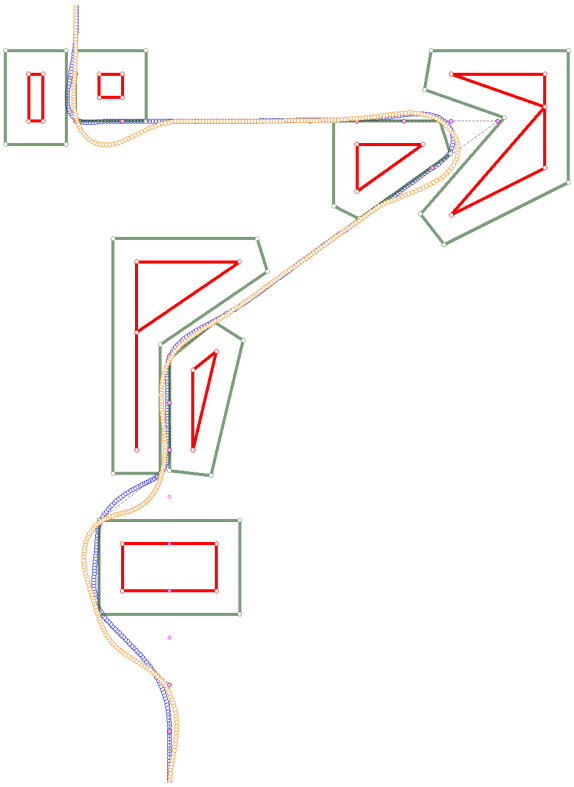


Trajectory Generation for Multiple Autonomous Transport Robots using Nonlinear Model Predictive Control

Master's thesis in Systems, Control and Mechatronics

Valdemar Krona
Noa Lindén

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Trajectory Generation for Multiple Autonomous Transport Robots using Nonlinear Model Predictive Control

Valdemar Krona
Noa Lindén

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Trajectory Generation for Multiple Autonomous Transport Robots using Nonlinear Model Predictive Control
Valdemar Krona
Noa Lindén

Supervisor: Per-Lage Götvall, Volvo Group
Examiner: Knut Åkesson, Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Trajectory planning for follower and leader in an environment with static and unexpected obstacles.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Trajectory Generation for Multiple Autonomous Transport Robots using Nonlinear Model Predictive Control
VALDEMAR KRONA
NOA LINDÈN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis proposes a Nonlinear Model Predictive Control (NMPC) approach to drive and carry objects between two Autonomous Transport Robots (ATRs) in a factory environment. Considering static obstacles and a pre-defined node network, a rough global path to goal is generated using visibility graphs and the A* algorithm. This path is used as a reference path when calculating a trajectory for one or two ATRs. The trajectory avoids static and dynamic obstacles such as forklifts, humans and pallets while keeping a fixed or minimum distance between the ATRs. Calculating the trajectory is done using NMPC using Proximal Averaged Newton-type method for Optimal Control (PANOC). The proposed behavior of the ATRs is adjustable by adjusting the weights in the NMPC formulations, and is able to change between safe and more aggressive behavior.

Obstacles are modelled using polygons. In the NMPC formulation convex constraints can be handled, thus the polygon is split into multiple triangles using constrained Delaunay triangulation resulting in a set of convex constraints. Formation control is done using a leader follower approach. First a trajectory is generated for the leader. It is then modified for the follower in such a way that the the trajectories are at a fixed distance apart at each time step.

The approach is evaluated through simulations and using physical ATRs by simulating a set of challenging scenarios. Most scenarios are handled very well by the proposed method, but for large dynamic obstacles the optimization problem might, in some situations, fail in finding a trajectory avoiding the obstacle. Also in scenarios when the leader has to adapt a lot to the follower it struggles. The physical evaluation support the evaluation done using simulations.

Keywords: NMPC, PANOC, formation control, trajectory planning, OpEn.

Acknowledgements

We would like to thank Chalmers University of Techonology and Volvo Group for allowing us to do this master thesis. It has been a spring of struggle but also of fun. Thank you Knut Åkesson and Per-Lage Götvall for your help and your passion, it has been inspiring. Thank you Dean Emmanuel for your help with ROS. We would not have gotten the communication running as well without you. Finally we would like to thank our friends who did sister master thesis for being so great to work with. Thank you!

Valdemar Krona & Noa Lindén, Gothenburg, June 2021

Contents

List of Figures

List of Tables

Glossary

ATR	Autonomous Transport Robot
MPC	Model Predictive Control
NMPC	Non-linear Model Predictive Control
PANOC	Proximal Averaged Newton-type method for Optimal Control
Path	A set of coordinates
Trajectory	A set of coordinates with time attached to each coordinate
A*	A-star algorithm.
k	timestep k

1
Introduction

This chapter introduces the thesis. Some background as to why the thesis was performed as well as a short description of the problem is given. After that the purpose and research questions are described as well as some related works. Finally the limitations of this thesis are discussed.

The problem of having several robots working together is not new. Various robots in production environments have done that for decades. This has been done with a great deal of manual labour. Humans have made sure that the robots could not interfer with each other. For example, programmers have made sure that trajectories of robot arms are such that they do not crash. This yields a solution which is rigid, but it cannot handle disturbances well. Suppose that two robots work closely together. If one of the robots is slightly worn so that it moves slower that it should. This would make it's movement out of sync with the pre-calculated trajectory. This could lead to the robots crashing. If a human would come in proximity to the robot it would have to stop because of safety regards. If the other robot does not account for this it could crash into the stationary one. This thesis aims to solve this issue by synchronising the movements of the robots by using non-linear model predictive control. This results in a flexible system which can handle disturbances such as obstacles well.

1.1 Background

Transportation of material and goods is a key cog in today's society. The same is true in production facilities. Volvo Group has an ongoing research project which aims at increasing the number of automated delivery tasks as well as replacing current autonomous techniques. They want to do this by utilising multiple small robots instead of single big robots. The idea is that the smaller robots could work together to carry big goods. If the object to carry is small then it would be inefficient to have a big robot carry it. Instead a small robot could carry it. If the object instead is big then multiple small robots could work together to carry the load. This principle could be expanded so an almost arbitrarily sized good could be carried by a swarm of small robots.

Volvo Group's current automatic delivery system uses magnetic lines to guide the robots. These lines are burrowed into the floor. This makes the system rigid since the robots have to follow these lines and cannot deviate to avoid potential obstacles.

It is also expensive to install and update these lines. In addition to replacing big robots with small robots Volvo also aims at removing the need for the magnetic lines and instead guide the robots via cameras. This would result in a cheaper system which is both more robust and flexible.

This thesis aims at solving one of the problems which arose from the above and is a continuation of a previous thesis, [1], by Pålsson and Svärling. Their thesis aimed at solving the trajectory planning of two ATRs which were to follow a path while keeping a fixed distance between them. Inputs to their system was a path to follow and the distance to keep. The output was control signals for the motors. This thesis aims at solving both the path- and trajectory planning of two ATRs while keeping a fixed distance between them and avoiding obstacles. The input is a goal location, a distance to keep, nodes which can be searched in order to create a path to the goal and continuously updated locations of any obstacles.

1.2 Purpose

The purpose of this thesis is to solve the problem of path- and trajectory planning of two ATRs while keeping a fixed distance between them and avoiding obstacles in a production enviroment.

1.3 Problem description

In order to fulfil the purpose of this thesis two steps are required, namely path planning and trajectory planning. Path planning should formulate a path from the starting position to the goal position. This should take static obstacles into account so that the path is traversable. It should also consider the boundary of the area the ATRs are allowed to be in as to not plan to move outside of the allowed area.

The second step is trajectory planning. This is the main bulk of this thesis. The trajectory planning should formulate a trajectory which follows the path from the path planner. The trajectory should take the dynamics of the ATRs into account so that the actual physical ATRs can follow the trajectory. This makes it so the trajectory planner has to consider time. In addition to following the path the trajectory should maintain a fixed distance between the ATRs so that a load can be balanced on the ATRs. Since the trajectory planner takes time into account it can avoid dynamic obstacles. By predicting the future positions of dynamic obstacles the trajectory can avoid crashing into these. Dynamic obstacles could force the ATRs off the path and into static obstacles or out of bounds. Therefore the trajectory planner also needs to consider the static obstacles and the boundary.

1.4 Research questions

- **Research question 1**
  Implement a global path planner that finds the shortest path between two points, given a road network. Also implement a complementary local path planner, using visibility graphs, which avoids obstacles along the path. Evaluate controller compatibly and quality of the solution.
- **Research question 2**
  Evaluate how well a PANOC NMPC controller solves the issue of having two ATRs follow a given path while obeying constraints and avoiding dynamic obstacles. Evaluate it with regards to robustness and violations of the constraints in practice compared to simulations. The constraints can be of varying kind, for example a fixed or greater than distance between the robots.

1.5 Related works

In [1], it is shown that it is a feasible solution to use PANOC NMPC controller with the framework from [2] to constrain the distance between two ATRs. The thesis compares different solvers and scenarios including photogrammetry sensor system with and without noise. However, the thesis did not consider path-planning nor obstacle avoidance.

In [3], a novel approach is presented, to collision free, trajectory generation for a single differential drive robot. Using visibility graph and PANOC NMPC controller to generate a trajectory. Obstacles are modelled as circles, polygons are avoided by placing a circle at each vertices. Presents a stable way to keep ATRs moving along a line.

In [2], a novel modelling framework for obstacle avoidance is presented. In this obstacles are represented as sets.

1.6 Limitations

A maximum of two ATRs are considered to move in formation. The robot model is a generalised differential drive model. Modelling of actual hardware is done as a separated module. The robots are considered as a point mass model during the obstacle avoidance.

2

Problem setup

2.1 Complete system

In this section, a overview of the system is described. This thesis aims to be a part of a bigger system, functional as a real world example. Different projects are working together to build the complete system.

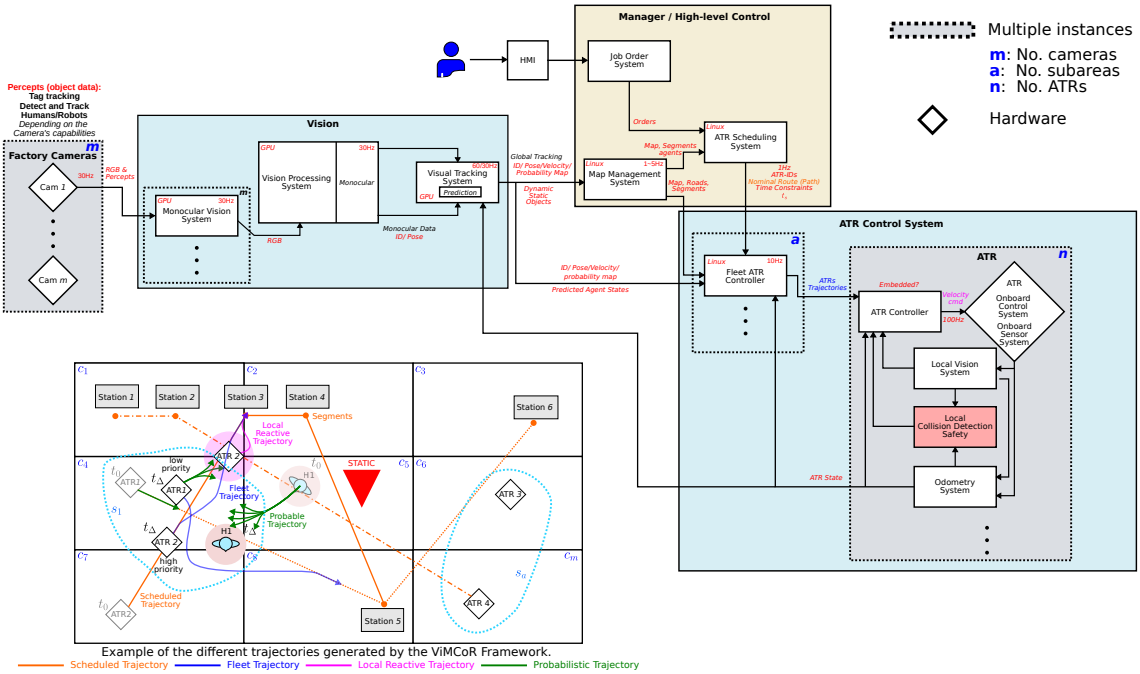


Figure 2.1: Brief overview of the complete system architecture.

Fixed cameras are placed in the ceiling, pointing down, recording the environment, and classifies static and dynamic obstacles and their occupied area as polygons. Each ATR has a unique AprilTag to enable the camera system to keep track of them with a unique ID. AprilTags are also placed on pillars spread across the factory.

Modified Husqvarna autonomous lawn mowers are used as ATRs. Each wheel on the robot is able to drive forward and backward independently from each other. Raspberry Pi is used as an onboard computer. The ATRs are also equipped with a local collision detection system.

Figure 2.2: A modified Husqvarna autonomous lawn mover equipped with a table and an AprilTag.

Communication between the different modules is done through ROS2 using custom ROS2 messages.

High-level Control

The High-level Control module manages and commands where each ATR should be and the rough path it should take to reach the goal. Decides which ATRs should move in formation.

Vision system

The Vision system module provides a list of categorised obstacles and boundaries as polygons, given camera feeds. Slightly alters the original pixel-polygons, to reduce the number of vertices each polygon has. It also provides a list of predicted positions at different time steps for each categorised dynamic obstacle.

ATR fleet control

The ATR fleet control module alters the velocity of the ATRs on a global scale to avoid collisions in the future.

ATR Control

The ATR Control module has a hardware close controller and is responsible for the actual signal output to the ATRs motors. Also implements an odometry system that estimates the robot position based on the control signals. Includes an emergency stop system.

2.2 Our system

In this section, a brief overview of the formation control system is described.

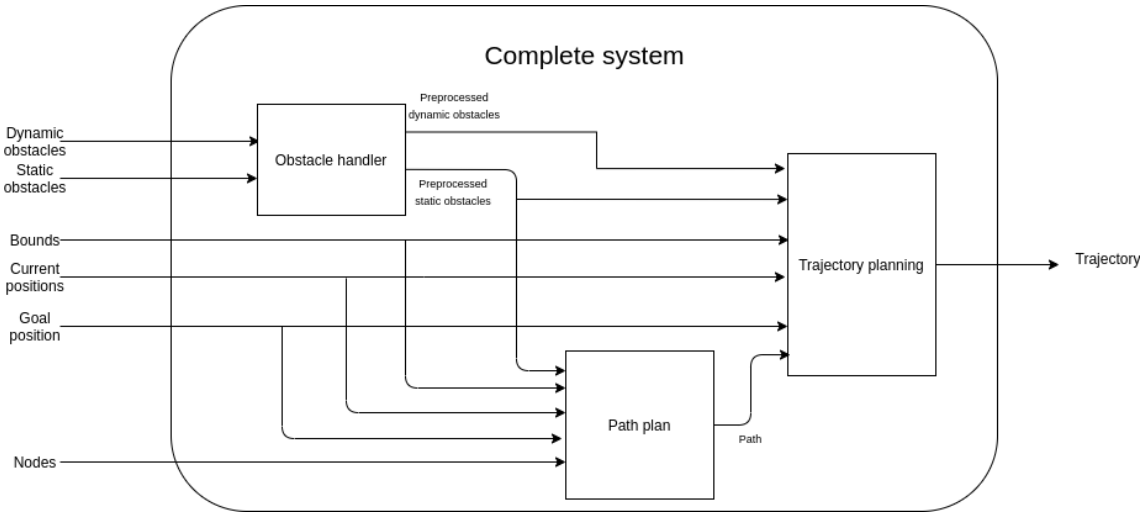


Figure 2.3: Brief overview of the system architecture.

The inputs to the complete system are dynamic and static obstacles, bounds, nodes, current state and a goal state. The system is divided into three different modules, obstacle handler, path plan and trajectory planning.

To be able to avoid collisions, the obstacle handler needs to process the obstacles, convert them into equations, pad the obstacles with a safety margin plus the robot size.

The path plan takes static obstacles, bounds, a set of nodes, current position, and a goal position. Calculates a collision free global path plan to the given goal state. The path is used in the trajectory generator as a reference path. The trajectory is calculated based on the current state, position of dynamic and static obstacles, bounds, system dynamics and constraints. The output of the trajectory generator is a trajectory on the form $[x_i, y_i, \theta_i, \Delta t_i]$ for each of the ATRs.

3
Path Planning

The goal of path planning in this thesis is to form a path which connects the starting position to the goal position while avoiding static obstacles. The area the robots can move in is fully known. The path planning is a two-step process which first finds the shortest path using the A* algorithm [4]. The second step is to avoid static obstacles. This is done using visibility graphs [5].

The shortest path in an area with polyhedron obstacles can be obtained via first creating a visibility graph then running A* [5] on the resulting nodes. Having the shortest path is not necessarily the goal though. The aim for this thesis is to be used in production environments. In a production facility there are predefined roads on which trucks, ATRs, are meant to move. So the goal is for the path to move along the predefined roads and only deviate where necessary to avoid obstacles. This is to make the system safe and predictable.

3.1 A*

A* is a well-known path planning algorithm that uses a directed graph to create a path from the start position to the goal position. In the case of this thesis the nodes of the graphs are formulated such that they lie on the roads of the production facility. There are faster and less memory intense path planning algorithms than A*. A* was chosen as it is a familiar algorithm, guarantees the shortest path if the heuristic is not over-estimated and execution time is not of particular importance.

In the graph A* searches through, each node has a cost associated to it which signals how expensive that node is. This cost is based on two things, the shortest path from start to this node and an estimate of how expensive it is to get to the goal from this node. The first cost is called the gscore and the second the heuristic. The cost of a node is shown in Equation (3.1).

$$f(n) = g(n) + h(n) \tag{3.1}$$

Here n is the node in question. How big $h(n)$ is depends on the heuristic function. As long as it is equal or smaller than the true cost of moving from node n to the goal A* will find the shortest path. A common choice of heuristic function is the euclidean distance.

3.2 Visibility-graph

Visibility graph [5] is a graph where all corners of polyhedrons have been extracted and connected. The corners are connected only if there exists a straight line between them which does not cross a polyhedron. By inserting a start and goal position into the graph, the shortest path can be found between them. The idea is to move along the corners of obstacles. This is guaranteed to be the shortest path in 2D. An example of a visibility graph is shown in Figure 3.1.

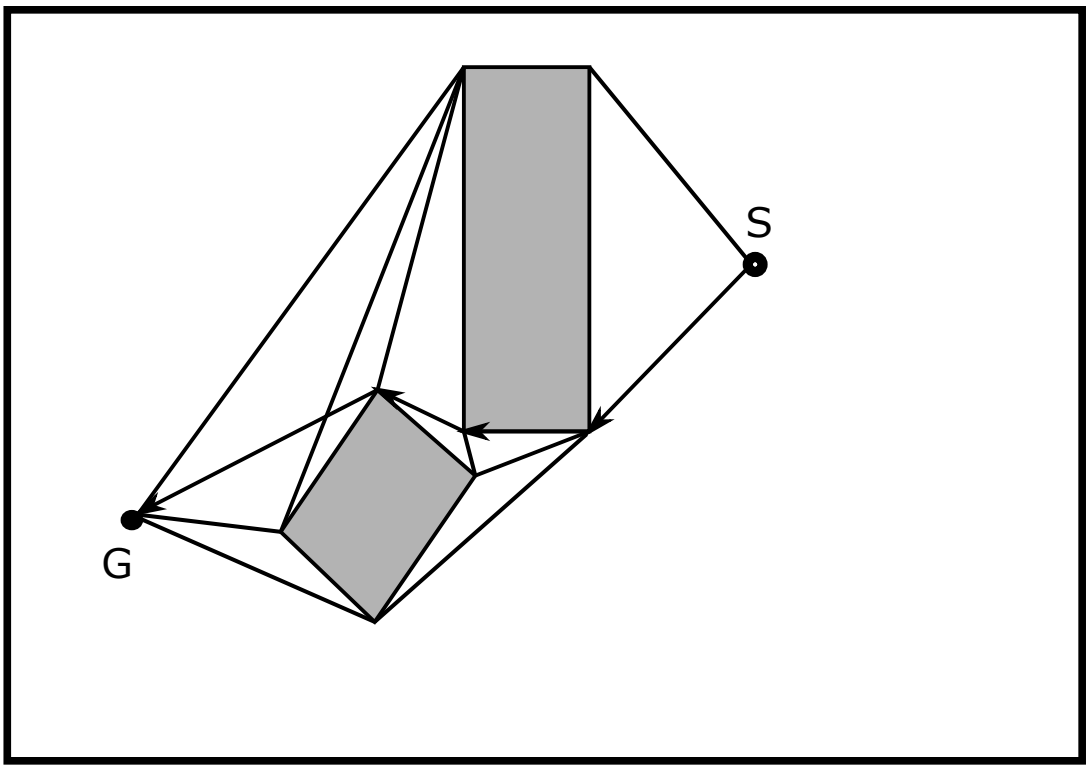


Figure 3.1: Visibility graph computation with polyhedron obstacles.

The position S is the start and G is the goal. The grey polyhedrons are the obstacles. The highlighted path is the shortest path from S to G.

A visibility graph reduces the robot down into a single point. This means that the physical size of the robot is ignored. As a result, the path may lead to the robot scratching the obstacles. To avoid this, the obstacles can be altered to accommodate for the size of the robot [5]. The easiest way of doing this is to approximate the robot as a circle and pad the obstacles with the radius. To pad means to increase the size of a geometric figure by a fixed amount.

3.3 Implementation

Path planning happens in two stages. The first stage ignores obstacles and finds the shortest path from start to goal using A*. The heuristic function chosen is the euclidean distance. The second step alters the path to avoid static obstacles using a visibility graph. The first stage works as a global path plan, and the second stage a local. The idea is that the path planner should be more predictable and robust against obstacles blocking the path. For example, if a pallet blocks a path 1 km ahead the planner should not react too early. This could result in the robot turning around and taking an unexpected route. Additionally, it could be so that the pallet is removed once the ATR gets close, making the rerouting pointless. In a production facility it is expected that obstacles are moved around the roads frequently, we do not want the ATRs to be constantly changing paths. Instead, The planner should plan to move towards the blocked path, and if the path is still blocked, then plan around the obstacle.

Consider the scenario shown in Figure 3.2.

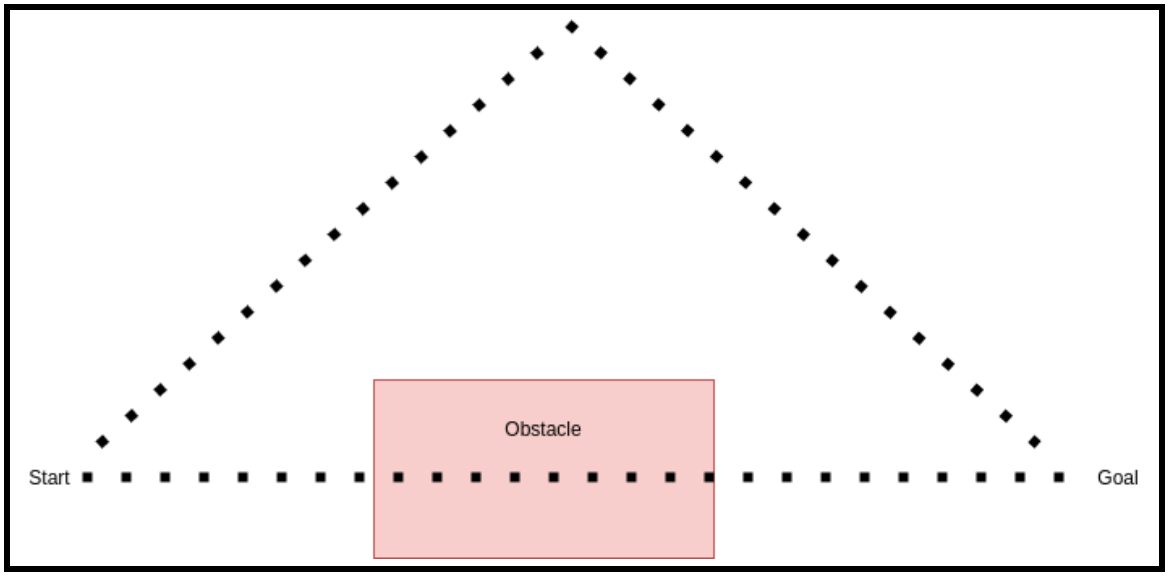


Figure 3.2: Path plan scenario.

