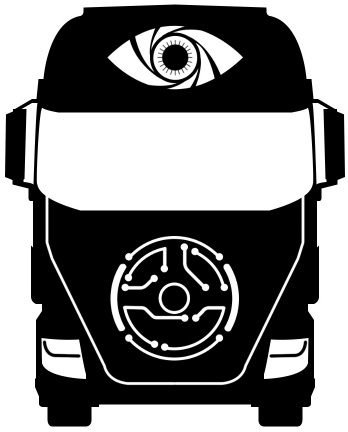
Design Specification

Autonomous Truck With a Trailer

November 9, 2020

Version 2.0



Status

Reviewed	Filip Jussila	2020-11-09
Approved	Carl Hynén	2020-11-10

Project Identity

Group E-mail:	vituv953@student.liu.se
Homepage:	https://tsrt10.gitlab-pages.liu.se/2020/rev_truck
Orderer:	Carl Hynén, Linköping University Phone: +46 73-562 34 34 E-mail: carl.hynen@liu.se
Supervisor:	Daniel Arnström, Linköping University Phone: +46 76-831 24 09 E-mail: daniel.arnstrom@liu.se
Customer / Course Responsible:	Daniel Axehill, Linköping University Phone: +46 13-28 40 42 E-mail: daniel.axehill@liu.se

Participants of the group

Name	**Responsibility**	**E-mail**
Per Antonsson		peran080@student.liu.se
Edvin Bourelius		edvbo949@student.liu.se
Gustav Erbing		guser450@student.liu.se
Johan Gustafsson		johgu927@student.liu.se
Anton Holgersson	Information Manager (IM)	antho223@student.liu.se
Ofa Ismail	Software Architect (SW)	abdis077@student.liu.se
Filip Jussila	Document Manager (DM)	filju425@student.liu.se
Per Liljeström	Test Leader (TL)	perli397@student.liu.se
Katherine Rajala		katra756@student.liu.se
David Salomonsson	Design Manager (DeM)	davsa358@student.liu.se
Viktor Uvesten	Project Leader (PL)	vituv953@student.liu.se

CONTENTS

DOCUMENT HISTORY

Version	Date	Changes made	Made by	Reviewer
0.1	2020-09-30	First draft	Project Group	Viktor Uvesten
0.2	2020-10-04	Fixed comments and added content	Project Group	Viktor Uvesten
0.3	2020-10-05	Clarified MPC structure with the help of supervisor	Project Group	Filip Jussila
1.0	2020-10-07	Improved models, figures, formalities and language	Project Group	Filip Jussila
2.0	2020-11-09	Final draft	Project Group	Filip Jussila

1 INTRODUCTION

This is the design specification for the project *Autonomous Truck With a Trailer* in the automatic control project course TSRT10 given at Linköping University in the fall of 2020.

The project has been running for several years and the main goal of the project this year is to improve the stability and robustness of the automatic control by implementing a Model Predictive Control (MPC) controller. A Raspberry Pi (RPi) will also be integrated to the already existing truck to replace the need of a laptop during run-time. The purpose of this document is to specify the design specifications for the final results of the project.

1.1 Background and purpose

Autonomy is a growing subject in our society and therefore are autonomous vehicles of great interest for the industry and thus, a relevant research area [1]. A difficult task that truckers face daily is to reverse their truck with a trailer. This tricky and arduous maneuver can be simplified through a driver assistance system such as Autonomous Path Planning & Parking Assistance System (APPAS). APPAS is not only used for assisting in reversing but also with general path planning, as the name implies.

1.2 Definition of terms

Terms and abbreviations used in this document are defined and explained below.

- **APPAS** – Autonomous Path Planning & Parking Assistance System. The name of the product developed during the project.
- **ROS** – Robotic Operative System. An open source software library, which simplifies writing modular code for robotic applications.
- **Visionen** – A research arena for robotics applications at Linköping University.
- **QualiSys** – Positioning system used in Visionen.
- **MPC** – Model Predictive Control, optimization based control method used to control a process while satisfying a set of constraints.
- **RPi** – Raspberry Pi. A small single-board computer.

2 SYSTEM OVERVIEW

The system consists of three main parts which are a positioning system called QualiSys, a computer that takes care of the projectors in Visionen and a LEGO truck equipped with an EV3 unit and an RPi.

2.1 Block diagram

An overview of the system and the internal communication can be seen in Figure 1. The subsystems inside the dashed red line is running on the RPi.

The LEGO truck consists of a tractor and a semitrailer where a dolly connects those two parts together. On the truck there is an EV3 unit, an RPi, servo motors and two angle sensors attached. The system colored red in Figure 1 handles the main computations, the ones colored grey are the subsystems hosted on the RPi and the blue ones are the external hardware.

There are two different computers in Visionen. One handles the QualiSys system and the other handles the projectors, here called visualization.

