Target Tracking of Manuvering Vehicle via Moving Horizon Estimation

Bordiga Raffaele
Corso Federico
Gemmani Giuliano

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Abstract

In this project the main goal was to design a Moving Horizon Estimation (MHE) filter to be integrated in the typical workflow of an object detection and tracking problem.
The solution is intended to handle optimally, in real-time, the uncertainties arising from limitations affecting sensors measurements, leveraging both a non-linear model of the tracked object and physical constraints.
The algorithm's performance is rigorously assessed through simulation in both Matlab and Simulink environments. Furthermore, it is compared against conventional methods such as the Extended Kalman Filtering (EKF) technique.

Contents

1. The Object-Detection and Tracking Problem

Driver assistance systems are engineered to enhance driver safety and reduce their operational burden. These technologies hinge on the precise interpretation of data obtained from sensors.
Notably, these systems heavily depend on the detection and subsequent tracking of objects in the proximity of the vehicle. These tasks usually rely on a robust and precise filtering algorithm, which is essential for conveying accurate environmental states to the control module, enabling timely and efficient actions on the controlled vehicle.
In the remaining of this chapter, a brief explanation of the target-tracking problem is provided. Moreover, we outline the core challenges that emerge while formulating a state estimation algorithm. These challenges represent the driving factors behind the formulation of our solution.

1.1 Vehicle Perception Main Challenges

To accomplish the perception task a vehicle need to overcome problems associated to incomplete sensory data, corrupted by measurement errors of various nature. Specifically, when dealing with tracking problems, it is of interest to develop accurate filtering algorithms that are robust with respect to inaccuracies in the measured physical variables, such as size, position, and speed of objects. Nevertheless, it's crucial to acknowledge computational limitations, especially in automotive applications.

1.2 Perception Task Workflow

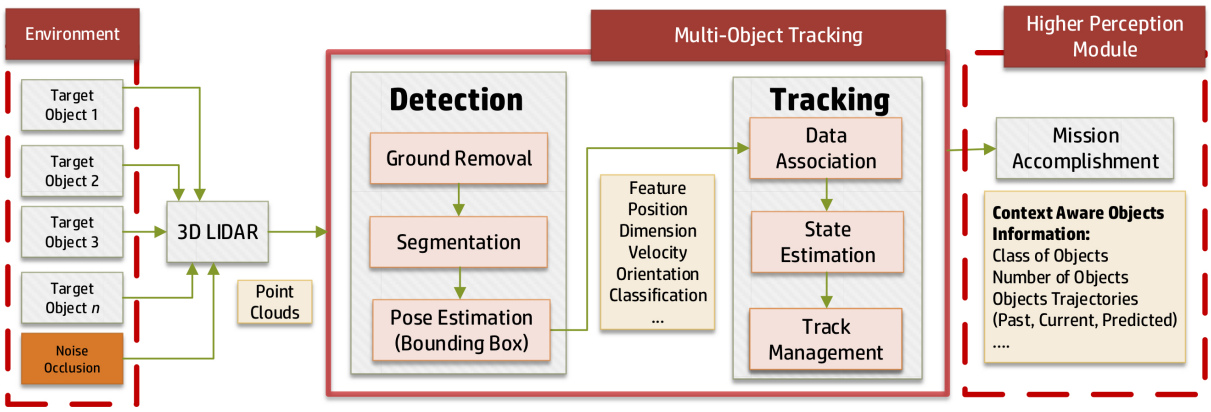


Figure 1.1: Object Detection And Tracking Workflow, From [10]

A widely spread approach to the tracking problem is the "tracking-by-detection" paradigm, whose workflow is represented, for lidar sensor measurements, in Figure 1.1.
In this process, the raw sensor data first undergo the detection stage, being pre-processed and elaborated to extract information about the objects of interest.
This information is then fed into a state estimation filter to predict their kinematic states based on a dynamic motion model. The tracker's main obstacles are measurement uncertainties such as noises, disturbances and outliers.
To address these uncertainties optimal Bayesian filters such as the Kalman Filter (KF) and Particle Filter are usually extensively employed.

1.2.1 State Estimation Fundamentals

A widely embraced technique for addressing the state estimation or tracking problem is the "point tracking" approach. In this paradigm, targets are conceptualized as mass-points navigating within the environment.
An essential motivation for prioritizing this approach over alternatives like "extended object tracking" is its superior scalability enabled by the use of a limited number of filtered states. This could be translated into a lower computational demand.
Following the above mentioned approach, the tracking target dynamic motion model is usually described by a discrete time, stochastic state-space model with additive noise components, as in the following:

$$\mathbf{z}(t+1) = f_{\mathbf{z}}(\mathbf{z}(t)) + \mathbf{d}(t) \tag{1.1}$$

In various conventional approaches, the problem is often simplified as planar, neglecting vertical dynamics. The state vector $\mathbf{z} \in \mathbb{R}^{n_z}$ usually encompasses position, velocity, and acceleration components along both dimensions. Meanwhile, $\mathbf{d} \in \mathbb{R}^{n_d}$ is typically regarded as a white Gaussian noise (WGN) with specified mean and variance, representing the inherent uncertainty associated with state knowledge.

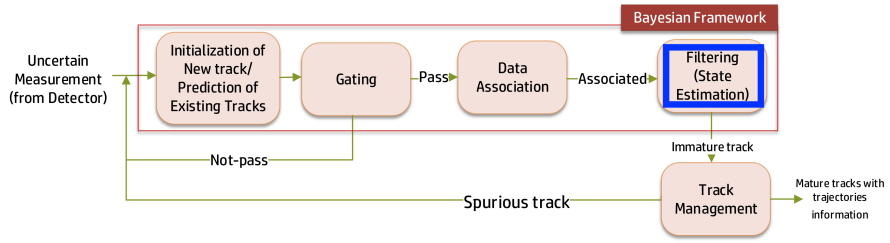


Figure 1.2: Object Tracking Workflow and Role of State Estimation, from [10]

Once the state and the model of the system has been defined, the tracking problem is addressed from a filtering perspective.
Across the literature, Bayesian filtering methodologies are predominantly adopted. Although they are simple and computationally efficient, these approaches (e.g. KF), exhibit several limitations which lead to sub-optimal solutions when applied to real systems.
For instance, in scenarios with nonlinear dynamics, linear approximations are necessary to build an optimal filter, such as EKF. Furthermore, it requires noise's whiteness, otherwise it might be affected by estimation divergence.
To deal with these shortcomings and maintaining real-time feasibility of the estimation process, we decided to tune an optimization-based algorithm in the framework of MHE.
Our proposed algorithm will undergo rigorous testing and comparison against a conventional EKF. This evaluation will be conducted in the context of tracking a maneuvering vehicle navigating along a curved road. Through this evaluation, we seek to demonstrate the effectiveness of our solution in managing complex scenarios marked by nonlinear dynamics and unstructured uncertainties.
In the remaining of our work, after carefully modelling the problem, we will present the MHE general framework. Subsequently, we will present an unconstrained formulation of our problem, followed by a constrained version. Lastly, the outcomes resulting from these various approaches will be examined and discussed.

2. Problem Modelling and Objectives

The project's core objective is to create a real-time optimization-based filtering algorithm that surpasses the primary limitations associated with classical Bayesian filtering approaches.
In this section we outline all the underlying modeling assumptions, ranging from the employed reference frames to the characteristics of synthetic measurement data. A consistent modeling allowed us to create a robust baseline to effectively compare the results of different estimation processes.

2.1 Reference Frames

Our tracking problem is assumed to be confined to a planar context, disregarding any vertical dynamics of the vehicles. Moreover, two principal reference frames are taken into account: a global frame and a local frame affixed to the ego-vehicle. We will refer to the world coordinates as X, Y, and Z, while the coordinates with respect to the vehicle frame will be denoted as x, y, and z.

2.1.1 World Frame

All vehicles, sensors, and their related coordinate systems are placed in the world frame. A global coordinate system is important in global path planning, localization, mapping, and driving scenario simulation. We assume a right-handed cartesian world coordinate system with the Z-axis pointing up from the ground. Distances in this frame are measured in meters m, while rotations, expressed in radians rad, are assumed to be positive in the counterclockwise direction.

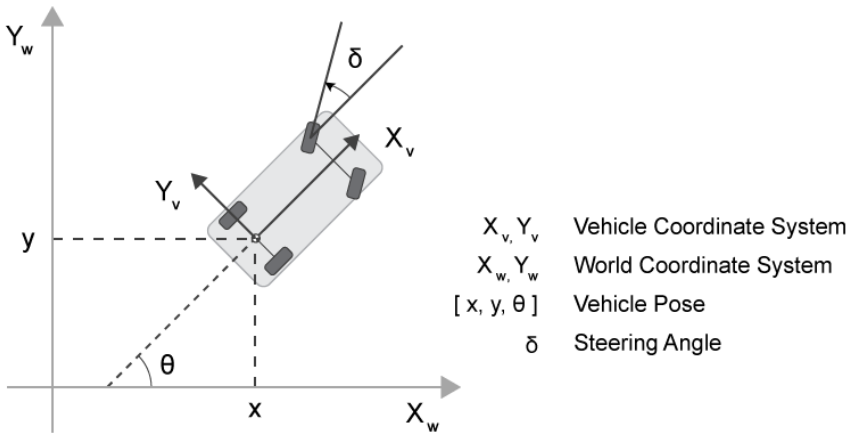


Figure 2.1: World Reference Frame

2.1.2 Vehicle frame

The vehicle coordinate system (x, y, z), represented in Figure 2.2, is right handed and anchored to the ego vehicle. Rotations and distances follows the same conventions mentioned in 2.1.1.

2.2 Driving Scenario Simulations

We performed simulations and synthetic data generation through the use of "Automated Driving Toolbox" from MATLAB [7]. The use of this toolbox simplified the generation of the desired scenario that allowed to gather data to test our filtering algorithms. Through the scenario generation process we formulated some assumptions to facilitate the formulation of physical constraints for the subsequent

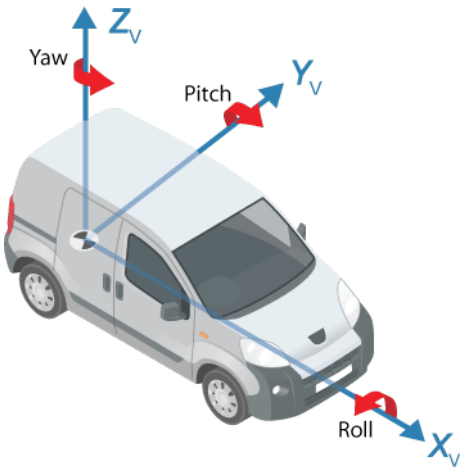


Figure 2.2: Vehicle Reference Frame

filtering problem. These assumptions are explained in the following subsection.