The path should go between start and goal. The rectangle is a padded static obstacle to be avoided. The squared lines are the graph in which the path can go. This graph is provided to the system, as indicated in section 2 Problem setup and the nodes are placed 1 m apart. The first step is A* and should find the shortest path between start and goal while ignoring any obstacles. It should be something like shown in Figure 3.3.

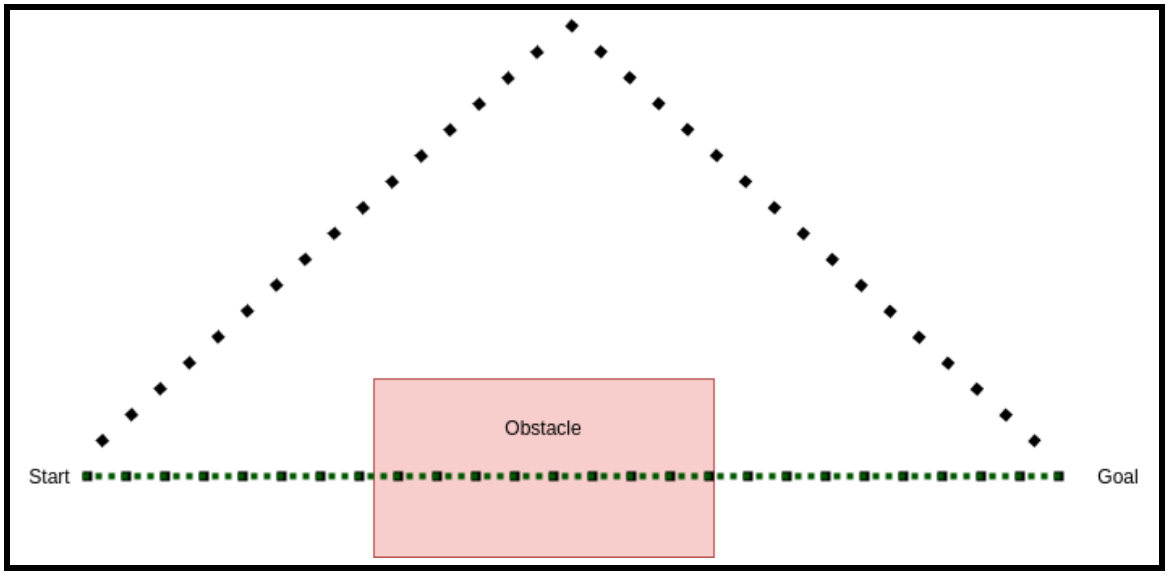


Figure 3.3: Path plan scenario with A* path.

The A* path is the smaller dotted green line going through the obstacle.

The next step is to avoid the obstacle using a visibility graph. To avoid the obstacle the robot has to turn and deviate off the original path. This is not desired as the path is actually a road in reality. So the robot should not start to deviate too early. Similarly it should not start to turn too late either as this would not result in smooth driving. The same applies to after the obstacle is passed. Then the robot should return to the original path at an appropriate point. How far ahead of the obstacle the robot should start to turn is a tuning parameter. After the visibility graph computation, the resulting path should be akin to Figure 3.4.

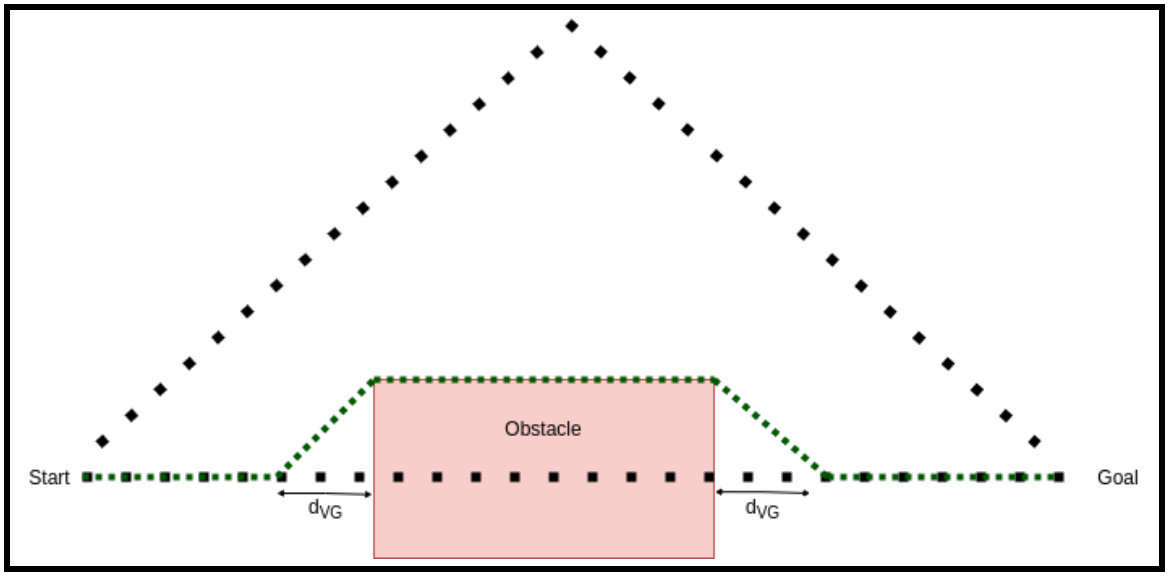


Figure 3.4: Path plan scenario after both A* and visibility graph computations.

The path has been updated to avoid the obstacle by moving along it's corners. The

new path starts to deviate off the old one at the first node which is at least a distance of d_{VG} away. Similarly it returns to the original path at the first node which is at least a distance of d_{VG} away. d_{VG} is chosen to be 1 m. The resulting path after path planning is not necessarily the shortest possible but should be one that is predictable for humans and close to the path a human would take.

3.4 Results

The first test was to see if the path planning worked at the most basic level. This scenario has no obstacle to avoid and only one possible path to take. Testing only A*. The result is shown in Figure 3.5.

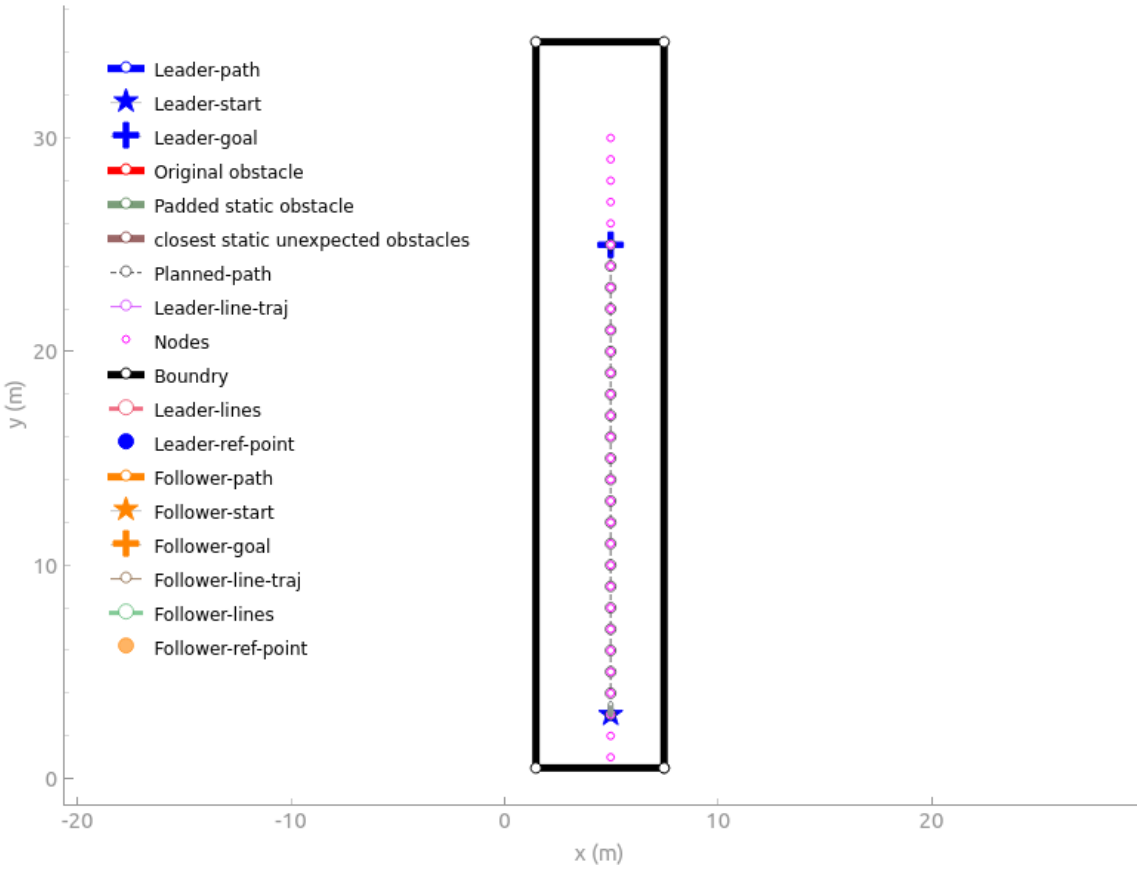


Figure 3.5: Simplest possible path plan.

The second scenario is more complicated and has several turns and two possible paths to take. Still no obstacle to avoid so again only testing A*. The result can be seen in Figure 3.6.

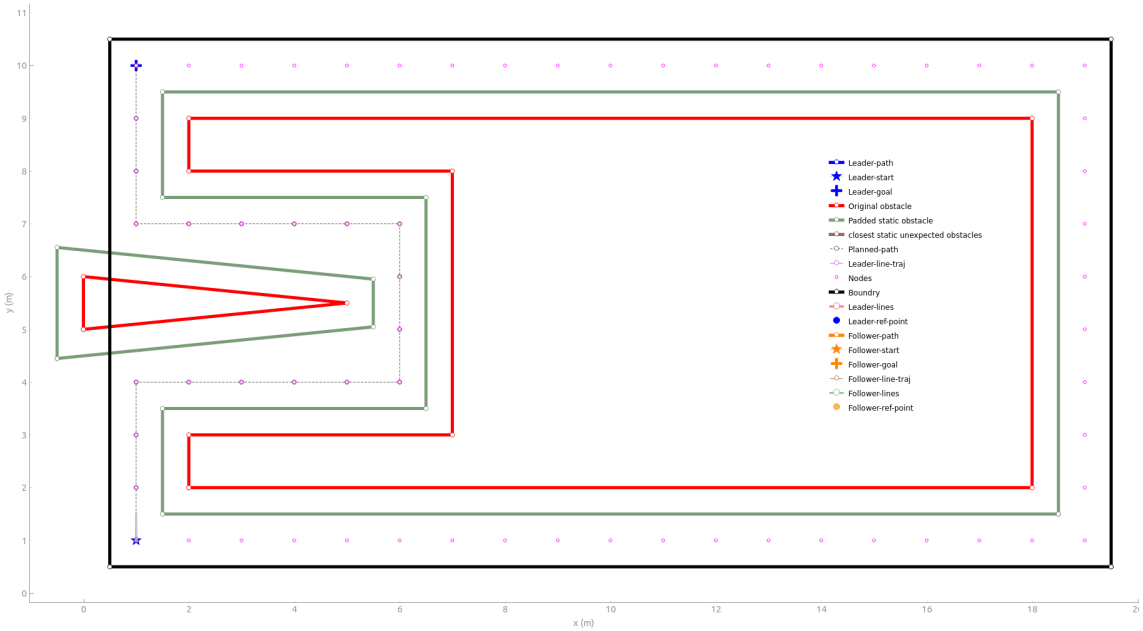


Figure 3.6: Complicated path plan without obstacle.

The third scenario has a similarly simple path as in the first scenario but with an obstacle to avoid. This tests the visibility graph. The results are in Figure 3.7.

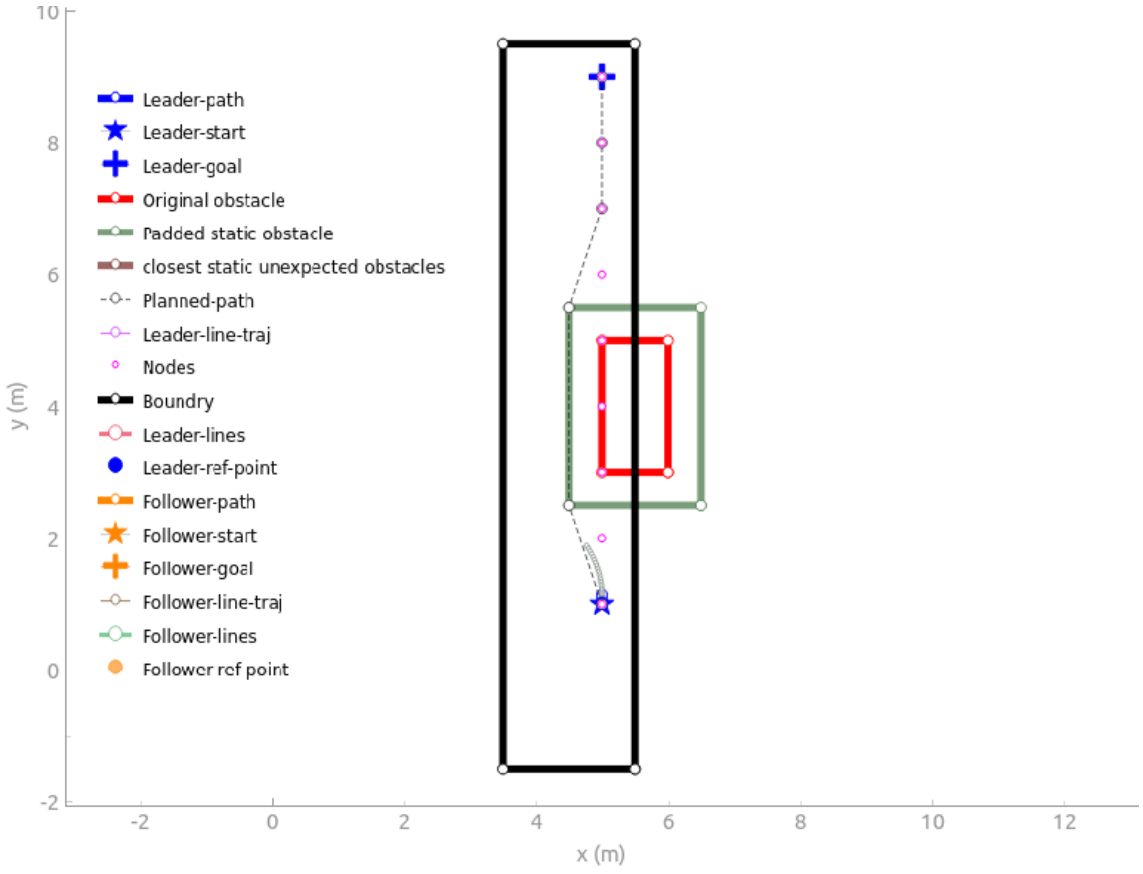


Figure 3.7: Path planned in simple scenario with obstacle.

The fourth scenario has two possible paths and two static obstacles to avoid. The result of the path planning is shown in Figure 3.8.

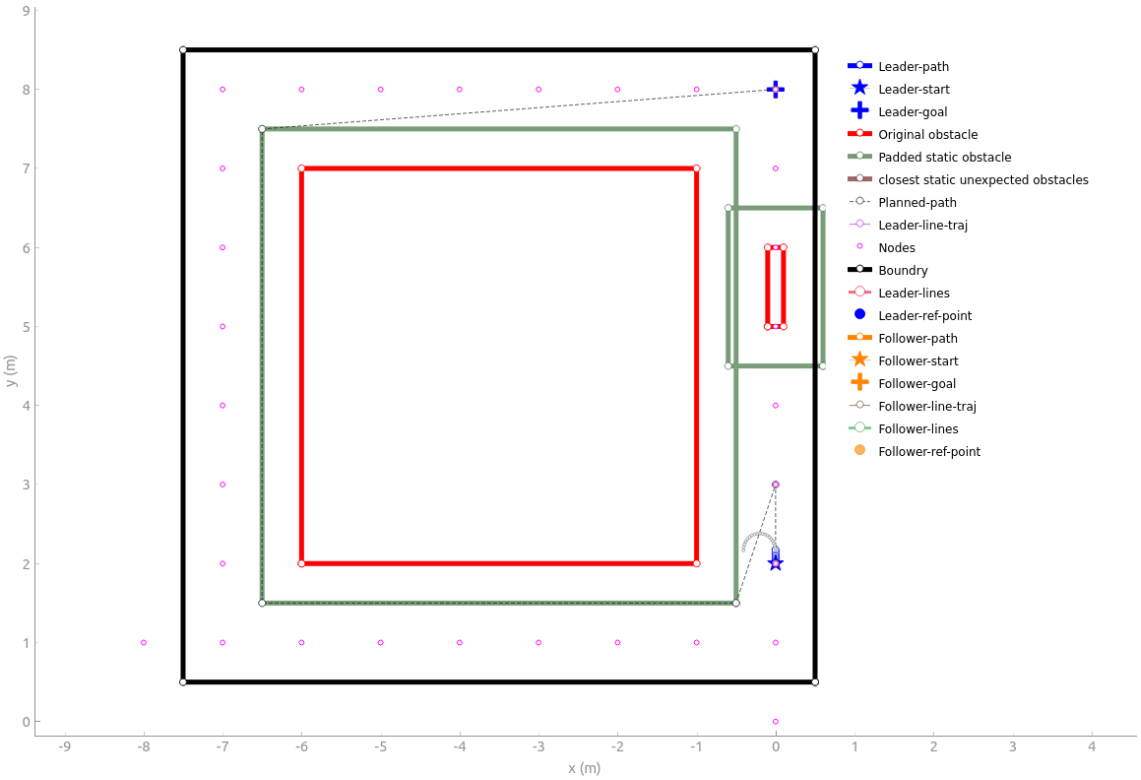


Figure 3.8: Path plan scenario after both A* and visibility graph computations.

4

Trajectory Planning

In this section, the theory behind trajectory planning is explained and the method regarding this thesis introduced.

4.1 Motion Model

This section shows the motion model of the ATRs and it's discretized counterpart. The ATRs are introduced in section 2.1 Complete system.

4.1.1 Differential drive

The continous motion model of the differential drive ATRs is deduced in [1] and is shown in Equation (4.1).

$$\dot{x} = v\ cos(\theta) \tag{4.1a}$$
$$\dot{y} = v\ sin(\theta) \tag{4.1b}$$
$$\dot{\theta} = \omega \tag{4.1c}$$

The positions of the ATR are x and y, θ is the rotation around it's center. These are the states of the robot. The control signals are v and ω. Linear velocity, v, is the moving the ATR forward and ω is the angular velocity which rotates the ATR. An illustration can be seen in Figure 4.1.

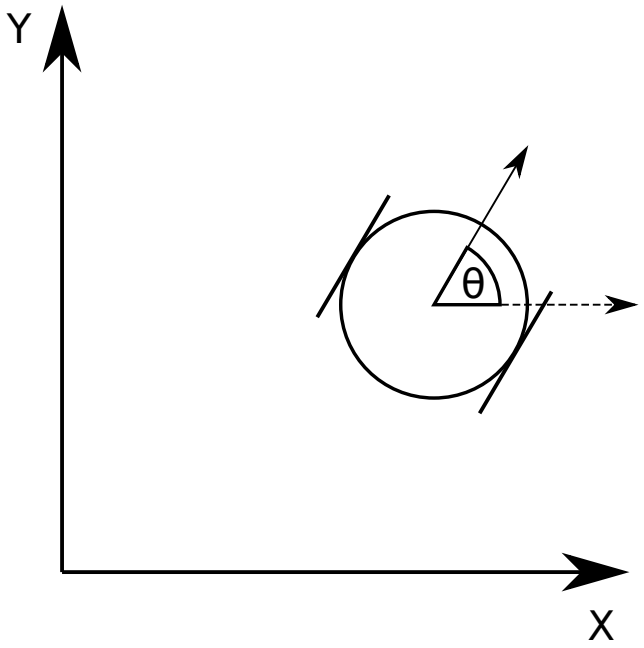


Figure 4.1: Complicated path plan without obstacle.

4.1.2 Discretization

Discretization of the ATR's motion model, shown in Equation (4.1), was discretized using explicit Runge-Kutta methods. Runge-Kutta is a well-studied integration method which provides good accuraccy vs computation complexity. It also provides different ways of increasing or decreasing the accuracy and complexity if that's needed.

The general formula for the Runge-Kutta methods, [6], is shown in Equation (4.2).

$$K_s = f(x_k + \Delta t \sum_{j=1}^{s} a_{ij} K_j, u(t_k + c_i \Delta t)) \tag{4.2a}$$

$$x_{k+1} = x_k + \Delta t \sum_{i=1}^{s} b_i K_i \tag{4.2b}$$

Where f is the continous motion model, x is the state vector, Δt is the sampling time and s is the order of the method chosen. In the case of Runge-Kutta 1, $s = 1$, with Runge-Kutta 4, $s = 4$. $u(t)$ is the control signal at time t and a, b and c are constants which define the method and are given in a Butcher tableau.

In this thesis Runge-Kutta of order 4 is used. This is because it provides especially good accuracy vs computational complexity [6]. There are several Butcher tableaus which result in Runge-Kutta 4. A common one is shown in Table 4.1.

Table 4.1: A Runge-Kutta 4 butcher tableau.

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

The resulting algorithm of using this tableau is shown in Equation (4.3).

$$K_1 = f(x_k + u(t_k)) \tag{4.3a}$$

$$K_2 = f\left(x_k + \frac{\Delta t}{2} K_1, u\left(t_k + \frac{\Delta t}{2}\right)\right) \tag{4.3b}$$

$$K_3 = f\left(x_k + \frac{\Delta t}{2} K_2, u\left(t_k + \frac{\Delta t}{2}\right)\right) \tag{4.3c}$$

$$K_2 = f\left(x_k + \Delta t K_3, u\left(t_k + \Delta t\right)\right) \tag{4.3d}$$

$$x_{k+1} = x_k + \Delta t\left(\frac{K_1}{6} + \frac{K_2}{3} + \frac{K_3}{3} + \frac{K_4}{6}\right) \tag{4.3e}$$

4.2 NMPC

Unlike a regular PID controller [7], Model Predictive Control (MPC) [8] is able to take multiple system states into account, input/output constraints and then optimize the desired system behaviour over a given time horizon. Most common is the use of linear models and constraints, or linearized models. In the case where the dynamics or constraints cannot be linearized, a Nonlinear Model Predictive (NMPC) [9] controller is used.

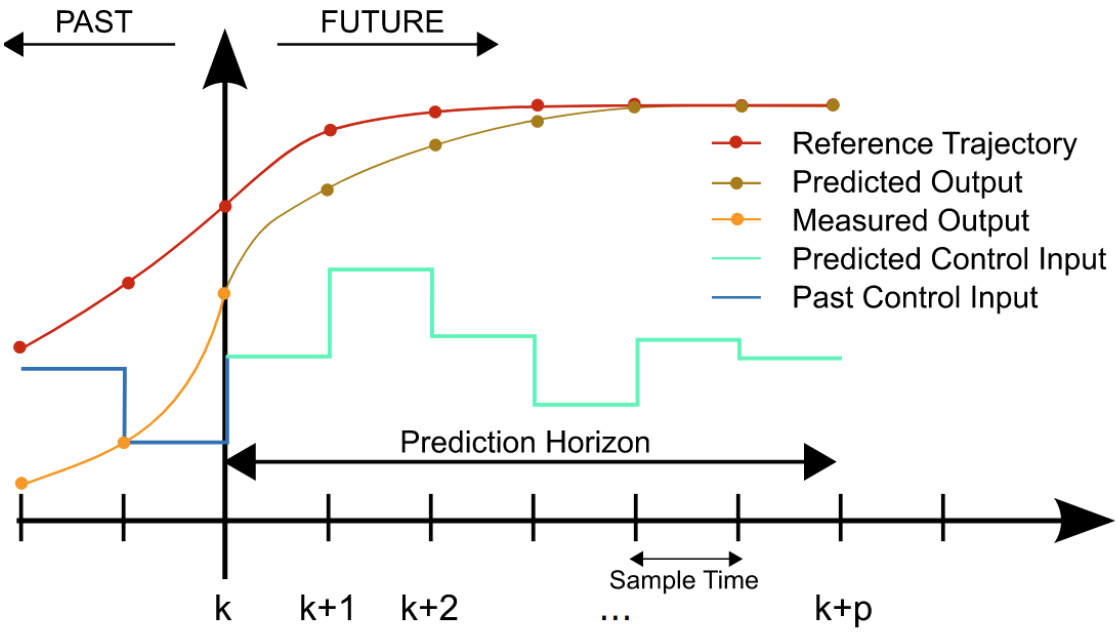


Figure 4.2: Basics of MPC.

