it needs to know where it is on that map. One of the most popular representations for a map is an occupancy grid map [4]. Each node of this grid can be assigned a value to represent its accessibility. These nodes can then be connected through edges which also hold a certain value describing the 'cost' of moving from one node to another. There are several path finding algorithms available like dynamic programming, Dijkstra's algorithm, or A-star, that can then determine the path from node A to node B with the lowest total cost [5]. These algorithms do however not consider the kinematics and dynamics of the robot, and do not consider semantic information about the environment. Therefore, outcomes of these algorithms can not directly be used to actuate the robot. Some of the following motion planning strategies do however rely on a predefined path like this.

- A potential field algorithm consists of several force vectors working on the robot. The robot first needs to generate a global path defined as waypoints. This then allows the robot to obtain a certain attraction force vector in the direction it should move. The robot is also subjected to repulsive force vectors from walls and other obstacles. The resulting vector could want the robot to move in any direction, but this is not always possible due to kinematic constraints. [6]
- Timed Elastic Band (TEB) is developed as a plugin for the ROS navigation package [7]. TEB can determine local trajectories for a non-holonomic platform. It converts an initial global path which is made up of a sequence of waypoints to a trajectory. The method will search for a shortest or fastest path based on an objective function which requires a number of parameters which the user will have to tune, resulting in a non-linear optimization problem. This results in a lack of explainability of the results, which is undesired.
- The DWA plugin for the ROS navigation package can handle non-holonomic platforms. It requires a global path or goal and will then locally around the robot create a grid map based on a cost function. It simulates a set of kinematically feasible trajectories to check the feasibility of the trajectories. After that the trajectory with the lowest cost and thus the optimal trajectory is chosen, and thus the control input for the robot is found. This method thus also results in some form of non-linear optimization problem due to the used cost function representation around the robot, which is undesired. [8]
- The motion planning algorithm of OMG-tools [9] is based on small dimensional optimization problems using a spline parameterization of the motion trajectory. It uses a series of waypoints as initial path. Splines consisting of a series of polynomial functions which can form nearly any shaped trajectory are generated on the convex hull of the initial path to define the trajectory. The method does however not consider semantic information about the environment.
- The control input for a robot can be obtained by expressing the task of the robot as an objective function subjected to a set of constraints. When this objective is then optimized the outcome theoretically is the optimal solution. This method requires a predefined path, or some form of goal, as this usually is incorporated in the objective function. As mentioned, many navigation strategies use some form of this optimization. However due to the non-linear nature of the problem this can lead to unexplainable results, and solutions going towards local minima.
- As opposed to trying to find an optimal solution for the control input one could put the focus on the constraints of the problem. Constraint satisfaction problems (CSP's) generally consist of a set of variables to which you want to find a feasible solution, the domains of these variables, and a set of constraints that bounds these variables [10]. Compared to optimization strategies the objective function is thus discarded. This also means that the solution does not necessarily depend on a goal or predefined path.
- In an effort to create a navigation strategy based on a semantic map allowing for the implementation of basic traffic rules Napoleon navigation [11] was created. The strategy is based on using a semantic map that indicates hallways, intersection, and turns. Depending on the location of the robot it will follow a specific set of rules.

Many of the above mentioned strategies focus on optimizing an objective function. As this project focuses on an environment where the main structure is known along with semantic information about this structure optimizing an objective function might be unnecessary. The robot does not need to follow an optimal path, but rather move in a sufficiently good manner, complying with the imposed task and constraints. Optimization methods can also result in the robot getting stuck around obstacles in local minima. When the robot takes actions which the user might not expect, the exact reason behind this action is hard to trace back when a standard optimization method is used. Furthermore, if the user desires to express the intentions of the robot to its environment it is also useful to be able to have specific reasoning behind the taken actions. CSP's form an interesting solution which do not require nonlinear optimization. Conventional CSP solvers are however not suitable for using semantic information. Napoleon Navigation provides an interesting basis for the proposed new strategy in this thesis. Using the map indicating hallways and junctions with polygons provides valuable information that can help with the reasoning why a specific control input is applied in a certain situation. The way the control input is then determined by Napoleon navigation based on this information is however lacking robustness and depends on many tuning parameters.

E. *Project goal*

Determine the control input for a mobile robot in a semi-structured environment, while being able to take a semantic behavior representation as input to represent the desired behavior of the robot. The navigation method must provide clear explainability and reasoning of why a control input is chosen. The navigation must also be able to deal with different footprints of the robot such that the method can be used on different types of mobile robots.

F. *Requirements*

To achieve the project goal, the following requirements must be met.

- The navigation strategy must be able to deal with semantic information about its environment. This way the behavior of the robot can be adapted to the local situation, and the robot has a broader understanding of its environment than only knowing if a space is occupied or not.
- When determining the control input the footprint of the robot has to be considered. This allows the method to be used on platforms with different footprints and kinematics, and allows for the robot to dock to carts which can also result in a different footprint and kinematics.
- The control input must be determined in a way such that the reasoning behind a control input can always be traced back and explained. Having this knowledge about the reasoning can help with expressing the intentions of the robot to its environment.
- The robot must make progress towards its goal whenever possible, whilst complying with the specified behavior and imposed constraints.
- For safe navigation the robot must never collide with obstacles.

G. *Outline*

The structure of this thesis is as follows. Section II provides an alternative strategy, Section III contains the results of the proposed alternative strategy, and the conclusion and recommendations for further research will be provided in Section IV.

II. Proposed navigation strategy

A new navigation strategy is proposed based on a non-holonomic platform. This is to allow the strategy to work on as many platforms as possible, where holonomic robots can behave in a non-holonomic matter but not the other way around. Also, combinations of a holonomic robot docked to non-holonomic carts result in a non-holonomic platform. The vehicle model is introduced, after which the structured environment semantic map is explained. Next, a method to represent parts of the world and desired behavior of the robot in semantic areas is provided. Then the strategy of SENSE is explained and applied on this provided world model.

A. *Vehicle model*

The vehicle model used for SENSE is a tricycle model as shown in 1.

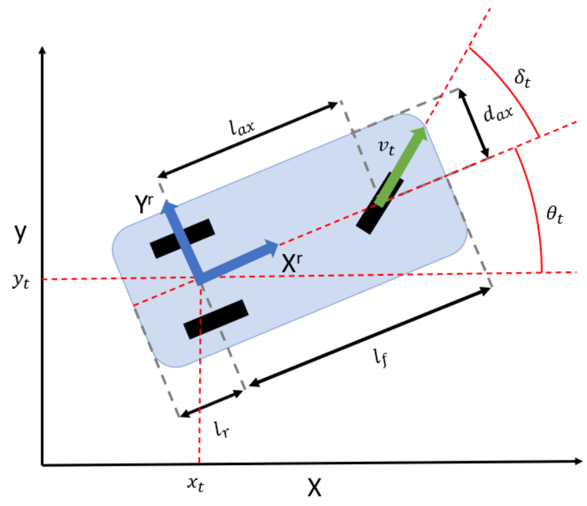

Fig. 1. Tricycle vehicle model. X and Y represent the inertial frame, X^r and Y^r the robot frame, θ_t the orientation of the vehicle, v_t the front wheel velocity, δ_t the front wheel steering angle.

The vehicle velocity v_t and steering angle δ_t are the control input and can be used to describe the motion of the center of the rear axle as:

$$\begin{bmatrix}\dot{x_t}\\ \dot{y_t}\\ \dot{\theta_t}\end{bmatrix} = v_t \cdot \begin{bmatrix}cos(\delta_t)cos(\theta_t)\\ cos(\delta_t)sin(\theta_t)\\ \frac{1}{l_{ax}}sin(\delta_t)\end{bmatrix} \tag{1}$$

where $\dot{x_t}$ and $\dot{y_t}$ are the x- and y-velocities of the vehicle with respect to the inertial frame and $\dot{\theta_t}$ the rotational velocity around the rear axle. θ_t represents the orientation of the robot in the inertial frame and l_{ax} is the distance between the front wheel and rear axle of the vehicle as indicated in Figure 1.

B. Structured environment semantic map

The robot operates in an environment of which the main structure is known. As with the Napoleon navigation strategy, the route that the user wants the robot to take is described as a set of connected polygons with semantic labels 'hallway' and 'junction'. These types of polygons will from hereon be referred to as route polygons. A task for the robot is defined as moving from route polygon A to route polygon B, where polygon A is the current polygon that the robot is in. A network of these polygons can be created to describe the complete environment, and routes could be determined using a simple a-star algorithm to find the consecutive route polygons. This global route planning is however not part of this thesis. Specifics on the conventions of the construction of the route polygons can be found in Appendix **??**.