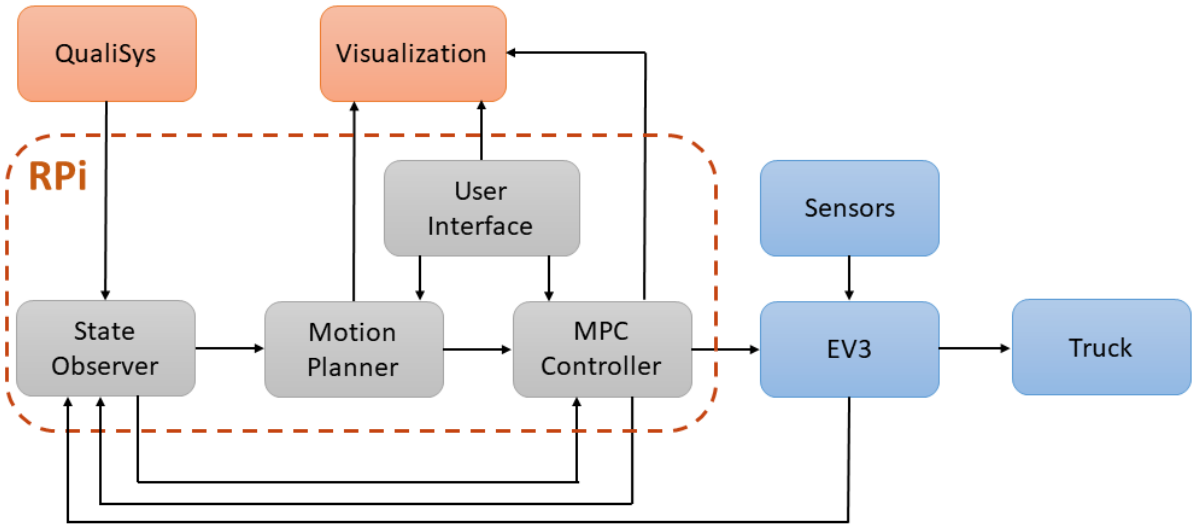


Figure 1: A block diagram of the system.

2.1.1 *EV3*

A LEGO EV3 unit is placed on the LEGO truck and is used to control the motors based on the communication with the RPi. The control signals are currents to control the two actuators, and these are sent from the RPi in real time through a USB cable. The task for the EV3 unit is to pass these signals to the actuators.

Furthermore, the EV3 system will receive sensor data from the two joint angle sensors. This data will then be sent to the state observer.

2.1.2 *Raspberry Pi*

An RPi 4 model B is going to be installed on the LEGO truck to host ROS. The purpose of using the RPi unit is to remove the need of an external computer performing the computations.

There are four different subsystems on the RPi, each handling major tasks. A state observer that takes sensor data and processes it before it is used. A motion planner that is planning a path that the truck should follow. An MPC controller that is responsible for controlling the truck so it can follow the path calculated by the motion planner. This is made in real time thanks to the RPi's calculation power. A steering angle and velocity are calculated by the MPC controller. However, these control signals must be converted into currents so that the actuators can be controlled. Hence, the calculated steering angle and velocity are sent to another controller (inside the MPC block) with the purpose of transforming these to currents that are used in the actuators. These currents are sent to the EV3 unit. Finally there is a user interface that enables the user to add obstacles to the world, set a goal point and change design parameters of the controller.

2.1.3 *Motors*

The truck has two servo motors, where one is in charge of steering while the other is in charge of propulsion. Both motors have an angle sensor for reference tracking.

2.1.4 *Motion sensors*

The truck has two HiTechnic NXT angle sensors equipped. The first sensor is placed between the truck and dolly while the other is placed between the dolly and trailer. The sensors can measure from 0° to -359°, with 1° accuracy. These angles will be used by the state observer.

2.1.5 *Visualization*

A visualization system will be used to display the path from the motion planner, the path from the MPC controller, where the truck has been and the obstacles of the world. The visualization system is hosted on a external computer in Visionen. This computer controls three different projectors.

2.1.6 *QualiSys*

QualiSys is a positioning system. It will track the truck's position and orientation by using 12 cameras that can detect reflective balls mounted on the truck. This information will be used by the state observer.

2.1.7 *Kinematic model*

The kinematic model in Figure 2 is given by system (1) which is described in detail in [2]. The model consists of three different parts: a front wheel steered tractor, a dolly connected to the tractor's hitching point and a semitrailer. The states x_3 and y_3 are the center position of the semitrailer's rear axle and θ_3 is the angle between its orientation and the x-axis. The joint angle β_3 is the angle between the semitrailer and the dolly and β_2 the joint angle between the dolly and the tractor. The length L_1 is the wheelbase of the tractor, L_2 is the distance between the tractor's hitch and the dolly's wheel axle and L_3 is the distance between the wheel axle of the dolly and the semitrailer. The length M_1 is the distance from the tractor's rear axle to the hitching point. The steering angle of the tractor is denoted with α and its curvature is $u = \frac{\tan\alpha}{L_1}$.

$$\dot{x}_3 = v_3 \cos\theta_3 \tag{1a}$$
$$\dot{y}_3 = v_3 \sin\theta_3 \tag{1b}$$
$$\dot{\theta}_3 = v_3 \frac{\tan\beta_3}{L_3} \tag{1c}$$
$$\dot{\beta}_3 = v_3 \left(\frac{\sin\beta_2 - M_1 \cos\beta_2 u}{L_2 C_1(\beta_2, \beta_3, u)} - \frac{\tan\beta_3}{L_3} \right) \tag{1d}$$
$$\dot{\beta}_2 = v_3 \left(\frac{u - \frac{\sin\beta_2}{L_2} + \frac{M_1}{L_2}\cos\beta_2 u}{C_1(\beta_2, \beta_3, u)} \right) \tag{1e}$$

Equation (1e) consists of $C_1(\beta_2, \beta_3, u)$ which is the relationship between the velocity of the tractor v and the semitrailer's velocity v_3 as shown in equation (2).

$$v_3 = vC_1(\beta_2, \beta_3, u) = v\cos\beta_3(\cos\beta_2 + M_1 \sin\beta_2 u) \tag{2}$$

The state model (1) can be presented in a more compact form as $\dot{x} = v_3 f(x, u)$.

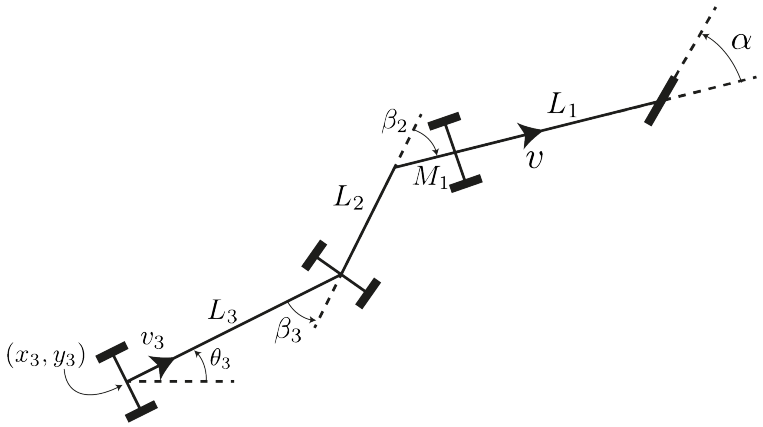


Figure 2: A kinematic model of the truck

2.2 ROS architecture overview

The ROS architecture consists of eight nodes, hosted by three different computing devices, the QualiSys computer and projector computer located in Visionen and the RPi that is attached on the LEGO truck. The communication flow and the ROS architecture are illustrated in Figure 3. The system can be controlled by any computer with an SSH connection to the RPi.