At each time step k, a prediction of system states over a given time horizon is performed, using a model of the system. A control input is chosen for each of the states in the horizon leading to the optimal cost. The best prediction is achieved by applying a optimisation algorithm on a objective function with the control signals as variables. The first control input is then applied and the prediction starts again.

4.2.1 Horizon

Horizon refers to the amount of time steps into the future the controller will optimise over. Lower horizon may lead to faster computational times and higher horizon may lead to increased closed loop stability [10].

4.2.2 System dynamics

When using NMPC, the system dynamics can be non-linear [8]. To be able to constrain the outputs of the system, the system states must be considered in the solver. States can be modelled in two main ways, direct single shooting and multiple shooting. In the single shooting method, the states only depend on the initial state and the past inputs, as shown in Equation (4.4).

$$x(k+1) = f(x(0), u(0), u(1), ..., u(k)) \tag{4.4}$$

In the multiple shooting method, the states are described as decision variables, depending on the past states and inputs, as in Equation (4.5).

$$x(k+1) = f(x(k), u(k)) \tag{4.5}$$

Since PANOC [2], requires a single shooting formulation, this thesis will only consider single shooting.

4.2.3 Objective function

The objective function is the cost function that should be minimised. A general formulation approach according to [8] is shown below in Equation (4.6).

$$\min_{\mathbf{u},\mathbf{x}} \quad V(x,\mathbf{u},\mathbf{x}) = x^T Q x + \mathbf{x}^T \tilde{Q} \mathbf{x} + \mathbf{u}^T R \mathbf{u} \tag{4.6a}$$

Where $\mathbf{u} \in \mathbb{R}^{N_u}$ is the input control signal with N_u number of control signals, $\mathbf{x} \in \mathbb{R}^{N_x}$is the state vector, with N_x number of states, Q, $\tilde{Q}$ and R are weight matrices.

4.2.4 Constraints

Constraints are conditions that the solution must satisfy. They can be inequality or equality conditions.

$$\begin{aligned} \min_{x} \quad & \|f(x)\|_2^2 && (4.7a)\\ \text{subject to} \quad & \alpha \geq 0, && (4.7b)\\ & \beta \geq 0. && (4.7c) \end{aligned}$$

Hard constraints

Generally used in NMPC formulations to constrain the inputs to the system. A common condition is that the input to the system must be nonzero, and thus the objective function in Equation (4.6) can be constrained as

$$\begin{aligned} \min_{\mathbf{u},\mathbf{x}} \quad & V(x,\mathbf{u},\mathbf{x}) = x^T Q x + \mathbf{x}^T \tilde{Q} \mathbf{x} + \mathbf{u}^T R \mathbf{u} && (4.8a)\\ \text{subject to} \quad & \mathbf{u}_k > 0, \forall k \in [0, N_u - 1] && (4.8b) \end{aligned}$$

Another common condition is that the control signal should be within a specific interval, $U_{min} \leq \mathbf{u} \leq U_{max}$, and can be formulated as

$$\begin{aligned} \min_{\mathbf{u},\mathbf{x}} \quad & V(x,\mathbf{u},\mathbf{x}) = x^T Q x + \mathbf{x}^T \tilde{Q} \mathbf{x} + \mathbf{u}^T R \mathbf{u} && (4.9a)\\ \text{subject to} \quad & \mathbf{u}_k \geq U_{min}, \forall k \in [0, N_u - 1] && (4.9b)\\ & -\mathbf{u}_k \geq -U_{max}, \forall k \in [0, N_u - 1] && (4.9c) \end{aligned}$$

Soft constraints

To make the problem easier, considering computations and convergence. It is possible to relax constraints [11], meaning approximating a constrained problem as an unconstrained problem by moving the constraints into the objective function and adding a penalty when they are violated. A problem on the form

$$\begin{aligned} \min_{x} \quad & f(x) && (4.10a)\\ \text{subject to} \quad & \alpha(x) \leq 0, && (4.10b)\\ & && (4.10c) \end{aligned}$$

Can be relaxed into a soft constraint formulation as

$$\min_{x} \quad f(x) + vX_s(x) \tag{4.11a}$$

$$X_s(x) := max(0, \alpha(x)) \tag{4.11b}$$

Where v is the penalty parameter, and should be set sufficiently high to avoid constraint violation. The term 4.11a is zero when no constraint violation occurs and nonzero when constraints are violated, with a factor proportional to how much the constraints are violated.

4.3 Trajectory planner implementation

This section deals with the implementation specific details regarding the trajectory planner in this thesis. The idea behind the system is that the ATRs should follow the roads in the production facility. The roads are provided to the trajectory planner in form of a path by the path planner. The job of the trajectory planner is then to follow the path while avoiding obstacles, keeping the distance between the ATRs and moving smoothly. The premise of the situation, location of obstacles etc, is fed into a NMPC which then calculates the optimal trajectory.

4.3.1 NMPC solver choice

Since [12] provides a framework for using PANOC NMPC solver for obstacle avoidance and [1] shows that it is a feasible solution to use PANOC for formation control with dual ATRs. PANOC was the choice due to the previous results. No other solver were considered.

4.3.2 First view of the trajectory planner

This subsection serves as an introduction into the trajectory planner and the NMPC used. The contents here are explained in further detail later on.

The path provided by the path planner is a list of positions. By taking a position and the following position, a line can be created. We will call these path lines. As long as the ATRs are close to the path lines they will move along the path. To keep the ATRs on the path, a cost is used in the NMPC, based on the distance to the closest path line. To ensure that the ATRs are facing the correct direction, the angle of the path lines are used. A cost is formulated based on these.

The distance to keep between the ATRs is formulated using two parts. First the distance and second the error margin. This distance is then kept using both a cost and a constraint. The cost is used to motivate the NMPC solver to keep the ATRs at the correct distance. The constraint is to keep the ATRs inside the error margin.

The obstacles and boundaries are formulated as sets of positions. The ATRs should keep out of these sets. The physical ATRs impose constraints in the form of maximal velocities, accelerations and jerk. The velocity is handled both as a constraint and a cost. The maximal acceleration and jerk depends on several factors, how heavy the ATRs are loaded for example. There is no such information available to this system. As such the acceleration and jerk are handled only as costs. That should provide a smooth trajectory. An initial formulation of the NMPC problem is shown in Equation (4.12).

$$\min_{u} \quad \sum_{k=1}^{N}(u_k^T Q_u u_k + \hat{z}_k^T Q_{z,k} \hat{z}_k + \hat{u}_k^T Q_{du} \hat{u}_k + \tag{4.12a}$$

$$Q_d(D_k - d^2) + a_k^T Q_a a_k + j_k^T Q_{jerk} j_k + \tag{4.12b}$$

$$Q_{line} D_{line,k}) \tag{4.12c}$$

$$\text{s.t.} \quad z_0 := z_{init} \tag{4.12d}$$

$$U_{min} \leq u \leq U_{max} \tag{4.12e}$$

$$D_{min} \leq D \leq D_{max} \tag{4.12f}$$

$$[x_{1,k}, y_{1,k}]^T \notin O_{k_i},\ \forall k \in \mathbb{N}_{[1,N]}\ \forall i \in \mathbb{N}_{[0,n-1]} \tag{4.12g}$$

$$[x_{2,k}, y_{2,k}]^T \notin O_{k,i},\ \forall k \in \mathbb{N}_{[1,N]}\ \forall i \in \mathbb{N}_{[0,n-1]} \tag{4.12h}$$

$$[x_{1,k}, y_{1,k}]^T \in B,\ \forall k \in \mathbb{N}_{[1,N]} \tag{4.12i}$$

$$[x_{2,k}, y_{2,k}]^T \in B,\ \forall k \in \mathbb{N}_{[1,N]} \tag{4.12j}$$

N is the horizon length. The number of obstacles is n. The control signals are u and U_{min} is the minimum value they can take and U_{max} is the maximum. The state variables are in z, $z = [x_1,\ y_1,\ \theta_1,\ x_2,\ y_2,\ \theta_2]^T$, and $\hat{z}$ is the difference between the reference state and the actual state, $\hat{z} = z_{ref} - z$. Similarly $\hat{u}$ is $\hat{u} = u_{ref} - u$. The distance to keep is d and the error margin is D_{min} and D_{max}, D is the distance between the ATRs. The distance to the closest path line is D_{line}. O and B are the obstacles and boundary the ATRs should be outside and inside of at all time steps. The weights matrices are Q_u, Q_d, Q_a, Q_{jerk}, Q_{line} and $Q_{z,k}$.

4.3.3 Obstacles

The obstacles input to the trajectory planner are only ellipses and polygons. As such only those kinds of obstacles need to be considered in this thesis. According to [12] obstacles can be described as sets. Polyhedral obstacles in dimension n_d can be described as an intersection of inequalities on the form:

$$h = \{p \in \mathbb{R}^n : b_j - a_j^T p > 0, j \in \mathbb{N}_{[0,m-1]}\} \tag{4.13}$$

Where m is the number of inequalities required to describe the polyhedron, $a_j \in \mathbb{R}^{n_d}$ and $b_j \in \mathbb{R}^{n_d}$. Ellipsoids can similarly be described as:

$$h = \{p \in \mathbb{R}^{n_d} : 1 - (p - c)^T E (p - c) > 0\} \tag{4.14}$$

Where $c \in \mathbb{R}^{n_d}$ and $E \in \mathbb{R}^{n_d * n_d}$.

In the case of a $n_d = 2$, each side is formulated as a half plane. To illustrate, a diamond polygon and it's half spaces are shown in Figure 4.3.

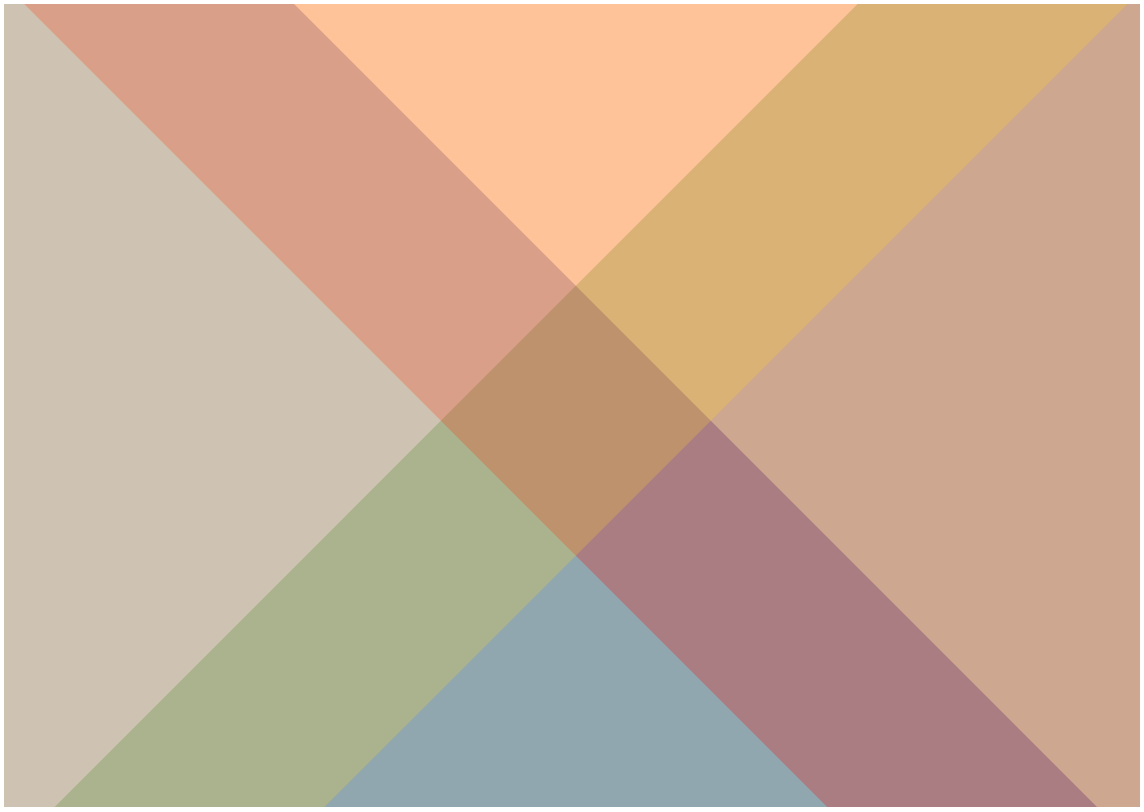


Figure 4.3: Convex obstacle described as a set. Each side of the polygon is represented as a half space. The resulting obstacle is the diamond in the middle where all half spaces overlap.

To more easily see the obstacle represented the excess half spaces can be cut. See Figure 4.4

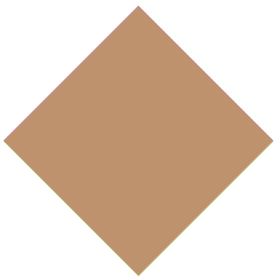


Figure 4.4: The resulting convex obstacle shown in Figure 4.3 when the excess half spaces are cut.

If the ATR is inside a half plane then $h_j(p) > 0$. In order to not be in the obstacle there must be at least one half space that the ATR is not inside. Let $[x]_+$ be defined as $[x]_+ = \max(0,x)$. To formulate an obstacle avoidance constraint the robot cannot be inside an obstacle set: $p \notin h$. This can be reformulated. For the robot to avoid obstacle h then $h_j(p) \leq 0 \; \forall \; h_j$. This can also be written as $[h_j(p)]_+^2 = 0 \; \forall \; h_j$. This can be further reformulated as:

$$\prod_{j=0}^{m-1} [h_j(p)]_+^2 = 0 \tag{4.15}$$

Concave obstacles

Expressing polyhedron obstacles as Equation (4.13) works well with convex obstacles. The formulation can not accurately describe concave obstacles. It underestimates the size of the obstacle. This is a result of representing every side of the polygon as a half space and requiring the ATR to be in all of the half spaces at the same time. An example of the problem is shown in Figure 4.5. Here the obstacle, outlined in black, is concave. It gets erroneously reduced into the green rectangle.

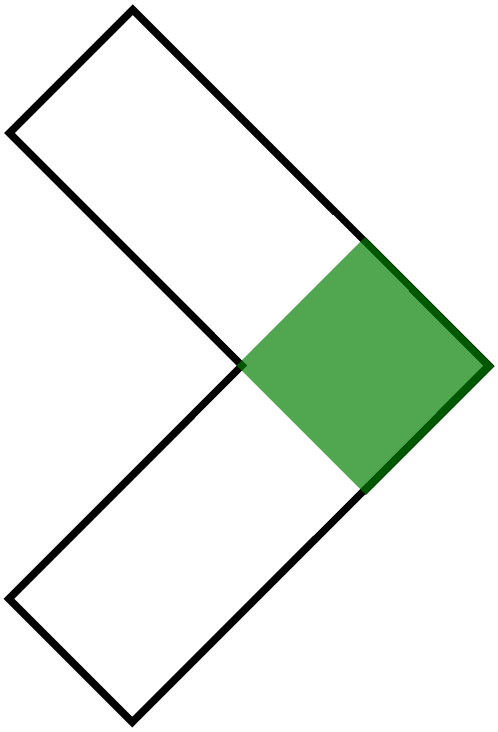


Figure 4.5: Concave obstacle described as a set. Original obstacle is outlined in black and the green rectangle is what it is erroneously reduced to.

The bottom polygon is the obstacle as represented by Equation (4.13). It is considerably smaller than the actual obstacle. This mean that the NMPC would see the obstacle as being smaller than it is and potentially plan to go through it. A crash is not acceptable.

There are multiple ways to address concave obstacles. One could switch to another way of representing the obstacles or one could convert the concave obstacle into a convex one. The latter option was chosen as it has proven to work in [12]. Two ways of converting a concave polygon to a convex polygon is to calculate the convex hull or split the polygon into smaller, convex ones. The convex hull would make the resulting polygon convex and it is computationally easy to do, but the result would be bigger than the actual obstacle. This could create an artificial barrier which blocks the ATR from going a certain path. Splitting the polygon into smaller, convex ones is computationally expensive and results in more polygons, it does however preserve the actual size of the obstacle. This method was chosen as the quality of the solutions outweighs the computation speed in this thesis.

Splitting a concave polygon into smaller convex ones can be done in a plethora of ways. Two of the possibilities are to split into fewest possible polygons or to split into triangles. Splitting into the fewest possible polygons can be done using Optimal Convex Decompositions (OCD) [13]. This results in the fewest possible polygons but is slow to compute and the number of vertices of the polygons is unbound.

Splitting into the fewest possible triangles can be done using Constrained Delaunay Triangulation(CDT) [14]. CDT is fast and produces polygons with a known amount of vertices, namely three but results in more polygons than OCD. CDT is chosen because OpEn requires pre-allocating the number of parameters, and fixed amount of vertices makes pre-allocating possible.

With triangulation the concave obstacle in Figure 4.5 becomes as shown in Figure 4.6.

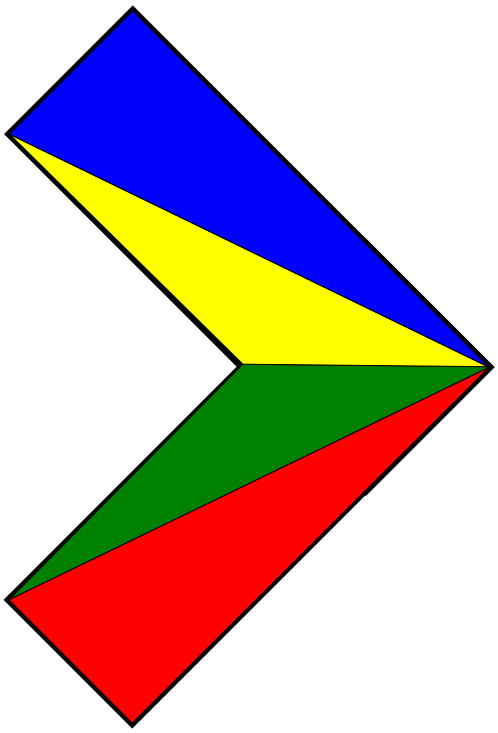


Figure 4.6: Concave obstacle split into triangles. The colored areas are the sub triangle the original obstacle is split into.

The area covered by the triangles is the same as the original concave obstacle.

Dynamic obstacles

Dynamic obstacles are obstacles that move with time and can be represented in various ways. One way is to have time varying parameters. This can be done by changing a, b and c in Equation (4.13) and Equation (4.14) to depend on time instead of being constant. Another way is to have individual parameters for each time step. So for time step k one set of parameters are used. In time step $k+1$ another set of parameters are used. This is the way the dynamic obstacles are input to the obstacle handler so that is how they are represented in the NMPC.

4.3.4 Boundary

The boundary of the map is another obstacle that needs to be avoided. The area inside of the boundary is called driveable area. The approach taken to the boundary is similar to the one of obstacles, in section 4.3.3 Obstacles, but differs with the condition that the ATRs needs to be inside the obstacle/boundary. The non driveable area is in general concave. The equation used for obstacles blocked the inside area of a polygon, see Equation (4.15). Modifications can be done to instead allow the inside area of a polygon, shown as the white space in Figure 4.7.

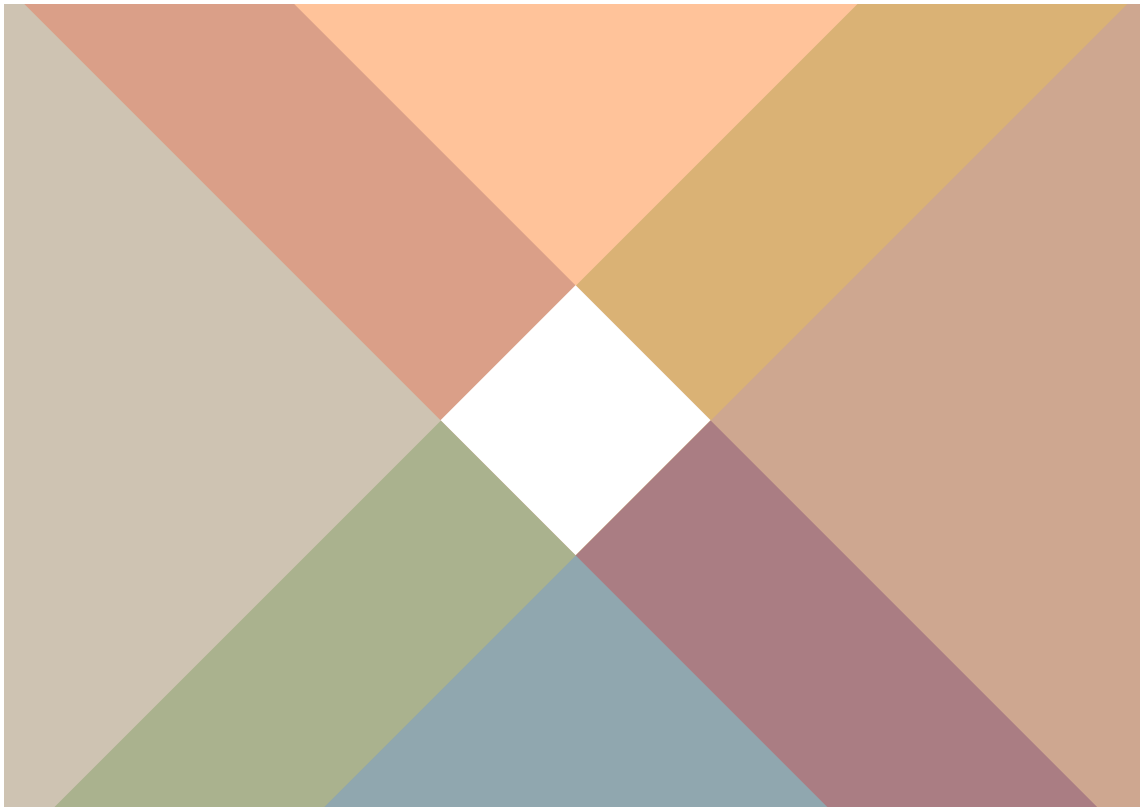


Figure 4.7: Polygon boundary described as a set of half spaces, coloured space is non driveable area

Multiplication can, in this case, be seen as an AND statement, meaning that Equation (4.15) is > 0 if the ATR is inside all of the halfspaces. The modification is to swap the AND statement to an OR statement by swapping the multiplication to a sum. Thus, the Equation (4.16) is > 0 if the ATR is outside any of the halfspaces.

$$B(p) = \frac{1}{2} \sum_{j=0}^{m-1} [h_j(p)]_+^2 \tag{4.16}$$

where $[h_j(p)]_+^2$ describes each half-space of the boundary.

Non-convex boundaries

To enable avoidance of concave boundaries, the approach was to calculate the convex hull of the boundary using Shapely [15], and use the new boundary with condition

4.16. The difference between the original concave polygon and the convex hull can also be described as a set of polygons using symmetric difference from the Shapely package. The set of polygons can then be considered as regular obstacles as in section 4.3.3 Obstacles.