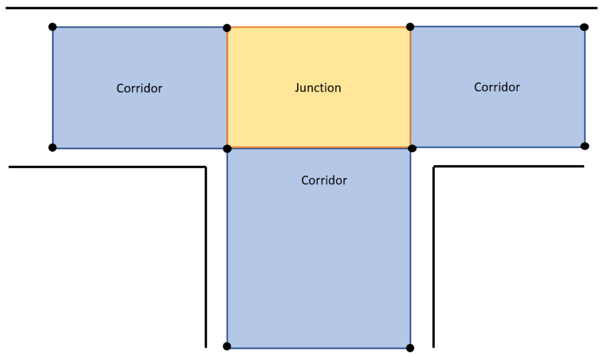

Fig. 2. Route polygons example. Black lines indicate physical structure, black dots represent polygon nodes.

Figure 2 shows a schematic example of the placement of route polygons with labels 'hallway' and 'junction' in a structured environment.

C. Elemental areas

Besides the route polygons a method is needed to make the robot follow the desired behavior based on semantic information about the environment. In the Napoleon navigation strategy this was done by defining tubes within the route polygons in which the robot has to stay and explicitly incorporate the desired rules in the navigation strategy. This is not a scalable way of representing desired behavior, as every new behavior that might be required has to be programmed into the navigation software. A much more scalable way of representing semantic behavior for a mobile robot is presented in [12]. It presents a method where the environment and desired semantic behavior is represented by placing polygons with semantic behavior labels attached to them in the world model. This method boils down desired semantic behavior into 'elemental areas' which indicate 'no-enter' and 'avoid' behavior. For this thesis areas with a 'no-stop' label are also introduced to prevent congestion of traffic and creating unsafe situations. Figure 3 shows a a simplified example of placing these elemental areas.

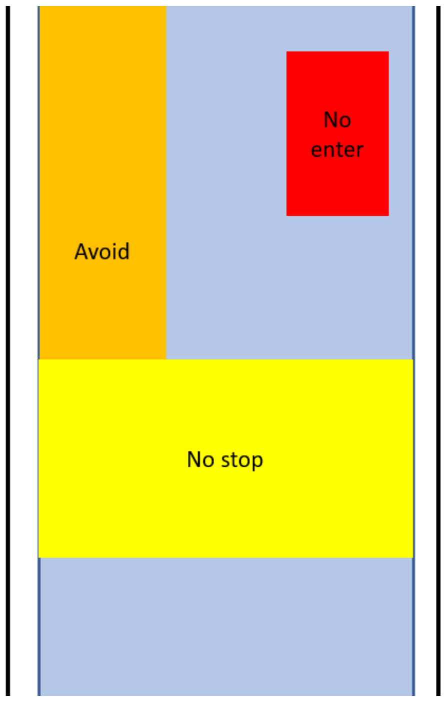

Fig. 3. Elemental areas. Red polygons indicate 'no-enter' areas, orange indicates 'avoid' areas and yellow indicates 'no-stop' areas.

If the navigation algorithm is able to deal with these elemental areas, new types of behavior can be constructed by expressing them with placement of combinations of these areas without needing to change the navigation strategy.

D. Main strategy

The main idea to determine the control input based on the route polygons and elemental areas is to make use of the information locally around the robot. Based on a physical horizon it is determined which polygons are taken into account by the navigation strategy. The goal then becomes to find a trajectory such that within this physical horizon the robot does not collide with no-enter elemental areas or exceeds the boundaries of the route polygons. In order to be able use the geometry of the environment and to say something about theoretical future poses of the robot without performing predictions this navigation strategy looks at trajectories with constant steering angles. When a steering angle for a tricycle model is constant the robot will follow a circular trajectory, independent of the velocity (assuming no slip). This circular trajectory allows us to have some information about theoretical future robot poses. Based on the local environment a set of steering ranges can be determined for which the robot will avoid a collision within the given horizon. Essentially this thus becomes its own specific form of CSP where a lower and upper bound for the steering angle is found. A schematic representation of the strategy is shown in 4.

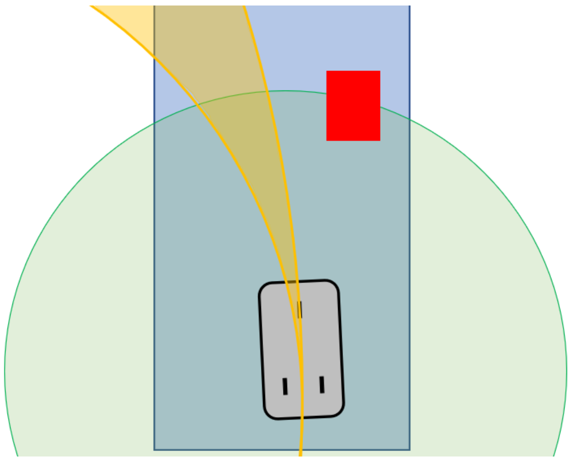

Fig. 4. Main strategy example. Blue polygon represents hallway route polygon. Red polygon represents an obstacle. Physical horizon is indicated in green. The highlighted yellow part indicates the set of safe trajectories within the physical horizon with constant steering angle.

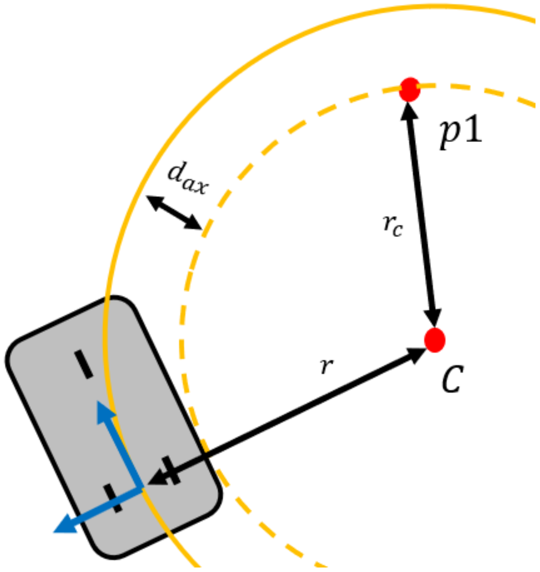

Fig. 5. Circular trajectory passing point *p1* on the concave side of the trajectory. Solid yellow line indicates the trajectory of the origin of the robot frame. Dashed yellow line represents the trajectory of the most inner point of the robot footprint

To determine the set of steering angle ranges there are a couple of core functions that are used.

- The first one being a function to avoid a point on the concave side of a circular trajectory. Based on the width of the robot it is determined what the radius of the trajectory of the robot must be to avoid a specific point using 2 where r is the radius of the trajectory of the robot frame origin, r_c the distance from the trajectory center C to point $P1$, and d_{ax} is half the width of the robot.

$$r = r_c(r) + d_{ax} \tag{2}$$

r_c can be determined as:

$$r_c(r) = \sqrt{(P1_x - C_x(r))^2 + P1_y - C_y(r))^2} \tag{3}$$

where the position of C can be determined as:

$$C(r) = (r * cos(\theta \pm \pi/2), r * sin(\theta \pm \pi/2)) \tag{4}$$

and the sign of $\pm\pi/2$ depends on which direction the trajectory is curved.
- The second function is to avoid a point on the convex side of a circular trajectory as shown in 6. Again, the radius of the trajectory can be determined knowing what the most outer radius of any point on the robot will be for a given trajectory. When the radius r of a trajectory is given the outer radius r_{outer} can be determined as:

$$r_{outer}(r) = max(\sqrt{(r + d_{ax})^2 + (l_f)^2}, \sqrt{(r + d_{ax})^2 + (l_r)^2}) \tag{5}$$

The steering angle can then be found by solving for r when r_{outer} equals r_c.

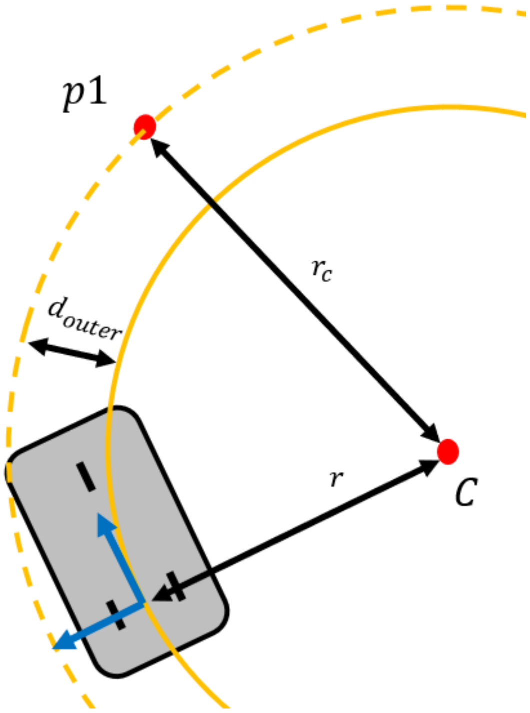

Fig. 6. Circular trajectory passing point *p1* on the convex side of the trajectory. Solid yellow line represents the trajectory of the origin of the robot frame. Dashed yellow line represents the trajectory of the most outer point of the robot footprint.