2.2.1 Road Geometry

Given our objectives, we assumed the two cars moving on a curved road made of 2 lanes, each one with a constant width of 3.6 m. The road center-line describes an arc of circumference delimited by an angle of $\pi/3$ rad and with a constant curvature equal to $1/R = 1/760$ $1/m$. The total length of the path is almost 796 m; its initial and terminal points have coordinates $(X, Y) = (0, -760)$, $(X, Y) = (658, -380)$, respectively. A visual representation is provided in Figure 2.3.

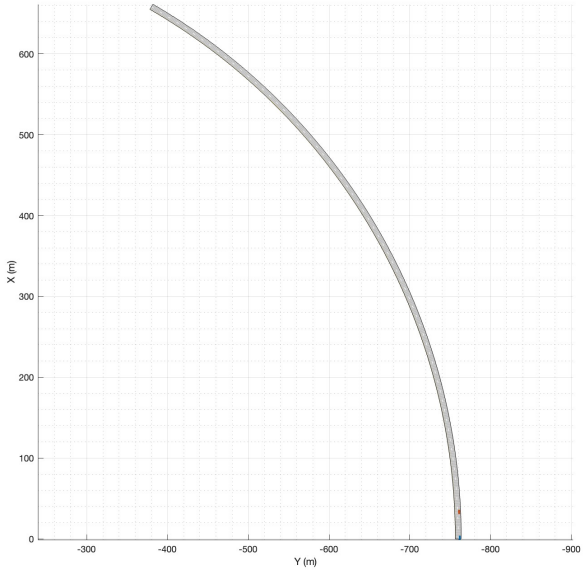


Figure 2.3: Road Geometry

2.2.2 Vehicle Geometry

Vehicles geometry has been defined according to the "Automated Driving Toolbox" as reported in Figure 2.4 and Table 2.1.
It's important to note that the origin of the local reference attached to the vehicle is placed in the

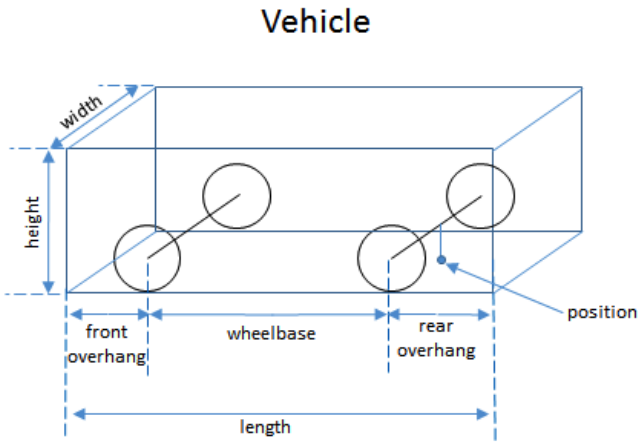


Figure 2.4: Cuboid Vehicle Dimensions

Length	Width	Height	Wheelbase	Front Overhang	Rear Overhang
4.7	1.8	1.4	2.8	0.9	1.0

Table 2.1: Cuboid Vehicle Dimensions, in meters m

projection of the middle point of the rear axis on the ground. Ground truth signals will refer to the position of this point.

2.3 Target Model

In the literature of target tracking problems, the tracking target dynamic motion model is usually described mathematically by a discrete time, stochastic state-space model as the one in 2.1

$$\begin{cases} \mathbf{z}(t+1) = f_{\mathbf{z}}(\mathbf{z}(t)) + \mathbf{d}(t) \\ \mathbf{y}(t) = g_{\mathbf{z}}(\mathbf{z}(t)) + \mathbf{w}(t) \end{cases} \tag{2.1}$$

Here, $f_{\mathbf{z}}$ represents a function $f_{\mathbf{z}} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ expressing the evolution of the state trajectories over time. $\mathbf{d} \in \mathbb{R}^{n_z}$ is a WGN process noise which represents discrepancies between the model and the actual system dynamics. $g_{\mathbf{z}} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ characterizes the measurement model, thus expressing dependency of sensor data on the system states. Finally $\mathbf{w} \in \mathbb{R}^{n_y}$ incorporates uncertainty inherent to measurements as WGN.

2.3.1 Motion Models

In line with the insights presented in [6], for road vehicle target tracking distinguishable driving patterns can be classified as a mixture of different motion models, for example constant velocity rectilinear motion and constant angular velocity curvilinear motion.
In our project, we assumed the ego-vehicle moving at a constant speed along the road mentioned in 2.2.1, while a target vehicle is moving right in front of it, with a constant speed as well, see Figure 2.5. Because of these assumptions, we chose a constant turn-rate velocity model (CTRV) to describe the target dynamics.

2.3.2 Constant Turn-Rate Velocity Model

The Constant Turn-rate Velocity Model is extensively employed to describe the motion of maneuvering targets, as detailed in [11]. In the following, we will initially present the inherent continuous dynamics of the model, followed by its discretized counterpart.

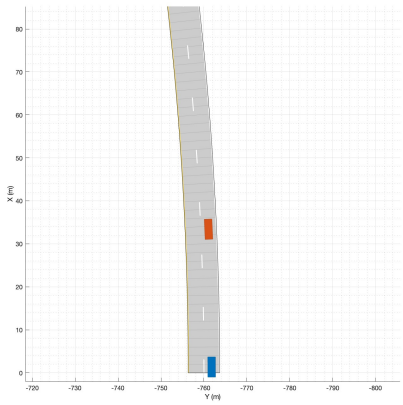


Figure 2.5: Ego Vehicle (blue) and Target (red) in the Scenario

Continuous Time CTRV Model in Cartesian Coordinates

Following the Cartesian parametrization we denoted the target state variables, defined with respect to the ego vehicle frame, as follows:

- $x(\tau)$ in m: target's longitudinal position
- $y(\tau)$ in m: target's lateral position
- $v_x(\tau)$ in m/s: target's longitudinal velocity
- $v_y(\tau)$ in m/s: target's lateral velocity
- $\omega(\tau)$ in rad/s: target's turn rate

Moreover, the relative heading of the target, $\theta(\tau)$, can be expressed as a function of the speed components by recalling that $\theta(\tau) = \text{atan2}(v_y(\tau), v_x(\tau))$, specified in rad.
Additionally, we consider the target to be influenced by linear and rotational acceleration inputs, denoted as $a(\tau)$ m/s^2 and $\alpha(\tau)$ rad/s^2 respectively. The linear acceleration a acts tangentially to the target's path, aligned with the total speed direction. In contrast, $\alpha(\tau)$ accounts for changes in the turn rate.
The full expression of the continuous time model is then given by 2.2.

$$\begin{bmatrix} \dot{x}(\tau) \\ \dot{y}(\tau) \\ \dot{v}_x(\tau) \\ \dot{v}_y(\tau) \\ \dot{\omega}(\tau) \end{bmatrix} = \begin{bmatrix} v_x(\tau) \\ v_y(\tau) \\ a(\tau)cos(\psi(\tau)) - v_y(\tau)\omega(\tau) \\ a(\tau)sin(\psi(\tau)) + v_x(\tau)\omega(\tau) \\ \alpha(\tau) \end{bmatrix} \tag{2.2}$$

Here, τ symbolizes the continuous time ($\tau \in \mathbb{R}$), serving as a distinction from the symbol t which is used to represent discrete time steps ($t \in \mathbb{Z}$).

Discrete Time CTRV Model

To employ discrete-time filtering methodologies, the equations in 2.2 need to be discretized. For the sake of clarity in notation, hereon, state variables will be denoted by the symbol z. Consequently, the state vector can be expressed as:

$$\mathbf{z}(t) = \begin{bmatrix} x(t) & y(t) & v_x(t) & v_y(t) & \omega(t) \end{bmatrix}^T \in \mathbb{R}^{n_z} \tag{2.3}$$

Assuming a sampling time denoted by T_s ($T_s \in \mathbb{R}^+$), the equation 2.2 can be reformulated into a discrete-time model with the general form:

$$\mathbf{z}(t+1) = f_{\mathbf{z}}(\mathbf{z}(t)) + G(\mathbf{z}(t))\mathbf{d}(t) \tag{2.4}$$

In this context $\mathbf{d}(t) = [a(t)\ \alpha(t)]^T$ represents the acceleration inputs as process disturbances.
For the purpose of discretization, it is assumed that $a(\tau)$ and $\alpha(\tau)$ are set equal to 0. This allows to get to the following compact and exact expression of $f_{\mathbf{z}}$:

$$f_{\mathbf{z}}(\mathbf{z}(t)) = \begin{bmatrix} x(t) + \frac{v_x(t)}{\omega(t)}\sin(\omega(t)T_s) - \frac{v_y(t)}{\omega(t)}(1-\cos(\omega T_s)) \\ y(t) + \frac{v_x(t)}{\omega(t)}(1-\cos(\omega(t)T_s)) + \frac{v_y(t)}{\omega(t)}\sin(\omega T_s) \\ v_x(t)\cos(\omega(t)T_s) - v_y(t)\sin(\omega T_s) \\ v_x(t)\sin(\omega(t)T_s) + v_y(t)\cos(\omega T_s) \\ \omega(t) \end{bmatrix} \tag{2.5}$$

To determine the matrix $G(\mathbf{z})$, it is assumed that $a(t)$ and $\alpha(t)$ are constant within each sampling interval, following a zero-order-hold discretization approach. Consequently, the resulting matrix is defined as:

$$G(\mathbf{z}(t)) = \begin{bmatrix} \frac{T_s^2}{2}\cos(\psi(t)) & 0 \\ \frac{T_s^2}{2}\sin(\psi(t)) & 0 \\ T_s\cos(\psi(t)) & 0 \\ T_s\sin(\psi(t)) & 0 \\ 0 & T_s \end{bmatrix} \tag{2.6}$$

Up to this point, due to the necessity of having a discretized model, the inputs $a(t)$ and $\alpha(t)$ have been presumed to be deterministic signals. However, once the system model has been discretized, it becomes feasible to offer a probabilistic characterization of these inputs, represented as $[a(t)\ \alpha(t)]^T$.
These signals are modeled as uncorrelated random variables, in particular as WGN , i.e. $a(t) \sim WGN(0, \sigma_a)$ in m/s^2 and $\alpha(t) \sim WGN(0, \sigma_\alpha)$ in rad/s^2. Choosing distributions with zero-mean aligns with the hypothesis of the CTRV Model.