4.3.5 Updated NMPC

With a way of formulating obstacles and the boundary as inequalities the somewhat abstract NMPC formulation in Equation (4.12) can be reformulated in a more concrete way.

$$\begin{aligned}
\min_{u} \quad & \sum_{k=1}^{N} (u_k^T Q_u u_k + \hat{z}_k^T Q_z \hat{z}_k + \hat{u}_k^T Q_{du} \hat{u}_k + && \text{(4.17a)}\\
& Q_d(D_k - d_k^2) + a_k^T Q_a a_k + j_k^T Q_{jerk} j_k + && \text{(4.17b)}\\
& Q_{line} D_{line,k}) && \text{(4.17c)}\\
\text{s.t.} \quad & z_0 := z_{init} && \text{(4.17d)}\\
& U_{min} \leq u \leq U_{max} && \text{(4.17e)}\\
& D_{min} \leq D \leq D_{max} && \text{(4.17f)}\\
& O_{i,k}([x_{1,k}, y_{1,k}]^T) = 0 \ \forall i \in \mathbb{N}_{[0,m-1]}, \forall k \in \mathbb{N}_{[1,N]} && \text{(4.17g)}\\
& O_{i,k}([x_{2,k}, y_{2,k}]^T) = 0 \ \forall i \in \mathbb{N}_{[0,m-1]}, \forall k \in \mathbb{N}_{[1,N]} && \text{(4.17h)}\\
& B([x_{1,k}, y_{1,k}]^T) = 0 \ \forall k \in \mathbb{N}_{[1,N]} && \text{(4.17i)}\\
& B([x_{2,k}, y_{2,k}]^T) = 0 \ \forall k \in \mathbb{N}_{[1,N]} && \text{(4.17j)}
\end{aligned}$$

Where D is the squared distance between the ATRs, calculated as

$$D = (x_{1,k} - x_{2,k})^2 + (y_{1,k} - y_{2,k})^2 \tag{4.18}$$

O is if the ATRs are inside any obstacle

$$O(p) = \frac{1}{2} \sum_{i=0}^{n-1} \prod_{j=0}^{m-1} [h_{j,i}(p)]_+^2 \tag{4.19}$$

where n is the number of obstacles and m the number of inequalities defining the obstacle h_j. B is if the ATRs are inside the boundary

$$B(p) = \frac{1}{2} \sum_{j=0}^{m-1} [h_j(p)]_+^2 \tag{4.20}$$

where m is the number of inequalities describing the boundary.
Acceleration a is calculated as:

$$a_k = \frac{v_{k+1} - v_k}{T_s} \forall k, \in \mathbb{N}_{[0,N-1]} \tag{4.21}$$

Jerk j is calculated as:

$$j_k = \frac{a_{k+1} - a_k}{T_s} \forall k, \in \mathbb{N}_{[0,N-2]} \tag{4.22}$$

A path line is defined as a a line segment between a position in the path subtracted by the previous position, $l(t) = p_n + (p_{n+1} - p_n)t$, $t \in \mathbb{R}_{[0,1]}$. The distance to a path line, d, is the distance from the current position, p, to l:

$$d(t) = \|l(t) - p\|_2 \tag{4.23}$$

To calculate the minimising t the derivative of d is used. The minimal t, called $\hat{t}$ is when $d'(t) = 0$. To make the computation easier the $d(t)^2$ is used, this gives the same $\hat{t}$ as $d(t)$ would.

$$\frac{dd(t)^2}{dt} = 0 \implies \hat{t} = \frac{(p - p_n) \cdot (p_{n+1} - p_n)}{\|p_{n+1} - p_n\|_2^2} \tag{4.24}$$

This can give a $\hat{t}$ outside of [0,1]. This suggests that the closest point is beyond p_{n+1} and p_n. Since l only exists between p_{n+1} and p_n $\hat{t}$ needs to be converted back to [0,1]. If $\hat{t} < 0$ then $\bar{t} = 0$ and if $\hat{t} > 1$ then $\bar{t} = 1$. Finally the shortest distance d between position p and path line l is given by d($\bar{t}$). The shortest distance to the closest path line, D_{line} in Equation (4.17), is calculated by calculating the shortest distance to all path lines and taking the smallest one.

Soft constraints

Reformulating the NMPC formulation using soft constraints results in Equation (4.25).

$$\begin{aligned}
\min_{u} \quad & \sum_{k=1}^{N} (u_k^T Q_u u_k + \hat{z}_k^T Q_z k \hat{z}_k + \hat{u}_k^T Q_{du} \hat{u}_k + && \text{(4.25a)} \\
& Q_d(D_k - d^2) + Q_{ds}[D_k - D_{max}^2]_+ + Q_{ds}[-D_k + D_{min}^2]_+ + && \text{(4.25b)} \\
& a_k^T Q_a a_k + j_k^T Q_j j_k + Q_{line} D_{line,k} + && \text{(4.25c)} \\
& Q_{os} O_k(x_{1,k}, y_{1,k}) + Q_{os} B(x_{1,k}, y_{1,k}) + && \text{(4.25d)} \\
& Q_{os} O_k(x_{2,k}, y_{2,k}) + Q_{os} B(x_{2,k}, y_{2,k})) && \text{(4.25e)} \\
\text{s.t.} \quad & z_0 := z_{init} && \text{(4.25f)} \\
& U_{min} \leq u \leq U_{max} && \text{(4.25g)}
\end{aligned}$$

Where Q_{os} is the soft constraint weight scalar for being inside an obstacle or outside of the boundary. Similarly Q_{ds} is for violating the distance error margin.

4.3.6 Line follow angle

To make the ATRs face the correct direction a reference angle, called line follow angle, is calculated based on path lines. This is to ensure that the ATRs do not move backwards along the path lines. The desired direction is the same angle as the closest path line. When taking a corner the closest path line will change. This results in a sudden change of line follow angle. To smooth this transition the line follow angle is based on both the closest path line and the upcoming one. If the next path line is within a distance of d_{turn} m then it's angle is considered. Figure 4.8 shows an example scenario.

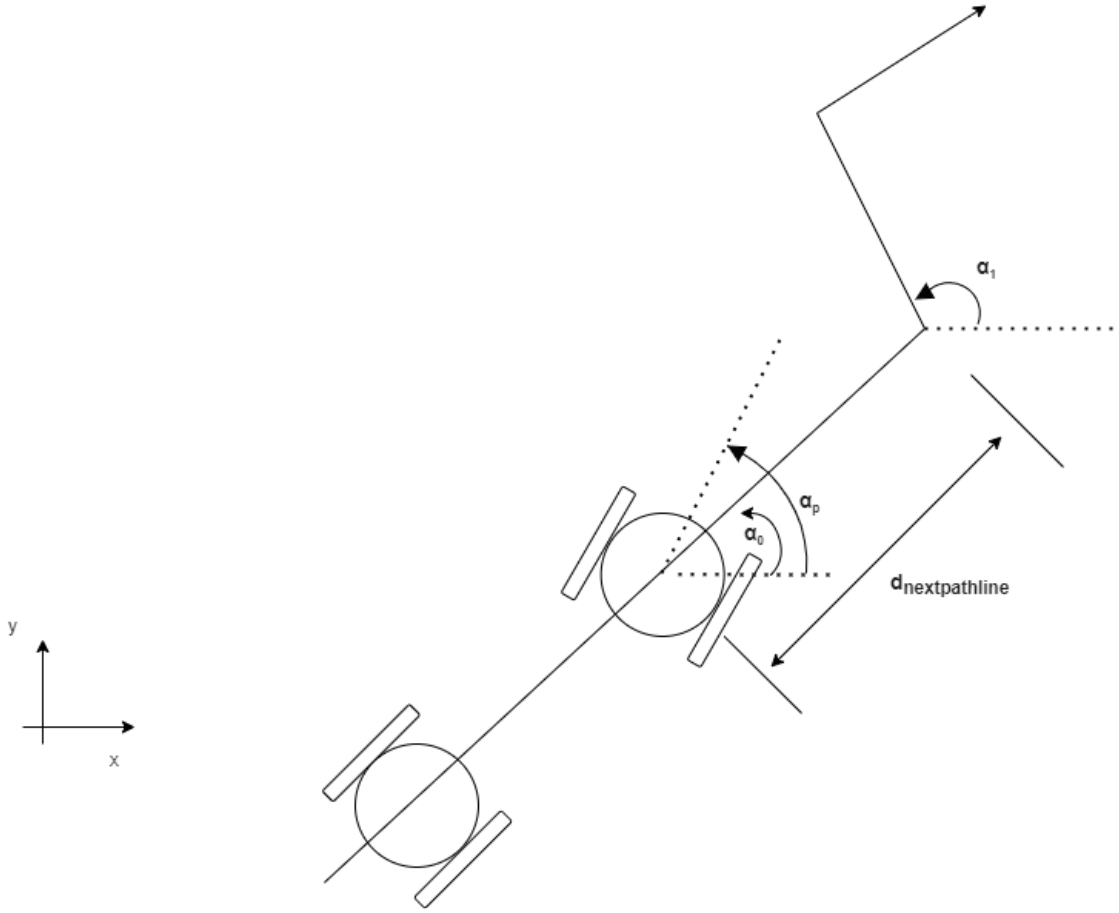


Figure 4.8: Line follow angle calculation scenario.

Equation (4.26) shows how the line follow angle, α_p is calculated.

$$\alpha_p = \alpha_0 a + \alpha_1(1-a) \tag{4.26a}$$

$$a = min\left(1, \frac{d_{nextpathline}}{d_{turn}}\right) \tag{4.26b}$$

Where α_0 and α_1 are the angles of the closest and upcoming path line respectively. The distance to the next path line is $d_{nextpathline}$ and is calculated as in Equation (4.23).

4.3.7 Formation

The formation control is based on a leader follow approach by [16], where there is a leader and a single or multiple followers. Given the trajectory of the leader the followers trajectories can be calculated as offset trajectories of this. In the case of two ATRs one is the leader and one is the follower. With the trajectory of the leader as X_L the follower trajectory X_F is calculated as Equation (4.27).

$$X_{F,k} = X_{L,k} + \delta_k, \forall k \in \mathbb{N}_{[1,N]} \tag{4.27}$$

Where δ is the offset. It should be chosen so that the distance constraint is fulfilled.

The offset δ is chosen such that the follower is always behind the leader. If the leader is going north the follower should be to the south. The angle between leader and follower, called α_f, should always be π, in the leader ATR reference frame. The leader is moving in the line follow angle, α_p, so the calculation of α_f is:

$$\alpha_f = \alpha_p - \pi \tag{4.28}$$

Figure 4.9 shows an example scenario.

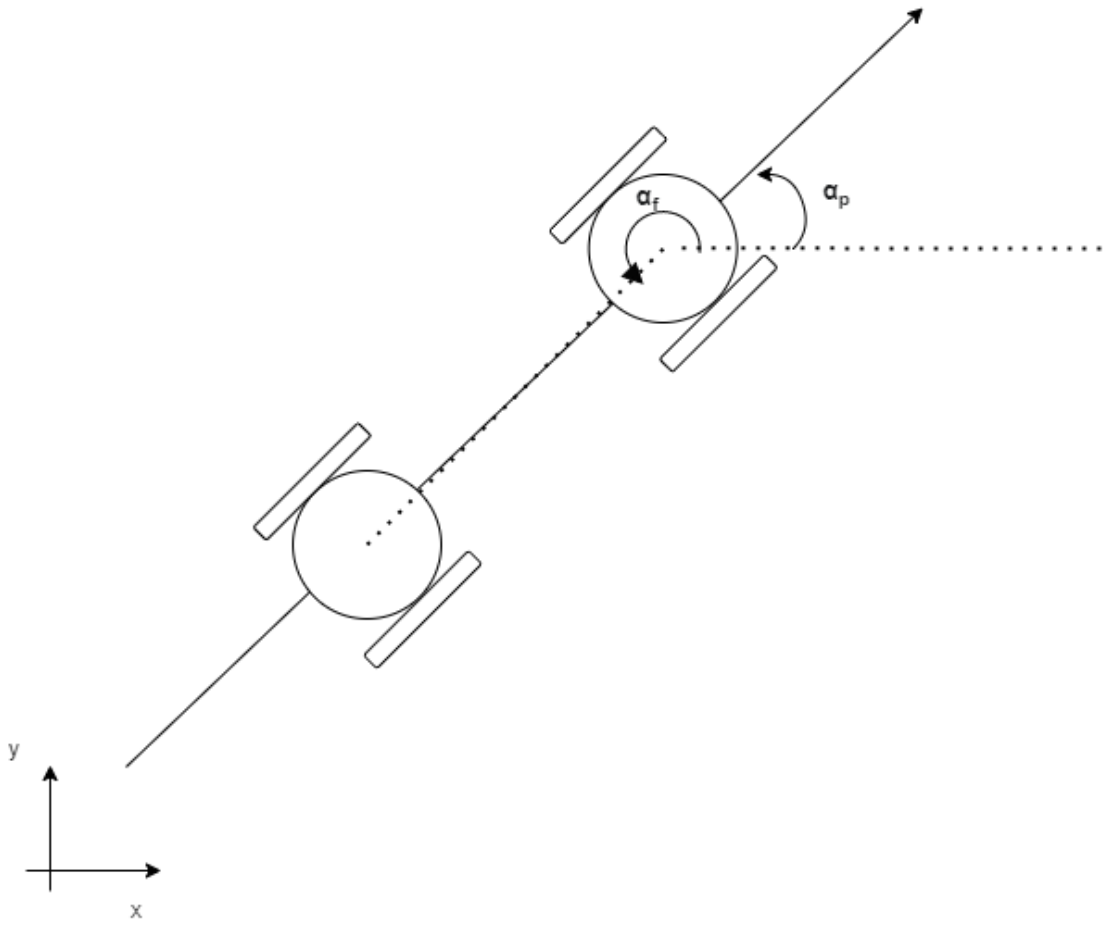


Figure 4.9: Formation angle calculation scenario.

With α_f the equation for δ is:

$$\delta_k = [dcos(\alpha_f),\ dsin(\alpha_f),\ 0]^T, \forall k \in \mathbb{N}_{[0,N-1]} \tag{4.29}$$

This way of calculating the follower trajectory requires the original leader trajectory as input. There is no leader trajectory input to the trajectory planner so it has to be generated. This is discussed in the next subsection.

4.3.8 Usage

When the ATRs are not in use they are in an idle location, a charging station for example. When a job arrives the ATRs should move from the charging station to a designated location to pick up their load. They should then follow the roads until the drop off position. Once the load has been unloaded they should move back into an idle location. This naturally introduces three states the ATRs can be in while moving. These states are named as coupling, formation and decoupling state. The coupling state is when the ATRs move from the charging station to the pick up position. The formation state is when they move from the pick up position to the drop off position. The decoupling state is when they move from the drop off position back to the charging stations.

The three states have different requirements. They all should avoid obstacles and moving within bounds. While in the formation state the ATRs should follow the roads, move as a leader and follower and keep a fixed distance between themselves. The coupling and decoupling states do not have to keep a fixed distance since there is no load to carry. Instead they should keep a minimum distance to avoid crashing into eachother. In this state the ATRs move independently. Since the charging stations are close to the pick up and drop off positions there is no notion of a road, so they cannot follow a road in these states. Instead they should move from their current position to the goal position.

The NMPC formulation shown in Equation (4.25) can be used to achieve the two different behaviours required by the states. By having the parts of Q_z corresponding to the x and y positions as 0 and the angle reference value as α_p in Equation (4.28) the line follow behaviour in the formation state is acquired. By instead having Q_{du}, Q_{line} and Q_d as 0 and D_{max} as infinity the position goal behaviour in coupling and decoupling states is acquired.

The aggressiveness of the ATRs can be tuned. By increasing Q_{du} and decreasing Q_{line} a more aggressive behaviour is achieved. This means that the ATRs cut in front of dynamic obstacles passing their path instead of waiting for them which results in the system moving forward. Avoiding obstacles, boundaries, having smooth accelerations, keeping the distance and being close to the path are all vastly more important than moving forward. As such the linear velocity reference is put at a medium value and the angular velocity reference is set as zero with Q_{du} being very low and Q_{line} high.

The trajectory planner should work with both two ATRs and one ATR. Two ATRs should only be used when the load is too big for one ATR to carry alone. The same NMPC can be used with either one of the ATRs, by tweaking the parameters. To remove the follower ATR from the NMPC the weights corresponding to it can be set to zero, this means Q_u, Q_z, Q_{du}, Q_a, Q_j, Q_{line}, Q_{os} and Q_{bs}. In addition all of the formation costs also have to be set to zero, these are Q_d, Q_{ds}.

The position goal behaviour of the NMPC with only one ATR is shown to work

by [12]. The line follow also known to work, shown by [3]. As mentioned in 4.3.7 Formation, the formation control chosen requires a leader trajectory as an input. This thesis acquires the leader trajectory by running the NMPC with only one ATR. The resulting trajectory is the leader trajectory. So first the NMPC is run with only one ATR. The resulting trajectory is modified to acquire a follow trajectory. These two trajectories are then the input to the NMPC via reference states, this time with two ATRs enabled. Now the line follow cost, Q_{line}, is set to zero.

4.4 Summary

The NMPC is formulated using soft constraints as shown in Equation (4.25). Convex obstacles as represented as inequalities as seen in Equation (4.13) and Equation (4.14). Concave obstacles are a problem and are solved by converting them into convex obstacles via triangulation. The boundary is represented as a combination of inequalities and obstacles. The ATRs move in leader follow formation with the leader in front of the follower. The ATRs can exist in three different states, coupling, formation and decoupling. Each state requires something different from the NMPC. The NMPC can also exist in three different modes, line follow, position goal and reference follow. Also the trajectory planner should work with both two and one ATRs.

4.5 Results

To evaluate the trajectory planner several test scenarios were used. First a general scenario is used. This scenario starts out easy and becomes harder. In addition to testing the general functionality, a few scenarios are used to test specific properties of the trajectory generator. How the simulations are performed is discussed in 5 Software architecture.

The general scenarios are shown first. First only one ATR is used and then the leader-follower combination is tested. The first scenario contains a long straight line, a 45 degree corner, a 135 degree corner, followed by a 90 degree corner into another straight line. The second scenario is similar to the first but contains static obstacles around the bends to test if the ATRs can navigate tight spaces. The third scenario contains a static obstacle on the path. Finally the last scenario has dynamic obstacles. As such it has all the components discussed in this thesis and tests the entire system. The dynamic obstacles move so seeing an image of how the ATRs moved does not show how the simulations played out. Both the positional plots and data plots are shown. The data plots contain the control signals, accelerations, jerk and distance between the ATRs.

After the general scenarios the more specific are shown. The first of which try test the ability of the leader to adapt to the follower. This scenario is doable and easy for a human. However it requires the leader to adapt it's trajectory to allow the system to progress. The second specific scenario tests how the ATRs react when

there is dynamic obstacles moving straight towards them.

4.5.1 General scenarios

The following scenarios was simulated using $D_{max} = 1.05$, $D_{min} = 0.95$, $T_s = 0.1$, $N = 20$ and a aggressive factor of 0.01.

The following Table 4.2 provides a brief explanation of the plotted results legends.

Legend	**Explanation**
Leader-path	The path the leader ATR has moved.
Leader-start	Starting position for leader ATR.
Leader-goal	Global goal position for leader ATR.
Original obstacle	Obstacle polygon without padding.
Padded static obstacle	Obstacle polygon with padding.
Closest static unexpected obstacles	Shows what obstacles are considered in the solver.
Planned-path	The planned path generated by the path-planner.
Leader-line-traj	Planned future trajectory with N steps.
Nodes	Nodes used by the path-planner A* and visibility graph algorithms.
Boundry	Border between driveable and non driveable areas.
Leader-lines	-
Leader-ref-point	Reference point for leader ATR, used in position goal control mode as a goal.
Follower-path	The path the follower ATR has moved.
Follower-start	Starting position for follower ATR
Follower-goal	Global goal position for follower ATR
Follower-line-traj	Planned future trajectory with N steps.
Follower-lines	-
Follower-ref-point	Reference point for follwer ATR, used in position goal control mode as a goal.

Table 4.2: Legend explanation

Leader no obstacles

First scenario, follows the given path, cuts the second sharp corner.

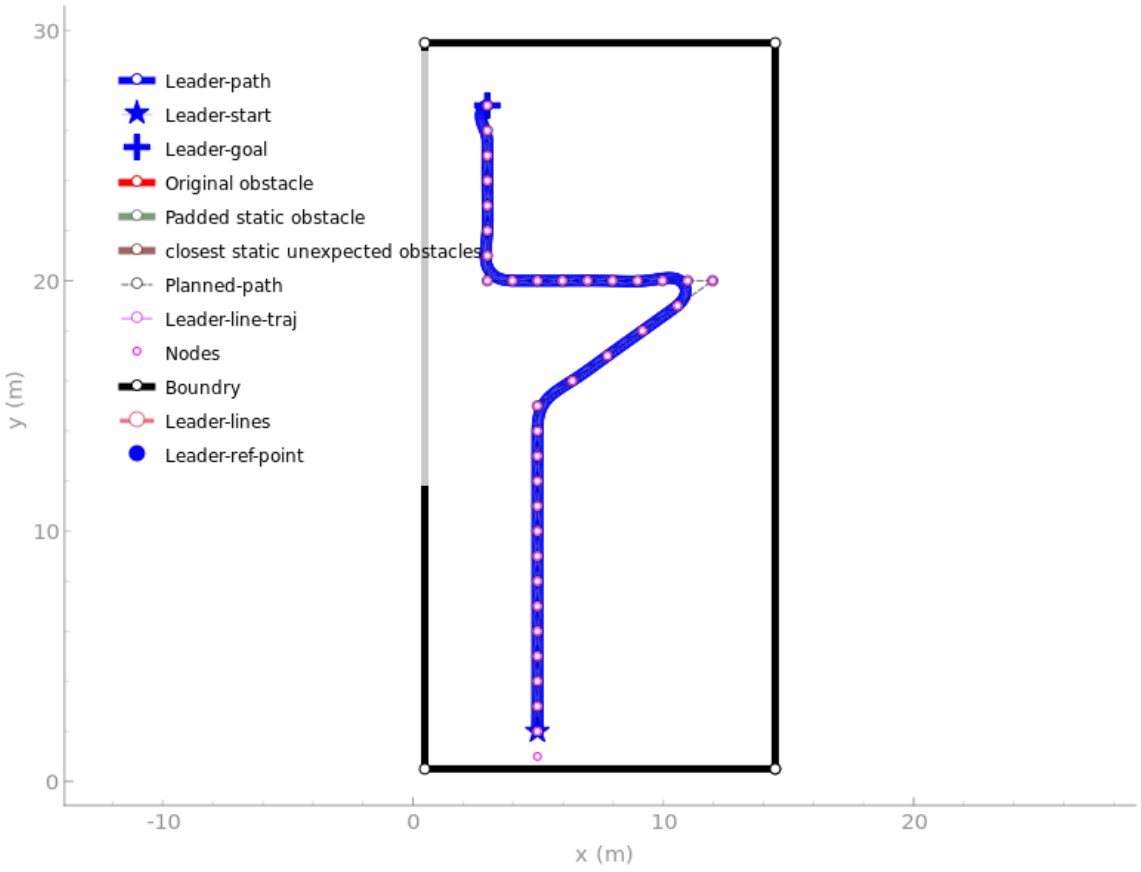


Figure 4.10: Trajectory planning positional results with only leader and no obstacles.

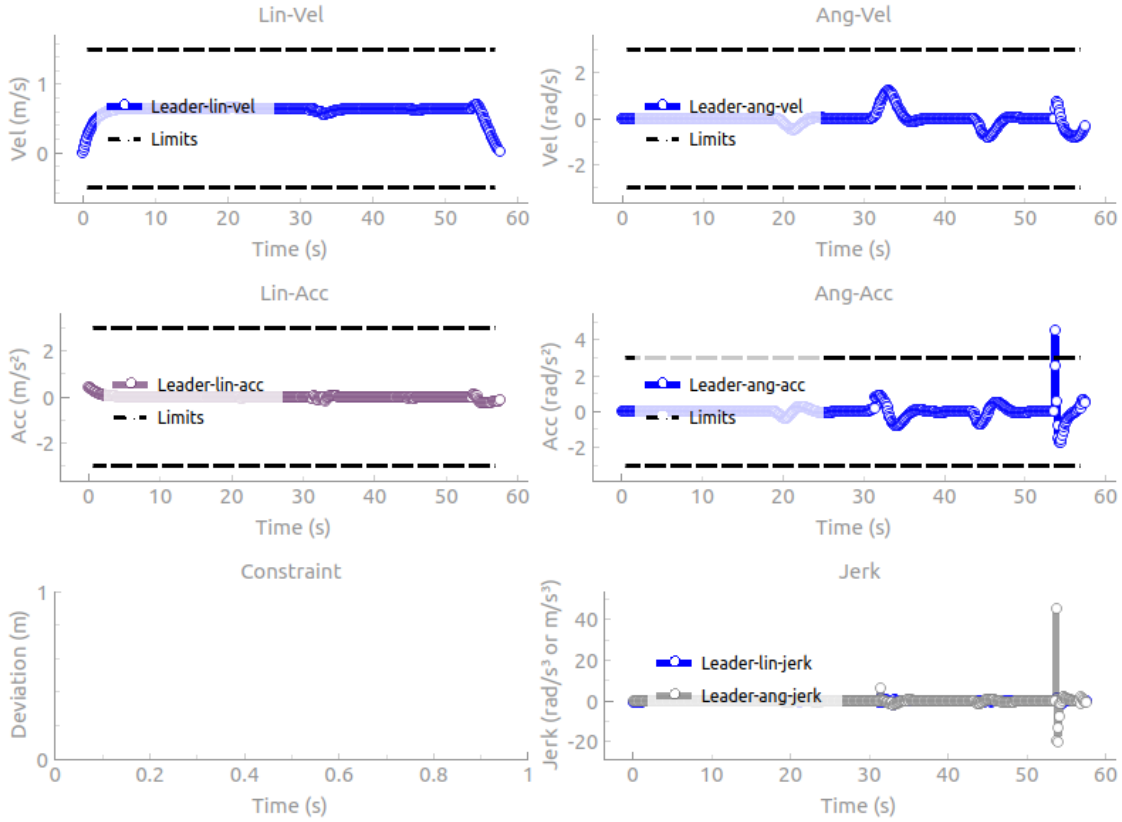


Figure 4.11: Trajectory planning data results with only leader and no obstacles.