- The third function is to avoid a line on the convex side of a circular trajectory. Avoiding a point on the convex side of a trajectory is not always enough as the robot is not allowed to cross some lines, like the edges of the route polygons. The trajectory can be determined in the same manner as avoiding a point on the convex side of the trajectory, but instead taking r_c as the distance from C to the line spanned by points $P1$ and $P2$, as shown in 7.

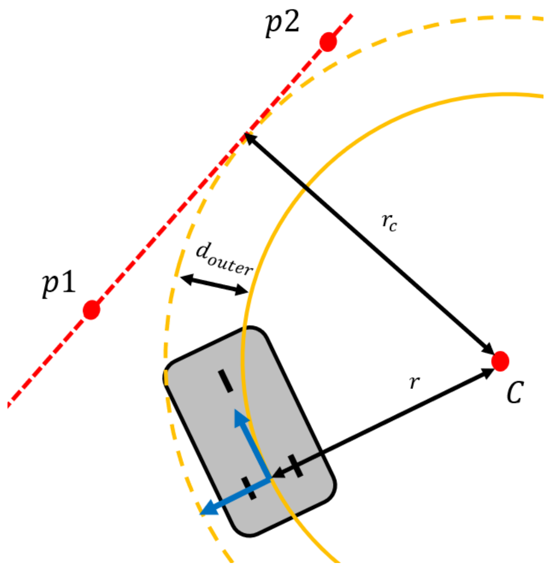

Fig. 7. Circular trajectory avoiding the line spanned by points *p1* and *p2*. Solid yellow line represents the trajectory of the origin of the robot frame. Dashed yellow line represents the trajectory of the most outer point of the robot footprint.

Based on the position of the robot with respect to the route polygons there are three situations that are recognized and acted upon. The first situation is where both the front wheel and center of the rear axle are in a hallway polygon. The second situation is when the front wheel is in a junction polygon. The third situation is when the front wheel is in a hallway polygon, while the center of the rear axle is in a junction polygon. Each of these situations and the application of the three core functions are explained next.

E. Cases

1) Hallway: Based on the size of the physical horizon intersection points with the edges of the route polygon can be determined. Based on the orientation of the robot with respect to those points there are several situations that can be distinguished. (i) Both points can be passed on the concave side of the trajectories. This gives the robot a range of steering angles between the steering angles corresponding with the two determined trajectories for which the robot will not collide with the edges of the hallway polygon. The steering angle in the middle of this range is then chosen as desired input. (ii) One point can be passed on the concave side of a trajectory, while the other one can not. Therefore the function to determine the steering angle that avoids the line of the hallway wall is used. If there is still a range of steering angles available the steering angle corresponding to the trajectory that steers around the concave point is used as desired input. (iii) If the same situation as in (ii) occurs, but the determined steering angles are such that no range is formed, the one corresponding with the avoiding of the line on the convex side of its trajectory is used. The reasoning behind this is that the robot is heading towards that line segment so it gets priority.

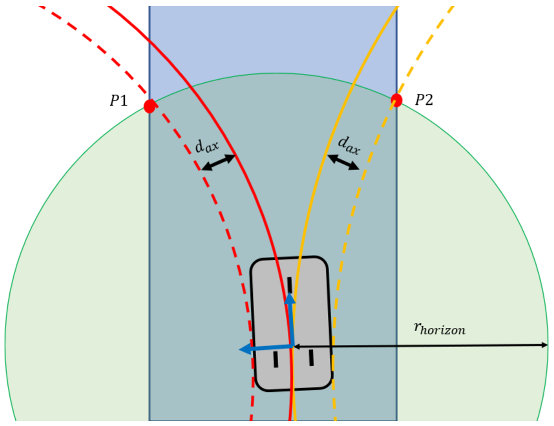

Fig. 8. Hallway situation 1. Trajectory range based on passing both $P1$ and $P2$ on concave side. The solid orange circle indicating the trajectory of the robot origin for the minimum desired steering angle. The solid red line indicates the robot origin trajectory for the maximum desired steering angle.

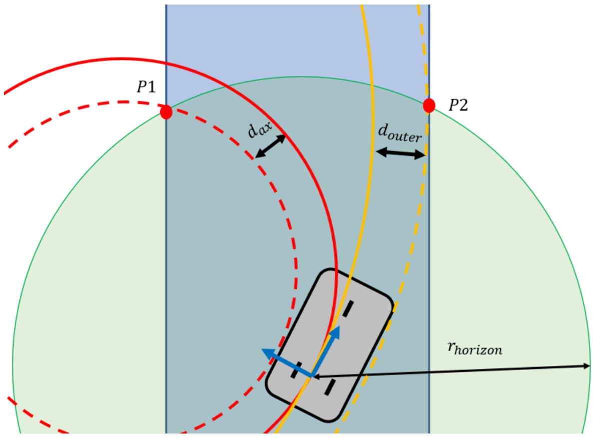

Fig. 9. Hallway situation 2. Trajectory range based on passing $P1$ on the concave side, and $P2$ on the convex side of the trajectories.

2) Junction entry: When the front wheel of the robot enters a junction route polygon the navigation strategy starts with the junction strategy. From the interpretation of the route polygons it is known which direction a turn is headed, and thus what the inner and outer edges of the turn are. Determining the trajectory that avoids colliding with the outer edges the function to avoid a line is used. This is done for the outer edges of the junction, but also the connecting edges of the previous and following hallway polygon as shown in 10, to get a smooth corner exit.

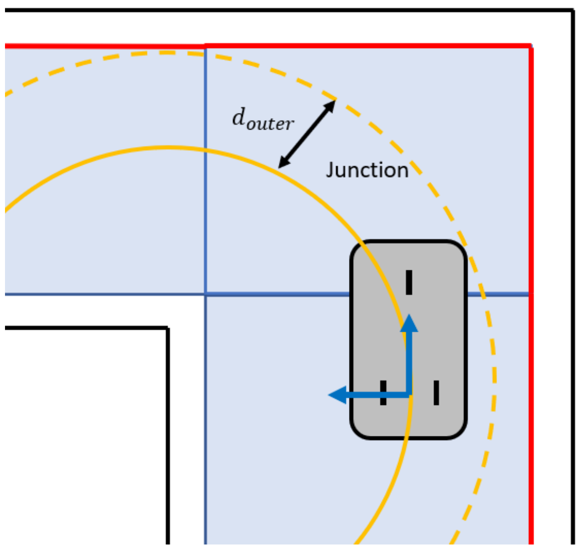

Fig. 10. Trajectory corresponding to minimum required steering angle to avoid the junction outer wall lines shown in red.

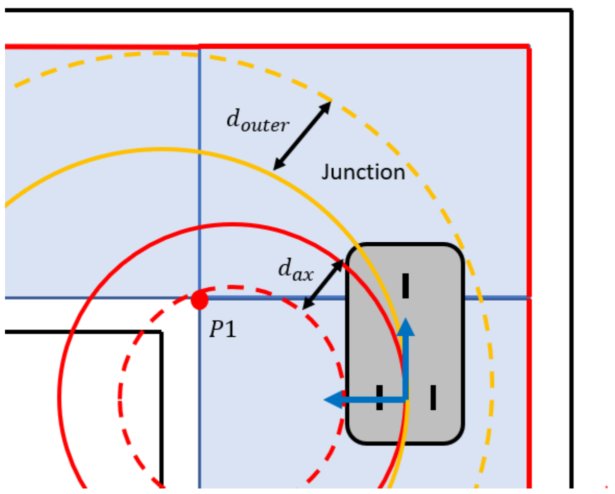

Fig. 11. Trajectory corresponding to the steering angle avoiding inner corner point $P1$.