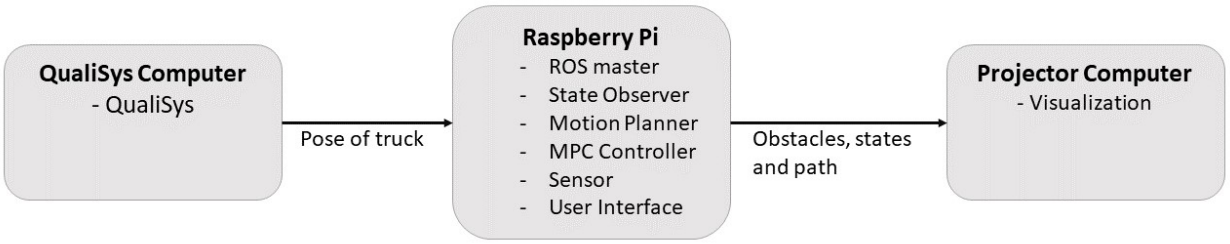


Figure 3: Overview of the ROS architecture.

2.2.1 *Nodes and topics*

The communication in Figure 3 is done by publishing and subscribing on different ROS topics. The name of the topic, an overview of its content and which nodes that publishes and subscribes to it can be seen in Table 1. This table is an example of topics in the system.

The nodes in the system are the State observer node (SO), Motion planning node (MP), MPC controller node (MPC), QualiSys node (QSys), Visualization node (Viz), Sensor node (Sens) and User interface node (UI). The ROS master will also serve as a node.

Table 1: A description of key ROS topics.

Name	Description	Publish	Subscribe
\est_state	Estimated states.	SO	MP, MPC
\path	Reference path.	MP	MPC, Viz
\pred_state	A vector X containing predicted states that the MPC controller want to steer the truck to.	MPC	Viz
\control_signals	Steering angle and velocity.	MPC	SO, Sens
\truck_pose	Pose of the truck from QualiSys.	QSys	SO
\sensor_angles	Angles from the sensors on the truck.	Sens	SO, MPC
\obstacles	Obstacle locations on the map.	UI	MP, MPC, Viz
\goal_pos	Goal position.	UI	MPC, MP
\q1q2	Weight matrices.	UI	MPC

3 MPC CONTROLLER

The MPC controller is responsible for the truck with its trailer being able to follow a given path. The idea with MPC is that the control problem is formulated as an optimization problem. To introduce feedback, the optimization problem is solved every time new measurements are given. This control strategy, unlike a PID controller, requires a model of the system that is being controlled. A model of a truck with a trailer, similar to the one used in this project, is described in Section 3.2. A drawback with an MPC controller is that it requires a lot of computational power, since it runs an optimizer in real time. The benefit with an MPC controller is that it can take limitations and constraints into account. It can also handle non-linear systems [3].

3.1 Problem description

The truck is currently controlled by a linear quadratic (LQ) controller. This controller had some issues last year with lateral and heading error, which could possibly be improved by an MPC controller with constraints [4]. Hence this year the main focus will be to implement the MPC controller. This controller shall make the truck follow a reference path and minimize the path error the truck has in correlation to the path.

3.2 Model description

Using and continuing from the previous year's path following error model [4, 5], the MPC problem should minimize the path following error state $\tilde{x} = [\tilde{z}_3, \tilde{\theta}_3, \tilde{\beta}_3, \tilde{\beta}_2]^T$ which are the errors of the truck with respect to the reference calculated by the motion planner. The signed lateral distance of the semitrailer's axle to the path is $\tilde{z}_3 = z_3 - z_{3r}$. $\tilde{\theta}_3 = \theta_3(t) - \theta_{3r}(s(t))$ is the orientation error, $\tilde{\beta}_2 = \beta_2(t) - \beta_{2r}(s(t))$ and $\tilde{\beta}_3 = \beta_3(t) - \beta_{3r}(s(t))$ are the joint angle errors, see Figure 4. The controlled vehicle's progress projected along the nominal path is $s(t)$. The MPC problem should also minimize the curvature deviation $\tilde{u} = u(t) - u_r(s(t))$, where $u(t) = \frac{tan(\alpha(t))}{L_1}$ and $u_r(s(t)) = \frac{tan(\alpha_r(s(t)))}{L_1}$.

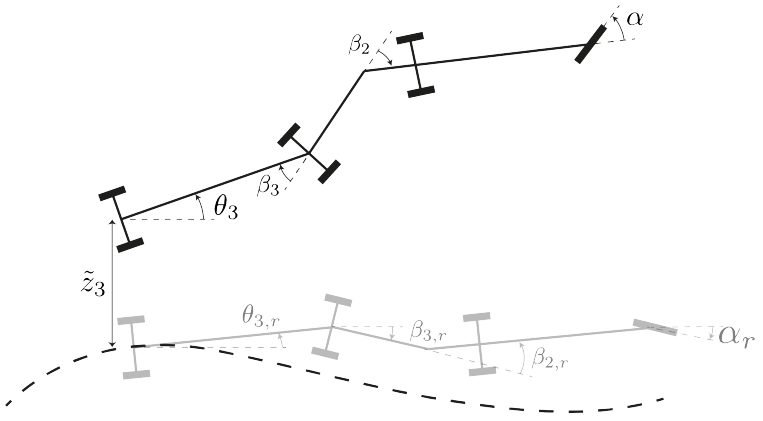


Figure 4: The semitrailer deviation from its path.