Leader with static, unexpected and dynamic obstacles

The leader robot starts outside a node, the NMPC is adjusted to position goal behavior to drive to the first node in the path. When reached, the controller switches to the line follow behavior. Avoiding the first dynamic obstacle moving left to right by slowing down and letting it pass. The second obstacle is a unexpected and the path-planner adjusts the path around the obstacle. The second dynamic obstacle also avoided by slowing down. The last node is reached and the controller switches again to the position goal behavior to move to the final buffer.

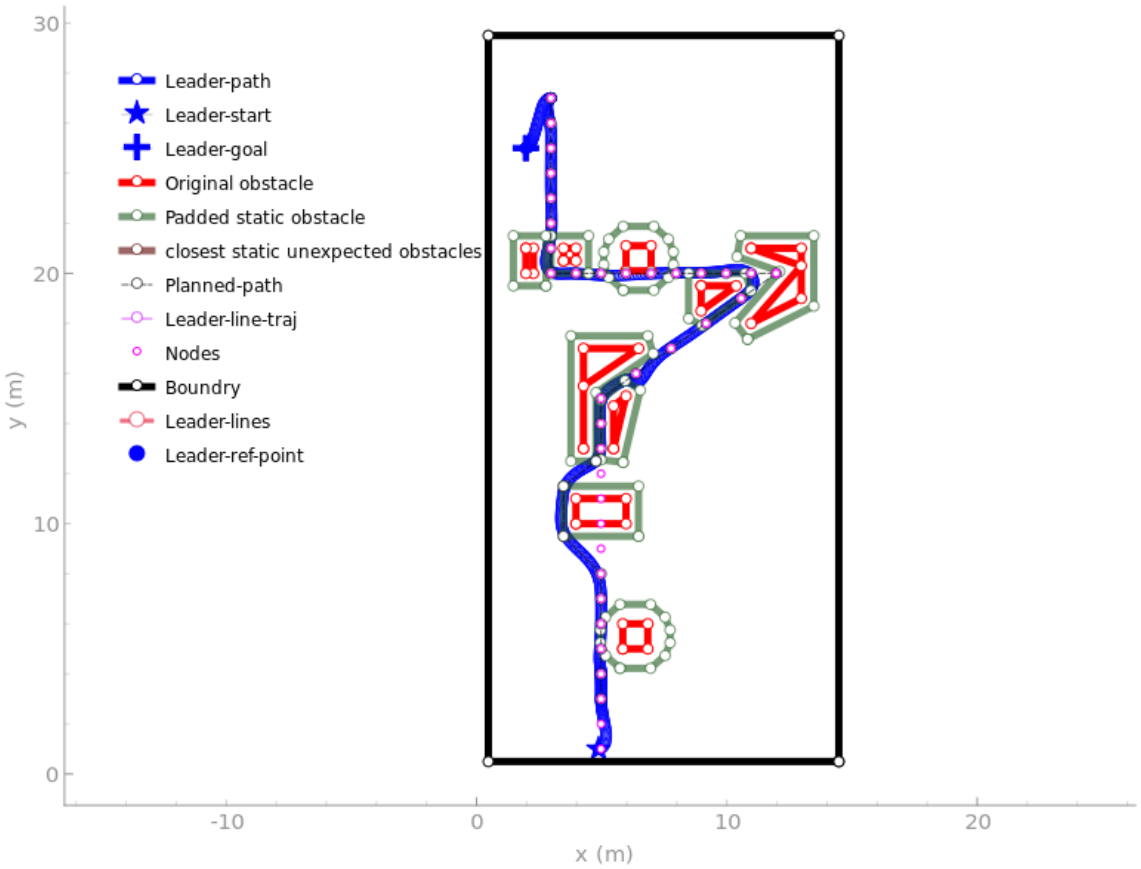


Figure 4.12: Trajectory planning results with only leader and static obstacles but with one to avoid and dynamic obstacles.

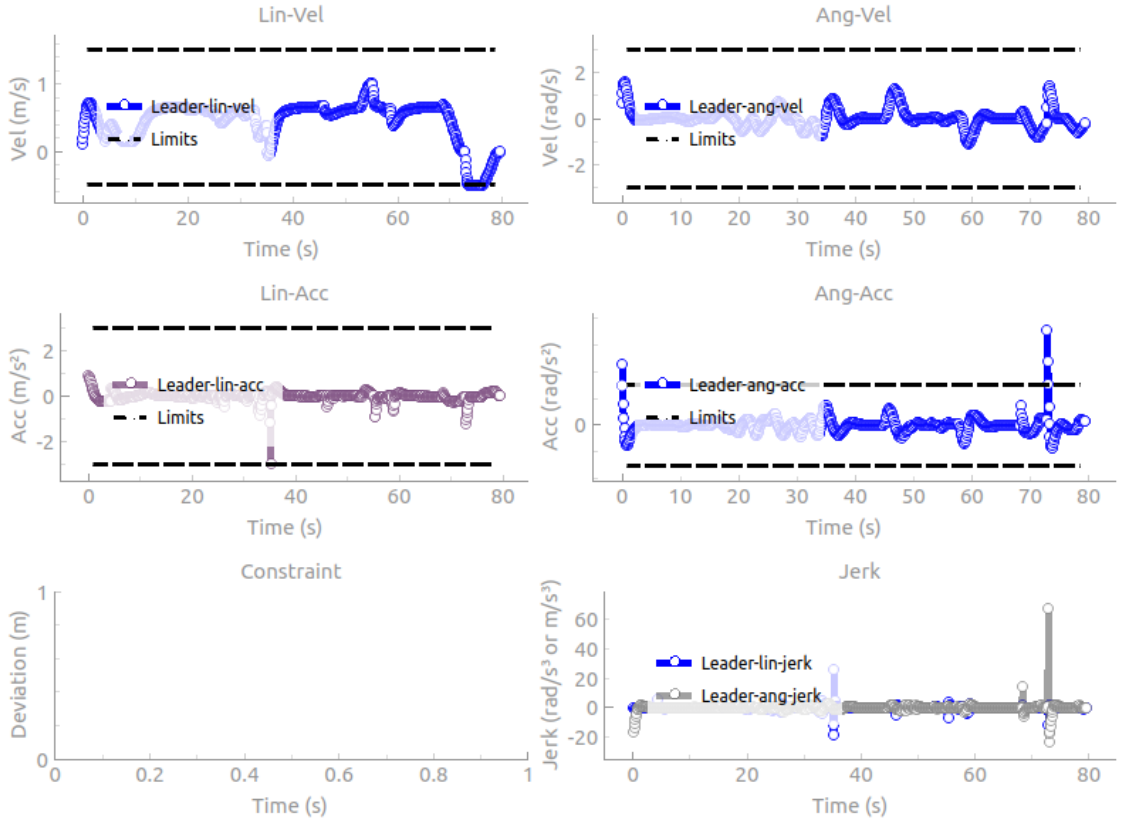


Figure 4.13: Trajectory planning results with only leader and static obstacles but with one to avoid and dynamic obstacles.

Leader and follower no obstacles

Same map as in Figure 4.10, but with formation control. Follows the given path and cuts the second sharp corner. Maximum distance between ATRs: (within constraints).

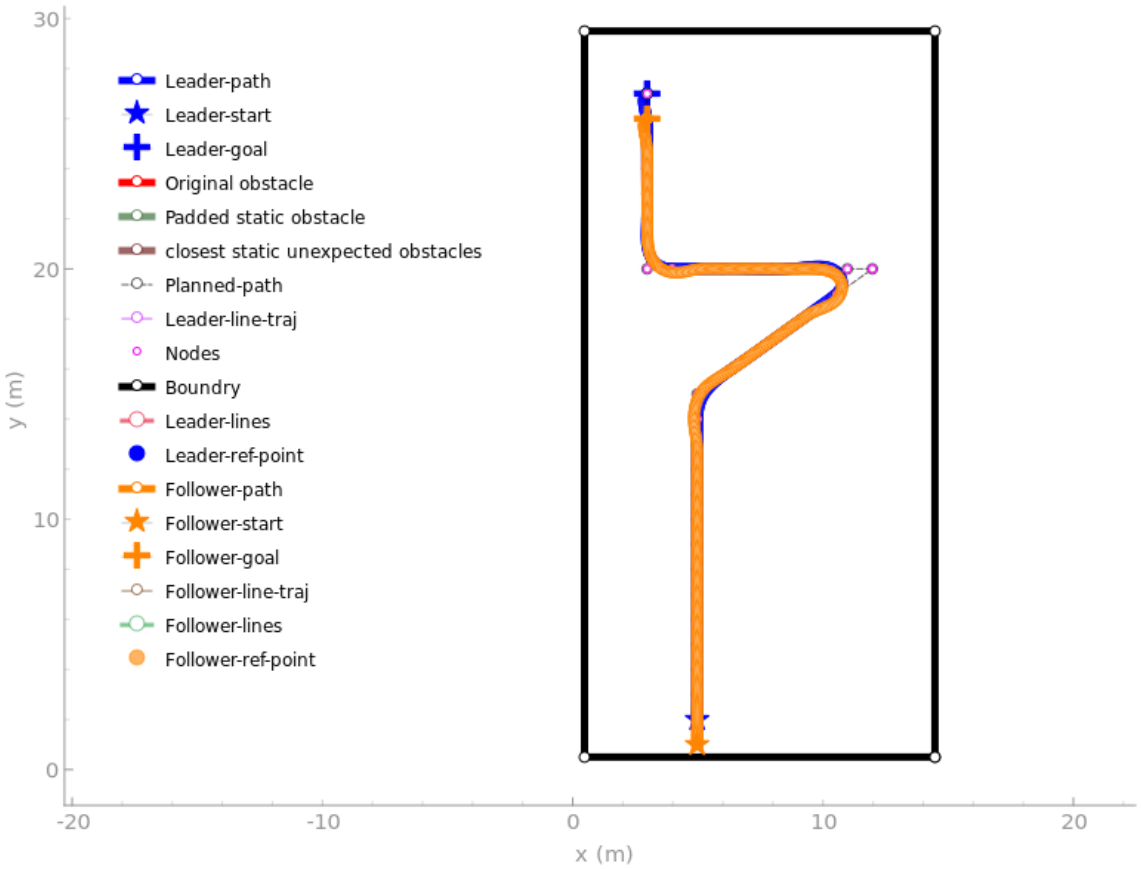


Figure 4.14: Trajectory planning results with only leader and follower and no obstacles.

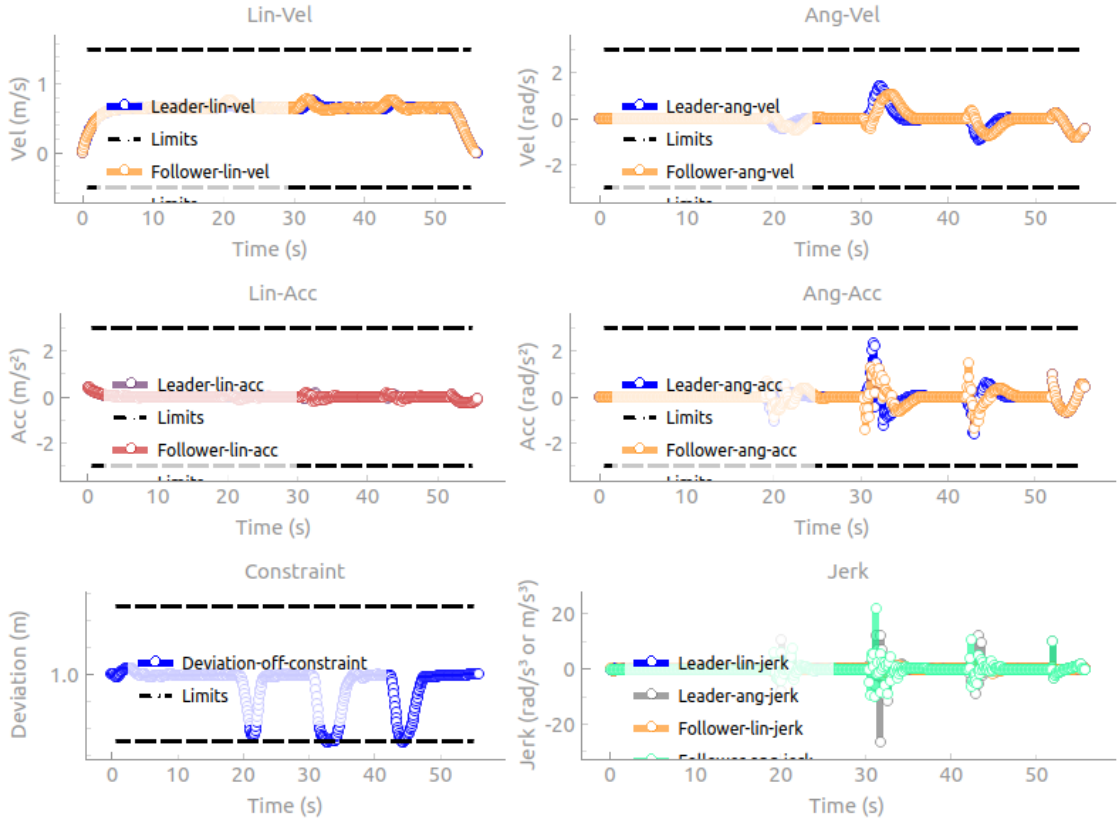


Figure 4.15: Trajectory planning results with only leader and follower and no obstacles.

Leader and follower with static, unexpected and dynamic obstacles

The leader and follower ATRs starts outside a node, the NMPC is adjusted to position goal behavior to drive to the first node in the path. When reached, the controller switches to the line follow behavior. Avoiding the first dynamic obstacle moving left to right by slowing down and letting it pass. The second obstacle is a unexpected and the path-planner adjusts the path around the obstacle. The leader ATR passes the second dynamic obstacle, the follower ATR follows but has to diverge from the path to avoid collision and to keep up with the leader robot. The same behavior can be seen after the first left turn. The last node is reached and the controller switches again to the position goal behavior to move the ATRs to the final buffer. As shown in Figure 4.18, the ATR cuts the corner and moves inside the static obstacle. As shown in Figure 4.19, the cost for cutting the corner is low, compared to being fully inside a obstacle.

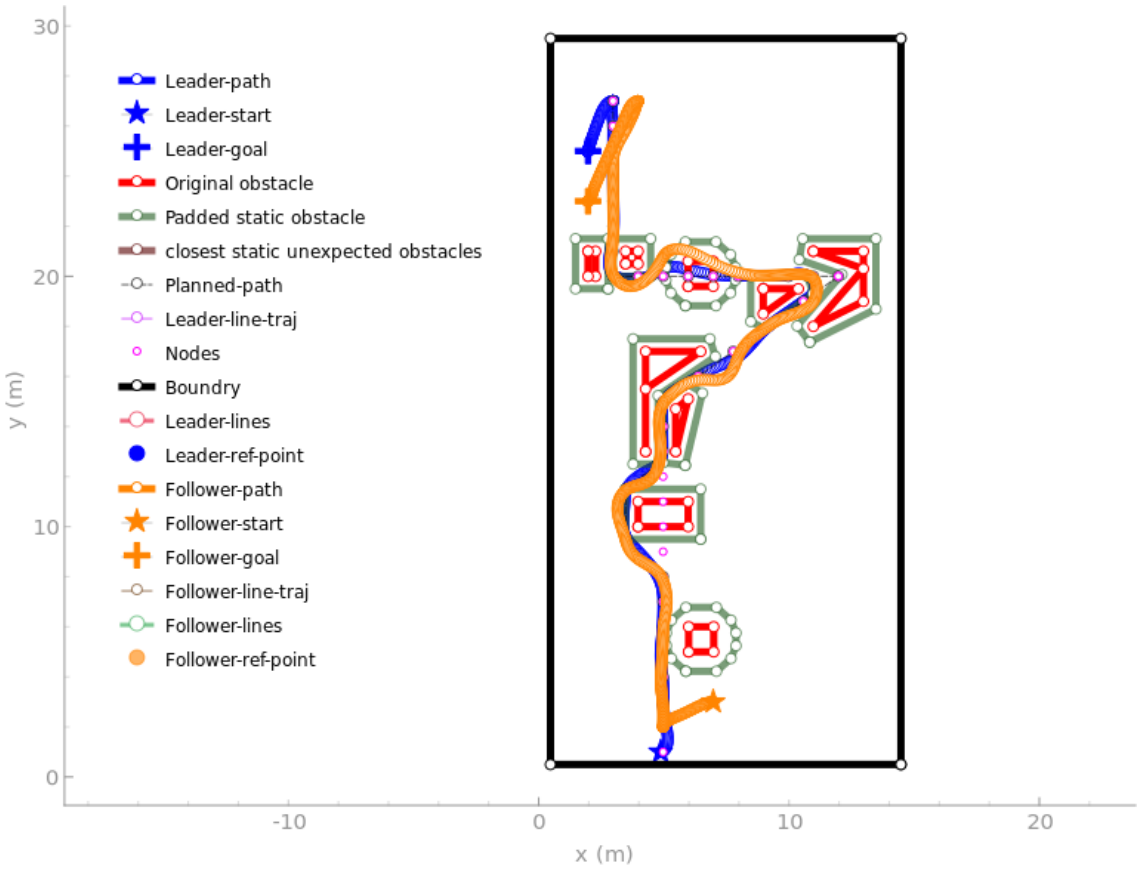


Figure 4.16: Trajectory planning results with only leader and follower and static obstacles but with one to avoid and dynamic obstacles.

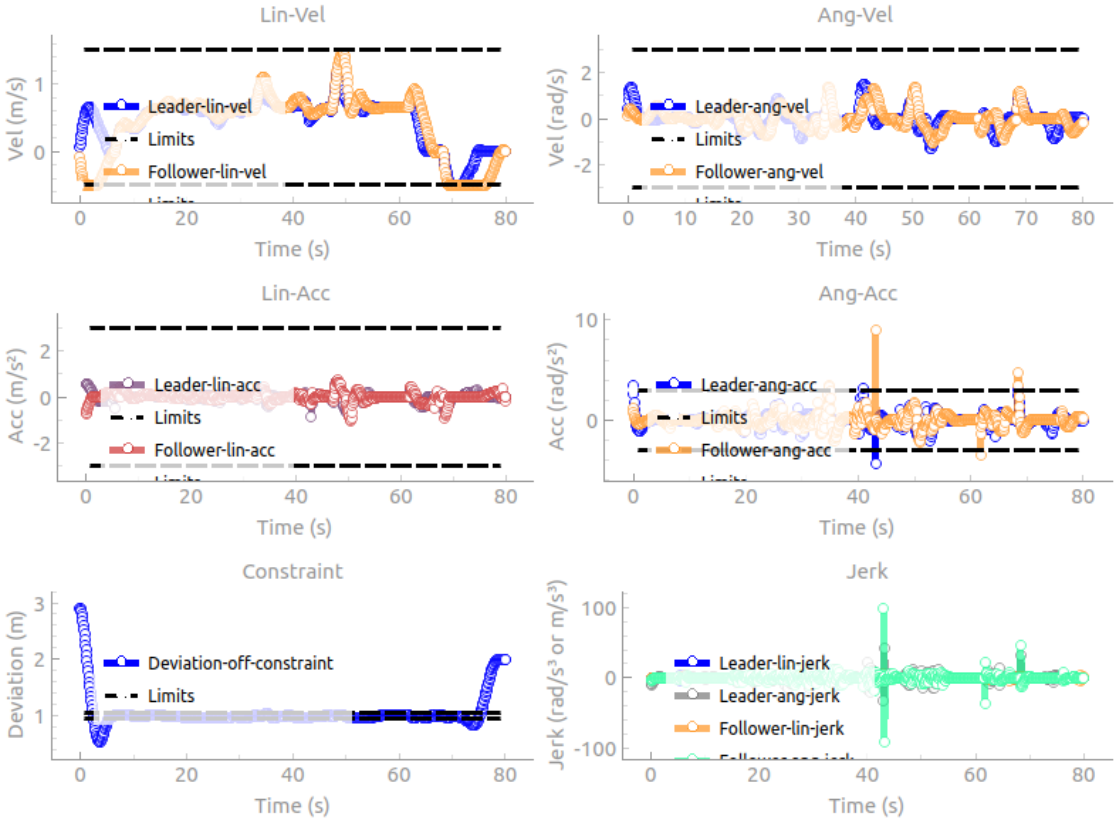


Figure 4.17: Trajectory planning results with only leader and follower and static obstacles but with one to avoid and dynamic obstacles.

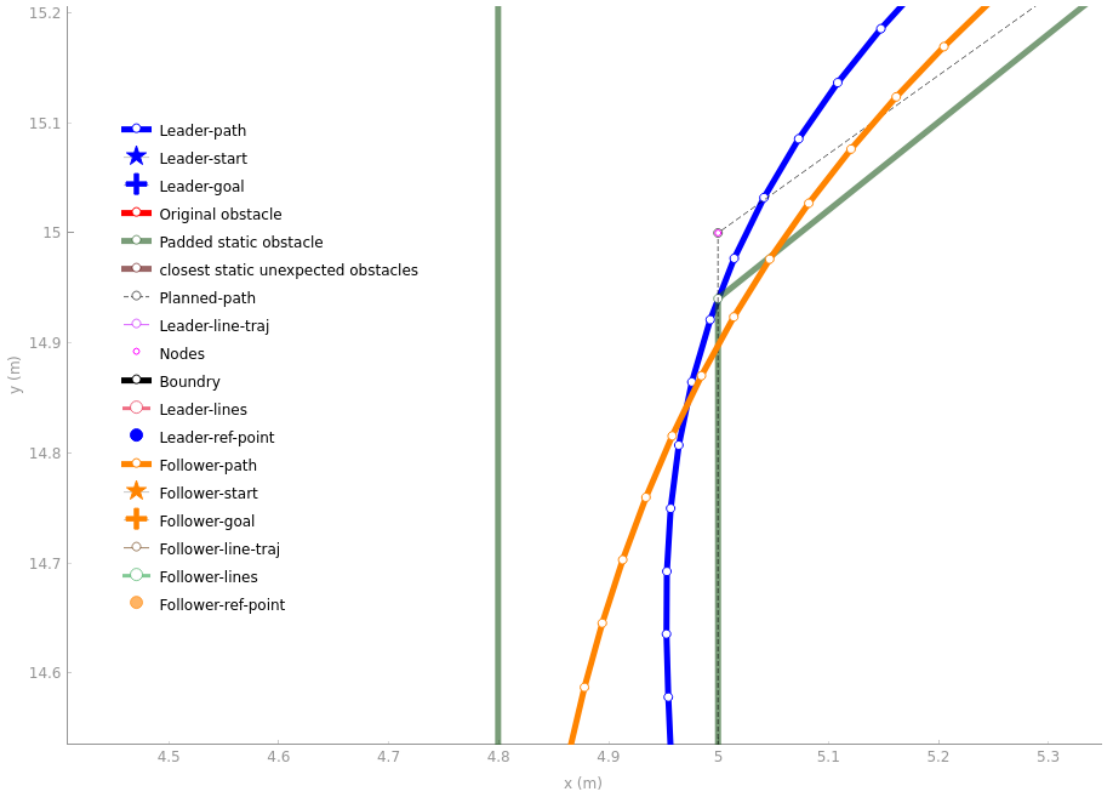


Figure 4.18: Trajectory planning results with leader, follower and static obstacles. Close up look at at the first corner.