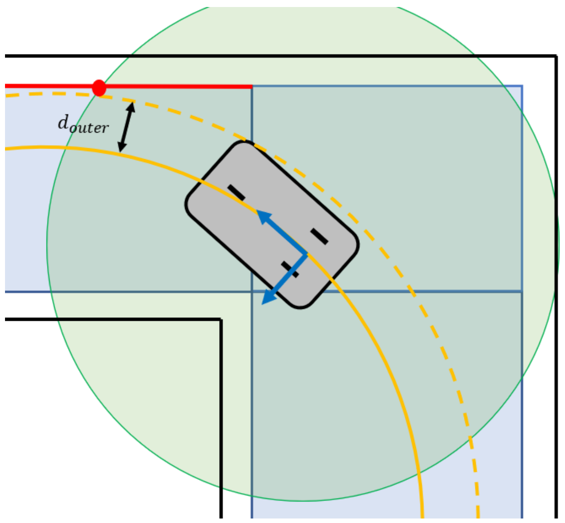

Fig. 12. Using the intersection point of the physical horizon and hallway edge connecting to the outer wall of the junction polygon for junction exit strategy.

For avoiding the inner edges the inner corner point is used in the function to avoid this point on the concave side of the trajectory. The intersection point of the horizon with the following hallway route polygon is also used with this function. Whenever the latter generates a steering range with the steering angle avoiding the outer edges the steering angle in the center of this range is chosen for a smooth junction exit. If this does not provide a steering angle range, but the trajectory based on avoiding the inner corner point does, then this range is used and the steering angle in the center of this range is chosen again. Another possibility is that both situations do not provide a steering angle range. In this case the steering angle corresponding to avoiding the inner corner point is used, as the steering angle based on the outer edges is more prone to conservatism. Predictions provide answers in whether there is a feasible path for the robot.

3) Junction exit: The case where the front wheel is in a hallway polygon, but the center of the rear axle is in a junction polygon requires a slightly different strategy than the other two cases. As the robot is still partially in the junction polygon the robot might still require a turning motion to smoothly fully enter the hallway polygon. For the hallway edge connecting to the outer edges of the corner the normal hallway approach of case I is used. To avoid the inner corner and connected hallway edge the average steering angle of both the junction as in case II and hallway as in case I will be taken. This is, until the trajectory based on the hallway edge connected to the outer wall passes the intersection point with the horizon on the concave side, then the normal hallway approach of case I will be used regardless if the center of the rear axle is still in the junction polygon.

F. Elemental areas

1) No-enter: The most important elemental area in this project is the no-enter area. This area indicates places where the robot may not go. They can be placed by the user to indicate off-limit places for the robot, and can indicate obstacles on its route. When a no-enter area enters the horizon it is determined first if it is within the route polygons. If this is not the case they are deemed not relevant and will be ignored.

If the no-enter area is within the route polygons it will roughly determine if it is physically possible for the robot to pass on either the right or the left side of the area, based on the width of the robot and the minimal clearance around the no-enter area. It will do so by first checking the minimum distance between the edges of the route polygons and each relevant no-enter area. Next, the minimum distance between

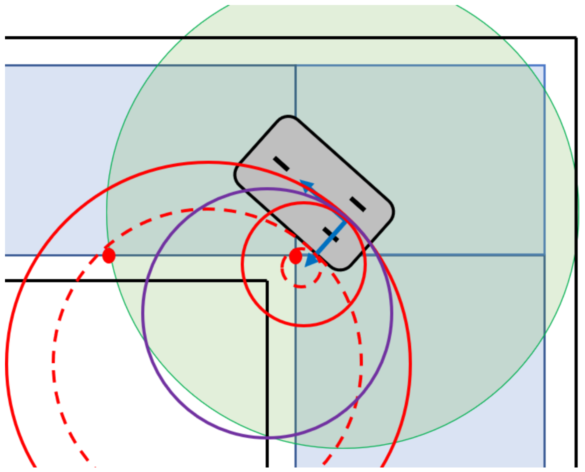

Fig. 13. Using the intersection point of the physical horizon and hallway edge connecting to the inner wall corner point to determine to compose intermediate maximum steering trajectory shown in purple.

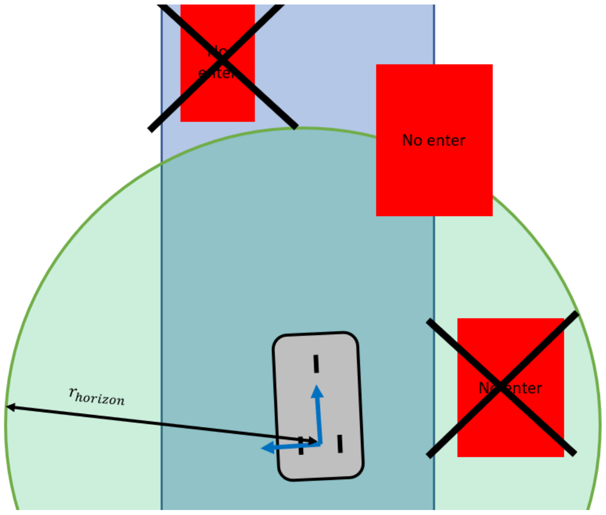

Fig. 14. No-enter areas filtered based on being inside the route polygon and physical horizon.

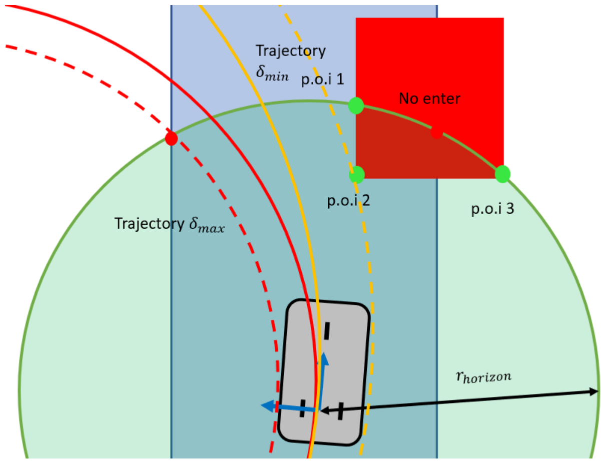

Fig. 15. Trajectory ranges with no-enter area in horizon. Trajectory corresponding with minimum steering angle based on 'no-enter' area.

no-enter areas is checked to see if areas are too close together. Next, the points of interest for the areas that are still passable on either the left or right side are determined. These points consist of the corner points of the polygon, and the points at which the polygon intersects with the horizon. With the knowledge about which side the robot could pass the area and the points of interest, the core functions can be used again. Based on the orientation of the robot with respect to the points of interest and knowing whether to pass the area on the left or the right, the corresponding steering angles for avoiding these points can be determined. This in combination with the steering angles determined based on the route polygon then can provide a new range of steering angles to pass the no-enter area while staying in the route polygons.

2) No-stop: No-stop areas are areas in which the robot may not stand still, or may no longer reside than a certain amount of time. This can apply in several situations. For example, blocking areas around doors or emergency exits is undesired. A solution would thus be to place a no-stop areas in the provided map along with the route polygons. Another option could be to retrieve this information at runtime from a perception module, such that no-stop areas can be placed in the environment at runtime. Also, the robot should not block junctions to avoid congestion of traffic. The approach for dealing with this area is fairly simple. When a prediction detects that the robot goes into a no-stop area a timer will start. The prediction will be extended until the prediction exits the no-stop area and determines that there is enough free space after this area, or the timer indicates that the robot is spending too long in the no-stop area, and thus is likely to cause too much congestion. When the latter is the case, the robot will come to a halt before entering the no-stop area. Predictions will keep being performed. When the predictions show that the robot can pass through the no-stop area within the prescribed time, the robot will start moving again. This area thus only affects the velocity of the robot, and does not influence the steering angle. There are however situations where altering the steering might be necessary. When a no-stop area is so large that the robot can never actually pass it within the prescribed time perhaps another strategy should be chosen. Perhaps its steering angle should be changed to guide it around the area, or it checks whether its velocity remains above a certain value such that it keeps making progress through the area, and it is thus not blocking the area. Ideally when the robot comes to a halt before a no-stop area it would also do so in a strategical way, where for example it will move closer to a wall to avoid congestion by blocking the way before a no-enter area. These alternative approaches are however not taken in to account in this project.