2.4 Sensors Modelling

The so called advanced driver assistance system (ADAS) rely on multiple types of sensors, such as sonar, radar, lidar, and cameras. To deal with an object-detection and tracking problem it's fundamental to devise a sensor fusion algorithm to combine the advantages provided by the various types of sensors included in the vehicle.
In our scenario, the Ego-Vehicle is equipped with both a front-facing radar and a camera, each positioned in the front of the vehicle. Considering the computational load of the detection stage, as seen in Figure 1.1, it's reasonable to assume that sensor readings occur at sampling frequency of $f_s = 10Hz$ [12]. This seemingly low frequency suits the real-time requirements of target tracking applications.

2.4.1 Vision Sensor Simulation

As explained in detail in [9], cameras offer accurate lateral position and velocity measurements, whereas they exhibit limitations in accurately capturing the longitudinal dynamics of a moving target. Moreover, their accuracy degrades proportionally with relative distance from the sensor.
In our case, we consider that the vision sensor returns at each sampling time the following measured target's states expressed into the ego vehicle frame, $\mathbf{y}(t) = \begin{bmatrix} x(t) & y(t) & v_x(t) & v_y(t) \end{bmatrix}^T$.

Camera Synthetic Data Generation Parameters

To accurately capture the above mentioned behaviour, camera data has been synthetically generated according to the following methodology. Firstly, we used the Automated Driving Toolbox to generate more realistic data. These data significantly deviates from the ground truth signals as they are corrupted both from a changing variance, a varying bias and the presence of false detections. However,

to tackle an easier problem, we decided to generate some bias free measurements by adding a WGN to the ground truth signals. In this way we avoided to include the sensor biases into the formulation of our state space model and filtering problem. Moreover, we considered the measurement noise variance as stationary, by averaging over the whole simulation steps the covariances of the Automated Driving Toolbox dataset. These average variances are reported in Equation 2.7. We also included false detections, which were generated by drawing, every 0.1 s, from a zero-mean white Gaussian noise signal with covariance kR^{meas}_{camera}, where k was suitably selected. At each sampling instant, we determined whether a measurement was an outlier or not based on a Bernoulli distribution. The probability of success in this distribution was designed to increase linearly with the distance of the target from the ego-vehicle.

$$R^{meas}_{camera} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_{v_x}^2 & 0 \\ 0 & 0 & 0 & \sigma_{v_y}^2 \end{bmatrix} = \begin{bmatrix} 0.901 & 0 & 0 & 0 \\ 0 & 0.0318 & 0 & 0 \\ 0 & 0 & 0.4851 & 0 \\ 0 & 0 & 0 & 0.0610 \end{bmatrix} \tag{2.7}$$

2.4.2 Radar Sensor Simulation

We assumed that the radar on the vehicle was returning measurements of the same state variables as the camera. Following the description in [8], radars provide accurate measurements of $x(t)$ and $v_x(t)$, while the quality of measurements degrades in terms of $y(t)$ and $v_y(t)$ variables.

Radar Synthetic Data Generation Parameters

We generated radar synthetic data following the same procedure as for camera. The covariance matrix reported in Equation 2.8 has larger entries to reflect the overall lower accuracy of the radar. However, radar false detections were drawn from a Bernoulli distribution with a slightly lower success rate to model the lower likelihood of having a false positive.

$$R^{meas}_{radar} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_{v_x}^2 & 0 \\ 0 & 0 & 0 & \sigma_{v_y}^2 \end{bmatrix} = \begin{bmatrix} 3.1143 & 0 & 0 & 0 \\ 0 & 1.4888 & 0 & 0 \\ 0 & 0 & 0.7242 & 0 \\ 0 & 0 & 0 & 9.92779 \end{bmatrix} \tag{2.8}$$

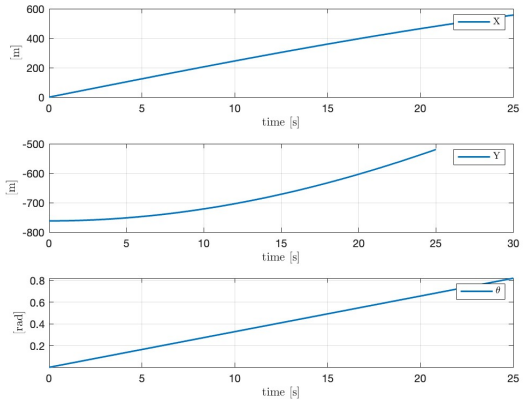
Summary

In summary, cameras excel in providing accurate lateral position and velocity measurements over extended ranges, yet exhibit diminished longitudinal accuracy at greater distances. Conversely, radar offers accurate longitudinal position and velocity measurements over extensive ranges, but its lateral accuracy diminishes at longer distances. In the design of our state estimation algorithm, we extensively considered these aspects to effectively fuse measurements from both sensors. This fusion is crucial to attain the most accurate estimate possible.

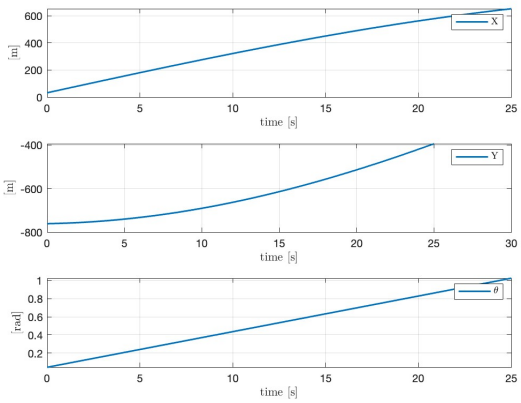
2.4.3 Other Simulation Info

All the following estimation algorithms are tested on the same data-set that is obtained following the above modelling assumptions. Here we also specify some additional information about the simulations:

- The ego-vehicle is assumed to go through the curved road by following the center of the rightmost lane with a constant tangential speed of $25m/s$.
- The target vehicle starts around $10m$ in front of the ego-car and moves with an higher constant speed of $30m/s$ along the same lane.
- The total simulation time is $T_{sim} = 25s$.

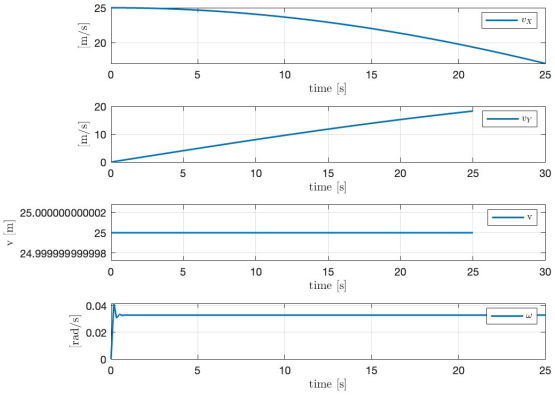


(a) Ego Vehicle Pose.

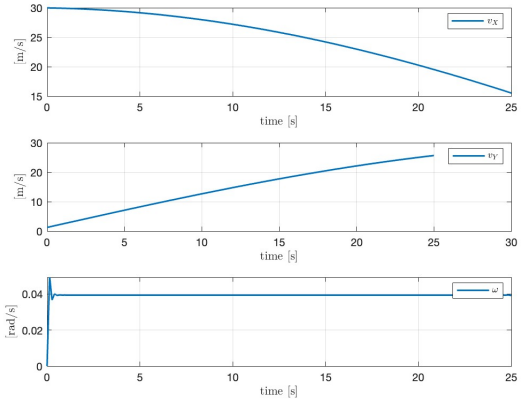


(b) Target Pose.

Figure 2.6: Ego Vehicle and Target Vehicle Pose in the World Frame



(a) Ego Vheicle Speeds.



(b) Target Vheicle Speeds.

Figure 2.7: Ego Vehicle and Target Vehicle Speeds in the World Frame

2.4.4 Metrics Used to Evaluate Performances

Throughout the report we decided to evaluate the performances of the various filtering algorithms by considering:

- The $RMSE$ (Root Mean squared error) between the ground truth and the estimated value of each state variable.
- An average computational cost for each estimation cycle lower than the sensor's sampling time of 0.1 s. In principle, it should ensure a real-time implementation of the estimation process.

3. Algorithm Design

In this chapter, we will present the target tracking problem in the context of optimization. At first, we'll abstract the tracking problem and then we'll formulate the MHE both in unconstrained and constrained form. Finally, we'll present Multiple-Shooting (MS) technique, which will be compared with Single-Shooting (SS).

3.1 The Moving Horizon Estimation Problem

MHE is an optimisation algorithm devoted to filtering process. Its primary objectives are the estimation of un-measurable states variables and the reduction of noise interference. In our case, what has to be estimated are both state variables, denoted as $\hat{\mathbf{z}}(t)$, and process noise denoted as $\hat{\mathbf{d}}(t)$.
The MHE algorithm possesses a unique capability to overcome the limitations of conventional filters like the KF and its extensions. It remains optimal even in the presence of system dynamics non-linearities and constraints. However, this advantage does come at the cost of increased computational demands. Nevertheless, due to our low sampling frequency of only 10 Hz, the computational load is manageable, making the implementation of a real-time MHE algorithm a reasonable and viable solution.

3.2 The Cost Function

In this subsection, we derive the objective function regarding the MHE problem from a probabilistic perspective. Subsequently, we will adapt it to our specific problem.

3.2.1 Probabilistic Derivation of the MHE Cost Function

Assuming a model as presented in Equation 2.4, a state estimator is tuned in order to solve the following optimization problem:

$$\mathbf{z}^*(t) = \underset{\mathbf{z}(t)}{\operatorname{argmax}}\ \mathbb{P}(\mathbf{z}(t)|\mathbf{y}(0), ..., \mathbf{y}(t)) \tag{3.1}$$

This program is commonly known as the *maximum likelihood estimation problem* [2]. The solution involves determining the most probable state at a given time instant, denoted as t, given all the measurements up to t.
An alternative approach to Equation 3.1 is to optimize based on the joint probability of observing an entire state trajectory.

$$\mathbf{z}^*(0), \dots, \mathbf{z}^*(t) = \underset{\mathbf{z}(0),\dots\mathbf{z}(t)}{\operatorname{argmax}}\ \mathbb{P}(\mathbf{z}(0), \dots, \mathbf{z}(t)|\mathbf{y}(0), \dots, \mathbf{y}(t)) \tag{3.2}$$

This formulation is not well-posed for our purpose, so we decided to switch to *log-likelihood estimation problem* and then considering the so called *Full information problem*. All the computations are referred to [2].

Assuming to know a priori the initial state estimate denoted as $\mathbf{z}^-(0) \sim \mathcal{N}(\mathbf{z}(0), P(0))$, we define our *Full information problem* via equation 3.3.

$$\begin{aligned}
\min_{\mathbf{z}(0),\dots,\mathbf{z}(t)} \quad & \Gamma(\mathbf{z}(0)) + \sum_{i=0}^{t-1} ||\mathbf{d}(i)||^2_{\ Q^{-1}} + \sum_{i=0}^{t} ||\mathbf{w}(i)||^2_{\ R^{-1}} \\
\text{subject to} \quad & \mathbf{z}(i+1) = f_{\mathbf{z}}(\mathbf{z}(i)) + G(\mathbf{z}(i))\mathbf{d}(i)),\ i = 0, \dots, t-1. \\
& \mathbf{y}(i) = g_{\mathbf{z}}(\mathbf{z}(i)) + \mathbf{w}(i),\ i = 0, \dots, t. \\
& \Gamma(\mathbf{z}(0)) = ||\mathbf{z}(0) - \mathbf{z}^-(0)||^2_{\ P_0^{-1}}
\end{aligned} \tag{3.3}$$

Here we can distinguish three terms constituting the cost function:

- $\Gamma(\mathbf{z}(0))$, arrival cost
- $\sum_{i=0}^{t-1} ||\mathbf{d}(i)||^2_{2\ Q^{-1}}$, it penalizes the effect of process noise on states
- $\sum_{i=0}^{t} ||\mathbf{w}(i)||^2_{2\ R^{-1}}$, it penalizes the effect of noise on the measurements

The term $\Gamma(\mathbf{z}(0))$ corresponds to the *Terminal Cost* (or *Arrival cost*) which penalizes how much the prior knowledge deviates from the initial value. Matrices Q and R are covariance matrices related to noise respectively on states and outputs. Other equality constraints both define state and output trajectory according to equation 2.4.
From this formulation the MHE is derived by assuming a finite horizon, thus a limited amount of past data. We denote the horizon length as $T = T_s \cdot M$, where $M \in \mathbb{Z}^+$ is a tunable parameter. By adjusting the value of M, the length of the horizon can be extended or shortened, affecting the number of data samples used for estimation.
The formulation of MHE optimisation program results in:

$$\begin{aligned} \min_{\mathbf{z}(-M|t),\dots,\mathbf{z}(0|t)} \quad & \Gamma(\mathbf{z}(-M|t)) + \sum_{i=1}^{M} ||\mathbf{d}(-i|t)||^2_{2\ Q^{-1}} + \sum_{i=0}^{M} ||\mathbf{w}(-i|t)||^2_{2\ R^{-1}} \\ \text{subject to} \quad & \mathbf{z}(i+1|t) = f_{\mathbf{z}}(\mathbf{z}(i|t)) + G(\mathbf{z}(i))\mathbf{d}(i|t)),\ i = -M, \dots, -1. \\ & \mathbf{y}(i|t) = g_{\mathbf{z}}(\mathbf{z}(i|t)) + \mathbf{w}(i),\ i = -M, \dots, 0. \\ & \mathbf{z} \in Z, \mathbf{d} \in D, \mathbf{w} \in W \end{aligned} \tag{3.4}$$

Where the sets Z, D, and W define the feasible ranges for the state variables, process noises, and measurement noises.

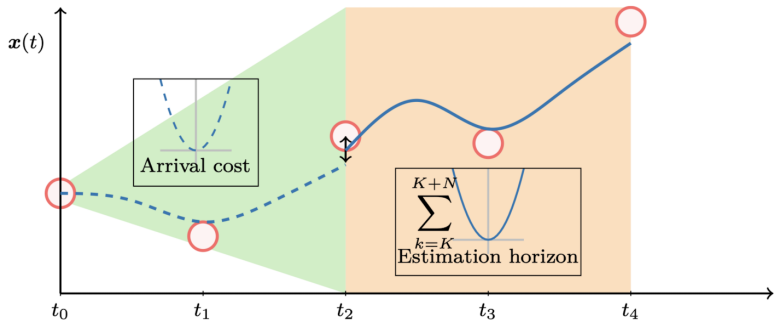


Figure 3.1: Visualization of an iteration of MHE algorithm. The dotted line represent the history, encoded in the terminal cost. The orange window instead represent the horizon on which current estimate is computed.

3.2.2 Formulating an MHE Problem for Vehicle Target Tracking

Measuremnent Noise

The cost function in 3.10 needs to be re-arranged to our specific context. Since we acquire measurement from two distinct sensors (radar and camera), a noise component should be highlighted in the cost function for each sensor. These components are weighted by their respective covariance matrices. This technique follows a *sensor fusion approach* and it aims to optimize the usage of the most precise information offered by both sensors.

$$\sum_{i=0}^{M} ||\mathbf{w}(-i|t)||^2_{2\ R^{-1}_{camera}} + \sum_{i=0}^{M} ||\mathbf{w}(-i|t)||^2_{2\ R^{-1}_{radar}} \tag{3.5}$$

Subsequently, in order to explicitly incorporate the influence of both measured data and state variables into the cost function, the measurement noise is expressed as follows:

$$\mathbf{w}(-i|t) = \mathbf{y}(t-i) - \hat{\mathbf{y}}(-i|t) = \mathbf{y}(t-i) - g_{\mathbf{z}}(\hat{\mathbf{z}}(-i|t))\ i = M, \dots, 1 \tag{3.6}$$

The term $\mathbf{y}(t-i)$ denotes the sensor measurements, as specified in Section 2.4, acquired at time $t-i$, whereas $\hat{\mathbf{y}}(-i|t)$ represents the corresponding filtered quantity. In our case, $g_{\mathbf{z}}$ is a linear mapping $g_{\mathbf{z}} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ expressed through $H \in \mathbb{R}^{n_y \times n_z}$ which extracts all the measurable states from the filtered state vector $\hat{\mathbf{z}}(-i|t)$. We recall that $n_z = 5$ as in 2.3, and $n_y = 4$.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.7}$$

Terminal Cost

We formulated the terminal cost as the weighted l_2-norm 3.8. By selecting the weighting matrix P we were able to change the algorithm's performances in terms of both accuracy and computational efficiency. For this reason, it will have a more detailed description in further subsections.

$$||\hat{\mathbf{z}}(-M|t) - \hat{\mathbf{z}}(-M+1|t-1)||^2_{2\ P^{-1}} \tag{3.8}$$

Input Disturbances

Finally, as is customary in MHE formulations, we have included process noise in the set of decision variables. In our formulation, the term associated to these variables is equal to the one in the general formulation 3.10.

The Cost Function

Bringing everything together and introducing the vector of decision variables $\mathbf{x} \in \mathbb{R}^n$ for simplicity, where $n = n_z \cdot (M+1) + n_d \cdot M$, we have:

$$\mathbf{x} = \begin{bmatrix} \hat{\mathbf{z}}(-M|t)^T & \dots & \hat{\mathbf{z}}(0|t)^T & \hat{\mathbf{d}}(-M|t)^T & \dots & \hat{\mathbf{d}}(-1|t)^T \end{bmatrix}^T$$

Now, we can rewrite our cost function as a sum of weighted quadratic terms, $f : \mathbb{R}^n \to \mathbb{R}$:

$$\begin{aligned} f(\mathbf{x}) = ||\hat{\mathbf{z}}(-M|t) - \hat{\mathbf{z}}(-M+1|t-1)||^2_{2\ P^{-1}} + \\ + \sum_{i=1}^{M} ||\hat{\mathbf{d}}(-i|t)||^2_{2\ Q^{-1}} + \sum_{i=0}^{M} ||\mathbf{y}(t-i) - H\ \hat{\mathbf{z}}(-i|t)\ ||^2_{2\ R^{-1}_{camera}} + \\ + \sum_{i=0}^{M} ||\mathbf{y}(t-i) - H\ \hat{\mathbf{z}}(-i|t)\ ||^2_{2\ R^{-1}_{radar}} \end{aligned} \tag{3.9}$$

Being the weighting matrices positive definite and being the norms always convex, our cost function is convex, more precisely it's a quadratic function.

Final Formulation

Finally the full MHE problem for our purpose can be written as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{z}(i+1|t) = f_{\mathbf{z}}(\mathbf{z}(i|t)) + G(\mathbf{z}(i))\mathbf{d}(i|t)),\ i = -M, \dots, -1. \\ & \mathbf{z} \in Z, \mathbf{d} \in D, \mathbf{w} \in W \end{aligned} \tag{3.10}$$

3.2.3 The Cost Function as a Quadratic Form

Since $f(\mathbf{x})$ is composed as sum of squared quantities, it is possible to write the cost function in the following form: $f(\mathbf{x}) = F(\mathbf{x})^T F(\mathbf{x})$, Where $F : \mathbb{R}^n \to \mathbb{R}^N$, with $N = n_z + 2 \cdot n_y \cdot (M+1) + n_d \cdot M$. This suggests using the Gauss-Newton method to compute the search direction for solving the optimization problem.

$$F(\mathbf{x}) = \begin{bmatrix} P(-M+1|t-1)^{-\frac{1}{2}} \cdot (\hat{\mathbf{z}}(-M|t) - \hat{\mathbf{z}}(-M+1|t-1)) \\ Q^{-\frac{1}{2}} \cdot \hat{\mathbf{d}}(-1|t) \\ \vdots \\ Q^{-\frac{1}{2}} \hat{\mathbf{d}}(-M|t) \\ R_{camera}^{-\frac{1}{2}}(\mathbf{y}(0|t) - H\hat{\mathbf{z}}(0|t)) \\ \vdots \\ R_{camera}^{-\frac{1}{2}}(\mathbf{y}(-M|t) - H\hat{\mathbf{z}}(-M|t)) \\ R_{radar}^{-\frac{1}{2}}(\mathbf{y}(0|t) - H\hat{\mathbf{z}}(0|t)) \\ \vdots \\ R_{radar}^{-\frac{1}{2}}(\mathbf{y}(-M|t) - H\hat{\mathbf{z}}(-M|t)) \end{bmatrix} \tag{3.11}$$

With:

$$\begin{aligned} P(-M+1|t-1) &= \text{diag}(p_x^2, p_y^2, p_{v_x}^2, p_{v_y}^2, p_\omega^2) \\ Q &= \text{diag}(q_a^2, q_\alpha^2) \\ R_{radar} &= \text{diag}(r_{r_x}^2, r_{r_y}^2, r_{r_{v_x}}^2, r_{r_{v_y}}^2) \\ R_{camera} &= \text{diag}(r_{c_x}^2, r_{c_y}^2, r_{c_{v_x}}^2, r_{c_{v_y}}^2) \end{aligned} \tag{3.12}$$

With this formulation, we can observe that the lower the variance of the i-th component of the covariance matrix, the more the corresponding element is penalized in the optimization problem. That is, the more important will be the information provided by that element in filtering. Consequently, those matrices can be seen as tuning parameters of the filtering algorithm. Additionally, they can be designed taking into account the different orders of magnitudes in the optimization variables.

3.2.4 Terminal Cost Update

Given the cost function 3.9, an accurate choice of the *Terminal cost* allowed us to balance computational effort with respect to estimation accuracy as this term collects past information not belonging to the considered horizon. In our project we tested two different approaches.

Constant Arrival Cost

Using a constant terminal cost is a straightforward but suboptimal choice.

$$||\hat{\mathbf{z}}(-M|t) - \hat{\mathbf{z}}(-M+1|t-1)||^2_{\bar{P}^{-1}} \tag{3.13}$$

We can interpret this arrival cost as a fixed penalty on the variation of the current estimate $\hat{\mathbf{z}}(-M|t)$ from an a priori value $\hat{\mathbf{z}}(-M+1|t-1)$. The latter is obtained computing the state trajectory, hence $\hat{\mathbf{z}}(-M+1|t-1) = f_{\mathbf{z}}(\hat{\mathbf{z}}(-M|t-1)) + G\hat{\mathbf{d}}(-M|t-1)$, where $\hat{\mathbf{z}}(-M|t-1)$ represent the solution from the previous filter iteration at step $t-1$.
The degree of penalization is chosen through the diagonal matrix $\bar{P}$, which represents the uncertainty on the state at the beginning of the horizon. As in the cost function we take its inverse, the lower the uncertainty $\bar{P}_i$ on the i^{th} state component, the less $\hat{\mathbf{z}}_i(-M|t)$ can change with respect to $\hat{\mathbf{z}}_i(-M+1|t-1)$.

For example, as velocities have inherently higher variance than positions, the corresponding entry could be weighted more.

EKF Arrival Cost

The idea of the *smoothed EKF arrival cost* has the aim to obtain recursively an a priori estimate of both $\mathbf{z}(-M+1|t-1)$ and $P(-M+1|t-1)$ based on past information. The algorithm that is designed to carry out this updates is analogous to the prediction and correction algorithm of a normal EKF. However, these steps are reversed, i.e. correction is performed before the update. The algorithm that we implemented is reported in [4]. Moreover, to be robust with respect to possible approximation errors, we also implemented the *Joseph Stabilized Update* of the state covariance matrix, reported in [1]. The EKF terminal cost update has enabled us to achieve a significant improvement in accuracy while keeping computational costs manageable compared to using a constant arrival cost as we've seen in section 3.2.4.

3.3 Unconstrained Problem

The first step towards obtaining an MHE algorithm outperforming a classical EKF approach is to solve the unconstrained problem. This allowed us to get a deeper understanding of how tuning parameters such as the window length M and the covariance matrices influence the final solution, both in terms of accuracy and computational cost.
Physical constraints on states and process noise are thus ignored and a SS approach allowed also to reformulate the problem with a lower number of decision variables ($n = n_z + M \cdot n_d$), i.e. only the state at the beginning of window is optimized, hence $\mathbf{x} = \begin{bmatrix}\hat{\mathbf{z}}(-M|t)^T & \hat{\mathbf{d}}(-M|t)^T & \dots & \hat{\mathbf{d}}(-1|t)^T\end{bmatrix}^T$.
In fact, constraints associated with the system dynamic are enforced through the open loop simulation over the whole estimation horizon, which is carried out with a time step $T_s = 0.1\ s$. Indeed, we observed that a lower simulation step was not providing much higher accuracy compared to the additional computational cost needed.
Given the above considerations, the problem can thus be written in the form:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x}) \tag{3.14}$$

Being $f(\mathbf{x})$ a strictly convex function and $\mathbb{R}^n$ an open set, we can use line search algorithms that guarantees convergence to a local minimizer independently of the choice of the initial guess of $\mathbf{x}_0$. The quadratic structure of the cost function, already discussed in 3.2, allowed to prefer the use of the Gauss-Newton approach to define effectively the search direction.

3.3.1 Unconstrained Optimization Results

Our unconstrained program 3.14 is characterized by some parameters to be tuned, namely the weighting matrices in the cost function 3.12 and the window length M.

Tuning the Weighting Matrices

To effectively tune the matrices, we initially solved the target tracking problem using an EKF. Through a trial and error approach, we found that the performances were acceptable with the following values 3.15. To ensure a meaningful comparison of different estimation methods, we kept these matrices unchanged throughout the entire project.

$$\begin{aligned} P(0) &= \text{diag}(10^3, 10^3, 10^3, 10^3, 10^3) \\ Q &= \text{diag}(10^{-1}, 10^{-1}) \\ R_{radar} &= \text{diag}(30, 20, 10, 90) \\ R_{camera} &= \text{diag}(150, 10, 60, 20) \end{aligned} \tag{3.15}$$

Optimization Routine

We solved this unconstrained NLP through an iterative algorithm, employing back-tracking line search with Armijo Condition. The search direction at each iteration of the optimization algorithm, $\mathbf{p}^k \in \mathbb{R}^n$, has been computed through a Gauss-Newton method, while all the gradients have been computed with the *Forward Differences* (FD) approximation to reduce computational load. In Table 3.1 and Table 3.2, we report the main parameters controlling the specified optimization procedure.

TOL_∇	$TOL_\mathbf{x}$	TOL_f	N_{max}
10^{-8}	10^{-12}	10^{-12}	10^3

Table 3.1: General Optimisation Parameters

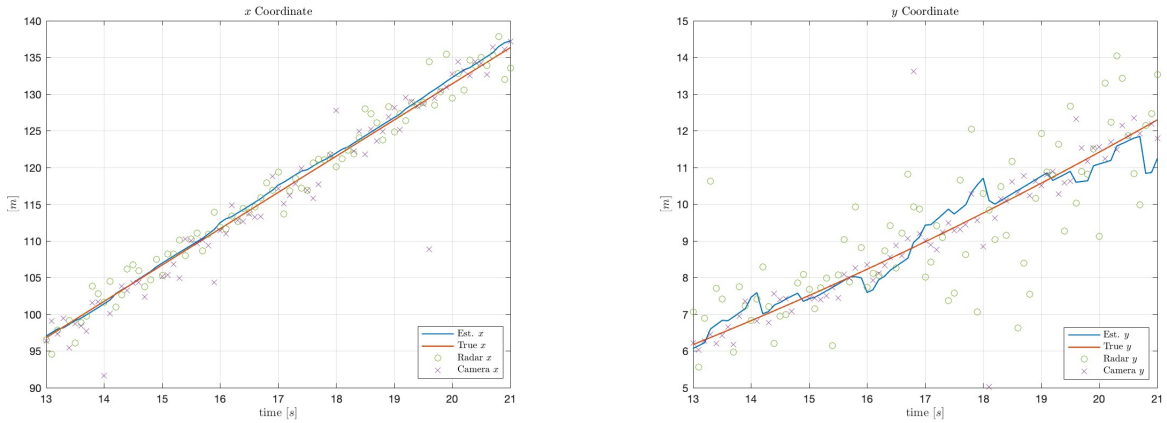
β	c	$\bar{t}$	$N_{LS_{max}}$
$2 \cdot 10^{-2}$	10^{-2}	1	15

Table 3.2: Line Search Parameters

After conducting several trial-and-error experiments, we determined that small values for both c and β led to low values of the cost function at the minimizer $f(\mathbf{x}^*)$ and resulted in satisfactory computational time.
These parameter choices, along with the low values of the directional derivative $||\nabla_\mathbf{x} f(\mathbf{x})^T \mathbf{p}^k||$, discouraged the solver from performing extensive line search routines. Consequently, we achieved an algorithm capable of reaching a local minimizer with fewer iterations. It's worth mentioning that, the solver typically satisfies the termination conditions based on either the relative step size, $TOL_\mathbf{x}$, or on the magnitude of the directional derivative, TOL_∇.

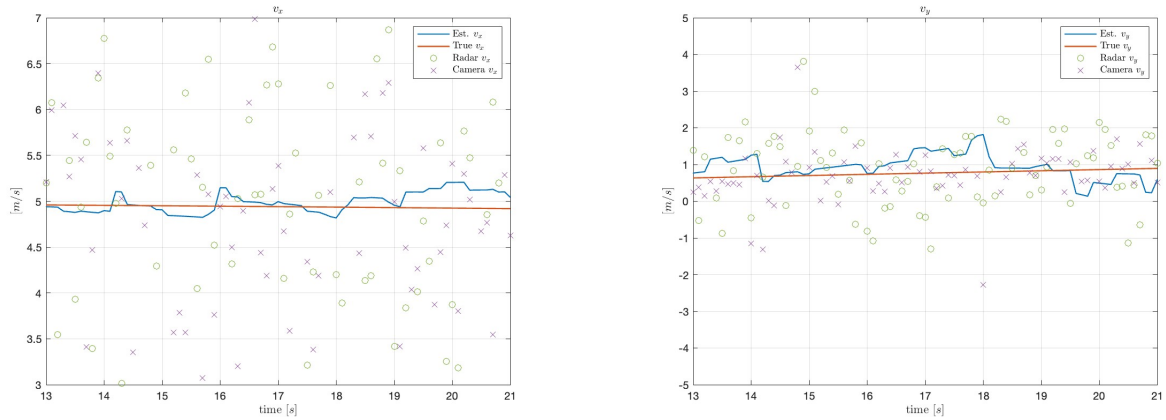
Estimation Performances

We observed that an $M = 2$, corresponding to an horizon window of $T = 0.2\ s$, provided the best balance between computational efficiency and accuracy, aligning with our project goals.



(a) x Coordinate. (b) y Coordinate.

Figure 3.2: Unconstrained MHE of x and y Relative Coordinates



(a) v_x. (b) v_y.

Figure 3.3: Unconstrained MHE of v_x and v_y Relative Speeds

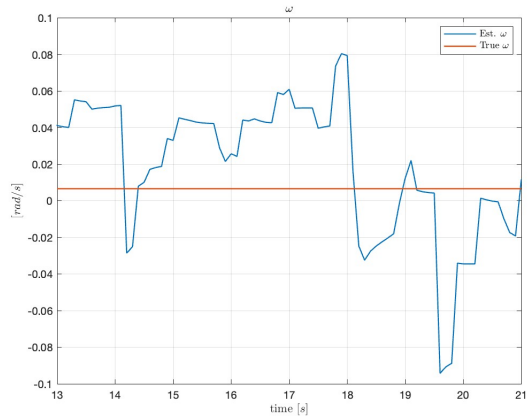
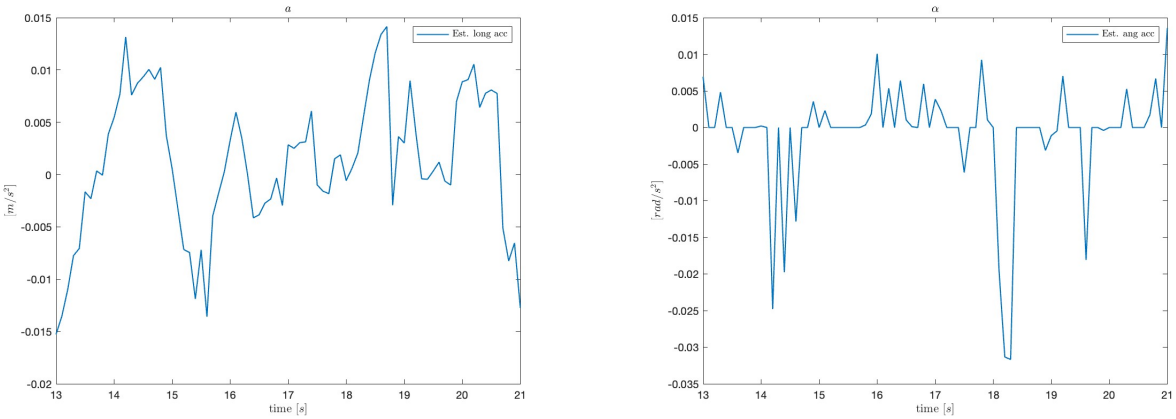


Figure 3.4: Unconstrained MHE of ω Relative Angular Speed



(a) Longitudinal Acceleration a. (b) Angular Acceleration α.

Figure 3.5: Unconstrained MHE of a and α

$RMSE_x$ (m)	$RMSE_y$ (m)	$RMSE_{v_X}$ (m/s)	$RMSE_{v_y}$ (m/s)	$RMSE_\omega$ (rad/s)
2.31	0.61	0.42	0.57	0.13

Table 3.3: Unconstrained RMSE for each estimated state variable letting $M = 2$

Our filter effectively rejects outliers and the most corrupted measurements. Additionally, the process noises are bounded at an order of magnitude of 10^{-2}, which is consistent with a constant uniform circular motion. Table 3.3 demonstrates the best estimation performance in state variables x, y, and v_x. The lower performance in v_y may stem from the higher uncertainty associated with the corresponding measurements provided by the sensors 2.4.
The average computational time, over the whole simulation, needed to provide the estimate of the state at current time $\hat{\mathbf{z}}(0|t)$ is within constraints, being it equal to 0.014 s.

Sensitivity to the Window Length M

As we learned from the literature [3], we expected an inversely proportional relationship between increasing values of M and the values of the $RMSE$. We found, experimentally that this trend held while solving the unconstrained problem with a constant arrival cost $\bar{P} = 10^3 \cdot \text{diag}\,(1, 1, 1, 1, 1)$. However, when introducing the EKF update to the terminal cost (as described in Section 3.2.4), the trend in Figure 3.6 emerged. We can explain this result assuming that an adaptive arrival cost represent a richer statistics of the history, providing at each new iteration of the filter a better estimation of the window initial state $\hat{\mathbf{z}}(-M|t)$.

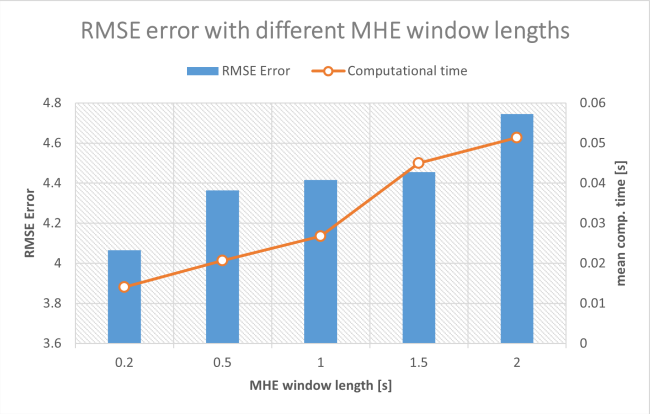


Figure 3.6: Sensitivity of the Unconstrained MHE to the Window Length M.

Remark: Each bin representing the $RMSE$ *Error* in Figure 3.6 is given by the sum of the $RMSE$ on each state variable.

3.3.2 Analytical Differentiation of The Cost Function $f(\mathbf{x})$

In an attempt to further reduce the computational time of our filtering algorithm we computed $\nabla_{\mathbf{x}} F(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^{n \times N}$ via Analytical Differentiation (AD). The primary drawback of this approach lies in its implementation complexity. It necessitates the complete coding of analytical derivatives, involving an iterative loop to account for the dependence on the variable parameter M, which affects both n and N. Further details on the implementation can be found in the code. Figure 3.7 illustrates the effectiveness of our implementation. For instance, with $M = 20$, corresponding to $T = 2$ s, the Automatic Differentiation (AD) approach results in a nearly 40% reduction in time.

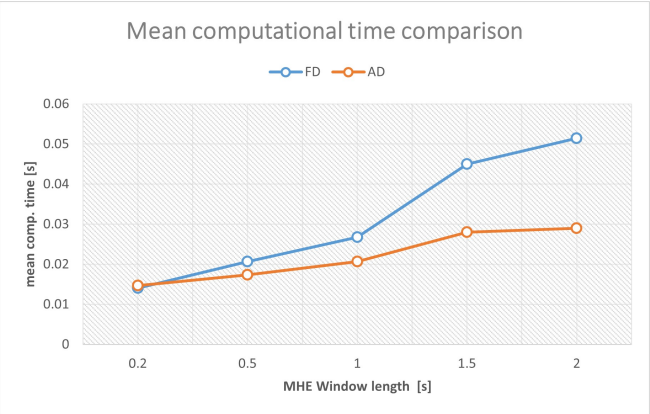


Figure 3.7: Mean Computational Time Comparison between FD and AD Methods.

3.4 Constrained Problem: Single Shooting Formulation

After fine tuning the unconstrained filter, we introduced domain knowledge in order to try to achieve an higher accuracy and being even more robust with respect to measurements corruption. In a single shooting formulation of the constrained problem the vector of decision variables $\mathbf{x}$ stays the same as in Section 3.3.

3.4.1 Non Linear Inequality Constraints

Knowing the road geometry, already described in Chapter 2, we formulated the non linear inequality constraints in 3.16.

$$r_{in} \leq \sqrt{X_{target}^2 + Y_{target}^2} \leq r_{out} \tag{3.16}$$

3.16 can be equivalently written as in 3.17 to conform with the notation $h(\mathbf{x}) \geq 0$.

$$\begin{cases} -\sqrt{X_{target}^2 + Y_{target}^2} + r_{out} \geq 0 \\ \sqrt{X_{target}^2 + Y_{target}^2} - r_{in} \geq 0 \end{cases} \tag{3.17}$$

Here, r_{in} and r_{out} are built from the radius of the road centerline and considering the lane width as in 2.2.1.

$$\begin{cases} r_{out} = 760 + 3.6\ m \\ r_{in} = 760 - 3.6\ m \end{cases} \tag{3.18}$$

Assuming to have information concerning the position of the ego vehicle in the world frame, whose coordinates are denoted as $X^{ego}, Y^{ego}, \theta^{ego}$ we can rewrite these constraints so that they depend explicitly on the optimization variables, more specifically on the position of the target in the ego vehicle reference frame, namely $(x(t), y(t))^T$ components.
It's straightforward to derive the coordinate transformation from local to global coordinate frame, being all reference frames right handed:

$$\begin{cases} X^{target}(t) = X^{ego}(t) + x(t)\ cos(\theta^{ego}(t)) - y(t)\ sin(\theta_w^{ego}(t)) \\ Y^{target}(t) = Y^{ego}(t) + y(t)\ sin(\theta^{ego}(t)) + y(t)\ cos(\theta^{ego}(t)) \end{cases} \tag{3.19}$$

After this coordinate transformation it's possible to substitute 3.19 into 3.16 obtaining the set of non linear inequality constraints in standard form by enforcing these constraints over the whole estimation horizon. This should prevented the simulation of the model to end up in positions that are out of the track bounds.

3.4.2 Constrained Optimization in Standard Form

We are now in the position of formulating the problem that must be solved at each iteration of the filtering algorithm.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h(\mathbf{x}) \geq 0; \end{aligned} \tag{3.20}$$

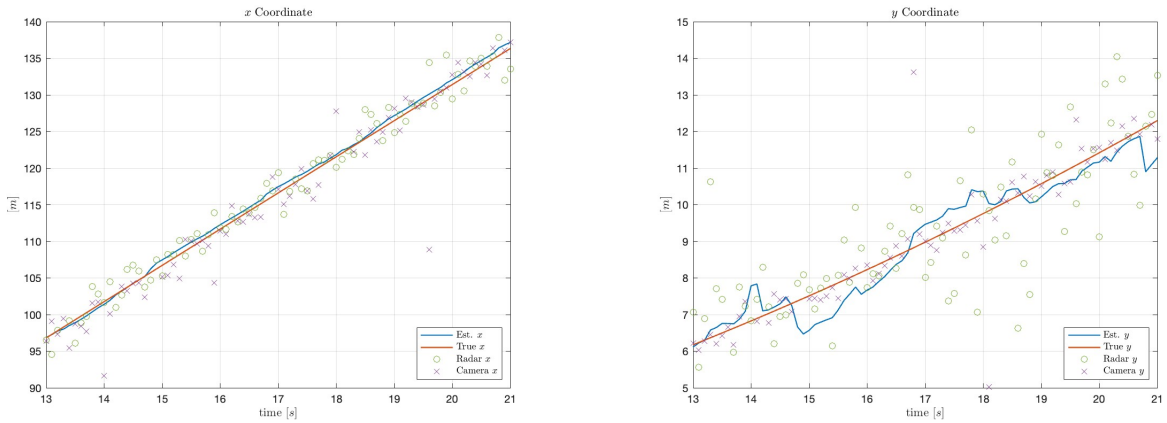
Where the function $h : \mathbb{R}^n \to \mathbb{R}^{2(M+1)}$. Being the problem a general continuous non linear program, we decided to solve it employing an SQP (Sequential Quadratic Programming) iterative solver with an interior point method. We still leverage a constrained Gauss-Newton approach to compute the search direction $\mathbf{p}^k$, given its obvious advantages already discussed. The general parameters of both the optimization procedure and the line search on the merit function are the same as in Table 3.1 and Table 3.2. Additionally, we set a tolerance on constraint satisfaction equal to 10^{-6}.

3.4.3 Results

The following results deriving from the above mentioned constrained optimization problem are obtained using a cost function with an EKF update on the terminal cost, as all the other experiments in the remaining of the report.

Estimation Performances

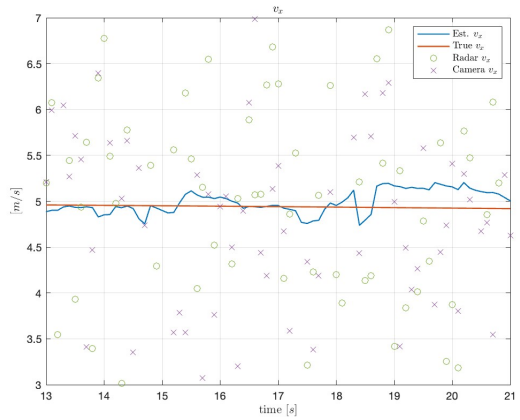
For the constrained problem, we observed that an $M = 5$, thus an horizon window of $T = 0.5\ s$ held the the best performances.



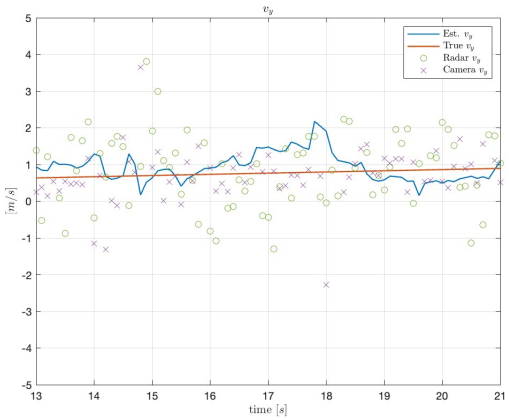
(a) x Coordinate.

(b) y Coordinate.

Figure 3.8: Constrained MHE of x and y Relative Coordinates



(a) v_x.



(b) v_y.

Figure 3.9: Constrained MHE of v_x and v_y Relative Coordinates

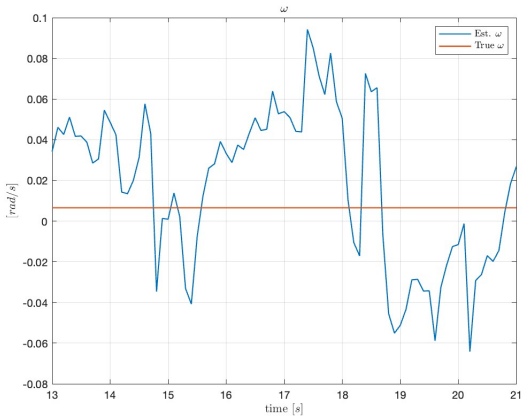
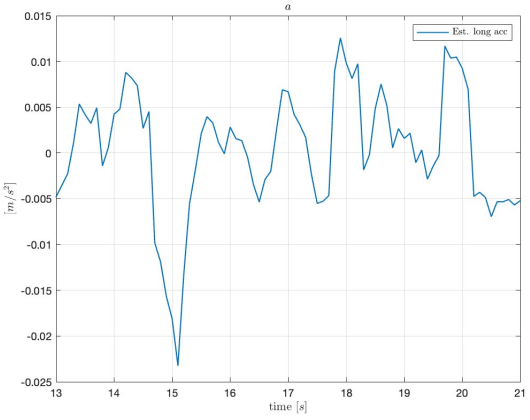
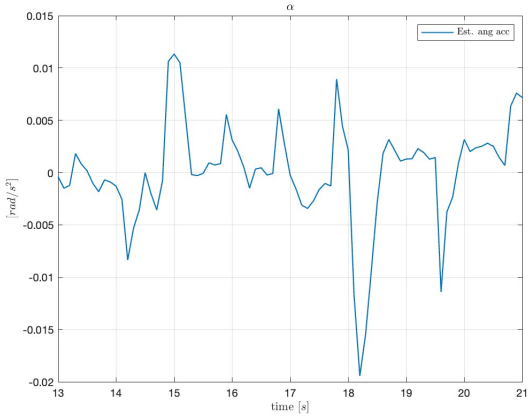


Figure 3.10: Constrained MHE of ω Relative Angular Acceleration



(a) Longitudinal Acceleration a.



(b) Angular Acceleration α.

Figure 3.11: Constrained MHE of a and α

In the following table are presented the RMSE regarding each state estimate with a window's length $M = 5$.

$RMSE_x$ (m)	$RMSE_y$ (m)	$RMSE_{v_X}$ (m/s)	$RMSE_{v_X}$ (m/s)	$RMSE_\omega$ (rad)
2.40	0.6240	0.3886	0.6138	0.1750

Table 3.4: Constrained RMSE for Each Estimated State Variable Letting $M = 5$

Again, from Table 3.7 we can see how x, y, v_x are better estimated than v_y and ω, as in Section 3.3.1. Upon close examination of the solver output, it is evident that, in general, nonlinear constraints are typically satisfied, and the minimizer does not lie on the boundary of the set that these constraints define. The average computational time of this solution was recorded to be 0.0296 s.

3.4.4 Constrained VS Unconstrained

To effectively test the importance of including road constraints in our estimation problem, we tried to isolate their effect on the estimated variables. To this aim we compared the results from an unconstrained MHE and a constrained formulation both employing a constant terminal cost $\bar{P} = 10^3$. Intuitively, we expected that for a fixed widow length M the constrained problem would provide much lower values of the $RMSE$. To verify our assumptions we checked the worst possible conditions, by setting $M = 2$. Unfortunately, the performance gains were less significant than predicted. Hence, because of its lower computational burden, we concluded that it is reasonable to prefer an unconstrained formulation. Results of the comparison are reported in Table 3.5.

	$RMSE_x$ (m)	$RMSE_y$ (m)	$RMSE_{v_x}$ (m/s)	$RMSE_{v_y}$ (m/s)	$RMSE_\omega$ (rad)	time (s)
Unc	2,70	1.0616	1.6518	2.3983	3.7893	0,0298
Con	2,6577	0,8040	1.6511	2.3735	3.8016	0,0732

Table 3.5: RMSE for each estimated state variable letting $M = 2$, Constrained vs Unconstrained

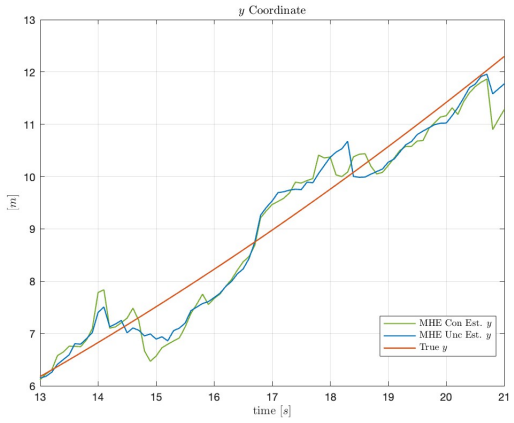


Figure 3.12: Comparison of Unconstrained and Constrained MHE on Estimation of y

3.4.5 Analytical Differentiation in The Constrained Problem

Being the constrained problem inherently computationally heavier, we tried to implement also for this formulation an analytical derivation of the gradients $\nabla_{\mathbf{x}} F(\mathbf{x})$, as previously stated in 3.3.2. We expected a similar trend to the one seen in the unconstrained problem, with a lower computational time with increasing values of M, with respect to the FD method.
The observed result did not satisfied our expectations, as the computational load was basically the same as the FD differentiation. This might be caused by the need to compute also the analytical derivatives of the constraints, $\nabla_{\mathbf{x}} h(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^{n \times 2(M+1)}$. In fact, this gradient needs to be computed through an iterative procedure whose length depend on M.

3.4.6 Sensitivity To Gradient Tolerance

Implementing analytical differentiation offered the possibility to gain some deeper understanding of the constrained optimization routine. More precisely we observed that changing the termination conditions to some less restrictive values, offered a lower computational time while providing the same accuracy in terms of RMSE. In Table 3.6 we can see how the Constrained Optimization is completed in less time with by just changing $TOL_{\nabla} = 10^{-3}$.

$RMSE_x$ (m)	$RMSE_y$ (m)	$RMSE_{v_X}$ (m/s)	$RMSE_{v_X}$ (m/s)	$RMSE_{\omega}$ (rad)	time (s)
2.4048	0.6240	0.3886	0.6139	0.1754	0.0177

Table 3.6: Constrained RMSE for Each Estimated State Variable Letting $M = 5$, $TOL_{\nabla} = 10^{-3}$

3.5 Constrained Problem: Multiple Shooting Formulation

As a final step of our project we decided to implement the Multiple Shooting Formulation of the MHE problem in 3.10. Thus we considered the set of optimization variables as in Section 3.2.2. Then we enforced the system dynamic equations by formulating the set of $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^{n_z M}$ equality constraints of the form 3.21.

$$\hat{\mathbf{z}}(i+1|t) - (f_{\mathbf{z}}(\hat{\mathbf{z}}(i|t)) + G(\hat{\mathbf{z}}(i|t))\mathbf{d}(i|t))) = 0 \quad i = -M, \dots, -1. \tag{3.21}$$

Thus, in standard form the MS formulation of our problem becomes the one in 3.22.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g(\mathbf{x}) = 0. \\ & h(\mathbf{x}) \geq 0 \end{aligned} \tag{3.22}$$

Here $h(\mathbf{x})$ represent the same non linear inequality constraints as explained in Section 3.4.1.

3.5.1 linear inequality constraints

We would have expected that the performances of the MS formulation would have provided a lower computational time with an increasing size of the window length, as reported in the literature [5]. However, solving 3.22 with the same optimization parameters detailed in Table 3.1, and Table 3.2 and using a general SQP solver with constrained Gauss Newton method, this trend was not verified. Moreover, the estimation performances were much deteriorated as we observed a divergence in the estimation of the course of the input disturbances.
We recall that the input disturbances $\hat{\mathbf{d}}$ are signals modeled as WGN in 2.3.2. Changing the components of matrix Q in the cost function, we are able to bound the values taken on by the vectors $\mathbf{d}$ across the horizon. In the MS formulation this seemed not enough. To restore satisfactory accuracy of the estimation, we decided to enforce in the problem some linear inequality constraints in the form of $C\ \mathbf{x} \geq \mathbf{a}$. This procedure effectively prevents the disturbances to diverge to huge values even by considering higher valued component in the matrix Q. In MS formulation of the constrained estimation problem we wrote these constraints as:

$$\begin{bmatrix} a_{min} \\ \alpha_{min} \end{bmatrix} \leq \begin{bmatrix} d_1(t) \\ d_2(t) \end{bmatrix} \leq \begin{bmatrix} a_{max} \\ \alpha_{max} \end{bmatrix} \tag{3.23}$$

Where $a_{max/min}$ and $\alpha_{max/min}$ represent the maximum and minimum longitudinal and angular accelerations respectively. Their values are $a_{max/min} = \pm 0.1\ m/s^2$ and $\alpha_{max/min} = \pm 0.1\ rad/s^2$. These constraints are assumed to hold $\forall t$ in the estimation horizon.
In the standard form, by extracting the course of the input disturbances from the set of optimization variables $\mathbf{x}$, through a matrix $C \in \mathbb{R}^{2n_d M \times (nz(M+1)+n_d M)}$ and expressing the vector $\mathbf{a} \in \mathbb{R}^{4 \times M}$ we obtain:

$$\begin{bmatrix} -d_1(-M|t) \\ \vdots \\ -d_1(-1|t) \\ -d_2(-M|t) \\ \vdots \\ -d_2(-1|t) \\ d_1(-M|t) \\ \vdots \\ d_1(-1|t) \\ d_2(-M|t) \\ \vdots \\ d_2(-1|t) \end{bmatrix} \geq \begin{bmatrix} -a_{max} \\ \vdots \\ -a_{max} \\ -\alpha_{max} \\ \vdots \\ -\alpha_{max} \\ a_{min} \\ \vdots \\ a_{min} \\ \alpha_{min} \\ \vdots \\ \alpha_{min} \end{bmatrix} \tag{3.24}$$

Final MS Standard Formulation

Finally we are able to provide the standard formulation of the MS problem that achieved the best performances.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & C\mathbf{x} \geq \mathbf{a} \\ & g(\mathbf{x}) = 0. \\ & h(\mathbf{x}) \geq 0 \end{aligned} \tag{3.25}$$

3.5.2 Results

With a careful analysis of the output of the solver, we observed that at each iteration of the filter the cost function was experiencing an increase, usually after the 10^{th} iteration of the optimization algorithm. We never experienced this trend in previous experiments, thus we decided to modify the terminal conditions of the optimization algorithm by bounding the maximum number of optimization iterations to $N_{max} = 10$.
After solving these issues, we still encountered an undesired trend: as M increased, the computational cost of the MS formulation was still higher than the SS. This might be due to the following reasons:

- The higher number of optimization variables.
- The computational demand to satisfy $g(\mathbf{x})$ might be higher than the one needed to simulate a non linear system with a time-step of just $T_s = 0.1\ s$.

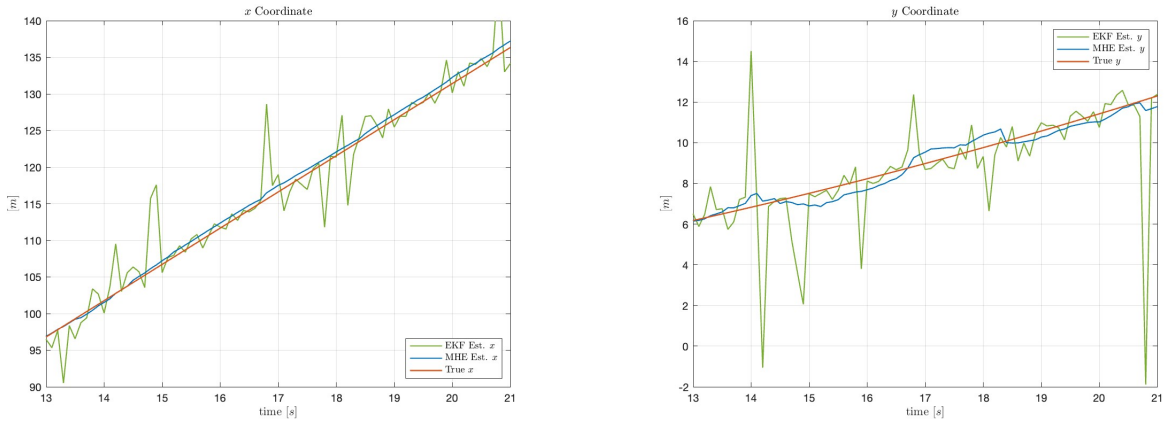
As we can see from the results in Table 3.7 the MS approach allow a slightly reduced computational time with only with a very small horizon, i.e $M = 2$ or lower.

	$RMSE_x\ (m)$	$RMSE_y\ (m)$	$RMSE_{v_X}\ (m/s)$	$RMSE_{v_X}\ (m/s)$	$RMSE_\theta\ (rad)$	time (s)
MS	2,4103	0,8265	0,5549	1,0190	0,5585	0,0185
SS	2,4125	0,8491	0,5654	1,0462	0,5659	0,0213

Table 3.7: Multiple Shooting RMSE for each estimated state variable letting $M = 2$, SS vs MS

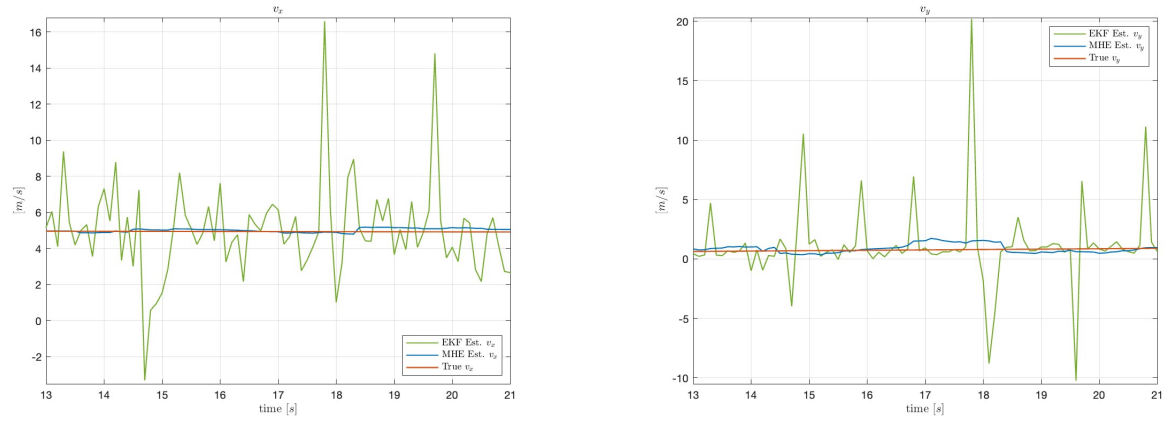
4. EKF Comparison and Final Considerations

In this chapter we provide an explicit comparison between an EKF, tuned as in 3.3.1, and our best MHE algorithm, namely the unconstrained MHE with $M = 2$.



(a) x Coordinate. (b) y Coordinate.

Figure 4.1: Unconstrained MHE estimation of x and y relative coordinates



(a) v_x . (b) v_y.

Figure 4.2: Unconstrained MHE estimation of v_x and v_y relative coordinates

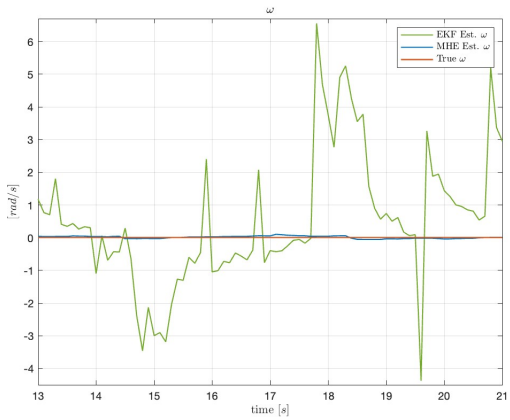


Figure 4.3: Unconstrained MHE estimation of θ relative yaw rate

	$RMSE_x$ (m)	$RMSE_y$ (m)	$RMSE_{v_X}$ (m/s)	$RMSE_{v_X}$ (m/s)	$RMSE_\theta$ (rad)	time (s)
MHE	2.7692	1.8286	2.2880	3.2389	2.2041	1.4×10^{-4}
EKF	2.31	0.61	0.42	0.57	0.13	1.4×10^{-2}

Table 4.1: RMSE for each estimated state variable letting $M = 2$, SS vs MS

Looking at the pictures and at the table 4.1 we can conclude that we achieved our goal on the designing a filtering algorithm able to satisfy our timing constraints of a maximum computational time of 0.1 s and achieving a significant noise reduction with respect to a conventional estimation approach. This last result allows to the implementation of MHE designed in a real-time context.

List of Figures

List of Tables

Bibliography

[1] *The discrete-time Kalman filter*, chapter 5, pages 121–148. John Wiley Sons, Ltd, 2006.

[2] J. B. R. Eric L. Haseltine. A critical evaluation of extended kalman filtering and moving horizon estimation. *Texas-Wisconsin Modeling and Control Consortium*, 03 2003.

[3] L. G. G. Sánchez n, M. Murillo. Adaptive arrival cost update for improving moving horizon estimation performance. *Research Institute for Signals, Systems and Computational Intelligence*, 2017.

[4] F. Girrbach. *Moving horizon estimation for inertial motion tracking : algorithms and industrial applications*. PhD thesis, 06 2021.

[5] J. L. H. D. H. C. F. X. Hanghang Liu, Ping Wang. Real-time longitudinal and lateral state estimation of preceding vehicle based on moving horizon estimation. *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, 09 2021.

[6] Y.-S. Kim and K.-S. Hong. An imm algorithm for tracking maneuvering vehicles in an adaptive cruise control environment. *International Journal of Control, Automation and Systems*, 2, 09 2004.

[7] MATLAB. Automated driving toolbox, 2023.

[8] MATLAB. Model radar sensor detections, 2023.

[9] MATLAB. Model vision sensor detection, 2023.

[10] A. Rachman. 3d-lidar multi object tracking for autonomous driving: Multi-target detection and tracking under urban road uncertainties. 2017.

[11] M. Roth, G. Hendeby, and F. Gustafsson. Ekf/ukf maneuvering target tracking using coordinated turn models with polar/cartesian velocity. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8, 2014.

[12] Waymo. Waymo open datasets, 2023.

Appendix: Acronyms

KF: Kalman Filter

EKF : Extended Kalman Filter

WGN: White Gaussian Noise

CTRV: Constant Turn-rate Velocity

RMSE : Root Mean Squared Error

MS: Multiple Shooting

SS: Single Shooting

SQP : Sequential Quadratic Programming

NLP: Nonlinear Program