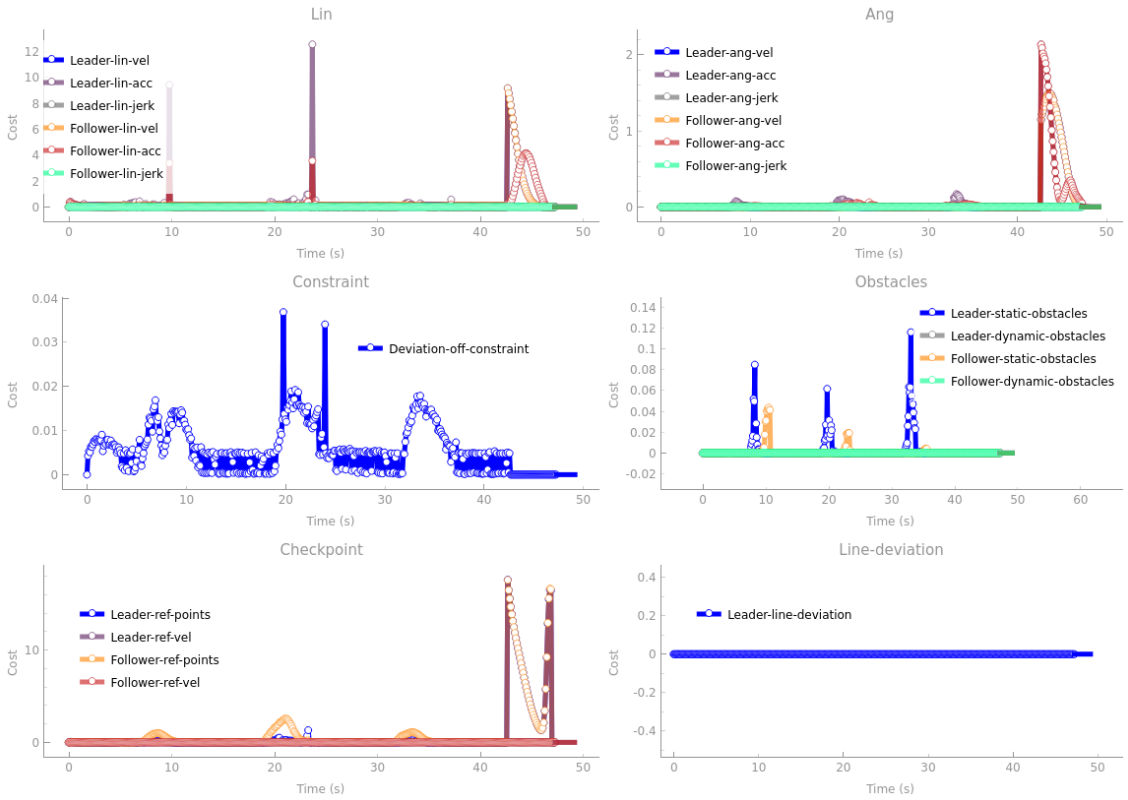


Figure 4.19: Trajectory planning results with leader, follower and static obstacles. Costs

4.5.2 Specific scenarios

Leader adjust

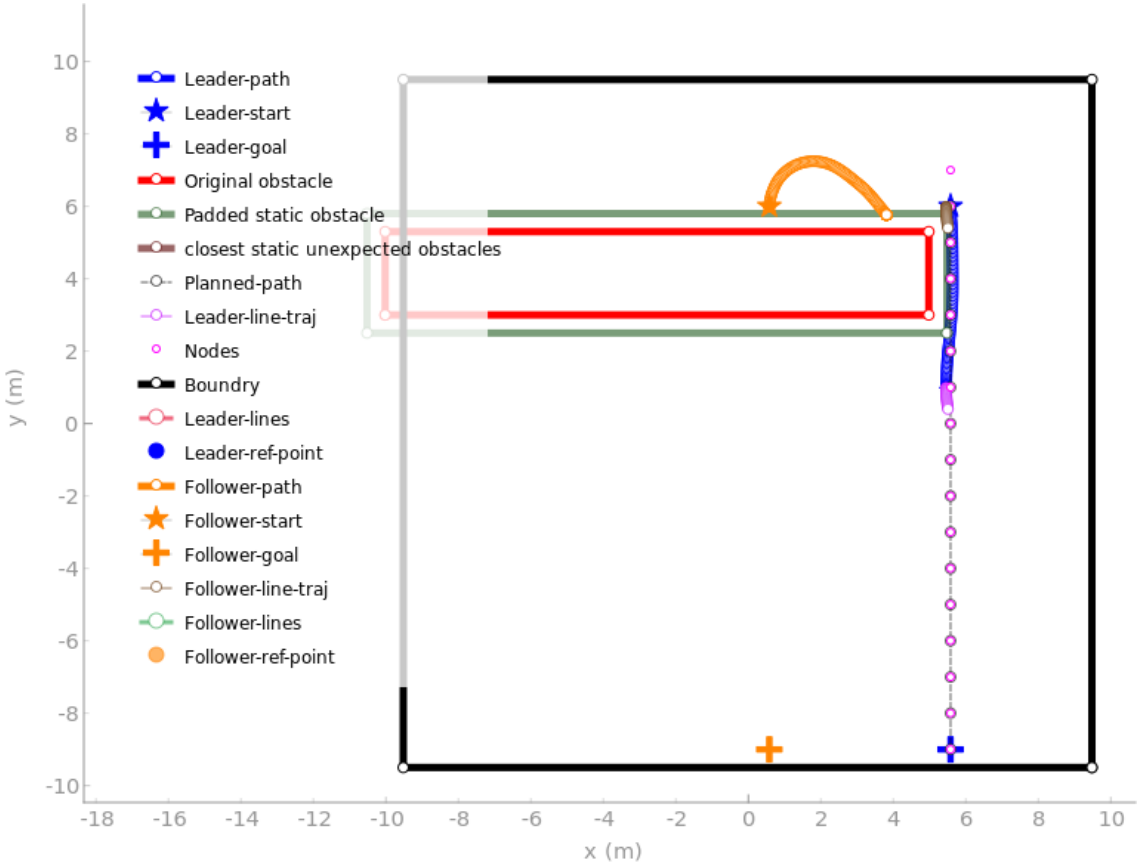
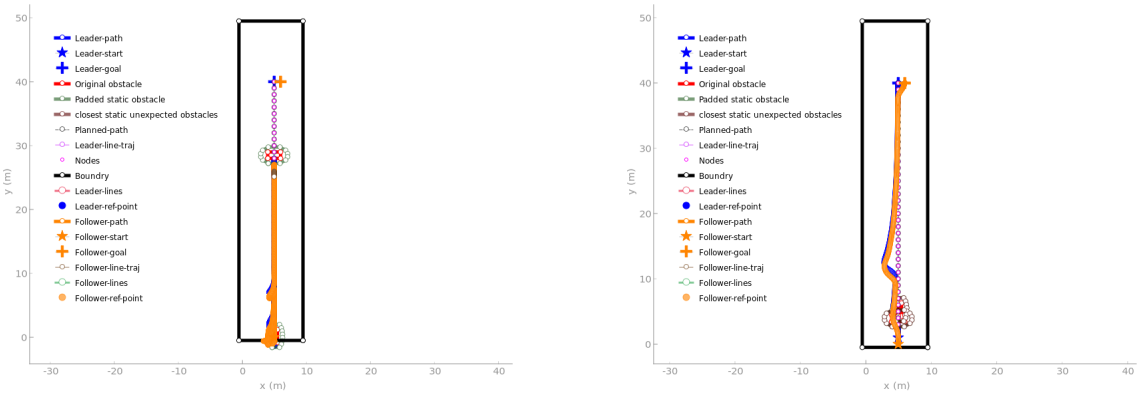


Figure 4.20: Trajectory planning results where leader has to adjust to follower, positional.

The scenario in Figure 4.20 is run with a distance of 5 m. The ATRs start 5 m apart and gets stuck when the follower ATR cant move around the static obstacle.

Dynamic obstacles straight ahead



(a) Low aggressiveness

(b) High aggressiveness

Figure 4.21: Trajectory planning results where dynamic obstacles are coming straight ahead, positional.

The scenario in Figure 4.21 was run with two different aggressiveness factors, one low and one high. In Figure 4.21a, the ATRs stops for the dynamic obstacle straight ahead, and when the obstacle is close enough, they start to reverse backwards and to the side, and wait until the obstacle has passed. The bigger obstacle forces the ATRs slightly into the boundary. In Figure 4.21b, the ATRs diverges from the planned path and avoids the obstacles without slowing down.

Unavoidable collision

The scenario in Figure 4.22 shows a dynamic obstacle moving towards the ATRs in a narrow passage. The ATRs first avoid collision by reversing towards the left wall. After the initial collision, they try to move inside the dynamic obstacle, violating the distance, dynamic obstacle and static obstacle constraint.

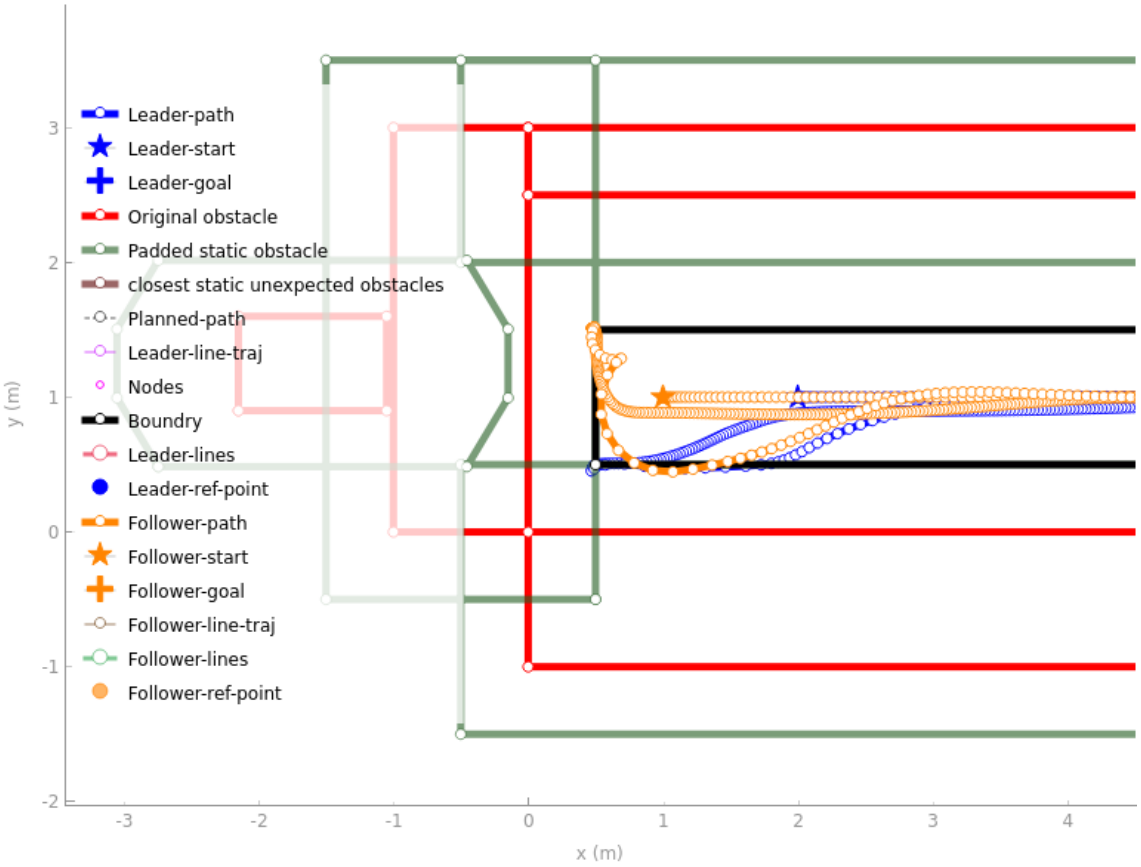


Figure 4.22: Trajectory planning results where a dynamic obstacle are coming straight ahead, with nowhere to go.

4.5.3 Constraint satisfaction

The maximum logged delta distance between the ATRs during simulations. The delta distance is the deviation from the defined fixed distance. The constraint during the simulations is set between 0.95 and 1.05 m.

Scenario	**Max ATR distance [m]**
Leader and follower, no obstacles	0.004759
Leader, follower, static and unexpected obstacle	0.041063
Dynamic obstacle straight ahead, (a)	0.049947
Dynamic obstacle straight ahead, (b)	0.031476
Leader adjust	0.050007
Unavoidable collision	0.406017

5
Software architecture

In this chapter the software used is explained. First the way the simulated results are achieved is explained. Then the chosen NMPC solver, PANOC, is explained. The code for the solver is automatically generated via OpEn. OpEn is lightly explained. Finally the communication between this thesis and the rest of the system is explained.

5.1 Simulation environment

The simulations were run with Python on a computer running Ubuntu 20.04 with an i7-6700K and 16 GB of ram. As shown in Figure 2.1 this thesis is a part of a bigger system. When simulating, the other systems also has to be simulated. The manager was simulated by hard coding nodes, start and goal states. The vision module provides positions of static and dynamic obstacles. The static obstacles were hard coded to fixed positions. The dynamic obstacles were moved backwards and forwards in a predefined path. This gave the predicted positions. The ATR module combined with the vision module provides the position of the ATRs. This was simulated by hard coding a start position for the ATRs and updating it by the control signals calculated by the trajectory generator.

5.2 PANOC

PANOC stands for proximal averaged Newton-type method for Optimal Control [17] and is a fast solver for NMPC problems. Normal NMPC solvers use sequential quadratic programming which requires cumbersome calculations. PANOC is a line-search method which uses forward-backward iterations and Newton-type steps. It has proven to work for similar kinds of work as this thesis [12]. Complex constraints are possible and can be formulated using either Penalty method or augmented Lagrangian method [11], their formulations are shown in Equation (5.1) and Equation (5.2) respectively.

$$F_1(u, p) = 0 \tag{5.1}$$

Where u is the control signals and p are states or combination of states, i.e. x_1 - x_2.

$$F_2(u, p) \in C \tag{5.2}$$

5.3 Opengen

OpenGen [2] and CasADi [18] combined provides a framework for creating optimisation problems and generating Rust [19] code for fast execution time.
The framework enables the use of Python to create and setup the optimisation problem in a structured way, and then generate configurable Rust code. The generated code can be called via TCP/IP, python binding for C/C++ or the actual Rust code. The number of parameters and variables needs to be pre-allocated. This impacts mostly the number of obstacles and vertices the solver should consider, and it needs to be a fixed number. To enable the ATRs to move in an environment with an arbitrary number of obstacles, a relatively small amount of obstacle parameters with various amount of vertices is pre-allocated. A module outside the solver calculates the distance to every obstacle and only sends the closest obstacles up to the fixed amount. If the amount of obstacles is less than the fixed amount, the parameters are filled with zeros. In the case of dynamic obstacles, filling with zeros results in a constraint that is always violated, therefore dynamic obstacles also have additional parameters to determine if the obstacle is active or not.

5.4 ROS

Robot Operating System (ROS^{TM}) is used to communicate between the different modules in the system, Figure 2.1 shows the complete system. Specifically ROS2 Foxy [20] is used. ROS2 supports two main ways of communicating. The first way is topic based and the second is service based. The topic works by having nodes subscribe to a topic. Then whenever a node publishes a message on that topic the subscribing nodes receive the message. The service communication works by a client requesting data. The request is serviced by a server. All of the communication is done via Ethernet or WiFi.

To generate a trajectory, the formation control module needs to subscribe to different topics and request from services. Information which changes rapidly is communicated via topics and slowly changing information is communicated via services. Information about static obstacles and the ATRs' positions is sent via topics. Non accessible areas (NONA) is static obstacles which can never move or disappear. Information about NONA, dynamic obstacles, A* nodes and start-stop positions was service based.

6

Use case

The real world example was constructed in Volvo Group's Pilot plant in Gothenburg, Tuve using the modified Husqvarna autonomous lawnmowers 2.2. The different master thesis modules were connected together and communicating via ROS2.

6.1 Pilot plant

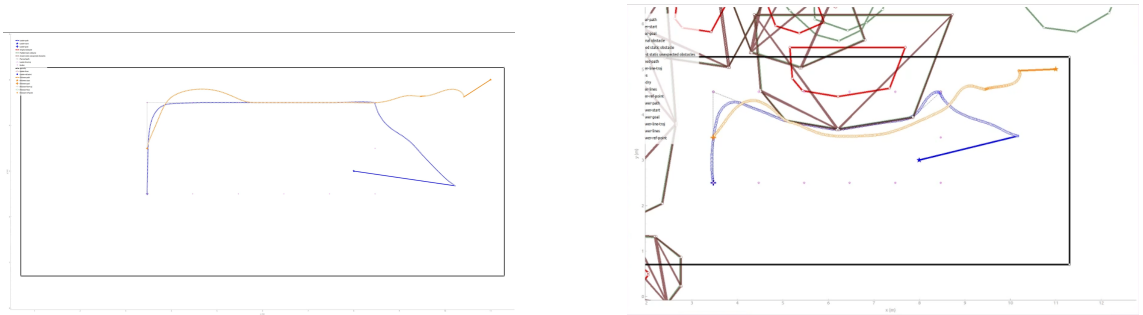
The demo consisted of a defined road map, seen in Figure 6.1 as the yellow tape. A node was placed every 1 m along the tape. In the following tests, our system will plan a trajectory from one side to the other, with and without a pallet blocking the path.

Figure 6.1: First view of the pilot plant

6.2 Test drive

By continuously sending a short trajectory with length two seconds, the ATRs did not move synchronized, due to the use of delta times, and delays in the complete system. To bypass this, Formation control was tested by sending a complete trajectory to the ATR control module.

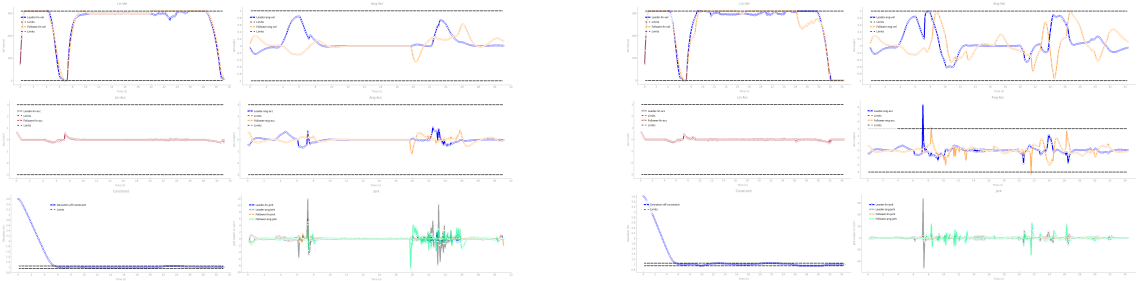
The formation tests included a coupling sequence, where the ATRs start at a distance greater than their formation distance, and then moved to the formation distance, then moving together in formation state, with a fixed distance between them. There were no decoupling positions. The simulated trajectories are shown in Figure 6.2.



(a) Without pallet (b) With pallet

Figure 6.2: Simulated trajectories for real world test scenarios.

Due to the simulated data being lost, a reconstructed simulation of the trajectory generation was done and shown in Figure 6.3. Maximum simulated distance from the fixed distance without pallet was 0.004661 and with pallet: 0.050070 m. Where the fixed distance $d = 1$ m, $D_{min} = 0.95$ m and $D_{max} = 1.05$ m.



(a) Without pallet (b) With pallet

Figure 6.3: Simulated data plots for real world test scenarios.

Figure 6.4 and Figure 6.5 show the observer data and the reference trajectory read by the ATR control module. The observer used the odometry from the wheels and the motion model to calculate the current position, this is the position that the ATR control module uses. The real world position could not be measured by the cameras in real time due to delays.

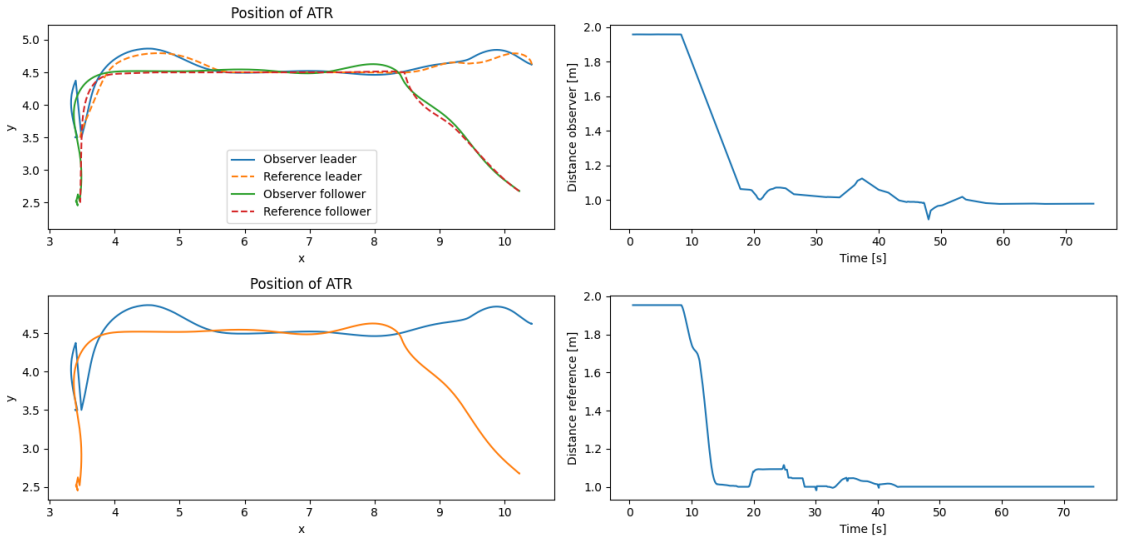


Figure 6.4: Real world test drive without pallet

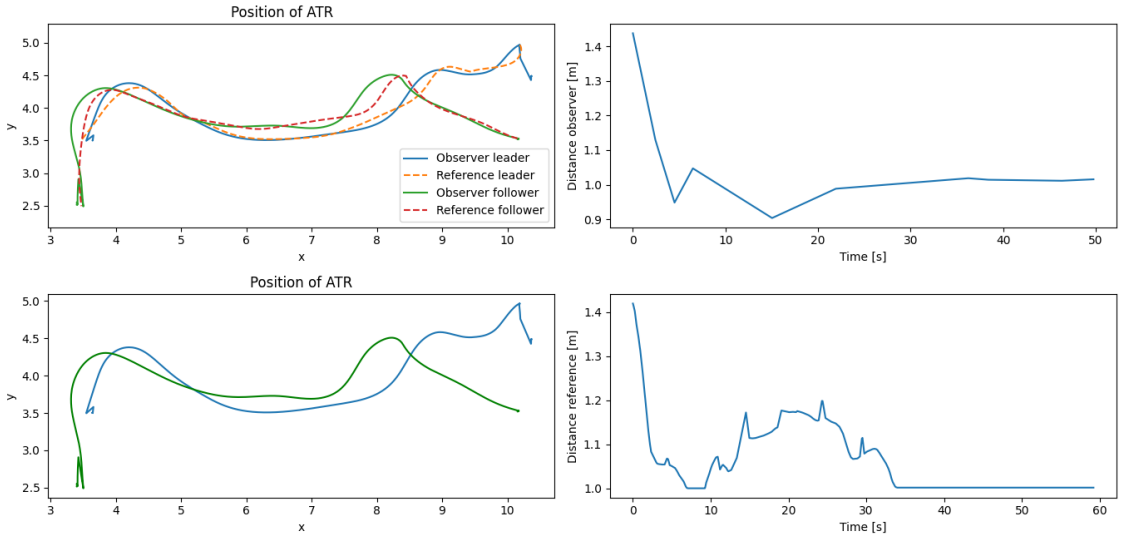


Figure 6.5: Real world test drive with pallet

The ATR Control module has some problems following the reference trajectory. This could be due to it's controllers or that the models of the ATRs are not perfect and as such the calculated control signals do not yield exactly the wanted results and may request physically impossible movements.

Due to delays in the system the ATRs must be stationary for three seconds before the positions are updated using the visual tracker positions. While moving the ATRs odometry positions drift and then at the end they update with the visual information. This results in the positions making a jump at the end of the run, noticeable in Figure 6.5.

The maximum real world distance from the fixed distance can not be computed exactly as it is not known when the ATRs moved to the formation state. It can be gauged from the plots that without the pallet this happened at $t \approx 13$ s and with pallet at $t \approx 8$ s. With this the maximum reference distance was 0.1 m and actual distance was 0.4 m without pallet. With pallet the maximum reference distance was 0.1 m and the maximum actual distance was 0.2 m.
and with this that the maximum distance is about 0.1 m and 0.2 m.