3) Avoid: The 'avoid' type elemental area is an a soft constraint. It is not prohibited to enter this area like a 'no-enter' area, but it is preferred to not enter this area. It is one of the building blocks of specifying higher order semantic behavior. When the traffic rule 'drive on the right hand side'

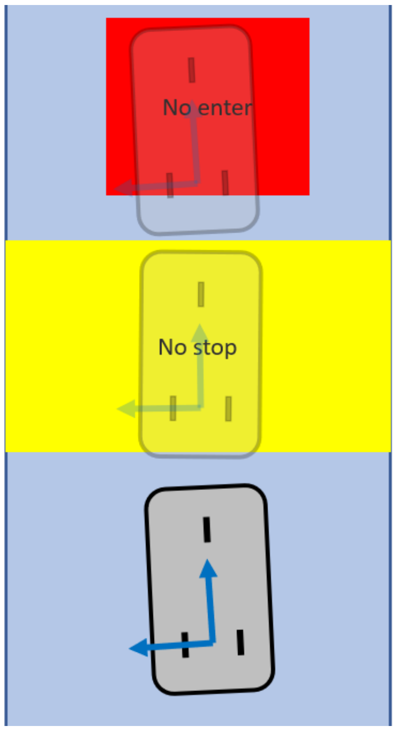

Fig. 16. Example situation with 'no-stop' area such that the robot can not pass the area due to the 'no-enter' area.

must be expressed in elemental areas this would mean placing an 'avoid' area on the left hand side. It is not forbidden to drive on the left, but it is preferred to drive on the right whenever possible. Due to time limitations the 'avoid' type elemental area is not available in the current implementation. Further details on a proposed strategy for the 'avoid' area can be found in Appendix V-A.

G. Horizon

Up until now we have assumed the physical horizon size to be a given. However, if for some reason a situation causes there to be no available steering range, it might be necessary to reduce the horizon as a shorter view could provide a feasible steering angle range. The physical horizon size goes hand in hand with the velocity of the robot. Intuitively, the faster the robot goes the further it should look ahead when determining its control input. The question then becomes how to determine this size. There are a couple of factors that contribute to the answer. We can start off by providing some lower and upper bounds. First, the *absolute* minimum size of the horizon is such that it covers the entire footprint of the robot. This is to ensure that the robot always considers the local information to be able to determine if it can rotate around its rigid axle without collisions. Secondly, as mentioned the velocity and horizon size are connected. However if the velocity is zero that must not mean that the horizon is also zero. Therefore, the *desired* minimum horizon size is defined as the horizon size covering the footprint, plus the distance it takes for the robot to get to its desired cruising velocity from a stand still. This makes sure that the robot can determine a steering angle that would theoretically guide onto a trajectory at which it can reach its desired velocity. Another *absolute* lower bound is covering the front end of the robot plus the distance it takes the robot to stop in an emergency situation. the robot should never determine a steering angle of which it is not sure if that steering angle results in an inevitable collision. A *desired* upper bound is defined by the distance it takes the robot to come to a halt with a user specified nominal deceleration. For smooth motion of the robot it should not make use of its full deceleration capabilities at all times when it slows down.

1) Scaling horizon: Some situations might require the physical horizon to be scaled down from its desired size. An example of this situation is given in 17 and 18. Figure 17 shows that the trajectories corresponding with the minimum steering angle based on $P3$ and maximum steering angle based on $P1$ do not form a safe range of trajectories in that situation. However, 18 shows that with a scaled down horizon there is a feasible range of steering angles. The scaling thus happens based on the no-enter areas in the horizon. It looks for the closest point on a no-enter area as reference for its new horizon, such that the no-enter area is still considered, but the reduced horizon possibly allows for a larger trajectory range. When multiple no-enter areas are in the horizon this scaling happens based on the furthest no-enter area, and can consist of multiple iterations until a suitable horizon size is determined, or it is determined that there is no suitable horizon size at all. When this scaling of the horizon occurs the velocity also should be adjusted accordingly. The velocity should decrease such that it matches to the *desired* horizon size.

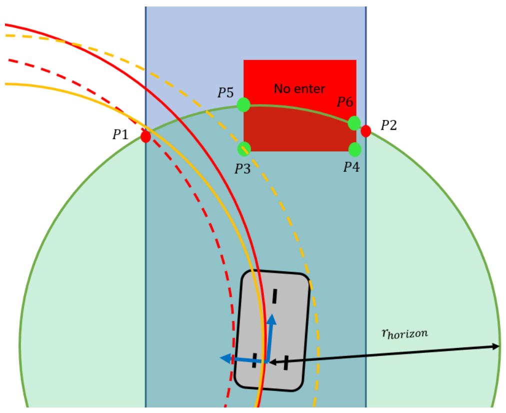

Fig. 17. Example situation. Physical horizon does not provide a feasible steering angle range based on points $P1$ and $P3$.

H. Predictions

As mentioned, the proposed strategy makes use of predictions. It does so to determine its velocity in a safe manner. Theoretically, by design the method would not require predictions in certain situations as the determined steering angle range would imply a set of safe trajectories for the distance specified

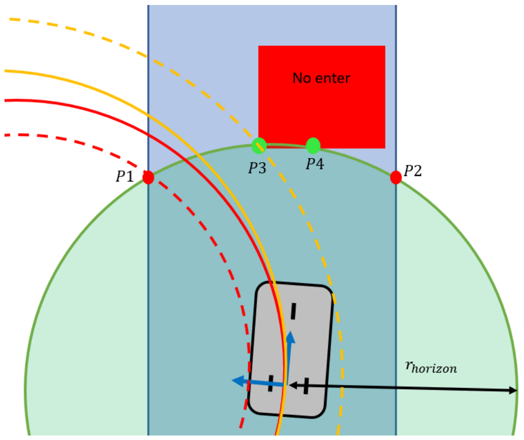

Fig. 18. Example situation. A scaled down horizon does provide a feasible steering angle range based on points $P1$ and $P3$.

by the physical horizon. However due to some situations requiring some compromise on the guarantee of no collisions within the physical horizon it is beneficial to use predictions to check if the determined control input actually leads to a safe trajectory. Each iteration of the navigation algorithm a new prediction is performed.
Each time step of the prediction the new robot pose is simulated. Based on this new pose the steering angle ranges are determined again and a desired steering angle is chosen. A check is performed if the robot is exiting the route polygons, colliding with no-enter areas, and if it has entered a no-stop area. The time horizon of this prediction should always be at least equal to the time it would take for the robot to come to a full stop using its maximum deceleration. This way it can be avoided that the robot comes in a situation where even with full deceleration it is unable to avoid a collision with a static obstacle. Longer time horizons could be used to be able to anticipate earlier on possible collisions, or based on a desired deceleration instead of maximum deceleration. This can result in more smooth driving of the robot. Another application of the predictions is predicting if a no-stop area is entered, as mentioned in the previous section. This allows the robot to come to a halt before actually entering the no-stop area. Finally, when a prediction is performed, the predicted control input can also be stored and reused in the next time step. This way the navigation only needs to check if with that predetermined control input it still complies with the specified constraints, which can reduce the required computational effort.

III. Results

The proposed method has been implemented as a ROS package, written in C++. Experiments have been performed on the Cura platform. First the result with respect to the requirements will be discussed. Next, the simulations will be validated through experiments. Finally the issues that have come up throughout this process will be discussed.

A. Requirements

Several scenarios have been tested in simulation to check whether the requirements stated in the introduction are met.

1) Dealing with semantic information: The first requirement is that the navigation strategy must be able to deal with semantic information. The representation of this information in the form of route polygons and elemental areas has been tested. Figure 19 shows a test scenario in which the robot finishes the route formed by the route polygons without elemental areas. The ’hallway’ and ’junction’ types are distinguished and acted upon in a logical fashion, where in junction polygons the trajectories are determined in the direction the route is headed after the junction polygon. No elemental areas are present and the horizon is static. A link to the GIF of the simulation can be found in Appendix V-B. It shows that the navigation strategy maneuvers the robot around the course.

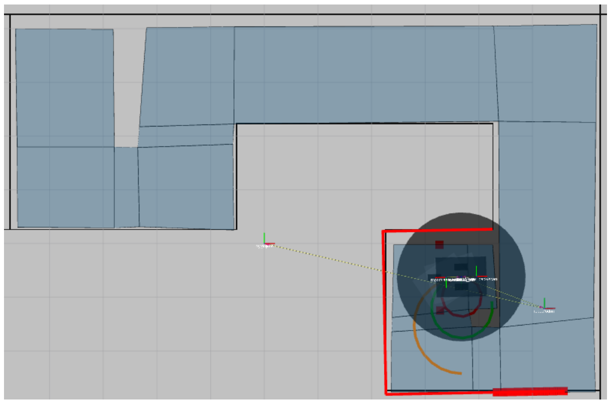

Fig. 19. Simulation scenario testing route polygons. Link to GIF in Appendix V-B

Next, the application of the elemental areas is simulated. The scenario in 20 shows placement of no-enter areas and a no-stop areas. It is expected that the robot should stop before the no-stop area as the no-enter area behind it blocks the way and thus leaves the robot no space. Once the blocking obstacle is removed the robot should start moving again. A GIF of the simulation shows that the strategy correctly identifies the points on no-enter areas that it needs to consider to determine the control input. The scenario shows the capability to pass a no-enter area on the left, and pass in between two no-enter areas. It also shows that the robot does indeed extend its predictions once it encounters the no-stop area, and stops before the area until the blocking obstacle is removed. The horizon is static.