By linearizing the path following error model around the origin $(\tilde{x}, \tilde{u}) = (0, 0)$, we obtain:

$$\frac{d\tilde{x}}{ds} = A(s)\tilde{x} + B(s)\tilde{u}, \tag{3}$$

where $\frac{d\tilde{x}}{ds} = \frac{d\tilde{x}}{dt}\frac{1}{\dot{s}}$. By discretizing with Euler-forward, we obtain:

$$\tilde{x}_{k+1} = F_k\tilde{x}_k + G_k\tilde{u}_k. \tag{4}$$

In (4), $F_k = I + \Delta_s A_k(s)$ and $G_k = \Delta_s B_k(s)$. $A(s)$ and $B(s)$ are taken from [2]. Δ_s is the sampling distance.

3.3 MPC problem description

For a path following controller the MPC problem can then, according to [2], be formulated as below:

$$\underset{\tilde{X},\tilde{U}}{\text{minimize}}\ V_N(\tilde{X},\tilde{U}) = V_f(\tilde{x}_N) + \sum_{k=0}^{N-1} l(\tilde{x}_k, \tilde{u}_k) \tag{5a}$$

$$\text{subject to } \tilde{x}_{k+1} = F_k\tilde{x}_k + G_k\tilde{u}_k, \qquad k = 0, 1, ..., N-1 \tag{5b}$$

$$(\tilde{\beta}_{3,k}, \tilde{\beta}_{2,k}) \in \tilde{\mathbb{P}}_k, \qquad k = 0, 1, ..., N-1 \tag{5c}$$

$$\tilde{u}_k \in \tilde{\mathbb{U}}_k, \qquad k = 0, 1, ..., N-1 \tag{5d}$$

$$|\tilde{z}_{3,k}| \leq \tilde{z}_3^{\max},\ |\tilde{\theta}_{3,k}| \leq \tilde{\theta}_3^{\max} \qquad k = 0, 1, ..., N-1 \tag{5e}$$

$$\tilde{x}_0 = \tilde{x}(s(t)) \text{ given}, \tag{5f}$$

where the terminal cost is given by $V_f(\tilde{x}_N) = \tilde{x}_N^T \bar{P}_N \tilde{x}_N$, the running cost is given by $l(\tilde{x}_k, \tilde{u}_k) = \tilde{x}_k^T \bar{Q}_1 \tilde{x}_k + \tilde{u}_k^T Q_2 \tilde{u}_k$ and N is the prediction horizon. Note that the constraints are different dependent on if the truck is driving forward or backwards. $\tilde{X}$ and $\tilde{U}$ are the predicted path following error state vector sequence and the curvature deviation sequence, respectively. These are defined as:

$$\tilde{X} = \begin{bmatrix} \tilde{x}_0 \\ \tilde{x}_1 \\ \vdots \\ \tilde{x}_{N-1} \end{bmatrix}, \qquad \tilde{U} = \begin{bmatrix} \tilde{u}_0 \\ \tilde{u}_1 \\ \vdots \\ \tilde{u}_{N-1} \end{bmatrix}. \tag{6}$$

The concept of MPC control is that the optimization formulation in (5) is solved in each time instance and only the first control deviation from the optimal control set $\tilde{U}^*$ is used. The control signal for that time instance is then calculated by equation (7).

$$u(t) = u_r(s(t)) + \tilde{u}_0^*, \tag{7}$$

Where $u_r(s(t))$ is the reference curvature calculated by the motion planner. The constraints (5c) - (5e) will be discussed further in Section (3.5).

Initially will CasADi [6] be used to solve the presented optimization problem, after which a Quadratic programming (QP) solver is planned to be used. To convert the presented notation to a fitting convention for QP, the following matrices are constructed:

$$\mathcal{F} = \begin{bmatrix} I \\ F \\ F^2 \\ \vdots \\ F^{N-1} \end{bmatrix}, \qquad \mathcal{G} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ G & 0 & 0 & \dots & 0 \\ FG & G & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ F^{N-2}G & \dots & FG & G & 0 \end{bmatrix}, \tag{8}$$

$$\mathcal{F}_N = F^N, \qquad \mathcal{G}_N = \begin{bmatrix} F^{N-1}G & \dots & FG & G & 0 \end{bmatrix}. \tag{9}$$

Through the definitions in equation (6) and (8), the recursive format given in equation (4) can be used to express the system states $\tilde{x}$ in terms of the initial state x_0 and the control sequence $\tilde{U}$ as:

$$\tilde{X} = \mathcal{F}\tilde{x}_0 + \mathcal{G}\tilde{U}. \tag{10}$$

Likewise, the final state $\tilde{x}_N$ can be expressed as:

$$\tilde{X}_N = \mathcal{F}_N\tilde{x}_0 + \mathcal{G}_N\tilde{U}. \tag{11}$$

Similarly to last year [4], the error state will be extended to include the signed lateral distance errors of the dolly and the tractor, $\tilde{z}_2$ and $\tilde{z}_1$ respectively, which expands the state vector to:

$$h(\tilde{x}) := z = [\tilde{z}_1\ \tilde{\theta}_1\ \tilde{z}_2\ \tilde{\theta}_2\ \tilde{\beta}_2\ \tilde{z}_3\ \tilde{\theta}_3\ \tilde{\beta}_3]^T, \tag{12}$$

where for a straight line the extended states are defined as:

$$\tilde{z}_2 = \tilde{z}_3 + L_3 \sin\tilde{\theta}_3, \tag{13a}$$
$$\tilde{z}_1 = \tilde{z}_2 + L_2 \sin\tilde{\theta}_2 + M_1 \sin\tilde{\theta}_1, \tag{13b}$$

and

$$\tilde{\theta}_2 = \tilde{\theta}_3 + \tilde{\beta}_3, \tag{14a}$$
$$\tilde{\theta}_1 = \tilde{\theta}_2 + \tilde{\beta}_2, \tag{14b}$$

where $h(\tilde{x})$ in equation (12) is nonlinear and needs to be linearized. This is done by Jacobian linearization around the origin as:

$$z \approx \frac{\partial h(0)}{\partial \tilde{x}}\tilde{x} := M\tilde{x}. \tag{15}$$

The result of equation (15) can be used to construct the quadratic weight matrix Q that penalizes all the extended states from (12). This matrix is expressed as:

$$\bar{Q}_1 = M^T Q_1 M, \tag{16}$$

$$\bar{P}_N = M^T P_N M, \tag{17}$$

where $\bar{Q}_1$ is a diagonal matrix which penalizes each state in $\tilde{z}$ and $\bar{P}_N$ is a diagonal matrix which penalizes the final state in $\tilde{z}$.
Once again is a conversion to a more fitting convention for QP done for previous definitions – equation (15) and (17) – which is presented in matrices (18) below:

$$\mathcal{Q}_i = \begin{bmatrix} Q_i & & & \\ & Q_i & & \\ & & \ddots & \\ & & & Q_i \end{bmatrix} \quad i = 1,2 \qquad \mathcal{M} = \begin{bmatrix} M & & & \\ & M & & \\ & & \ddots & \\ & & & M \end{bmatrix} \tag{18}$$

where all three matrices – $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{M}$ – are of dimension $N \times N$.

3.4 Quadratic programming description

In QP, there is a cost function that is quadratic and constraints that are linear. The cost function can be described as:

$$\min_w \frac{1}{2} w^T H w + f^T w \tag{19a}$$

$$Aw \leq b. \tag{19b}$$

If the MPC problem is formulated with the matrices in Section 3.3, we obtain:

$$\min_U \frac{1}{2} \tilde{U}^T (\mathcal{G}_N^T \mathcal{M}^T \mathcal{P}_N \mathcal{M} \mathcal{G}_N + \mathcal{G}^T \mathcal{M}^T \mathcal{Q}_1 \mathcal{M} \mathcal{G} + \mathcal{Q}_2) \tilde{U} + (\mathcal{G}_N^T \mathcal{M}^T \mathcal{P}_N \mathcal{M} \mathcal{F}_N \tilde{x}_0 + \mathcal{G}^T \mathcal{M}^T \mathcal{Q}_2 \mathcal{M} \mathcal{F} \tilde{x}_0)^T \tilde{U} \tag{20}$$
$$A_u U \leq b_u.$$

3.5 Constraints

There are several constraints that the MPC controller will have to take into account. These are brought up in the subsections below.

3.5.1 *Obstacle avoidance*

Figure 5 illustrates obstacle avoidance handled by the truck. Since it is a possibility that the truck and trailer will deviate from the path planned by the motion planner (green line), the MPC controller must be able to avoid obstacles. This is achieved by putting constraints on the truck's deviation from the path.

$\tilde{z}_1$, $\tilde{z}_2$ and $\tilde{z}_3$ is defined as the lateral deviations of the tractor, dolly and trailer position from the path, respectively. These distances corresponds to the red lines in Figure 5. The distances between the points on the path where the red lines crosses it to the nearest obstacle, are called $d_{j,i}$. These distances are represented by the blue lines in Figure 5 and are parallel to the red lines. The index j states the left or right side of the truck and the index i states which part of the truck that the blue line will cross (tractor, dolly or trailer). The lines $d_{j,i}$ can be calculated by ray tracing. All mentioned distances are calculated at each future state $k = (1, \ldots, N-1)$. The constraint can be written as:

$$d_{1,1} \leq \tilde{z}_1 \leq d_{2,1} \tag{21a}$$

$$d_{1,2} \leq \tilde{z}_2 \leq d_{2,2} \tag{21b}$$

$$d_{1,3} \leq \tilde{z}_3 \leq d_{2,3}. \tag{21c}$$

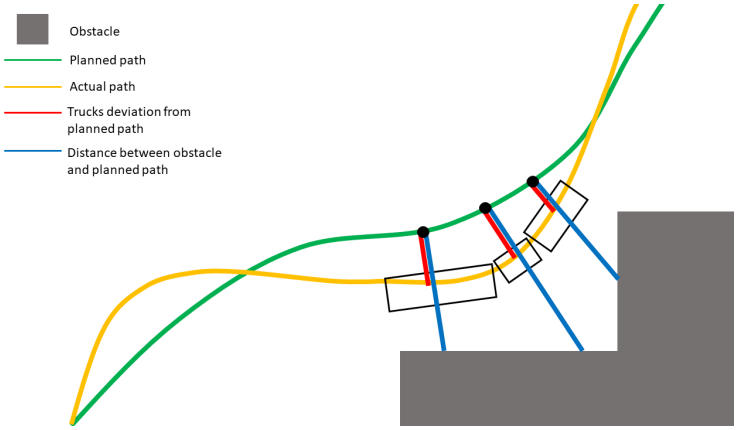


Figure 5: How obstacle avoidance should be implemented.

Note that the points on the three parts of the vehicle, where the red lines starts, can be changed to arbitrary positions for better obstacle avoidance. It might for example be better to use points on the trailer's outer end instead of the center of the trailer to be sure that the vehicle will not hit an obstacle.

To use this in QP the constraints in (21) need to be expressed in the following form: $A_d U \leq b_d$. This is obtained as follows:

$$\bar{d}_1 \leq C_d z \leq \bar{d}_2 \tag{22}$$

where C_d is defined as:

$$C_d z = \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \end{bmatrix}. \tag{23}$$

Constraint (22) can be divided into:

$$C_d z \leq -\bar{d}_1 \tag{24a}$$
$$C_d z \leq \bar{d}_2. \tag{24b}$$

By only expanding (24a), for the whole prediction horizon the constraint can be written as:

$$\begin{bmatrix} C_d z_0 \\ \vdots \\ C_d z_{N-1} \end{bmatrix} \leq \begin{bmatrix} -\bar{d}_{1,1} \\ \vdots \\ -\bar{d}_{1,N-1} \end{bmatrix} \tag{25}$$

$$\mathcal{C}_d \mathcal{M}(\mathcal{F}\tilde{x}_0 + \mathcal{G}\tilde{U}) \leq \begin{bmatrix} -\bar{d}_{1,1} \\ \vdots \\ -\bar{d}_{1,N-1} \end{bmatrix} \tag{26}$$

$$\mathcal{C}_d\mathcal{M}\mathcal{G}\tilde{U} \leq \begin{bmatrix} -\bar{d}_{1,1} \\ \vdots \\ -\bar{d}_{1,N-1} \end{bmatrix} - \mathcal{C}_d\mathcal{M}\mathcal{F}\tilde{x}_0 \tag{27}$$

where $A_d = \mathcal{C}_d\mathcal{M}\mathcal{G}$ and $b_d = \begin{bmatrix} \bar{d}_{1,1}^T & \dots & \bar{d}_{1,N-1}^T \end{bmatrix}^T - \mathcal{C}_d\mathcal{M}\mathcal{F}\tilde{x}_0$.

3.5.2 *Jack knifing*

There are several constraints to take into account when formulating the MPC problem. The first two constraints are regarding jack knifing. To avoid this, the angle between the truck and dolly (β_2) as well as the angle between the dolly and the semitrailer (β_3) can not be too large. Furthermore, there are limitations in the relation between these two angles to avoid jack knifing. Hence, the following constraints are obtained:

$$|\beta_2| \leq \beta_{2,\max} \tag{28a}$$
$$|\beta_3| \leq \beta_{3,\max} \tag{28b}$$
$$\beta_2 \leq C\beta_3. \tag{28c}$$

Since $\tilde{\beta}_2 = \beta_2(t) - \beta_{2r}(s(t))$ and $\tilde{\beta}_3 = \beta_3(t) - \beta_{3r}(s(t))$ we obtain:

$$-\beta_{2,\max} - \beta_{2,r} \leq \tilde{\beta}_2 \leq \beta_{2,\max} - \beta_{2,r} \tag{29a}$$
$$-\beta_{3,\max} - \beta_{3,r} \leq \tilde{\beta}_3 \leq \beta_{3,\max} - \beta_{3,r} \tag{29b}$$
$$\tilde{\beta}_2 - C\tilde{\beta}_3 \leq -\beta_{2,r} + C\beta_{3,r}. \tag{29c}$$

Therefore $\tilde{\mathbb{P}} = \{\text{all } (\tilde{\beta}_3, \tilde{\beta}_2) \text{ s.t. (29a), (29b), (29c) are fullfilled}\}$.

Formulating the constraints (29a) and (29b) in QP form ($A_\beta U \leq b_\beta$) we obtain:

$$A_\beta = \begin{bmatrix} A_{1,\beta} \\ A_{2,\beta} \end{bmatrix}, \quad A_{1,\beta} = \mathcal{C}_\beta\mathcal{M}\mathcal{G}, \quad A_{2,\beta} = -\mathcal{C}_\beta\mathcal{M}\mathcal{G}, \quad b_\beta = \begin{bmatrix} b_{1,\beta} \\ b_{2,\beta} \end{bmatrix}$$

where:

$$b_{i,\beta} = \begin{bmatrix} \bar{\beta}_i \\ \vdots \\ \bar{\beta}_i \end{bmatrix} - \mathcal{C}_\beta\mathcal{M}\mathcal{F}\tilde{x}_0, \; i = 1,2, \; \bar{\beta}_1 = \begin{bmatrix} \beta_{2,\max} - \beta_{2,r} \\ \beta_{3,\max} - \beta_{3,r} \end{bmatrix}, \bar{\beta}_2 = \begin{bmatrix} \beta_{2,\max} + \beta_{2,r} \\ \beta_{3,\max} + \beta_{3,r} \end{bmatrix}$$

and:

$$\mathcal{C}_\beta z = \begin{bmatrix} \tilde{\beta}_2 \\ \tilde{\beta}_3 \end{bmatrix} \tag{30}$$

$$\mathcal{C}_\beta = \begin{bmatrix} C_\beta & & & \\ & C_\beta & & \\ & & \ddots & \\ & & & C_\beta \end{bmatrix}. \tag{31}$$

When formulating the constraint (29c) in QP form, we obtain:

$$A_\beta = \begin{bmatrix} 1 & -C \end{bmatrix} C_\beta \mathcal{I}\mathcal{M}\mathcal{G}, \quad b_\beta = \begin{bmatrix} -\beta_{2,r} + C\beta_{3,r} \\ \vdots \\ -\beta_{2,r} + C\beta_{3,r} \end{bmatrix}.$$

.

3.5.3 *Steering*

Due to limitations in the the steering, the steering angle α can not be too large. Hence, the following constraint must be taken into account:

$$-\alpha_{\max} \leq \alpha \leq \alpha_{\max}. \tag{32}$$

Therefore, since the curvature $u = \frac{\tan(\alpha)}{L_1}$ and $u_{\max} = \frac{\tan(\alpha_{\max})}{L_1}$,

$$-u_{max} \leq u \leq u_{max} \iff -u_{\max} - u_r \leq \tilde{u} \leq u_{\max} - u_r. \tag{33}$$

The tractor curvature rate also has to be limited. Hence, we obtain:

$$-\dot{u}_{max} \leq \dot{u} \leq \dot{u}_{max}. \tag{34}$$

Using Euler forward discretization, this is done by constraining the slew rate:

$$-c_{\max,k}\Delta_s \leq (\tilde{u}_k - \tilde{u}_{k-1}) - (u_{r,k} - u_{r,k-1}) \leq c_{\max,k}\Delta_s \tag{35}$$

where $c_{\max} = \frac{\dot{u}_{\max}}{|v|C_1(\beta_2,\beta_3,u)\cos\tilde{\theta}_3}$. Therefore $\tilde{\mathbb{U}} = \{\text{all } \tilde{u} \text{ such that (33) and (35) are fulfilled}\}$.

Expressing (33) and (35) in QP form we obtain:

$$A_u = \begin{bmatrix} A_{1,u} \\ A_{2,u} \end{bmatrix}, \quad A_{1,u} = \begin{bmatrix} I & \dots & 0 \\ \vdots & \ddots & \\ 0 & & I \end{bmatrix}, \quad A_{2,u} = -A_{1,u}, \quad b_u = \begin{bmatrix} b_{1,u} \\ b_{2,u} \end{bmatrix},$$

$$\text{where } b_{1,u} = \begin{bmatrix} u_{\max} - u_r \\ \vdots \\ u_{\max} - u_r \end{bmatrix}, \; b_{2,u} = \begin{bmatrix} u_{\max} + u_r \\ \vdots \\ u_{\max} + u_r \end{bmatrix}.$$

Expressing (35) in QP-form we obtain:

$$A_u = \begin{bmatrix} A_{1,u} \\ A_{2,u} \end{bmatrix}, \quad A_{1,u} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ -I & I & & \\ \vdots & & \ddots & \\ 0 & & -I & I \end{bmatrix}, \quad A_{2,u} = -A_{1,u}, \quad b_u = \begin{bmatrix} b_{1,u} \\ b_{2,u} \end{bmatrix},$$

$$\text{where } b_{1,u} = \begin{bmatrix} 0 \\ c_{\max,k}\Delta_s + (u_{r,k} - u_{r,k-1}) \\ \vdots \\ c_{\max,k}\Delta_s + (u_{r,k} - u_{r,k-1}) \end{bmatrix}, \; b_{2,u} = \begin{bmatrix} 0 \\ c_{\max,k}\Delta_s - (u_{r,k} - u_{r,k-1}) \\ \vdots \\ c_{\max,k}\Delta_s - (u_{r,k} - u_{r,k-1}) \end{bmatrix}.$$

3.6 Input and output structure

To control the truck and trailer, the MPC system will need relevant states and a reference path as inputs. The outputs of the MPC system will be the control signals and predicted states, which are calculated based on the inputs.

The MPC node will subscribe to several topics, presented in Table 1, to get the inputs. This data will be a state vector for the truck including position, orientation and joint angles. A reference path including multiple future states together with reference curvatures u_r. Furthermore the MPC will get information of virtual obstacles in the world together with a goal position. Finally the MPC will get the weight matrices to enable the operator to easily weigh the controlled signals.

The control signals sent to the EV3 unit are the steering angle of the truck's front wheels and the velocity orientation of the truck. This orientation will either be 1, -1 or 0 depending on if the truck should move forwards, backwards or not move. Another option is to use a controller that controls the velocity in the interval $[-1, 1]$ instead. The EV3 unit will then convert the steering angle and velocity orientation to appropriate currents. These are sent to the two actuators on the truck that control the steering angle and velocity. The control signals are sent via an USB cable, connected between the RPi and the EV3 unit. Signals will be sent when the MPC system have calculated an optimal control signal.

The visualization system is supposed to display the planned path from the MPC controller. To enable this, the MPC node will publish a state vector X, containing all positions needed to display the path, on a topic. The vector will be published when the MPC system has calculated an optimal control signal.

The state observer has to use the control signals from the MPC controller to estimate states. Hence, the MPC node will not only send them through USB to the EV3 unit but also publish them on a topic.

3.7 Calibration of steering angle

To eliminate the possible bias of the steering angle of the tractor's front wheels a calibrations phase will be implemented at the start of deployment. This is done to capture potential behaviors, e.g. wheel slip, that bring about the angular bias. This calibration phase will be executed as follows:

1. The truck is placed in an arbitrary position and orientation.
2. A control signal will be sent to the truck with the command to drive straight forward.
3. After an appropriate time, a control signal will be sent to the truck with the command to stop: an appropriate deceleration until $v = 0$ is achieved.
4. As the longitudinal velocity v is constant and known and the truck's pose at the start point and end point are also known, the bias of the steering angle can be calculated by taking the angular difference φ between the tractor's orientation and the planned path α_r using the definitions in Figure 6.

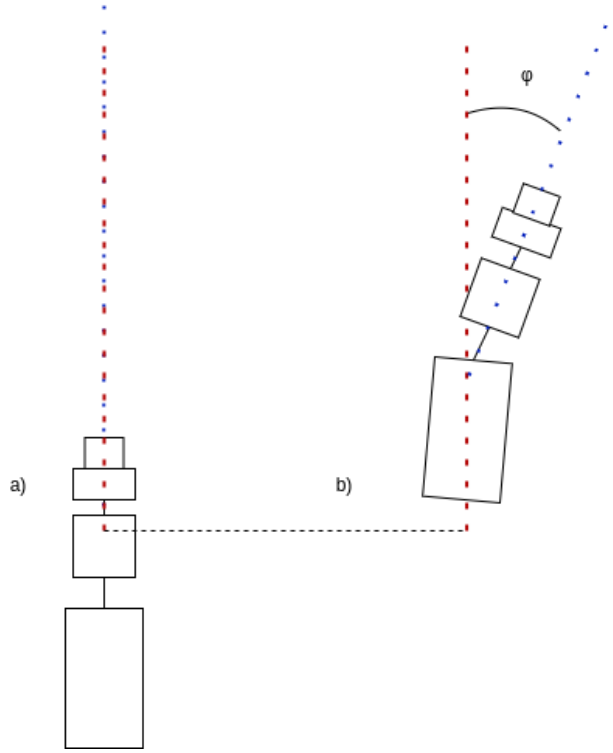


Figure 6: Illustration of the calibration phase of the steering angle. Red dashed line represent the planned path straight ahead. Blue dotted line represent the orientation of the tractor. **a)** The starting point of the calibration phase. The wheels are presumed to face straight forward at start. **b)** Presents a probable endpoint. The appropriate time interval between **a)** and **b)** is given as ΔT. φ is the bias of the steering angular after ΔT.

As the sampling characteristics of all involved subsystems are known can the bias of the steering angle be scaled down to each sample iteration yielding the steering bias per sample iteration φ_s.

4 COMMUNICATIONS SYSTEM

The communications system consists of the computing unit, RPi 4, along with a power supply unit.

4.1 Problem description

There is no implementation of the RPi on the truck currently. For this year's project the task will be to implement the computing unit, install all the required software and migrate existing nodes in ROS. Another problem is to figure out where to place the RPi and the power supply on the truck.

4.2 Communication with EV3

Sending and receiving data from an EV3 can be made using different types of technologies. Since both the EV3 and the RPi have Bluetooth, using a standard Bluetooth protocol is possible. However, sending and receiving data using any type of wireless connection is a suboptimal solution. A wired solution is to prefer for this application. There are several already existing interfaces for both wired and wireless connection.

4.2.1 *USB connection*

The EV3 has a 2.0 Mini-B port which the RPi can connect to. The EV3 USB device has a configuration with one interface and two end points, one for sending data and one for receiving data. Every message sent and received has a size of 1024 bytes and follows a specific format, otherwise, the message will be ignored or not sent at all by the EV3. The speed of the communication is up to 480 MBit/s when using a wired USB connection.

4.2.2 *Bluetooth connection*

The RPi and the EV3 must have Bluetooth enabled to be able to couple the two devices. The coupling of the devices can be initiated either from the EV3 or from the RPi as it will be done in this project. Bluetooth 2.1+ EDR is used on the EV3 which has a communication speed of 2.1 MBit/s, which is significantly lower than the USB connection.

5 VISUALIZATION SYSTEM

This section will describe the visualization system. The visualization system will be implemented on a computer which will be connected to a projector in Visionen, this to be able to project the map provided by the system.

5.1 Problem description

The main goal for the visualization system is to illustrate the map that the truck is navigating within. It will display the different virtual obstacles as well as the goal state for the truck, which for example could be a virtual loading bay, as Figure 7 shows. Furthermore, the system will show the optimized, calculated path for the truck, path from MPC controller, improved path from motion planner and also the path the truck has taken so far. Figure 7 shows an example of how it will look.

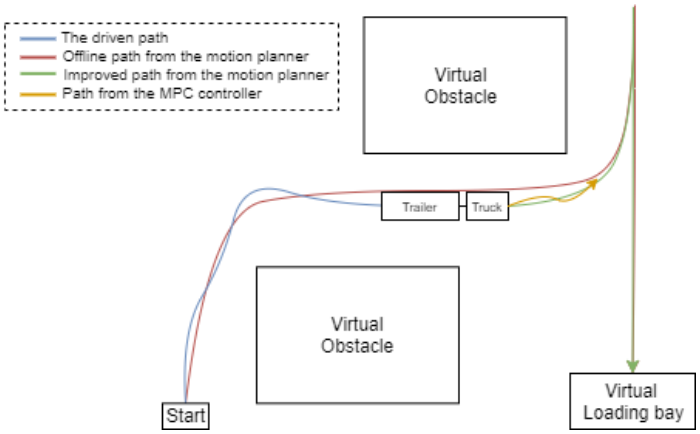


Figure 7: Visualization system

5.2 Wireless connection with RPi

The calculated and improved updated path from the planner will be communicated from the RPi to the computer connected to the projectors in Visionen via ROS topics.

5.3 Calibration

To make sure that the obstacles are visualized in the right positions a calibration has to be made. The reason for this is to make sure that the coordinates of the visualized obstacles match the positions known to the RPi. If the positions does not match some error message will be sent to the RPi on the truck.

5.4 Visualized paths

The visualization system will show the path driven by the truck. This implementation will import the position of the truck from the state observer and update the position of the truck in the map. As the mission proceeds the updated

position will generate a path which will be shown in the map by the visualization system, seen as the blue path in Figure 7. During the mission a plan will be calculated and updated by the MPC controller, this path will be displayed in the visualization, seen as the orange path in Figure 7. In the beginning of a mission the planned motion path will be sent from the RPi to the visualization system, this plan will be shown so the audience can see which path the truck is supposed to take, seen as the red path in Figure 7.

5.4.1 *Offline*

The initial path, given by the RPi to the visualization system, in the beginning of the mission will be shown in the visualization throughout the whole mission.

5.4.2 *Online*

As the mission for the truck proceeds a new updated path will be calculated by the motion planner at some frequency. This new path will then be displayed in the visualization system just like the initial path, seen as the green path in Figure 7.

REFERENCES

[1] K. Stubbs, P. J. Hinds, and D. Wettergreen, "Autonomy and common ground in human-robot interaction: A field study," *IEEE Intelligent Systems*, vol. 22, no. 2, pp. 42–50, 2007, [Online accessed: 21 September 2020].

[2] O. Ljungqvist, D. Axehill, H. Pettersson, and J. Lofberg, "Estimation-aware model predictive path-following control for a general 2-trailer with a car-like tractor," https://arxiv.org/abs/2002.10291, 2020, [online accessed: 30 September 2020].

[3] S. G. Enqvist Martin, Glad Torkel, *Industriell reglerteknik Kurskompendium*. Reglerteknik, Institutionen för systemteknik, Liköpings universitet, 581 83 Linköping, 2014.

[4] T. Fridén, L. Junler, A. Källström, O. L. Jonsson, T. Nyberg, and T. Westny, "Technical documentation, autonomous reversing truck," http://www.isy.liu.se/edu/projekt/tsrt10/2019/rev_truck/, 2019, [online accessed: 23 September 2020].

[5] T. Busk, M. Dushku, D. Forsberg, M. Hal, O. Karlsson, K. Larsson, J. Mourad, and S. Nilsson, "Technical documentation, liu racetrack 2018," http://www.isy.liu.se/edu/projekt/tsrt10/2018/racetrack/doc/Technical_Documentation.pdf, 2018, [online accessed: 28 September 2020].

[6] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.