7
Discussion

This chapter aims to discuss the results and methods used and how to improve them.

7.1 Simulation

Here the simulated results of the path planner and trajectory planner are evaluated.

7.1.1 Discussion of path planner results

Figure 3.7 shows that the path planner works as intended and behaves as we would like it to. It first performs A* and then does a visibility graph around the static obstacle. A problem with our path planning approach is shown in Figure 3.8. This path deviates off the road a lot and firstly goes into a dead end. The reason for it being bad is our path plan approach. First A* finds the shortest path without considering the obstacles. This is a straight line from start to goal. Then a visibility graph is computed. The graph is computed from the first node which is at least a distance of d_{VG} away. This is why it moves into the dead end. As the visibility graph does not consider the nodes the resulting path is one which almost completely disregards the nodes.

The problem described above gets worse and worse the bigger the obstacle is. It is especially bad when the obstacle is big enough to block a passage completely. Then the visibility graph is forced to find a new path which goes through a new passage. When going through this passage the ATRs should be close to the nodes but the visibility graph does not consider the nodes so the resulting path is not close to the nodes.

7.1.2 Discussion of trajectory planner results

General scenarios

The scenarios when there is no dynamic obstacles work well. The ATRs move around the static obstacles and follow the path well. They keep relatively low accelerations and keep the distance between themselves. They cut the corners somewhat, especially in sharp corners. This provides a smooth behaviour similar to how a human would move.

Dynamic obstacles provide a tougher problem. As safety is important, the aggressiveness of the NMPC is tuned to be low. The passive behaviour can be seen in the start in Figure A.7. There the leader and follower movements over lap with the path. So they have waited for the dynamic obstacle to pass before moving. At the middle of this scenario the follower has deviated from the path while the leader has not. This is because of the formation approach used. The line following trajectory only considers the leader so when the follower trajectory is calculated it is inside the dynamic obstacle. To prevent collision the reference following trajectory adjusts the follower states. This leads to the follower deviating off the path. The preferred behaviour would be for both the ATRs to speed up or wait for the obstacle to pass.

At the end of the scenario there is another dynamic obstacle which also is not handled well. Here both the leader and the follower deviated. This behaviour is not seen when only the leader is being run, see Figure A.10. This suggests that it is related to timing. Depending on where the obstacle is when the ATRs arrives to it, the ATRs do or do not deviate off the path. It could be so that if the horizon in the NMPC was longer this could be avoided as the longer planning would wait for the obstacle to move to a better position before moving.

The constraints imposed on the ATRs are obeyed to varying degrees. The velocities are within their bounds in all the scenarios. The accelerations go outside of the limits shown in the graphs but these limits are not constraints but rather suggestions. All constraints, including obstacles, are formulated as soft constraints. So there is no guarantee that the trajectories will obey them. The corresponding weight matrices are high so the NMPC wants to reduce to violations but as long if the violations are small the cost is insignificant.

The distance constraint violations were at max 0.000007 m. This is within the tolerance of the NMPC solver and small enough to probably not matter in practice. The other interesting constraints are the obstacles.

Due to the nature of the obstacle formulation the violations of being inside an obstacle is low even when the ATRs are multiple centimeters inside of them. This means that the cost may be low enough to favour being inside of an obstacle by a significant amount, see Figure 4.19. This problem is further enhanced by the NMPC not considering how the ATRs travels between time steps. This means that between two time steps, the ATRs can be inside obstacles without the NMPC knowing and therefore not being able to do anything about it. Increasing the padding can prevent both of these problems. This can however introduce artificial barriers and it is difficult to know by how much to increase the padding. Also there is still no guarantee that this avoids crashes, meaning it possible that the optimal solution is inside an obstacle.

Specific scenarios

The scenario in Figure A.7 shows that trajectory planner struggles when the leader has to adjust to the follower. In this scenario the line following trajectory would take the leader straight down to the goal. What the system needs to progress how ever is quite different from that. It needs the leader to be almost stationary while the follower moves to the path. As the system is based on the follower adjusting to the leader this is naturally a hard task.

Another problem highlighted in Figure A.7 is that the load is not modelled in the NMPC. Any load the ATRs are carrying together has crashed into the obstacle by the time they are standing as they do.

The trajectory planner struggles when dynamic obstacles come straight ahead. This is shown in Figure A.7. When the dynamic obstacle get close the ATRs could move around it. This would mean deviating from the path. This means an increased cost and the choice with lower cost is to stay close to the path and avoid the obstacle. So the ATRs start going backwards. Once they are pushed back all the way to the boundary the start deviating from the path to avoid it. This behaviour is probably perceived as dangerous and erratic in real life.

The preferred behaviour is that the ATRs either stop and wait for the obstacle to pass before continuing forward or move around the obstacles. By increasing the aggressiveness the ATRs can move around them, see Figure 4.21b. The drawback of this is that the ATRs do not only become more aggressive when dynamic obstacles come straight ahead but in all scenarios.

Due the implementation of soft constraint, the ATRs are able to move inside obstacles and the solution converges, but at a high cost. This behavior is shown in Figure 4.22, Where a collision is unavoidable, the ATRs finds a intrusive solution, but violating almost all constraints. This might not be the preferred behavior, a safer option might be to just stop.

7.1.3 Discussion of formation approach

The formation approach has two key problems. Firstly, it is possible that the offset positions are inside an obstacle or out of bounds. When this happens the NMPC can avoid the collisions but the resulting trajectory can be unexpected, see Figure A.7. This is not desired in a production environment. The second problem of the formation approach is that the leader has a hard time adjusting to the follower. This can lead to unexpected dead locks where human intervention is required. As a result the system feels untrustworthy and is not autonomous.

7.1.4 Discussion of line deviation cost

Humans tend to slow down in corners, it feels naturally safe to do. The ATRs tend to speed up in corners, even above the reference velocity. This is probably due to a variety of costs but especially the line deviation cost. Figure 4.10 shows the trajectories cutting the corners. In doing so the line deviation cost increases. To minimize the cost the NMPC wants to quickly get close the line again and speeds up, see the data plots 4.15. Speeding up in corners feels unnatural and unsafe to humans. It is likely that the speed up in corners behaviour can be damped by tuning the NMPC weights but it is an inherent property of the line follow approach.

7.2 Practical application

A test case between all of the groups was constructed in the Pilot plant. To test the communication, a simplified version of each system was used to record a demo. This thesis part in the demo was to plan a path for a single ATR to a given goal position, avoiding a static pallet. The code was modified to fit the needs of the physical system and the other modules. Different weights to produce a slow and smooth trajectory, but only 10 % of our trajectory points was used in the Fleet-control module.

The formation control was also tested on two physical ATRs. The plots in Figure 6.4 and 6.5 shows that the ATRs does not completely follow the given path, nor that the reference path can hold the fixed distance. Since the simulated data shows no constrain violation, an investigation about the error could be done. First off, it is possible that the ATR trajectories is not synced, since only delta times are used, and this could lead to the ATRs following trajectories at different time steps. This possible error could be resolved by introducing timestamps in each message, to sync the trajectories. The ATR control module slightly alters the given trajectories by introducing more points, this should not result in a big error, since the trajectories from our module is with a resolution of $t_s = 0.1$ s and the ATR control module $t_s = 0.01$ s.

7.3 Research questions

This sections evaluates the the research questions of this thesis.

7.3.1 Research question 1

- **Research question 1**
  Implement a global path planner that finds the shortest path between two points, given a road network. Also implement a complementary local path planner, using visibility graphs, which avoids obstacles along the path. Evaluate controller compatibly and quality of the solution.

The quality of the solution provided by the path planner depends on the scenario. The global path planner ignores obstacles and produces the shortest possible path which follows the roads. So if there are no static obstacles the solution is optimal. The local path planner modifies the path to avoid obstacles. The path around an obstacle is optimal in that it is the shortest possible. Due to the obstacle avoidance starting a fix distance before the obstacle, the path can go into a dead end and turn around. The same can happen after the obstacle. In general the solution quality after the local path planner is good if the obstacles are well shaped.

The trajectory planner is able to follow the path provided by the path planner. The path should not be followed exactly so it is difficult to give an analysis on how compatible it is. We feel that the trajectory planner is following the path as a human would do. It cuts corners and deviates when needed to avoid obstacles but returns when possible. So we conclude that the compatibility is good.

7.3.2 Research question 2

- **Research question 2**
  Evaluate how well a PANOC NMPC controller solves the issue of having two ATRs follow a given path while obeying constraints and avoiding dynamic obstacles. Evaluate it with regards to robustness and violations of the constraints in practice compared to simulations. The constraints can be of varying kind, for example a fixed or greater than distance between the robots.

As mentioned in 7.3.1 Research question 1 the PANOC NMPC follows the given path well. How much it violates the constraints and obstacles varies. While simulating normal scenarios, the distance constraint violations were at max 0.000007 m. We deem this insignificant and that the NMPC obeys the distance constraint fine. During an unavoidable collision, the distance constraint was violated with 0.356017 m. This is a significant distance, but it happened during a scenario were the collision constraints were already violated. The static obstacle constraint where violated by a couple of centimeters, common around the obstacle corners. Since the padding is over-estimated around the corners, this was considered fine.

Due to time limitations the real world results achieved were not satisfactory enough. This is not due to the PANOC NMPC however. The PANOC NMPC worked well in practice. It was fast enough for real time control, reliable and obeyed the constraints.

7.4 Future work/improvements

In this section, we discuss different improvements on the system that we think might be the next step.

Transported object collision

A known problem is if the leader and the follower is driving around a sharp corner with large enough distance constraint between them, the transported object between the robots might collide with the obstacle. Similar if a sharp dynamic obstacle moves fast in between the robots and into the transported object. This is due to the point mass modelling of the robots as two points with padded obstacles, i.e the solver does not have any information about the transported object and its size. In [21] a collision avoidance method is shown where the controlled object is described as the union of convex sets. Implementing this would enable the solver to constrain an arbitrary large object from colliding with objects. The thesis also mentions that constraining additional points along the edges of the controlled object, which might be a good enough solution.

Minor constraint violation

Due the implementation of soft constraint, the solution might violate the constraints by a small amount, but at a very high cost in the objective function. The benefit is faster and easier converge. This leads to the ATRs might take the least intrusive trajectory when the objective function already have a high cost. If the ATRs is to be moving alongside humans, putting hard constraints on obstacle avoidance and/or distance might be an extra level of safety. This thesis did consider both the built in penalty and augmented Lagrangian constraints, but the results where not stable enough to justify not having soft constraints. Actual hard constraints only works directly on the decision variable in OpEn.

Leader adapting to follower

As seen in Figure 4.20 the leader might have trouble adjusting to the follower, this might be due to the implementation of the two-stage solver as a leader-follower approach, the leader does not have any constraints or information about the follower in the first solution. Further development might be to investigate a single stage solver without the leader-follower approach.

Safe collision

As shown in Figure 4.22, when a collision is unavoidable the ATRs still move inside the obstacles and violating constraints. A safer option would be to just stop. Various solutions to this problem where tested, such as adding a cost for velocity proportional to the closest distance to an dynamic obstacle inverted, or multiply the velocity with the obstacle avoidance condition 4.15. Due to time limitations, a stable solution was not implemented.

7.5 Conclusion

This thesis handled the problem of path- and trajectory planning for one or two ATRs working in tandem to transport goods in a production environment. The path planning was solved using a combination of A* and a visibility graph. This proved to work well when the static obstacles were relatively small and not too intrusive on the available areas. The trajectory planning was done with a NMPC which could be in several modes. This worked relatively well with both one and two ATRs. It had problems with dynamic obstacles but as long as they did not come straight ahead it worked well. The essence of the formation control was that the follower adjusted to the leader. When the opposite was required to achieve a good solution the system struggled. The physical testing provided a proof of concept, but more work is needed to make it a stable solution.

Bibliography

[1] K. Pålsson and E. Svärling. Nonlinear model predictive control for constant distance between autonomous transport robots. Gothenburg, Sweden, 2020.

[2] P. Sopasakis, E. Fresk, and P. Patrinos. OpEn: Code generation for embedded nonconvex optimization. In *IFAC World Congress*, Berlin, Germany, 2020.

[3] A. Karlsson J. Berlin, G. Hess and W. Ljungbergh. Long-range trajectory generation with obstacle avoidance using non-linear model predictive control. 2021.

[4] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[5] Tomas Lozano-Perez and M. A. Wesley. An algorithm for planning collison-free paths among polyhedral obstacles. *Communications of the ACM*, 22:560–570, 1979.

[6] Sebastien Gros. *Modelling And Simulation*. 08 2019.

[7] Karl Johan Åström and Björn Wittenmark. *Computer controlled systems : theory and design (1 ed.)*. Prentice-Hall, 1984.

[8] Bo Egardt. *Model Predictive Control*. 01 2020.

[9] Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. 01 2002.

[10] Carlo Novara. *Nonlinear control and aerospace applications, Nonlinear model predictive control, lecture notes*. 08 2019.

[11] Michael Patriksson, Niclas Andréasson, Anton Evgrafov, Zuzana Nedělková, Kin Sou, and Magnus Önnheim. *An Introduction to Continuous Optimization - Foundations and Fundamental Algorithms*. 11 2016.

[12] Ajay Sathya, Pantelis Sopasakis, Ruben Van Parys, Andreas Themelis, Goele Pipeleers, and Panagiotis Patrinos. Embedded nonlinear model predictive control for obstacle avoidance using panoc. In *2018 European Control Conference (ECC)*, pages 1523–1528, 2018.

[13] David P. Dobkin Bernard Chazelle. Optimal convex decompositions. *Computational Geometry*, pages 63–133, 1985.

[14] L. Paul Chew. Optimal convex decompositions. 1989.

[15] Sean Gillies et al. Shapely: manipulation and analysis of geometric objects, 2007–.

[16] Philippe Feyel. Safe formation control with constrained linear model predictive control. pages 70–81, 01 2019.

[17] Lorenzo Stella, Andreas Themelis, Pantelis Sopasakis, and Panagiotis Patrinos. A simple and efficient algorithm for nonlinear model predictive control, 2017.

[18] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[19] Nicholas D Matsakis and Felix S Klock II. The rust language. In *ACM SIGAda Ada Letters*, volume 34, pages 103–104. ACM, 2014.

[20] Dirk Thomas, William Woodall, and Esteve Fernandez. Next-generation ROS: Building on DDS. In *ROSCon Chicago 2014*, Mountain View, CA, sep 2014. Open Robotics.

[21] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 29(3):972–983, 2021.

A

Appendix 1

Leader and static obstacles

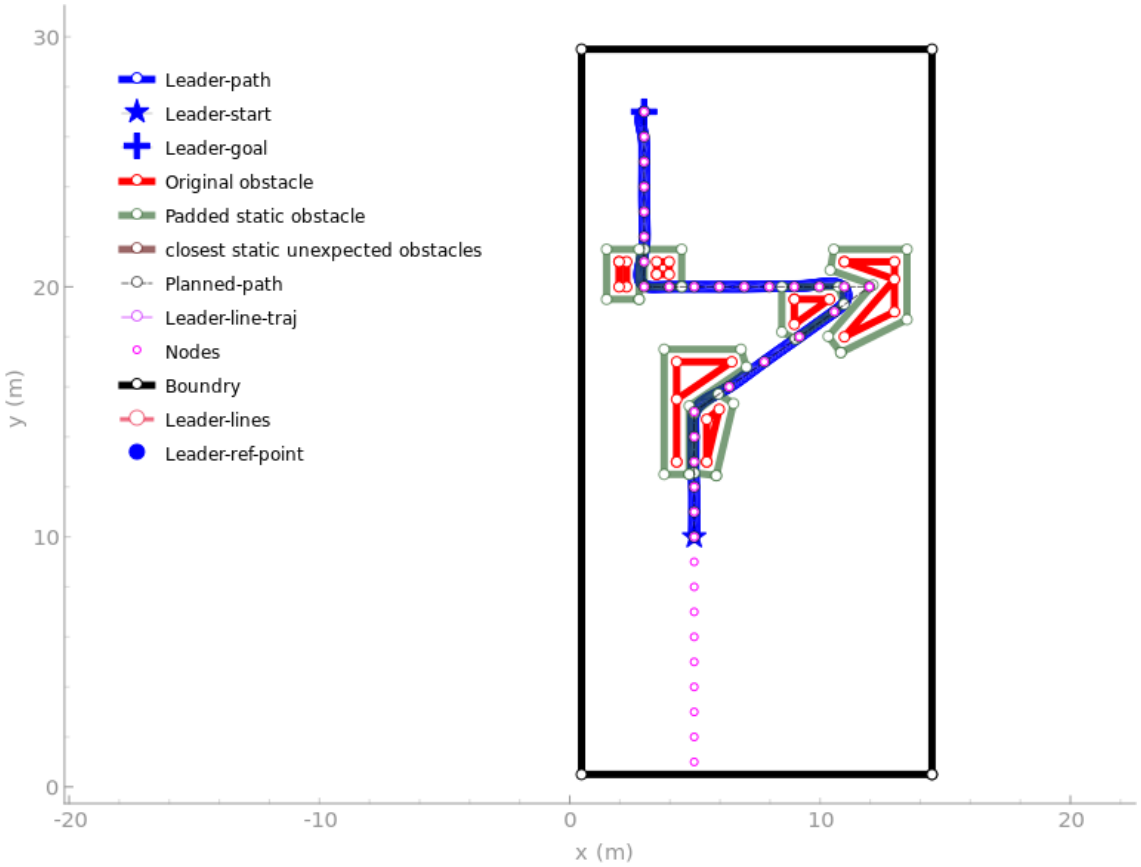


Figure A.1: Trajectory planning results with only leader and static obstacles.

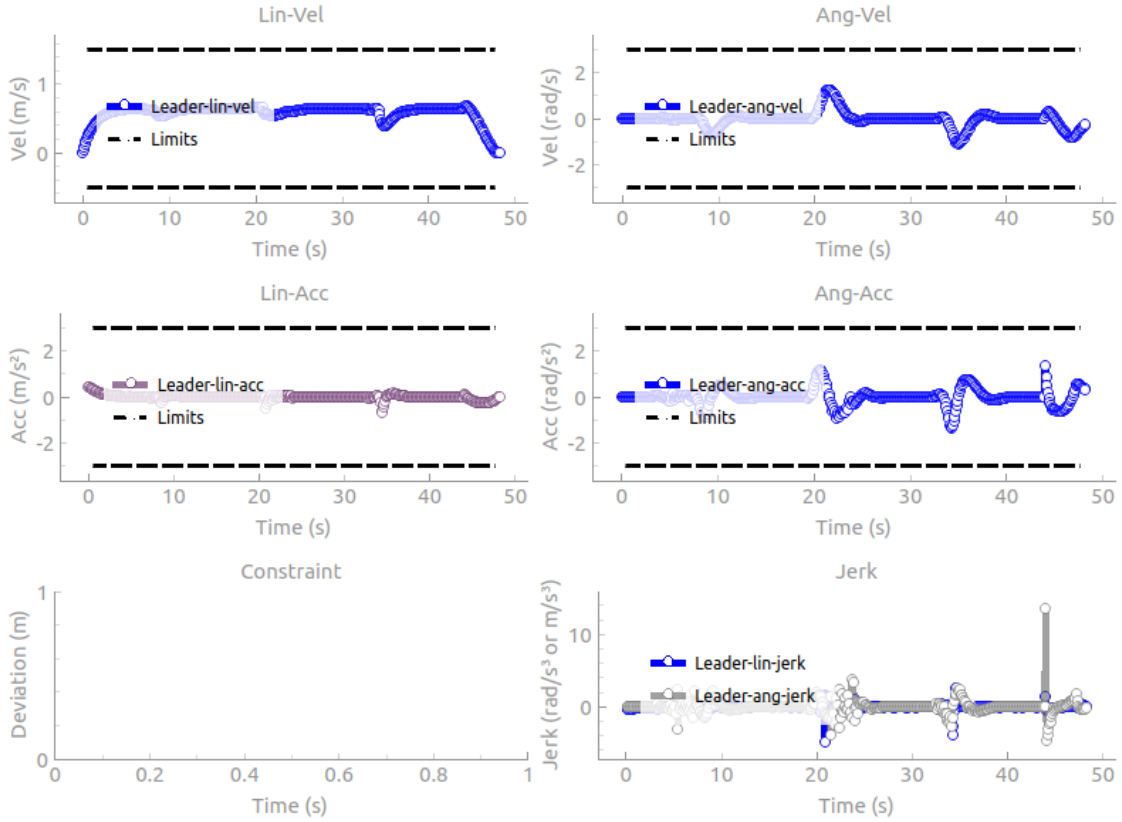


Figure A.2: Trajectory planning results with only leader and static obstacles.

Leader and static and unexpected obstacles

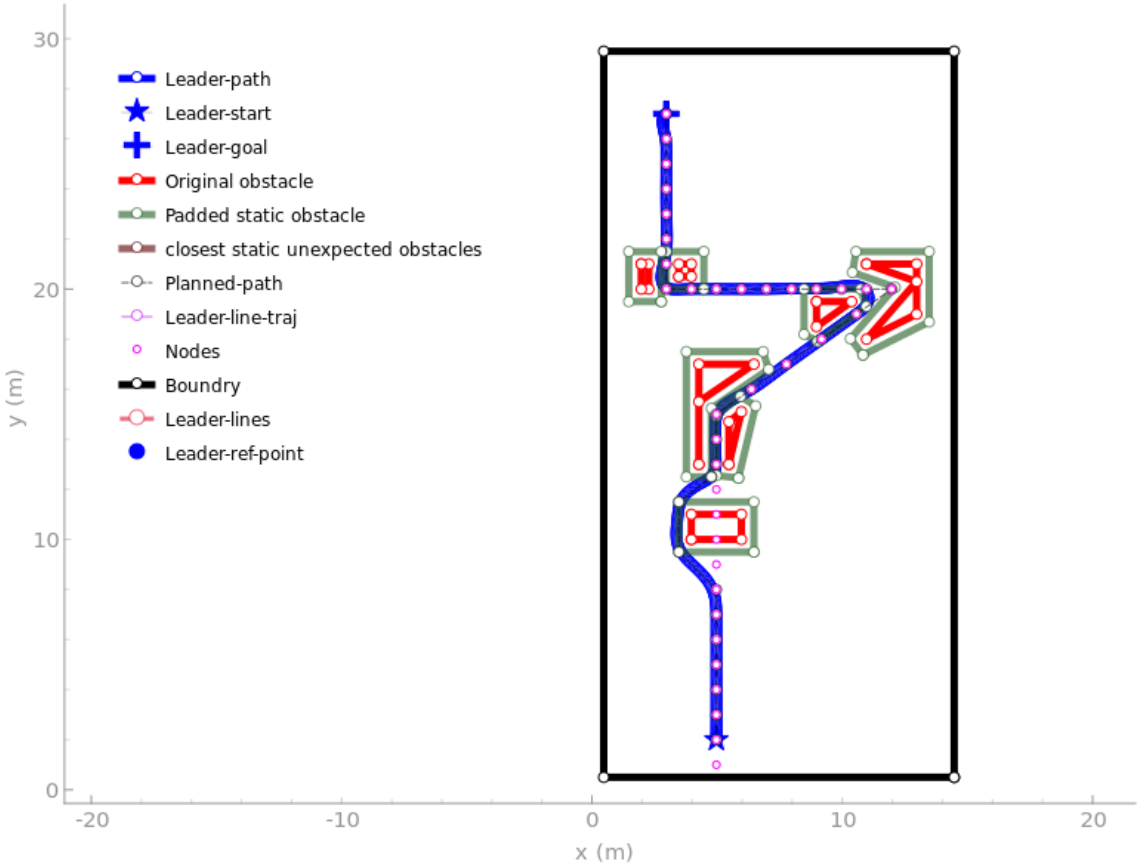


Figure A.3: Trajectory planning results with only leader and static obstacles but with one to avoid.