2) Changing footprint: The second requirement is that the method must work for different footprints. As the strategy is based on the parameters of the robot it does work for different footprints. Therefore the method is suitable for different type of platforms, and the footprint can be changed whenever the

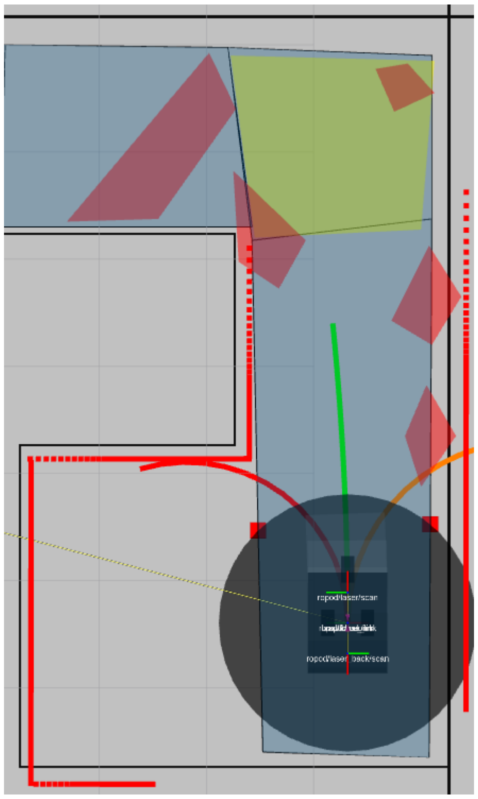

Fig. 20. Simulation scenario with yellow 'no-stop' area and red 'no-enter' areas. Link to GIF in Appendix V-B

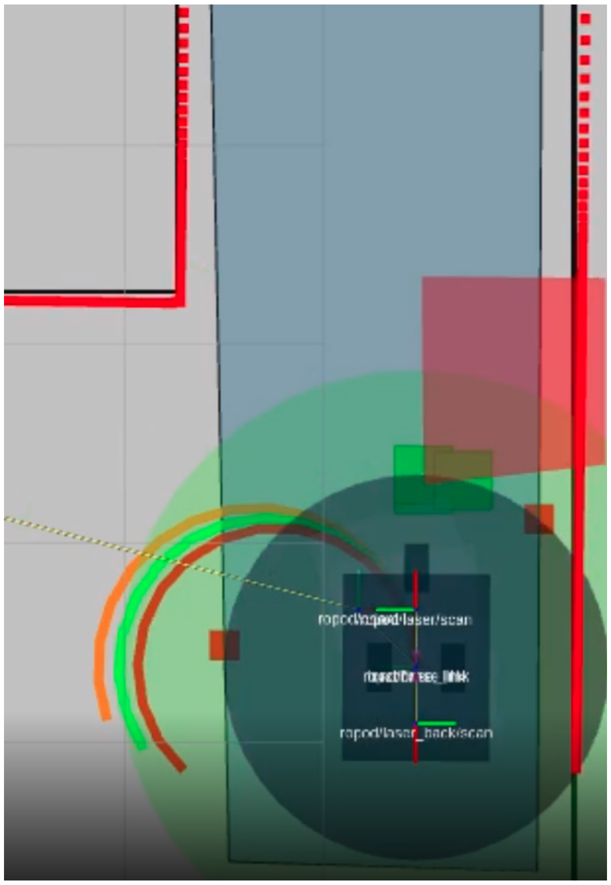

Fig. 21. Initial horizon highlighted as green circle found no feasible trajectory range. Re-scaled horizon represented as dark grey circle does find a solution. A Gif in Appendix V-B shows the simulation of this situation.

robot docks to a cart. GIFS listed in Appendix V-B show the scenario of 19 for several footprints. the horizon is static.

3) Explainability of robot actions: As a result of the method, each action of the robot can be traced back to a specific function or decision. The visualization in rviz provides visualization of the perceived environment, route polygons, elemental areas, and determined steering angle range. Relevant features on which the robot bases its actions can also be visualized to make it clear what the current action of the robot is based on. The output to the console can be modified such that the user can receive desired information about what the current intent is or possible future intent of the robot will be. This also allows for further expressing of the intentions via audio or visual clues from the robot.

4) Making progress towards goal: By defining the route polygons in which the robot has to stay some conservatism is introduced. Also, as the method does not use a standard optimization strategy, but rather calculates the control input in a deterministic manner based on the geometry and semantics of the environment some additional conservatism is introduced. In order to make progress towards the goal in as many situations as possible the scaling of the physical horizon is of importance, as a certain static horizon does not always provide a solution. Figure 21 shows a situation in which the initial desired horizon does not provide a solution, however when the physical horizon is scaled appropriately it is able to find a solution.

5) Do not collide with obstacles: The obstacle collision part goes hand in hand with the elemental area part. An algorithm is used to convert laser scan data into polygons which receive the 'no-enter' label. Therefore, as the robot is able to avoid and/or stop in time before these no-enter elemental areas it is also capable of avoiding collisions on its route.

B. Simulation validation through experiments

To validate the simulations with experiments the scenario of 22 has been tested on the Cura platform. The experiment was successful in the sense that the robot manages to execute the task. When the trajectory of the real robot is compared versus the simulated trajectory there is a difference. In the experiment the robot seems to start steering at a lower rate when entering a junction polygon, and overshoot its steering compared to the simulation. Reasons for this could be that the steering rate in the kinematic model differs from the actual steering rate of the robot. Another issue that could contribute to this difference is that the robot requires some threshold value for the wheels to start moving. Also a slight difference in starting position could lead to differences in the trajectory.

The GIF of the rviz visualization of the experiment is listed in Appendix V-B, along with a video of the real life situation.

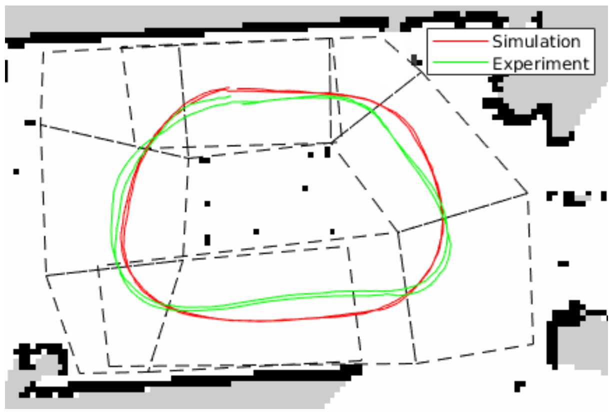

Fig. 22. Trajectories of simulation and experiment for a testing scenario. Structured environment is represented as solid black dots. The task for the robot consists of two halves of a route, indicated as a series of polygons represented with the dashed lines.

C. Alternative prediction usage

The basic implementation of the strategy runs the prediction every iteration of the navigation algorithm. In order to reduce the computational effort, and go more towards a 'lazy' navigation strategy it might be beneficial to use the prediction data of the previous iteration. The control input during each prediction step is stored. Then, next iteration of the navigation algorithm this set of previously determined control input is used in a prediction to see whether the prediction still satisfies the imposed constraints. If so, the next control from the prediction of the previous iteration can be applied without without redetermining this control input. The comparison is done in simulation without elemental areas. The behavior in both cases is the same as the kinematics model used in the prediction is the same as to model the actual robot. The prediction time horizon is 2 seconds. The maximum velocity is set to $0.5m/s$ and acceleration/deceleration is limited to $0.5m/s^2$, meaning the robot requires at most 1 second of prediction info to avoid collisions. The navigation algorithm and predictions run at 10 hz. This means that each prediction contains 20 time steps and control inputs, of which theoretically 10 can be applied before needing to recalculate the prediction. Table I shows the average computation time for different usages of the predictions in simulations of the scenario shown in 22. This shows that the required computational effort is roughly 6 times lower when reusing the prediction data. The computational benefit depends on velocity and acceleration limits, as well as the prediction time horizon and disturbances on the system. A more meaningful test would be to compare the average calculation time and behavior on the Cura platform such that the predictions are not perfect, and real world disturbances can be applied. Unfortunately due to time constraints this has not been done yet.

Strategy	**Average calculation time [s]**
Predict each iteration. Time horizon = 2s	3.39e-03
Only make new prediction when necessary. Time horizon = 2s	5.76e-04

TABLE I
AVERAGE CALCULATION TIME FOR DIFFERENT USAGES OF THE PREDICTION DATA ON 'HP ZBOOK STUDIO G3' RUNNING UBUNTU 20.04.

D. Comparison Napoleon navigation

As mentioned in Section I-D the Napoleon navigation strategy was build on roughly the same intentions, however the actual strategy lacked robustness. This also shows in the case of a route without obstacles as in 22. The parameters and settings are as much as possible the same. Napoleon navigation does however not succeed to complete the assignment. It goes outside of the route polygons and gets stuck.

E. Encountered issues

There are also some drawbacks with the current implementation of the strategy. One example is that in hallway situations where the robot is really close to or just overlapping with a route polygon edge the core function based on a line does not provide a feasible solution. In that case it determines the steering angle based on the intersection point of the horizon and route polygon edge. This does however result in a lower or upper bound trajectory that goes slightly outside of the route polygon. Thus the current implementation is not able to guarantee that the robot does not go partially outside of the route polygons. This creates a safety issue as it does not consider obstacles outside of the route polygons in its collision prediction. Also considering those no-enter areas outside of the route polygons can however provide a solution. The approach of certain situations should perhaps be reconsidered in order to increase the safety guarantee based on the determined trajectory range.

Another issue with the current implementation is when the horizon does not intersect with a hallway edge. The behavior in this situation is not defined correctly yet. Theoretically the steering range would go to its maximum as no collisions are foreseen, but from a navigation perspective this is not a great solution, as you want some way to know if you are making progress towards the goal. In this case the horizon could be enlarged to get intersections with the route polygon edges. When the horizon is scaled down due to no-enter areas perhaps some kind of directional information can be extracted from the orientation of the route polygon.

Also, the behavior when encountering a 'no-stop' area needs to be defined further. Once the robot enters the area, but predicts a collision in its extended prediction it will come to a halt in the area. There are several factors that can contribute to this decision making process, for example the percentage of the footprint of the robot that is in the area, or if there is a more strategic way to decide which stopping position inside the area is more preferable.

IV. Discussion and conclusion

This section will provide the conclusion of the thesis, along with a discussion and recommendations for future work.

A. Discussion

Simulations and experiments have shown that the proposed strategy is capable of moving a robot through a specified route. The method relies on identifying several situations that the robot can be in, and apply the core functions in an appropriate manner. There are however many situations that can be distinguished, and certain situations could be dealt with by considering different elements of the environment. Due to the time constraint on the project not all situations are dealt with such that safety is guaranteed by the determined trajectories, but some rely on predictions to check this safety. Ideally the safety would already be guaranteed by the determined steering angles as much as possible to reduce the need of predictions. Since the chosen method is constructed around a set of core functions which determine the control input, the method is also limited by the properties these functions. Some situations might cause the method to not find a suitable control input, resulting in the robot getting stuck, while other methods like MPC might be able to find a suitable control input for the robot to progress further towards its goal. Furthermore, if the robot goes outside of the route polygons for any reason it will predict that it is colliding and therefore get stuck, even though there might not be a physical wall or obstacle near the robot. As long as the route polygons are placed with the intention of not overlapping with walls or obstacles this introduces a degree of conservatism.
In experiments the robot is more likely to collide with the inner corner point of a junction route polygon. This could be caused by the localization not being perfect, especially when making a rotating motion. The proposed strategy often guides the robot fairly close by this inner corner point which can thus cause issues.

B. Conclusion

This thesis proposed a new navigation strategy which uses elemental areas as building blocks for higher order semantic behavior, and an assignment provided in the form of route polygons. The current implementation is capable of dealing with 'no-enter' and 'no-stop' areas.
The developed navigation strategy is capable of determining the control input in an explainable way. Due to the chosen method and usage of semantic information about the environment the strategy can distinguish many different situations and provide clear output for the user to see on why a certain control input is chosen.
The method is intended to get the robot from a starting area to a finish area, and thus not to a specific point or pose. U-turn maneuvers to obtain the correct initial heading are also not included.
The navigation strategy is developed for (semi-)structured environments where the robot is able to determine its pose. Simulations and experiments have shown promising results of applying this method in those environments.

C. Recommendations for further work

The proposed strategy has shown its first signs of success, however there is still plenty of work to be done. Some of the possible future work is provided in this section.

1) One of the major parts that should be added to the strategy is dealing with the 'avoid' elemental area type. This is crucial in being able to let the robot follow certain desired behavior like driving on the right by placing an avoid area on the left side of a hallway. This elemental area type along with the no-enter area type forms the building blocks of describing higher order semantic behavior.
2) Overall the method still requires refinement. Solutions to situations where within the horizon there are no clear points to base the control input on, or situations where the robot has to make a u-turn have to be defined. Also, additional decision making and possible reverse motion when the robot gets stuck or comes to a halt in a no-stop area is not incorporated yet.
3) Further research can be done into the effect of placing route polygons in such a way that it overlaps with the physical structure. This way the physical structure can be treated as no-enter polygons. This approach might reduce conservatism with respect to placing the route polygons to not overlap the physical structure.
4) Further work can also be done on making this navigation method compatible with the work of [12]. This requires the interfacing between the world model representations to be matched.
5) Finally, research can be done into expressing new desired behaviors in elemental areas, and if perhaps new types of areas are required for certain behaviors which then would need to be added to the navigation strategy.

References

[1] A. Sharma. "The mobile robot market in 2022 – our predictions." (), [Online]. Available: https://www.interactanalysis.com/the-mobile-robot-market-in-2022-our-predictions/.

[2] S. v. O. Martijn Driessen. "Verpleegkundigen bezorgd over tekort ic-personeel na anderhalf jaar pandemie: 'je bent bang dat de patiënt je door de vingers glipt'." (), [Online]. Available: https://eenvandaag.avrotros.nl/item/verpleegkundigen-bezorgd-over-tekort-ic-personeel-na-anderhalf-jaar-pandemie-je-bent-bang-dat-de-patient-je-door-de-vingers-glipt/.

[3] NOS. "Onderzoek: Tekort aan zorgpersoneel op lange termijn alleen maar groters." (), [Online]. Available: https://nos.nl/artikel/2413851-onderzoek-tekort-aan-zorgpersoneel-op-lange-termijn-alleen-maar-groter.

[4] N. Correll, *Introduction to Autonomous Robots, v1.9.* Magellan Scientific, 2020, ISBN: 978-0692700877.

[5] 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.

[6] Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, 1398–1404 vol.2. DOI: 10.1109/ROBOT.1991.131810.

[7] “Timed elastic band.” (), [Online]. Available: http://wiki.ros.org/teb_local_planner.

[8] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *Robotics Automation Magazine, IEEE*, vol. 4, pp. 23–33, Apr. 1997. DOI: 10.1109/100.580977.

[9] T. Mercy, W. Van Loock, and G. Pipeleers, “Real-time motion planning in the presence of moving obstacles,” in *2016 European Control Conference (ECC)*, 2016, pp. 1586–1591. DOI: 10.1109/ECC.2016.7810517.

[10] S. C. Brailsford, C. N. Potts, and B. M. Smith, “Constraint satisfaction problems: Algorithms and applications,” *European journal of operational research*, vol. 119, no. 3, pp. 557–581, 1999.

[11] M. de Wildt, *Tube driving mobile robot navigation using semantic features*, Master thesis, Eindhoven University of Technology, 2019.

[12] H. L. Chen, B. Hendrikx, H. Bruyninckx, and R. van de Molengraft, “Behavior adaptation for mobile robots via semantic map compositions of constraint-based controllers,” *Frontiers in Robotics and AI*, 2022.

V. Appendix

A. Elemental area 'avoid'

the same principles as with the no enter areas can be used when encountering avoid areas. One of the differences will be that it is no hard constraint. Also, a different situation might occur where the robot is actually inside an avoid area, unlike no enter areas where the robot by design should never be. This thus requires an additional method of determining the control input in that specific situation.
Also, key part of decision making when encountering avoid areas is being able to determine how long the robot spends in avoid areas, as the robot might have several options for the control input, and this can play an important role in the decision-making which control input is preferred.
A method for this can be to discretize the circular trajectory. Knowing the radius corresponding with a certain trajectory, the pose of the robot, and thus the center point of the circular trajectory can be used. Determining for a certain number of points inside the horizon in which type of areas these points reside can provide the information needed for the decision making.
Another possibly better option is to determine exactly the intersection points of the trajectory with the edges of the avoid polygon (if there are any) to determine the distance in each region in a more precise way, avoiding discretization and thus a tuning parameter for the discretization size.
When the robot is moving slowly and the horizon might be relatively small it might make the wrong decision when needing to enter an avoid area if there are for example two ways to go, but one way would lead the robot through a much larger avoid area, which it is unaware of due to the size of the horizon. Predictions might offer a solution here, as the robot moves forward in the predictions, and thus the horizon also goes further than at the current situation. Considering the total time/distance spent in avoid areas during prediction could thus offer useful information for the decision making.

When the robot is not inside an avoid area the same strategy as with no-enter areas can be used to determine a steering angle range which would possibly avoid the avoid area completely. If this additional restriction of the steering angle range causes the base steering angle range from the route polygons and no-enter areas to disappear this forms a challenge. One could perform the same strategy with the reduction of the horizon, but there are situations where going through avoid areas is unavoidable, so this would not be a generalized working strategy.

Possible solution
As avoid areas are basically 'soft constraints' it might be an option to just take the steering angle ranges from the route polygons and no-enter areas as absolutes and based on the time/distance those trajectories/predictions spend in avoid areas shift the actual desired steering input towards one of the bounds, instead of going somewhat through the middle.

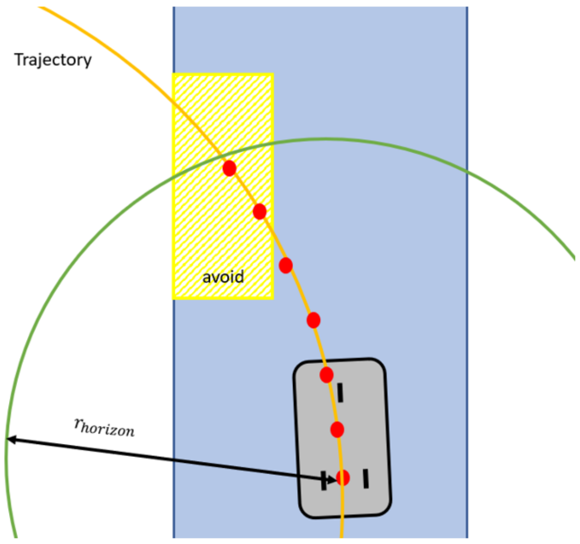

Fig. 23. 'Avoid' area example 1

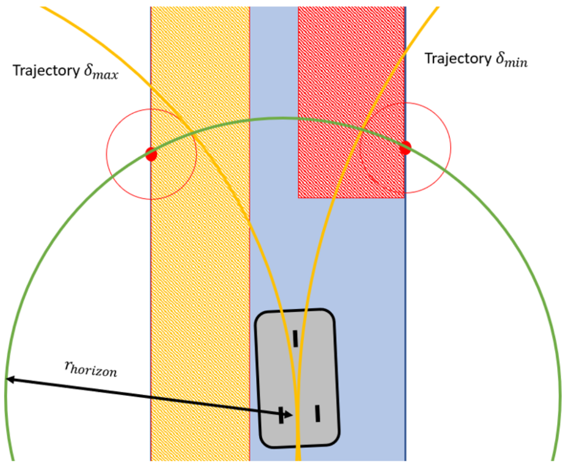

Fig. 24. 'Avoid' area for driving on the right hand side. The situation shows why it has to be treated as a soft constraint

Figure 25 shows an example of a situation where the trajectory corresponding with $_{min}$ is preferred, as this has the largest portion of the trajectory within the horizon outside of the avoid area. This approach on its own would however not change the horizon/steering angle range based on the fact that it is inside this avoid area, while this might be preferable to exit the area more quickly.
Another solution could be to perform somewhat the same strategy as with no-enter areas, even when the robot is actually inside an avoid area. Figure 26 shows such a case. $'_{max}$ is placed in such a way that the robot would be guided out of the avoid area. Of course the clearance circle around the p.o.i. now does not hold as much meaning as no actual collision needs

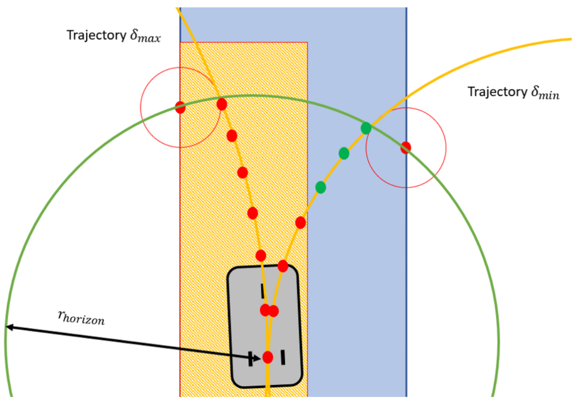

Fig. 25. Robot being inside 'avoid' area. The trajectory to the right has the least distance inside the 'avoid' area.

to be avoided. This does however provide a more generalized working method, as when the robot is outside of the avoid area the clearance circle does hold a meaning again.

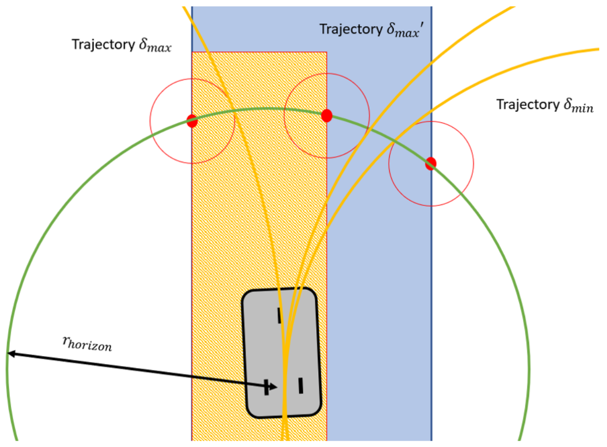
Trajectory δ_{max}
Trajectory δ_{max}'
Trajectory δ_{min}
$r_{horizon}$

Fig. 26. Determining points of interest on the 'avoid' area for more knowledge about exiting the area.

Figure 27 shows a case where there is no intersection of the avoid area with the horizon, so a point on the avoid area polygon has been taken as p.o.i.. This would guide the robot to the right of the avoid area. However, in some cases it might be more beneficial to just keep driving forward, as the robot will exit the avoid area there as well, perhaps even quicker, and with less disruption of the trajectory.

B. GIF links

Link to media:
https://tuenl-my.sharepoint.com/personal/c_a_lopez_martinez_tue_nl/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fc%5Fa%5Flopez%5Fmartinez%5Ftue%5Fnl%2FDocuments%2FStudents%2F2022%2FRinse%20H%2FRinse%27s%20thesis%20media&ga=1

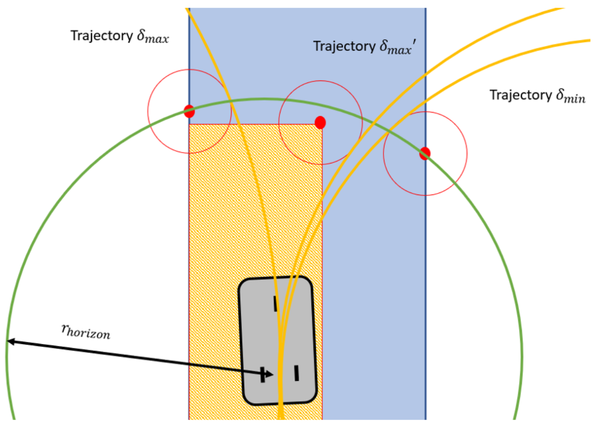
Trajectory δ_{max}
Trajectory δ_{max}'
Trajectory δ_{min}
$r_{horizon}$

Fig. 27. Same situation as Figure 26 but with different horizon. Importance of points of interest change.

Description of media:

- *route_polygon_result_speedx2.gif*: GIF of simulation scenario shown in Figure 19. Replay speed of the GIF is 2x the original speed.
- *elemental_areas_result.gif*: GIF of simulation scenario shown in Figure 20.
- *route_polygon_result_lwb_speedx2.gif*: GIF of simulation scenario show in Figure 19 with enlarged robot footprint. Replay speed of the GIF is 2x the original speed.
- *route_polygon_result_swb_speedx2.gif*: GIF of simulation scenario show in Figure 19 with short robot footprint. Replay speed of the GIF is 2x the original speed.
- *result_horizon.gif*: GIF of simulation scenario shown in Figure 21.
- *impulse_demo_rviz.mp4*: MP4 screen recording of rviz while performing an experiment on the Cura platform in the environment of the scenario of Figure 22.
- *impulse_demo_video.MP4*: Video footage corresponding with the rviz visuzlization of *impulse_demo_rviz.mp4*.