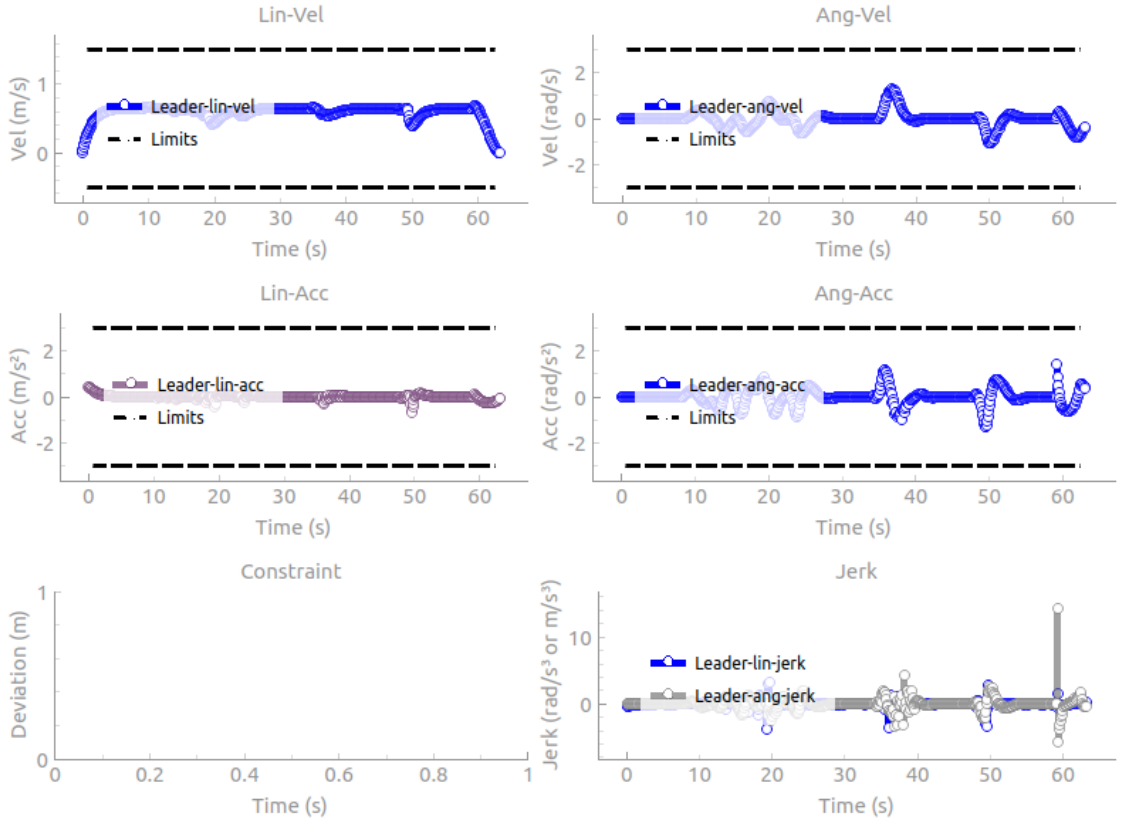


Figure A.4: Trajectory planning results with only leader and static obstacles but with one to avoid.

Leader and follower and static obstacles

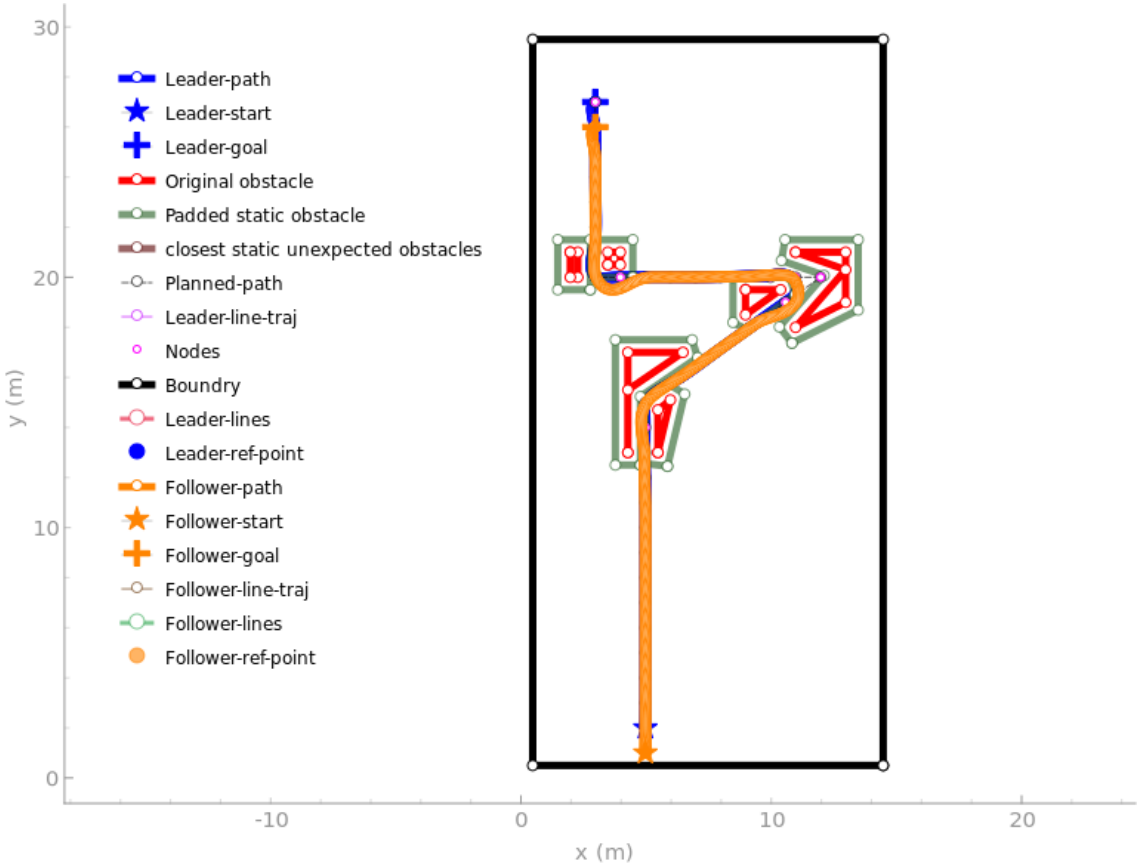


Figure A.5: Trajectory planning results with only leader and follower and static obstacles.

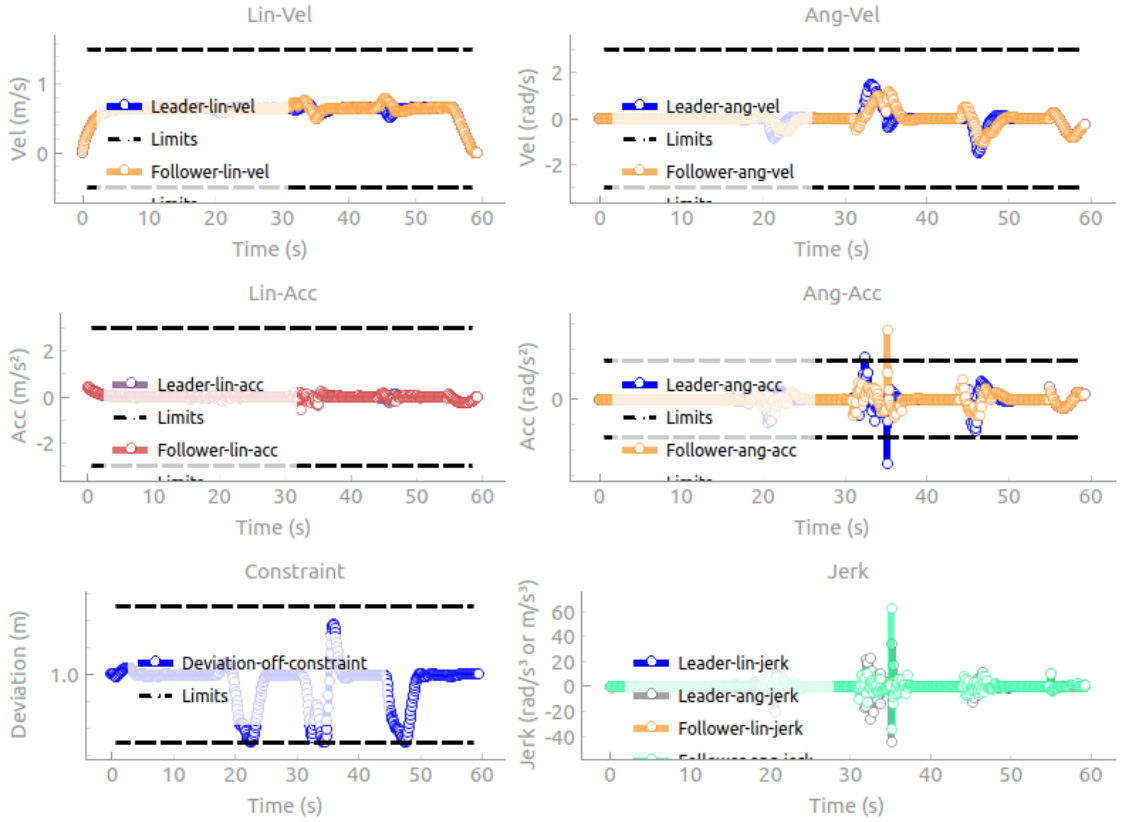


Figure A.6: Trajectory planning results with only leader and follower and static obstacles.

Leader and follower and static and unexpected obstacles

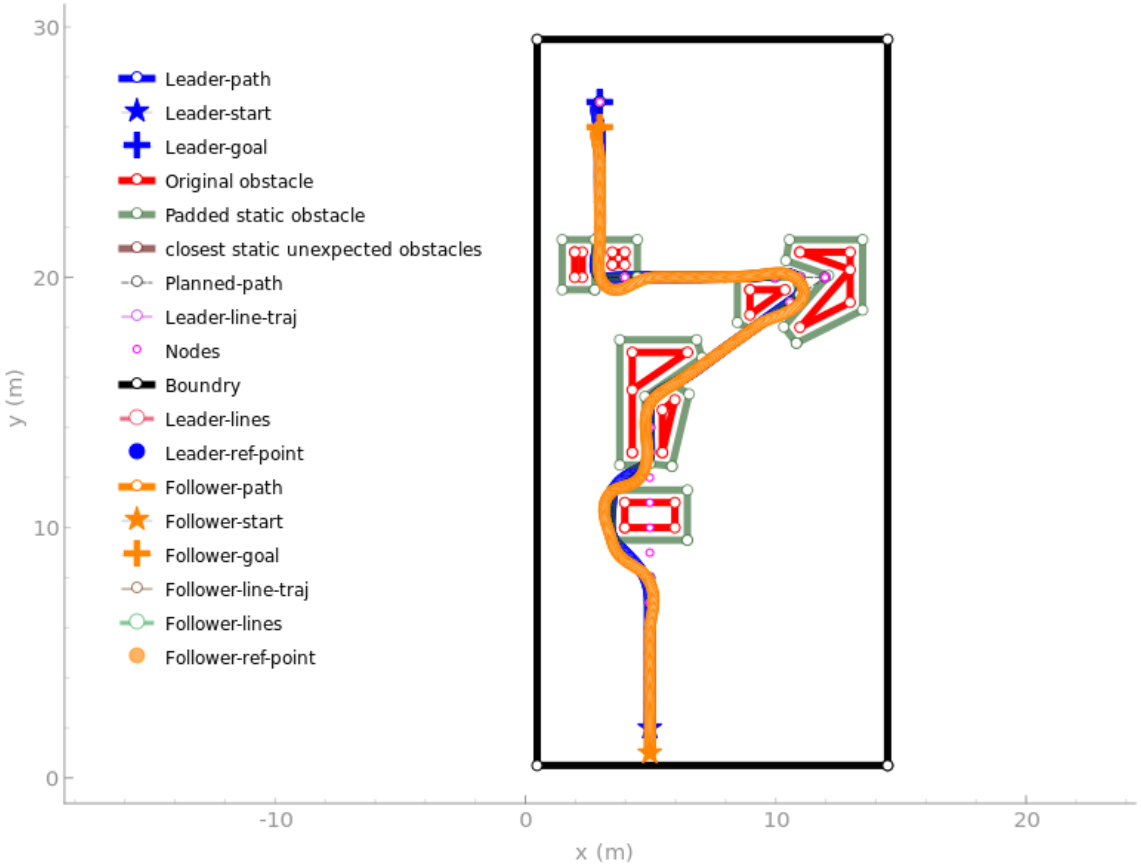


Figure A.7: Trajectory planning results with only leader and follower and static obstacles but with one to avoid.

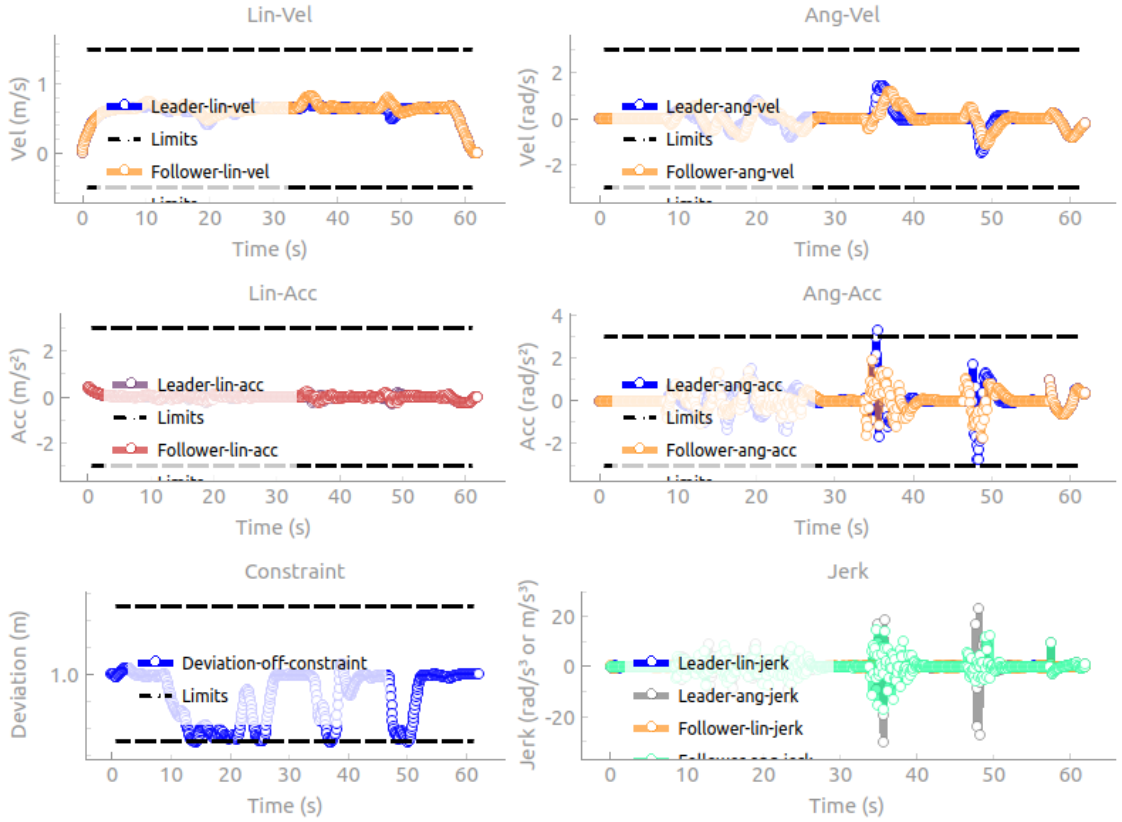


Figure A.8: Trajectory planning results with only leader and follower and static obstacles but with one to avoid.

constraint violation: -0.008102027487654873

Leader adjust

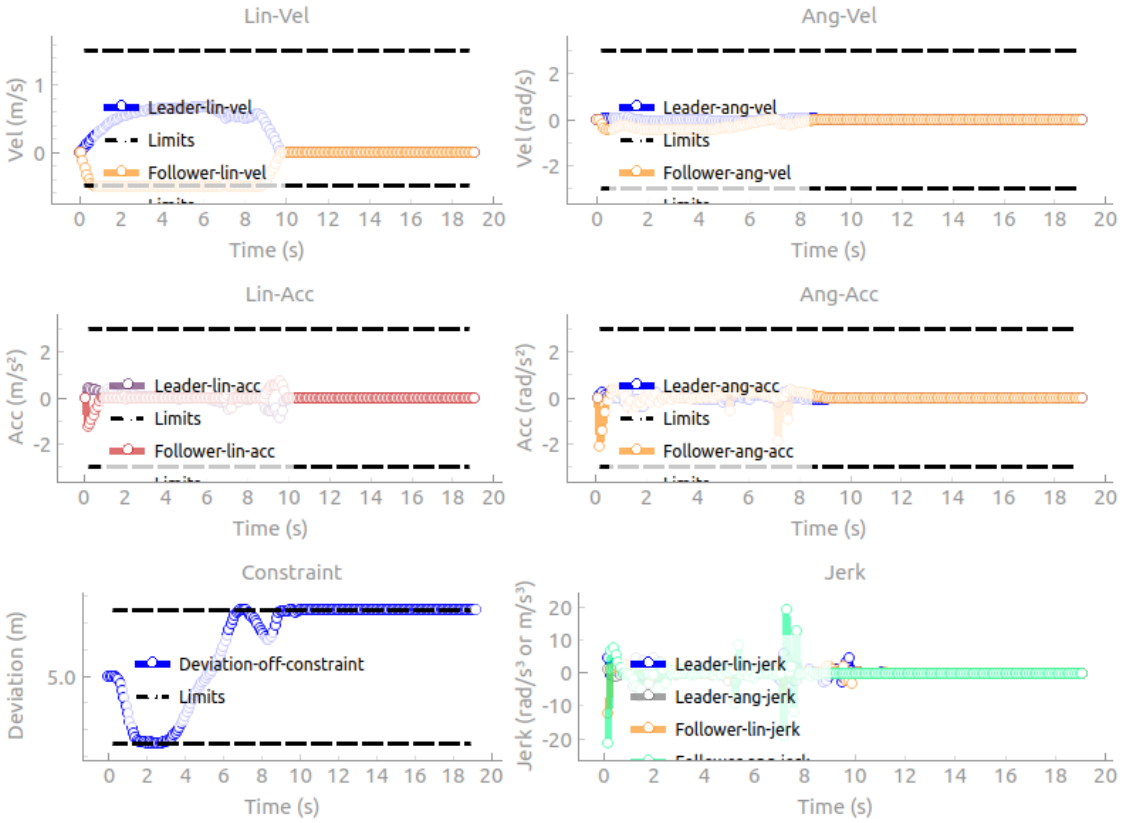


Figure A.9: Trajectory planning results where leader has to adjust to follower, data.

Dynamic obstacles straight ahead passive

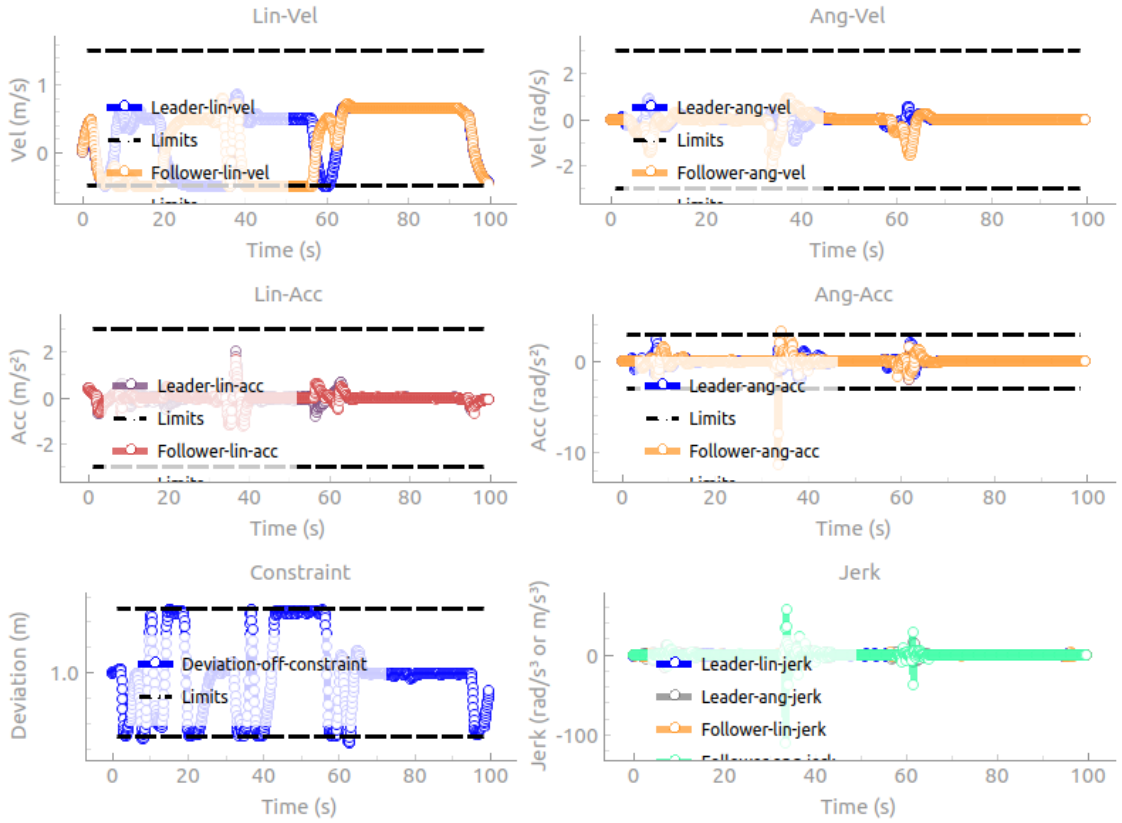


Figure A.10: Trajectory planning results where dynamic obstacles are coming straight ahead, data.

Dynamic obstacles straight ahead aggressive

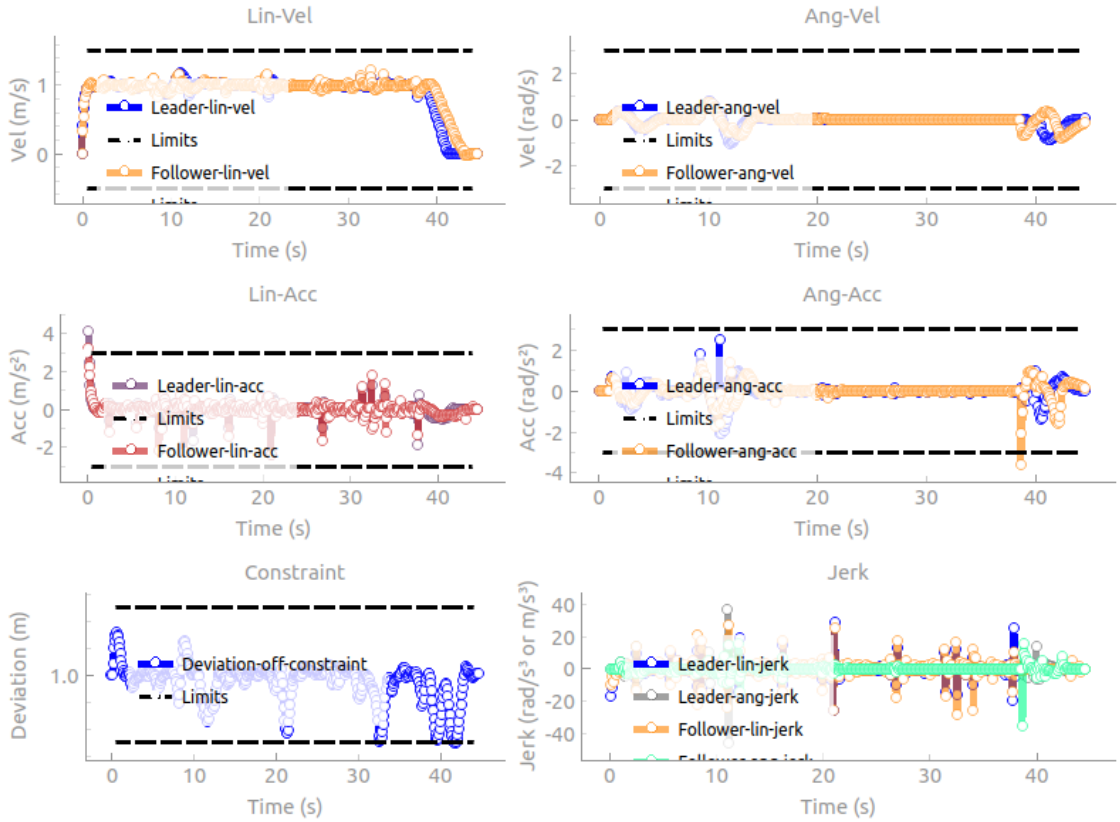


Figure A.11: Trajectory planning results with only leader and follower and static obstacles but with one to avoid and dynamic obstacles.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY