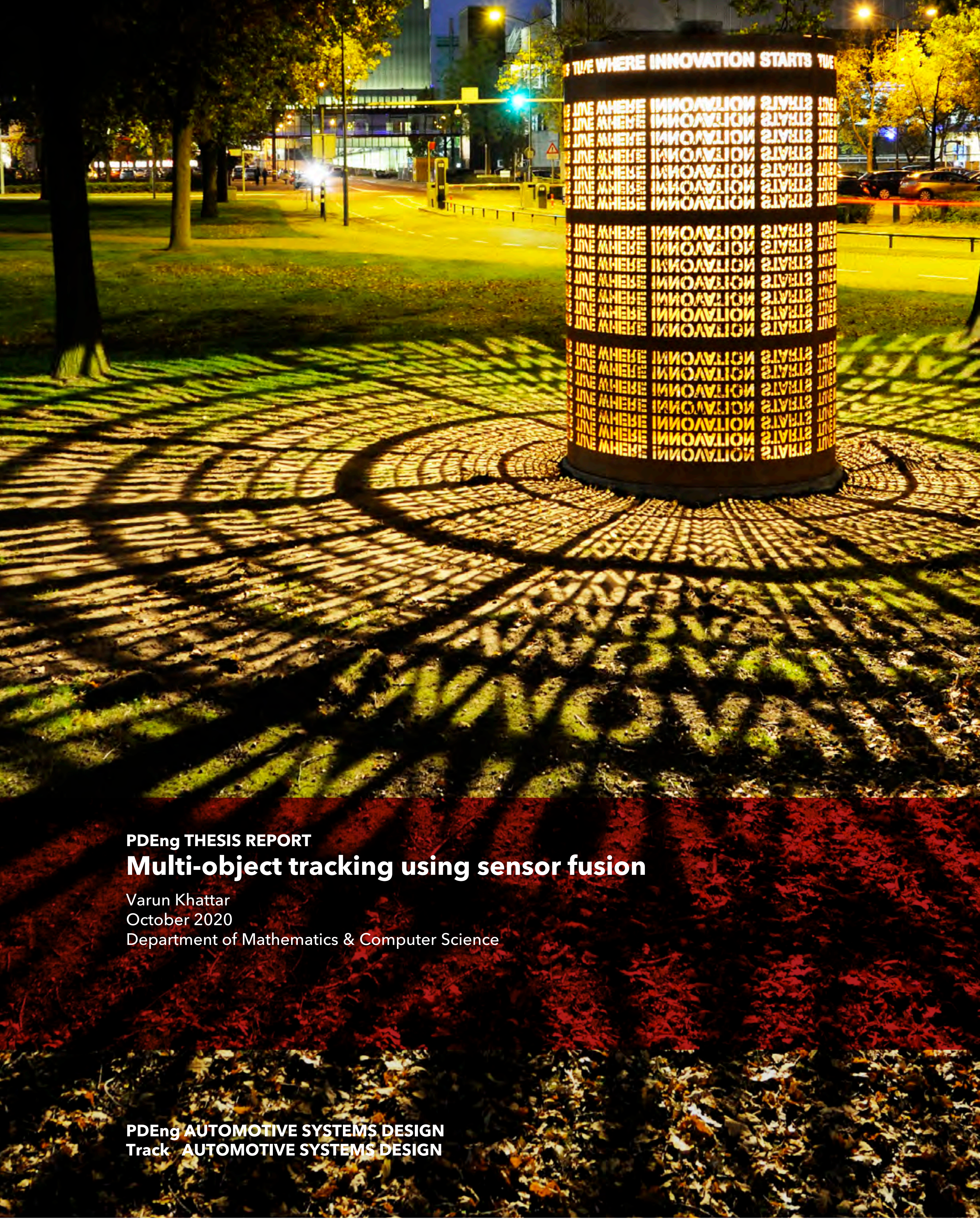
Multi-object tracking using sensor fusion



PDEng THESIS REPORT

Multi-object tracking using sensor fusion

Varun Khattar

October 2020
Department of Mathematics & Computer Science

PDEng AUTOMOTIVE SYSTEMS DESIGN
Track AUTOMOTIVE SYSTEMS DESIGN

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Multi-object tracking using sensor fusion

PDEng graduation project report

Varun Khattar

October 2020

Eindhoven University of Technology
Stan Ackermans Institute - Automotive Systems Design

PDEng Report; 2020/078

Confidentiality Status: Open access

Partners	i-CAVE consortium	Eindhoven University of Technology
Steering Group	Tom van der Sande Mohsen Alirezaei Peter Heuberger	
Date	October 2020	

Date	October 2020
Contact address	Eindhoven University of Technology Department of Mathematics and Computer Science Automotive Systems Design MF 5.073 P.O. Box 513 NL-5600 MB Eindhoven, The Netherlands +31 402743908
Published by	Eindhoven University of Technology
PDEng Report	2020/078

Abstract

Cooperative platooning involves a group of vehicles following a leader vehicle autonomously. To follow the lead vehicle directly in front, the ego vehicle first must track all road users within the FOV (Field-of-View) of its sensors such as radars and cameras, and then select the target to be followed from all tracked road users. These tasks have been the focus of this PDEng project as part of the i-CAVE consortium. The positions, velocities and accelerations of all road users in the FOV of radars and cameras have been tracked using measurements from radars, cameras, GNSS (Global Navigation Satellite System) and the IMU (Inertial Measurement Unit). From these tracked road users, the MIO (Most Important Object) i.e. the vehicle to be followed has been selected, and its yaw angle and yaw rate have also been tracked to improve its motion prediction. The methods have been first verified in MATLAB simulations and then also validated in MATLAB through reconstruction of tracks from data measured in the real world.

Keywords	Target tracking, Sensor fusion, Radar, Camera, IMU, GNSS, Unscented Kalman Filter
Preferred reference	Multi-object tracking using Sensor Fusion, Eindhoven University of Technology, PDEng Report October 2020.
Partnership	This project was supported by Eindhoven University of Technology and **i-CAVE**

Foreword

Varun has been working in the i-CAVE project, which is a large NWO (Netherlands Organization for Scientific Research) project, focused on automated and cooperative vehicles. One of the key challenges in achieving successful vehicle automation is environmental perception. In practice this means that a vehicle is equipped with various sensors such as RADAR, camera and LIDAR. Since all these sensors have pros and cons, fusing them is important. In addition, it creates a coherent image of the surroundings. The main objective of Varun's research was to develop a system that tracks objects around the vehicle and predicts their motion. In addition to that, Varun focused on the vehicle that is right in front of the Twizy, which is most often considered as a lead vehicle. To follow this lead vehicle with greater accuracy, Varun derived a method that uses the past locations of the lead vehicle to create an estimate for the yaw-rate and heading angle.

Besides the theoretical work, Varun also worked on the implementation of his algorithms on a Renault Twizy test vehicle. For that, he worked with an NXP Bluebox, several radars and the camera detections provided by fellow PDEng researcher Ashton Menezes. He showed that it is possible to track a lead vehicle while moving through real-life traffic.

With the work of Varun, another step has been set to achieve fully automated and cooperative driving vehicles. Together with the work of fellow PDEng researcher Ashton Menezes, the environmental perception of the Renault Twizy has taken another step in the right direction.

Tom van der Sande
30-09-2020

Preface

This report is written for the graduation project done as part of the Professional Doctorate in Engineering (PDEng) program in Automotive Systems Design (ASD) at the Eindhoven University of Technology (TU/e). The duration of the program is 2 years. The first year involves homologation courses and module projects done in collaboration with the industrial partners of TU/e. The module projects are done as group projects with PDEng trainees from multiple PDEng programs i.e. Automotive Systems Design (ASD), Mechatronic Systems Design (MSD) and Software Technology (ST). The second year is reserved for the graduation project that is also done in collaboration with industrial partners but is done individually.

The goal of the PDEng project described in this report was to track all road users in the FOV (Field-of-View) of the sensors of an autonomous vehicle through the fusion of measurements from sensors fitted on the autonomous vehicle. The reason for tracking all road users is that the lead vehicle to be followed has to be selected as the primary target so that it can be followed by the ego vehicle autonomously. This report describes the principles used for developing the tracking system and the methods used for verification and validation of the developed tracking system.

Varun Khattar
12-10-2020

Acknowledgements

I would like to start by thanking Tom van der Sande and Peter Heuberger for giving me the opportunity to work on this project. I also want to thank Mohsen Alirezaei for joining the Project Steering Group (PSG). The guidance, feedback, and support of the PSG has been extremely helpful in achieving the result of the project.

I want to thank Han Raaijmakers, Andrii Buriachevskyi and Olav Bruggeman for their help with the NXP Bluebox and the NXP Cocoon radar.

I want to thank Robbin van Hoek, Wouter Scholte, Wouter Schinkel and Frans Hoogeboom for their help with the Simulink models and the hardware on the experimental vehicles.

I want to thank Wietse Loor and Gerard van Hout for making the electrical connections, and the mountings for the radars and the NXP Bluebox respectively.

I want to thank Recep Firat Tigrek for sharing his knowledge of the radar.

I want to thank Narsimlu Kemsaram and Anwesan Das for their help with the camera hardware.

I want to thank Aditya Kamath and Ashton Menezes for their help in using the Linux operating system.

Finally I want to thank my wife Tarika Chopra and my family, without whose support this would not have been possible.

Varun Khattar
22-09-2020

Executive Summary

Autonomous and cooperative driving has been an area of research since the first half of the 20th century because it has the potential to increase safety since human error causes the majority of the road accidents worldwide, and has been shown to increase fuel efficiency since the combined aerodynamic drag forces on the platoon are reduced when the platoon vehicles follow the lead vehicle at a fixed time and distance gap. To achieve vehicle automation, the i-CAVE (Integrated Cooperative Automated Vehicles) project was started, which created a group of universities and companies in the Netherlands that is researching methods to reach level 5 autonomy as defined by SAE (Society of Automotive Engineers) in autonomous platooning. The PDEng project described in this report was done as part of i-CAVE.

A cooperative platoon is a group of vehicles moving autonomously in coordination with each other while following a lead vehicle. Each platoon vehicle must continuously track all road users in the FOV (Field-Of-View) of its sensors so that it can select the vehicle directly in front from the tracked road users and follow it. Target tracking is challenging because the number of targets is unknown and is changing continuously, measurement noise is present in sensors and due to the environment e.g. bad weather, the motion of the targets has to be modelled accurately, and the processing has to be done in real-time because the environment is continuously changing as the vehicle is moving most of the time and a large delay in processing can lead to an accident. Since real-time processing is required, the method used for target tracking should not be computationally expensive but still be reasonably accurate.

Keeping these challenges in mind, all road users in the FOV of the sensors fitted on an ego vehicle were tracked by fusing measurements from the radars, cameras, GNSS (Global Navigation Satellite System), and the IMU (Inertial Measurement Unit). From these tracked road users, the MIO (Most Important Object) i.e. the lead vehicle was identified based on the assumptions that it remains mostly in front of the ego vehicle and is one of the closest objects to the ego vehicle. The position, velocity and acceleration of the MIO were tracked using an Unscented Kalman Filter (UKF) and Global Nearest Neighbour (GNN) data association, and the yaw rate and yaw angle of the MIO were tracked using polynomial path tracing. The design of the tracker was verified by simulations and validated through reconstruction of tracks from measurements obtained by driving in a real-world scenario in MATLAB. The results show that GNN data association with the constant acceleration motion model based UKF and polynomial path tracing are able to track the position, velocity, acceleration, yaw angle and yaw rate of the lead vehicle with reasonable accuracy for a long time at one stretch in a real-world dense traffic situation.

Glossary

ASD	Automotive Systems Design
PDEng	Professional Doctorate in Engineering
PID	Project Initiation Document
PSG	Project Steering Group
TU/e	Eindhoven University of Technology
RMS	Root Mean Squared
CV	Constant Velocity
CA	Constant Acceleration
i-CAVE	Integrated Cooperative Automated Vehicles
FISST	Finite State Statistics
SAE	Society of Automotive Engineers
T2T	Track-to-Track
T2TFP	Track-to-Track fusion with Fused Prediction
FMCW	Frequency Modulated Continuous Wave
CFAR	Constant False Alarm Rate
DOF	Degree of Freedom
UT	Unscented Transform
NHTSA	National Highway Traffic Safety Administration
DARPA	Defence Advanced Research Projects Agency
DDT	Dynamic Driving Task
OEDR	Object Event Detection and Recognition
ODD	Operational Design Domain
ADAS	Advanced Driver Assistance Systems
ABS	Anti-lock Braking Systems
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
V2V	Vehicle to Vehicle
SLAM	Simultaneous Localization and Mapping
C-DMAT	Cooperative Dual Mode Automated Transport
NWO	Netherlands Organization for Scientific Research
RTT	Real Time Target
FOV	Field of View
MIO	Most Important Object
DBT	Detect before Track
TBD	Track before Detect
HMM	Hidden Markov Models

GM	Gaussian Mixture
GNN	Global Nearest Neighbour
MAP	Maximum a Posteriori
MHT	Multiple Hypothesis Tracking
PHD	Probability Hypothesis Tracking
GGIW	Gamma Gaussian Inverse Wishart
OSPDA	Order Statistics Probabilistic Data Association
JPDA	Joint Probabilistic Data Association
MCMCDA	Markov Chain Monte Carlo Data Association
EM	Expectation Maximization
PDF	Probability Density Function
KF	Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
PF	Particle Filter
SMC	Sequential Monte Carlo
RFS	Random Finite Set
GMSL	Gigabit Multimedia Serial Link
IC	Integrated Chip
RF	Radio Frequency
LHS	(Latin Hypercube Sampling)

List of symbols

X	Global X axis
Y	Global Y axis
Z	Global Z axis
x	Local x axis
y	Local y axis
z	Local z axis
S	Number of scans in MHT tracking
Z_k	Measurement vector at the k^{th} time step
Z_r	Measurement vector from the radar
Z_c	Measurement vector from the camera
$\hat{Z}_k$	Predicted measurement at the k^{th} time step
X_k	State vector at the k^{th} time step
$\hat{X}_k$	Predicted measurement at the k^{th} time step
T_s	Sampling time
$f(\cdot)$	State transition function
$h(\cdot)$	Measurement function
x_k	Position in the x direction at the k^{th} time step
y_k	Position in the y direction at the k^{th} time step
z_k	Position in the z direction at the k^{th} time step
$\sigma^2_{x,k}$	Variance in the position in the x direction at the k^{th} time step
$\sigma^2_{y,k}$	Variance in the position in the y direction at the k^{th} time step
$\sigma^2_{z,k}$	Variance in the position in the z direction at the k^{th} time step
$\dot{x}_k$	Velocity in the x direction at the k^{th} time step
$\dot{y}_k$	Velocity in the y direction at the k^{th} time step
$\dot{z}_k$	Velocity in the z direction at the k^{th} time step
$\sigma^2_{\dot{x},k}$	Variance in the velocity in the x direction at the k^{th} time step
$\sigma^2_{\dot{y},k}$	Variance in the velocity in the y direction at the k^{th} time step
$\sigma^2_{\dot{z},k}$	Variance in the velocity in the z direction at the k^{th} time step
$\ddot{x}_k$	Acceleration in the x direction at the k^{th} time step
$\ddot{y}_k$	Acceleration of the y coordinate at the k^{th} time step
$\ddot{z}_k$	Acceleration of the z coordinate at the k^{th} time step
$\sigma^2_{\ddot{x},k}$	Variance in the acceleration in the x direction at the k^{th} time step
$\sigma^2_{\ddot{y},k}$	Variance in the acceleration in the y direction at the k^{th} time step
$\sigma^2_{\ddot{z},k}$	Variance in the acceleration in the z direction at the k^{th} time step
$q_{x,k}$	Zero mean white noise in the x direction at the k^{th} time step
$q_{y,k}$	Zero mean white noise in the y direction at the k^{th} time step

$q_{z,k}$	Zero mean white noise in the z direction at the k^{th} time step
$\mathcal{X}_{i,k}$	i^{th} Sigma point for the state vector at the k^{th} timestep
$\mathcal{Z}_{i,k}$	i^{th} Sigma point for the measurement vector at the k^{th} timestep
$\hat{\Sigma}_{X,k}$	Covariance matrix of the predicted state vector at the k^{th} time step
$\hat{\Sigma}_{Z,k}$	Covariance matrix of the predicted measurement vector at the k^{th} time step
$\hat{\Sigma}_{XZ,k}$	Cross-correlation covariance matrix of the predicted state vector and measurement vector at the k^{th} time step
$\mathcal{K}$	Kalman gain
$\mathbb{R}$	Set of real numbers
N	Dimension of the state vector
M	Dimension of the measurement vector
λ_N	Scaling parameter for the state vector
λ_M	Scaling parameter for the measurement vector
α	Parameter to decide spread of sigma points
κ	Secondary scaling parameter
β	Parameter to incorporate prior knowledge of the distribution
U_k	Model input at the k^{th} time step
W_k	Process noise at the k^{th} time step
V_k	Measurement noise at the k^{th} time step
Q_k	Process noise covariance matrix at the k^{th} time step
R_k	Measurement noise covariance matrix at the k^{th} time step
$\mathcal{W}_{i,k}$	i^{th} Sigma point for the process noise at the k^{th} timestep
$\mathcal{V}_{i,k}$	i^{th} Sigma point for the measurement noise at the k^{th} timestep
$w_{N,i}^{(m)}$	i^{th} weight for calculating mean of transformed sigma points of the state distribution
$w_{N,i}^{(c)}$	i^{th} weight for calculating covariance of transformed sigma points of the state distribution
r	Radial distance
$\dot{r}$	Radial speed
θ	Elevation angle
$\dot{\theta}$	rate of change of elevation angle
φ	Azimuth angle
$\dot{\varphi}$	Rate of change of azimuth angle
b	Baseline of stereo camera
ψ	Half-angle subtended by stereo camera at a point
Δf	Frequency shift between the received and transmitted signal
Δt	Time of flight of the radar signal
$Z_{res,k}$	Residual vector at the k^{th} time step
$d_{n,k}$	Normalized distance at the k^{th} time step
d_{th}	Clustering threshold
$p_1 - p_7$	Parameters used in LHS (Latin Hypercube Sampling)
v_{lat}	Lateral velocity
F_{lat}	Lateral force
v_{long}	Longitudinal velocity
F_{long}	Longitudinal force
v_{net}	Net velocity
β_{side}	Sideslip
ω_r	Yaw rate

α_r	Yaw acceleration
M_{rot}	Rotational moment about the Z axis
$\beta_{s,k}$	Relative sideslip at the k^{th} time step
$\dot{\beta}_s$	Relative sideslip rate at the k^{th} time step
$v_{x,k}$	Relative velocity in the x direction at the k^{th} time step
$v_{y,k}$	Relative velocity in the y direction at the k^{th} time step
$v_{z,k}$	Relative velocity in the z direction at the k^{th} time step
$a_{x,k}$	Relative acceleration in the x direction at the k^{th} time step
$a_{y,k}$	Relative acceleration in the y direction at the k^{th} time step
$a_{z,k}$	Relative acceleration in the z direction at the k^{th} time step
ϕ_k	Relative yaw angle at the k^{th} time step
$\dot{\phi}_k$	Relative yaw rate at the k^{th} time step

List of Tables

List of Figures

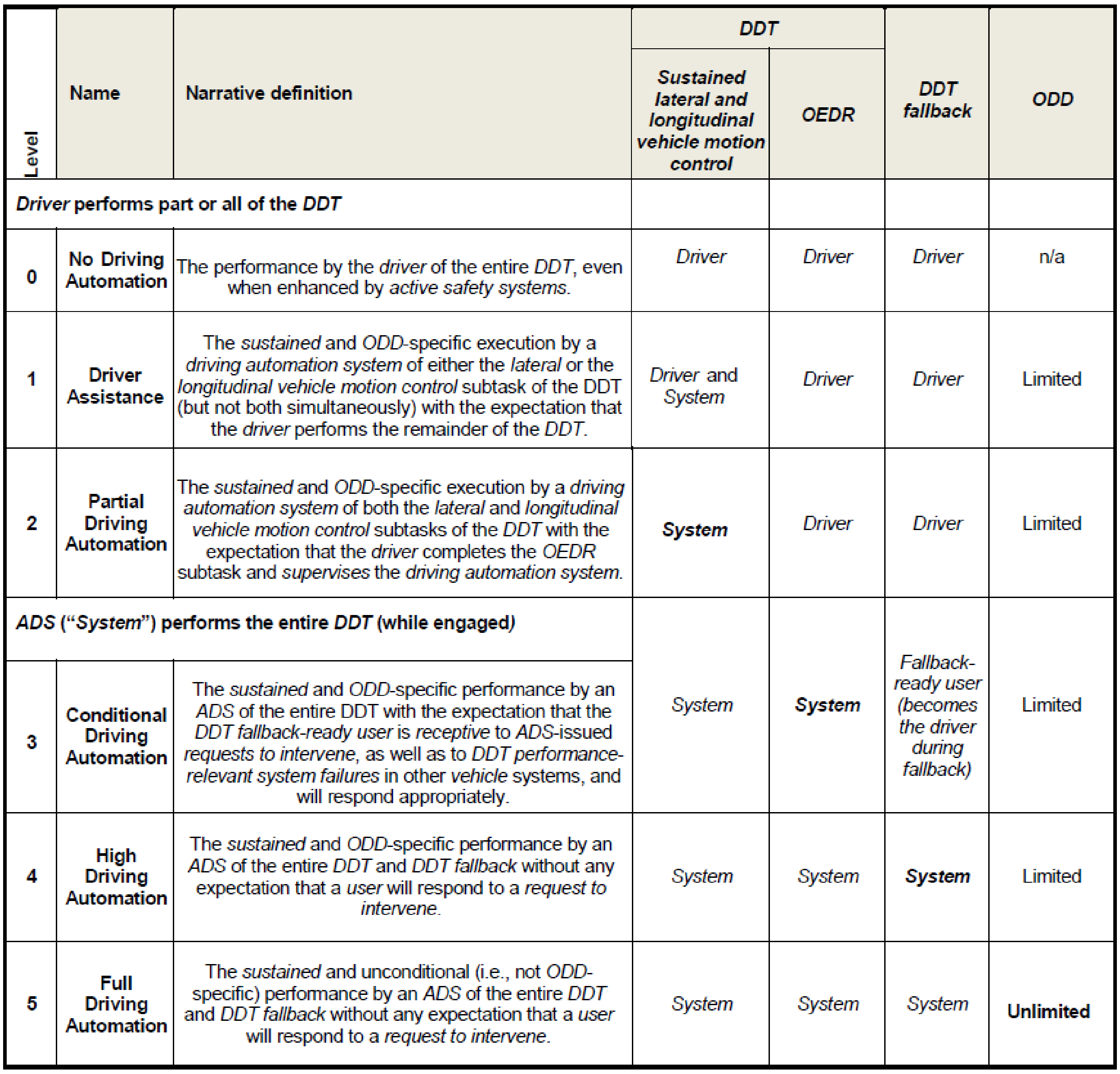
Contents

1 Introduction

Autonomous or self-driving vehicles have multiple advantages over manually driven vehicles. Since they are less prone to human error, they are more safe, more fuel-efficient, cause lesser traffic congestion than humans and thus increase the capacity of the road. For example, according to the National Highway Traffic Safety Administration (NHTSA) [1] and the European Commission [2], more than 90% of all motor vehicle accidents are caused by human error due to speeding, fatigue, distractions and drunk driving. This percentage can be reduced significantly through the use of autonomous driving. Autonomous vehicles can also improve human productivity by reducing the amount and improving the utilization of the time spent on commuting, and can provide better mobility for children, elder people and disabled people.

For these reasons, autonomous driving has been an area of research since the 1920s. In 1939, General Motors presented the concept of the first self-driving car [3]. It was an electric vehicle guided by radio controlled electromagnetic fields generated by magnetized metal spikes in the road. In 1958, the real version of the concept was shown to the world. The car's front end had sensors called pickup coils which could detect current flowing through an embedded wire in the road. The current could be changed to tell the vehicle to move the steering left or right. In 1977, the Tsukuma Mechanical Engineering Lab in Japan presented a system in which a computer in the vehicle processed images of the road from a camera to drive the vehicle, but only up to a speed of 32 km/hr. 10 years later, Daimler and Mercedes-Benz collaborated on a project called VaMoRs. It was a Mercedes-Benz van equipped with a computer, cameras and other sensors. The van could drive itself upto a maximum speed of 90 km/hr. The Defense Advanced Research Project Agency (DARPA) self-driving competitions started in 2004 in the USA and boosted the worldwide efforts to create autonomous vehicles.

Currently many companies and universities across the world are working towards the advancement of autonomous driving with the ultimate aim of level 5 autonomy as defined by the Society of Automotive Engineers (SAE) shown in Figure 1.1. The Dynamic Driving Task (DDT) consists of two parts: Longitudinal/Lateral control and Object Event Detection and Recognition (OEDR). The Operational Design Domain (ODD) represents the domain of operation e.g. fixed routes with fixed times, all routes at any time etc. Depending on the level of automation, the two parts of the DDT might be performed by the system and/or the driver. DDT fallback is an emergency situation e.g. in case an Advanced Driver Assistance System (ADAS) module e.g. the Anti-Lock Braking System (ABS) module stops working, the system cannot perform the DDT and thus requests the driver to take control. In levels 1-3, the DDT fallback has to be handled by the driver whereas in Levels 4-5, the DDT fallback is handled by the system itself. Thus, in levels 4 and 5, all tasks are executed by the system. The only difference between levels 4 and 5 is that the ODD is limited for level 4, whereas there is no limitation on the ODD for level 5. Currently, the maximum level of autonomy achieved by humans is level 4 [4].



Level	Name	Narrative definition	*DDT*		*DDT fallback*	*ODD*
			Sustained lateral and longitudinal vehicle motion control	*OEDR*		
Driver performs part or all of the *DDT*						
0	No Driving Automation	The performance by the *driver* of the entire *DDT*, even when enhanced by *active safety systems*.	*Driver*	*Driver*	*Driver*	n/a
1	Driver Assistance	The *sustained* and *ODD*-specific execution by a *driving automation system* of either the *lateral* or the *longitudinal vehicle motion control* subtask of the DDT (but not both simultaneously) with the expectation that the *driver* perfoms the remainder of the *DDT*.	*Driver* and *System*	*Driver*	*Driver*	Limited
2	Partial Driving Automation	The *sustained* and *ODD*-specific execution by a *driving automation system* of both the *lateral* and *longitudinal vehicle motion control* subtasks of the *DDT* with the expectation that the *driver* completes the *OEDR* subtask and *supervises* the *driving automation system*.	***System***	*Driver*	*Driver*	Limited
ADS ("*System*") performs the entire *DDT* (while engaged)						
3	Conditional Driving Automation	The *sustained* and *ODD*-specific performance by an *ADS* of the entire DDT with the expectation that the *DDT fallback-ready user* is *receptive* to *ADS*-issued *requests to intervene*, as well as to *DDT performance-relevant system failures* in other *vehicle* systems, and will respond appropriately.	*System*	***System***	*Fallback-ready user (becomes the driver during fallback)*	Limited
4	High Driving Automation	The *sustained* and *ODD*-specific performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	***System***	Limited
5	Full Driving Automation	The *sustained* and unconditional (i.e., not *ODD*-specific) performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	*System*	**Unlimited**

Figure 1.1: Levels of autonomy defined by SAE [5]

Autonomous driving consists of five primary tasks as shown in Figure 1.2 [6]:

1. **Sensing**: Sensors like Lidars, Radars, Cameras, Ultrasonic sense or 'see' the world around the vehicle i.e. measure the state of static and dynamic objects i.e. buildings, traffic lights and road users etc. Sensors like Global Positioning System (GPS)/GNSS (Global Navigation Satellite System) and IMU (Inertial Measurement Unit) measure the state of the ego vehicle in the global coordinate frame.

2. **Mapping and Localization**: Based on the measurements received from the sensors, the position of the ego vehicle is estimated on an available map e.g. from GPS/GNSS and/or a calculated map e.g. high definition 3-D map from Lidar by SLAM (Simultaneous Localization and

Mapping) [7], [8].

3. **Sensor fusion**: Measurements from all sensors are fused together to obtain a coherent image of the scenario.

4. **Path planning**: Based on the understood scenario i.e. perceived world and localized position, a short term path (for maneuvering) and a long term path (to the destination) are planned by the system.

5. **Motion control**: Based on the planned path, the system sends control inputs to actuators i.e steering, accelerator, brake etc. to move the vehicle in the desired manner.

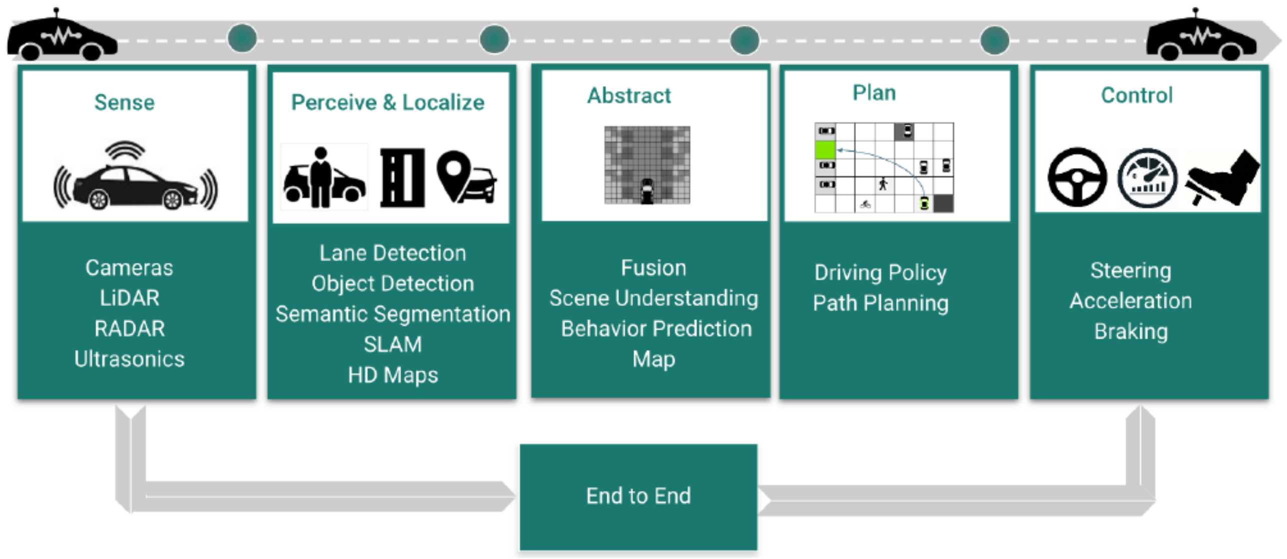


Figure 1.2: Autonomous driving tasks [6]

This PDEng project is concerned with the third task i.e. using information from all sensors on the vehicle e.g. cameras, radars etc. to track all road users i.e. vehicles and pedestrians around the vehicle. In the Netherlands, the i-CAVE (Integrated Cooperative Automated Vehicles) consortium [9] is leading autonomous driving research. This PDEng project has been done as part of the project groups in i-CAVE. The details of the project groups are explained in the next section.

1.1 PDEng project context

I-CAVE is a consortium of 4 universities (Eindhoven University of Technology, Delft University of Technology, University of Twente, and Radboud University) and automotive companies (NXP, DAF trucks, TNO, TomTom etc.) in the Netherlands. The consortium is working on design and development of a C-DMAT (Cooperative Dual Mode Automated Transport) system which can switch seamlessly between autonomous and manual driving modes for flexibility and user acceptance. The target is to achieve Level 5 autonomy with a high level of safety and reliability through technical design based on vision and radar based sensing for environment mapping and localization, and distributed cooperative control algorithms for autonomous platooning and fleet management. It is funded by the

NWO (Netherlands Organization for Scientific Research). There are 7 project groups as shown in Figure 1.3. Since the output of this PDEng project i.e. road user tracking information is eventually used in path planning and control of the vehicle (not goals of this PDEng project), this PDEng project is part of Project Group 2 and demonstrating the result is part of Project Group 7.

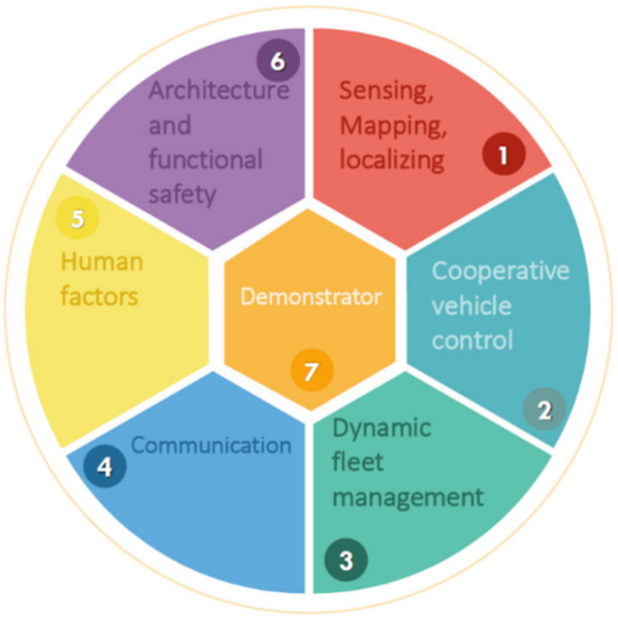

6
Architecture and functional safety
Sensing, Mapping, localizing
1
5
Human factors
Demonstrator
Cooperative vehicle control
7
2
Dynamic fleet management
Communication
4
3

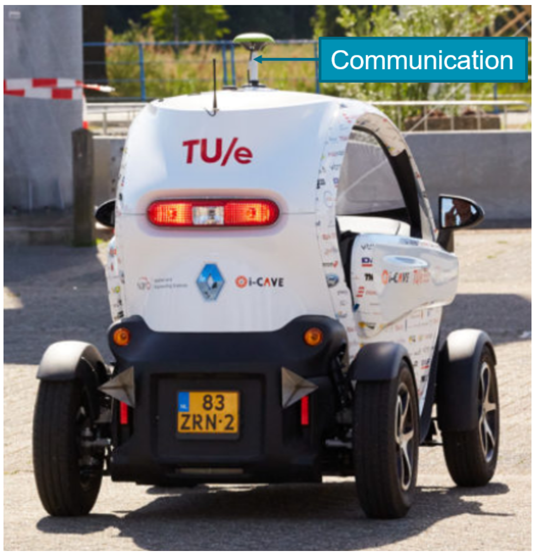
Figure 1.3: Projects in i-CAVE [9]

1.1.1 Project Group 2

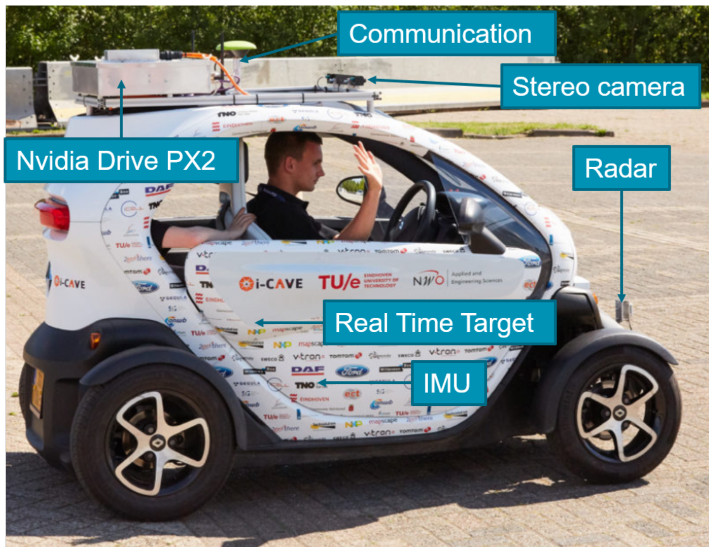
The main goal of this project group are developing controllers for cooperative and autonomous driving [9]. To control a vehicle individually and also as part of a platoon, the first task is to create an accurate picture of the environment around the vehicle. To do that, information from all sensors has to be used so that advantages of all sensors can be combined and the accuracy can be maximized. The result of this information fusion is that all objects around the vehicle can be tracked continuously. The second task is to choose the vehicle to be followed as part of the platoon and plan a dynamic path for following it. For the third task, the vehicle controller has to make decisions i.e. steer, brake or accelerate and implement these decisions through actuators i.e. steering wheel, accelerator and brake pedals. This PDEng project is focused on the first task i.e. map a picture of the environment around the vehicle.

1.1.2 Project Group 7

The aim of this project group is to implement and develop the findings of the other 6 project groups. For this purpose, Renault Twizys have been chosen as test vehicles. The Renault Twizy is a small, electric, city vehicle with a maximum speed of 80 km/hr and a range of 90 km on one charge. It is not equipped with ADAS systems and thus is ideal for installing preferred sensors and actuators for the purpose of adding autonomous driving functionality. The two Twizys have already been used by members involved in the i-CAVE project from Eindhoven University of Technology for their respective Project Groups. Before this PDEng project, Twizy 2 was fitted with 2 additional CAN buses (default configuration has 2 CAN buses), one stereo camera, one radar, Nvidia Drive PX2 [10] for camera data processing, a Real-Time Target (RTT) machine, a wireless communication router for Vehicle-to-Vehicle (V2V) communication, a Global Navigation Satellite System (GNSS) receiver, and an Inertial Measurement Unit (IMU) as shown in Figure 1.4.



(a) Twizy 1



(b) Twizy 2

Figure 1.4: Test vehicles

1.1.3 PDEng project goal

The primary goal of this PDEng project is:

To develop an algorithm to fuse information from two radars, one stereo camera, V2V communication, GNSS and the IMU to estimate the state i.e. the position, velocity, and acceleration of all road users in the Field of View (FOV) of the sensors of Twizy 2.

The following criteria have to be kept in mind:

1. The method has to be accurate in the real-world considering noise an uncertainties in sensors.
2. It has to be computationally efficient to be able to run the algorithm in real-time.
3. The Most Important Object (MIO) i.e. Twizy 1 has to be selected from the tracked objects in the FOV of the sensors.
4. The yaw angle and yaw rate of Twizy 1 also has to be tracked in real-time.

5. The RTT (Real-Time Target) machine can only support upto the R2017b version of Simulink Real-Time. Therefore, MATLAB and Simulink versions upto only R2017b can be used for development.

1.2 Related work

Automatic target tracking i.e. estimating the position, velocity and acceleration of a target object by using measurements from sensors, started in the early 1970s for aerospace applications like radar, sonar, guidance, navigation and air traffic control [11]. Since early 2000s, it is also being used for robotics, image processing, automotive systems, oceanography, remote sensing and biomedical research. It is primarily used for tracking multiple objects at once rather than a single object and thus poses many challenges:

1. **Uncertainty in data association**: Since the number of targets is unknown and is continuously changing, associating measurements with tracks is a difficult task.
2. **Clutter disturbance**: Surrounding objects, sensor noise and weather conditions can cause false tracks to be initialized or incorrect track updates.
3. **Maneuvering target motion**: The targets can have varying motions and sensors are moving with the ego-vehicle frame, so target motion needs to be modelled accurately.
4. **Real-time processing**: The computational complexity increases exponentially with the number of targets and processing needs to be done in real-time.

The strength of the measurement signal is very important for differentiating between automatic tracking and automatic detection. Given noisy measurements, tracking involves state estimation i.e. speed, position, acceleration whereas detection involves determining the presence or absence of an object. If the signal strength is sufficiently high, it can be assumed that the detections are true and tracking methods can be implemented. These methods are called Detect-Before-Track (DBT) methods [12]. DBT methods generally use a threshold to remove measurements which are low in strength. The second group of methods which do not use thresholding because the strength of measurements is already low are called Track-Before-Detect (TBD) methods [13]. TBD methods can be classified into 2 types: Bayesian methods e.g. Bayes-Markov filtering [14] and Non-Bayesian approaches e.g. Hidden Markov Models (HMM) [15]. This PDEng project has focused on DBT methods since the sensors used here provide sufficiently strong measurements. DBT methods can be classified into 2 categories: Data association methods and Finite Set Statistics (FISST) methods.

1.2.1 Data association methods

Data association methods are classical methods which do not use finite sets but matrices for data storage. They explicitly deal with data association i.e. associating measurements with tracks, and target motion modelling problems. Tracks are associated with measurements immediately after preprocessing the measurements e.g. removing low strength measurements as shown in Figure 1.5. Motion modelling and state filtering is used to predict the current measurement. The predicted measurement is corrected using the actual measurement to obtain a more accurate estimate of the current state.

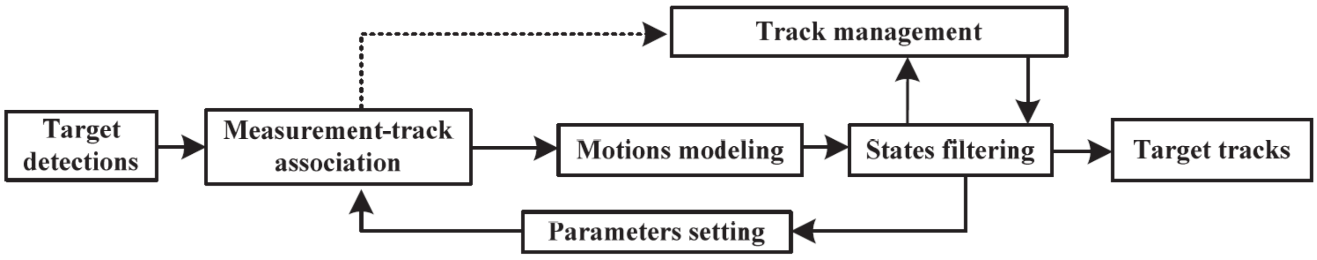


Figure 1.5: Steps in data association methods [12]

There are many criteria for classifying data association methods:

1. **Batch and Recursive methods**: In recursive methods, processing is done each time a measurement scan is received. In contrast, batch methods process data from all measurements together.
2. **Single-scan and Multi-scan methods**: Single scan methods use only the current scan along with the past state estimates to estimate the next state. Multi-scan methods can re-evaluate past scans while processing new scans and can revise previous associations in view of new information.
3. **Hard association and Soft association methods**: Hard association methods assume that only one measurement can originate from one point target whereas soft association methods ignore this assumption.
4. **Heuristic and Bayesian methods**: Heuristic methods are deterministic methods whereas Bayesian methods are stochastic.

1.2.1.1 Heuristic data association methods

These are deterministic algorithms which minimize a cost function and use hard measurement-track association. Examples are Nearest Neighbour Filter (NNF), Strongest Neighbour Filter (SNF), track splitting filter, fuzzy data association, neural networks, Dynamic Programming, genetic algorithms, game theory and graph theory. The disadvantage of these methods is that tuning of heuristic optimization algorithms is time consuming. Also, the performance drops significantly if there are false alarms from the sensors and clutter in the environment. But the advantage of these methods over Bayesian methods is that they are computationally less expensive. In this PDEng project, the Global Nearest Neighbour (GNN) method [16] has been used for data association.

1.2.1.2 Bayesian data association methods

Ideally, all feasible associations between tracks and measurements should be taken into account but the number of combinations increases rapidly with increasing number of targets and the problem becomes computationally expensive. Thus, approximations have to be made.

Maximum a Posteriori (MAP) methods The most well known MAP method is Multiple Hypothesis Tracking (MHT) [17]. It is a multi-scan algorithm that maintains multiple hypotheses by associating past measurements with targets. Every time a new set of measurements are available, a new set of hypotheses are formed from each of the previous hypotheses. The hypothesis with the highest posterior is returned as output. The disadvantage of this method is that the number of hypothesis grow exponentially with time and thus the problem becomes computationally expensive.

An alternative to the MHT is known as the integer programming algorithm [18] or multidimensional assignment algorithm [19]. Measurements in the last S scans are associated with the list of tracks. This is called S-Dimensional association or S-D association. For $S = 2$, the Jonker-Volgenant-Castanon algorithm [20] is used. For $S > 2$, data association becomes an NP-hard problem [21], for which solutions like generalized S-D association [22], Lagrangian relaxation based techniques [23] and biologically inspired ant colony optimization [24] have been devised. Since the core problem is NP-hard, the algorithms are not very efficient.

Bayesian estimator methods These methods generate optimal filtering predictions by summing over all possible associations, weighted by their probabilities. Because calculating truly optimal solutions is computationally expensive, sub-optimal approaches have been found. The simplest method is Probabilistic Data Association (PDA) [25] which takes into account all measurements that might have originated from a target. If this is done for all measurements and all tracks, the algorithm is known as Joint Probabilistic Data Association (JPDA) [26]. JPDA is better than PDA [27] and better than NNF [28] in tracking multiple objects in clutter but is computationally much more expensive since the exact calculation of association probabilities is an NP-hard problem. Heuristic alternatives to JPDA are cheap JPDA [29], sub-optimal JPDA [30] and near optimal JPDA [31]. The Order Statistics Probabilistic Data Association (OSPDA) [32], [28] algorithm combines the advantages of these 2 methods i.e. good performance in clutter and fast computation.

Another alternative to JPDA is Markov Chain Monte Carlo Data Association (MCMCDA) [33] which uses Markov Chain Monte Carlo sampling i.e. random sampling in posterior concentration region instead of taking into account all possible associations. Unlike JPDA and MHT, MCMCDA is a true approximation to the optimal Bayesian filter i.e. when run with unlimited resources, it converges to the Bayesian solution. It works well with large number of targets in a dense region, low detection probabilities and high false alarm rates.

Probabilistic Multi-Hypothesis Tracker (PMHT) is another method for approximating Bayesian filtering [34], [35]. PMHT iteratively calculates the data association probabilities using Expectation-Maximization (EM). It is a soft association batch method which ignores the constraint that one target can only generate one measurement per scan. This method is computationally faster than JPDA but performs worse.

1.2.1.3 Motion modelling and Bayesian state filtering

After the measurements have been associated to tracks, the next state of the track needs to be predicted. A large number of motion models are available for use e.g. e.g. constant velocity model, constant acceleration model, Singer acceleration model and its variants [36]. To account for uncertainties, the Multiple-Model (MM) method uses modes to represent the mixture order of the system Probability Density Function (PDF) [37]. The number of modes increase exponentially for optimal MM methods, thus requiring use of sub-optimal methods like Generalized-Pseudo Bayesian (GPB) merging [38], Gaussian Mixture Reduction (GMR) [39], and Interacting Multiple Models (IMM) [40]. When MR is used with MHT, ad-hoc joining and clustering is used [41]. This preserves the mean and covariance of the original distribution. IMM has been used with JPDA and MHT for IMMJPDA [42] and IMMMHT [43] respectively. The drawback of MM methods is that they use a closed architecture. Variable structure MM (VSMM) algorithms use an open architecture i.e. select an admissible model set at any time instance, but have too much computational complexity for practical applications [44].

After prediction, the next step is dynamic state filtering, for which 3 types of Bayesian filters can be used: Kalman filters, Grid based filters, and Particle filters.

Kalman filters When the state transition model and measurement model are assumed to be linear and Gaussian, the Kalman Filter (KF) provides the optimal Bayesian solution [45]. In practice, most transition and measurement models are non-linear. For these systems, the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) are used [46]. If the system can not be assumed to be Gaussian, Gaussian Sum Filters (GSF) can approximate the non-Gaussian distribution as a sum of Gaussian distributions [47]. For example, the Gaussian Mixture Kalman Filter (GMKF) is used for linear non-Gaussian systems [48]. It uses the greedy expectation maximization method to solve the problem of exponential model order growth.

Grid based filters Grid based filters use a discrete representation of the target density. Complicated densities can be estimated since there are no restrictions on the assumptions about the form of the density. For example, a method approximates the non-Gaussian density numerically without making any linear assumptions and applies numerical integration for the prediction step and Bayesian filtering for the update step [49]. The disadvantages of these methods are that they are computationally expensive if the state space has high dimensionality and that discrete modelling might be too simplistic some times.

Particle filters Particle Filtering (PF) is a Sequential Monte Carlo (SMC) method that can be used for non-linear and non-Gaussian models. The distribution is given by a weighted sum of samples/particles and is propagated using importance sampling to sequentially update the posterior distribution. The standard implementation is the Sequential Importance Sampling Resampling (SISR) filter [50]. Due to resampling, it is computationally complex. The Gaussian Sum Particle Filter (GSPF) overcomes this limitation by skipping the resampling step [51].

For efficient sampling, posterior independence of target states can be used i.e. posterior density of the state can be written as a product of densities of clusters of individual target states e.g. when the targets are far apart. Depending on this use, PFs can be classified into two groups:

1. **Independent samplers**: The Independent Partition PF (IPPF) samples the states of target clusters independently and thus has low computational cost even in the presence of many targets [52]. Measurement to target association is not done explicitly. For explicit data association, PF has been combined with data association methods like PMHT [53], JPDA [54], Rao-Blackwell theorem [55] and MCMC [56].

2. **Joint samplers**: When the target states are not far apart enough to be sampled independently, they have to sampled jointly. This increases the computational load. Sequential Sampling Particle Filter (SSPF) samples the individual targets sequentially by using a factorization of importance weights [57]. Particle Filters and Monte Carlo methods face the problem of maintaining

multi-modality when there are multiple targets. To solve this problem, the target distribution is modelled as a non-parametric mixture model in [58]. Mixture Particle Filters can also be used in combination with boosting [59]. Boosting is an approach to creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules [60].

1.2.2 FISST (Finite State Statistics) methods

They are relatively new methods that make use of Random Finite Sets (RFS) [61]. They treat the collection of individual targets as a set-valued state and the collection of individual measurements as set-valued measurements. The posterior intensity of the RFS is propagated in time and data association is not required before the motion modelling step as shown in Figure 1.6. Still, to keep a record of the identities of the targets for track formation, an estimation-track association step is performed after state filtering.

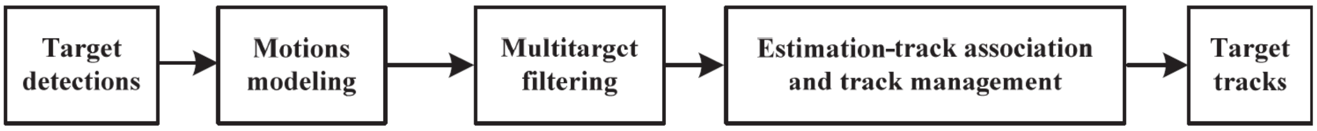


Figure 1.6: Steps in FISST methods [12]

1.2.2.1 Bayesian FISST approaches

In single target problems, the simplest and fastest filter used is the constant gain Kalman Filter. It propagates the first order moment i.e. mean of the posterior distribution instead of the posterior distribution itself. Similarly, for multi-target problems, the Probability Hypothesis Density (PHD) filter propagates the first order moment of the multi-target posterior distribution by using recursive Bayes filtering [62]. The integral of the PHD function in any region of space is the expected number of targets in the region. PHD recursion is a first order approximation, so it uses only parameter to propagate the cardinality distribution i.e the probability distribution of the number of targets. The distribution is Poisson where the mean and variance are equal, meaning that when the number of targets is high, the cardinal distribution has high variance. To account for this limitation, the cardinalized PHD (CPHD) filter does not assume the cardinal distribution to be Poisson but arbitrary [63]. It is more accurate than the PHD filter in estimating the number of targets but is more computationally complex.

1.2.2.2 Application of FISST approaches with Bayesian filters

For practical applications, the PHD method has been used with Gaussian Mixtures (GM) and Sequential Monte Carlo (SMC)/ Particle Filter (PF) in GM-PHD [64] and SMC-PHD [65] respectively. The default SMC approach is computationally complex. Instead of simply using the dynamic model of the system as importance function, data-driven importance functions and corresponding weight functions can be used for survival targets and spontaneous birth targets [66]. The auxiliary SMC-PHD filter [67] is based on the auxiliary particle filter [68]. It uses auxiliary variables which allow the particle filter to the adapted in a more efficient way. As another improvement, measurements can be used to determine the placement of newborn object particles [69]. Multiple Motion (MM) models can also be used with PHD/CPHD filters. For example, linear Jump Markov Systems (JMS) with GM-PHD [70] and SMC-PHD [71], non-linear JMS with GM-PHD [72], IMM with GM-PHD [73] and MM with

GM-CPHD [74].

In practice, the clutter and detection profile parameters are tuned manually or estimated from offline data. Adaptive learning of clutter rate and detection profile parameters using Beta and Gaussian mixtures has been done using Beta and Gaussian mixtures [75]. The clutter density can also be estimated using Finite Mixture Models (FMM) by Expectation Maximization (EM) or Markov Chain Monte Carlo (MCMC) instead of being assumed as uniform [76].

The standard PHD/CPHD filters assume that target birth intensity is known a priori. In practice this does not give good results where targets can appear from anywhere. Adaptive design of target birth intensity can be done at each scan using the measurements to differentiate between persistent and new born targets [77]. Contextual information can also be used to learn the distributions of birth and clutter [78]. Doppler information can also be used with GMPHD to improve clutter rejection and birth rate estimation [79].

For track management of FISST approaches, two methods are used: peak-to-track method and labelling method. In peak-to-track methods, PHD/CPHD is used with MHT [80], fuzzy logic [81], and 2-D assignment [82], [83]. In labelling methods, partitions are made in the position domain. All elements in a partition receive the same label. While resampling i.e. pruning or merging, children of elements are given the same label as the parents. After re-partitioning, if the majority of the particles in one partition have the same label, the partition is associated with that label. Labelling can be done through particles in SMC [84] or Gaussian elements in GM [85]. The advantage of labelling methods is that they can be implemented directly in the filtering step.

1.2.3 Multi-sensor fusion

Using information from multiple sensors can improve accuracy of estimation as compared to the accuracy of estimation from one sensor only [86]. There are two ways to fuse measurements from all sensors: Low-level sensor data fusion and High-level sensor data fusion [87].

1.2.3.1 Low-level sensor data fusion

Low-level data is measurement data which does not go temporal filtering i.e. averaging or tracking. A common approach is to fuse all measurements together and then use a single Bayesian filter and a single measurement model to track the target state as shown in Figure 1.7. The advantages are that simultaneous information processing is more robust and gives better performance [88]. It also minimizes the information loss as data processing leads to data reduction [89]. It also reduces the risk of taking incorrect modelling assumptions [90].

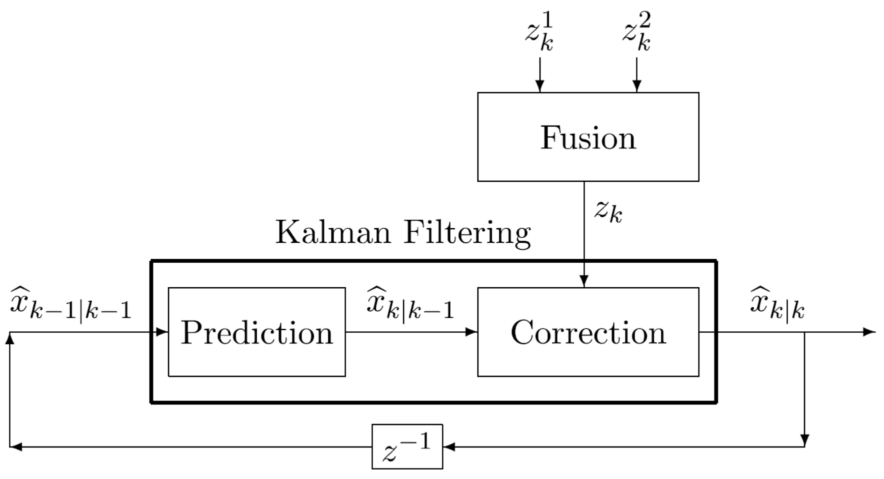


Figure 1.7: Low-level sensor data fusion [91]

1.2.3.2 High-level sensor data fusion

High-level sensor data is measurement data obtained after temporal filtering i.e. tracking. Tracked data from different sensors is correlated because of shared modelling assumptions [92], the double counting problem i.e. when data is unintentionally used multiple times [93], common measurement noise on the tracked object [94] and measurements coming out of sequence [95]. Track-to-Track fusion methods can be divided into 3 categories on the basis of correlation between tracks: Assuming no correlations, calculating correlations, and assuming known correlations.

Assuming no correlations This group of algorithms does not assume any correlations between the tracks. The Covariance Intersection (CI) algorithm [94] is the most known algorithm from this group. It performs consistently but is computationally demanding due to an optimization step and provides conservative estimates [96]. Fast CI [97] and improved fast CI [98] are faster versions of the CI algorithm. The largest ellipsoid [99] and the internal ellipsoid approximation algorithm [100] are less conservative. Most of these methods cannot prove that the fused estimate has an estimation error covariance matrix that is equal or lower than the original estimate's error covariance matrix.

Calculating correlations The Information Matrix Fusion (IMF) method is the most known algorithm from this group [101]. It memorizes individual measurements and then fuses only new measurements by de-correlating each track. It is not usable for dynamic network configurations e.g. in cooperative driving with ad-hoc communication networks. The Cross-Covariance Method (CCM) [102] is an alternative to the IMF method, but it requires more than object-level measurements e.g. parameters of the Kalman Filter. It is not usable for black box sensors i.e. sensors which do not expose internal filter settings.

Assuming known correlations The Best Linear Unbiased Estimate (BLUE) [103] can be used when the correlations are fully known. When the correlations are partially known, CCM and and cross correlation with minimization of Mahalanobis distance [104] can be used. Kalman filters based solutions are part of the second and third groups [95]. Algorithms from the second and third groups are more accurate than algorithms in the first group, but are not always practical.

1.2.3.3 Sensor fusion architectures

The first Track-to-Track fusion architecture [103], [28] uses separate filters for each track to predict and update the states, and then fuses the updated states as shown in Figure 1.8.

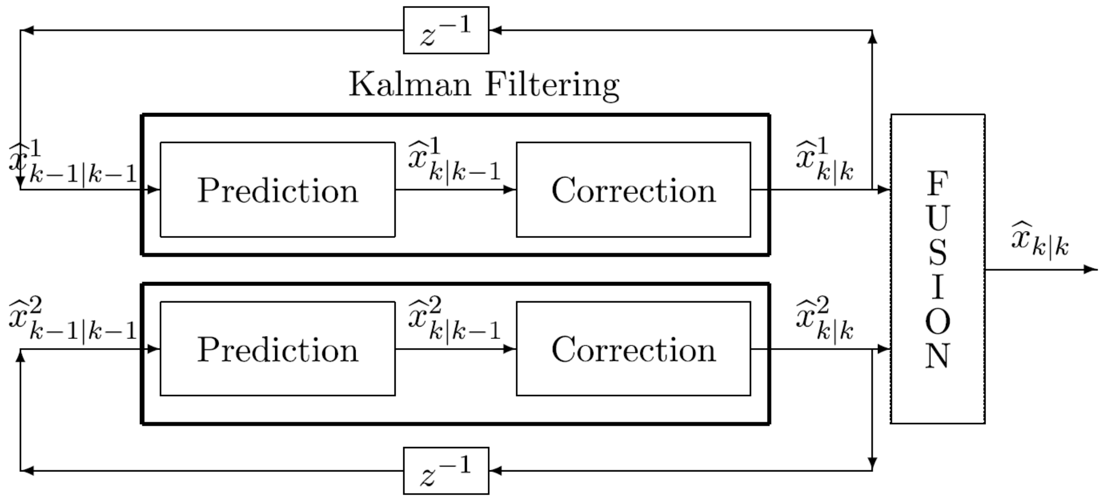


Figure 1.8: Track-to-Track fusion [103], [28]

A modification to this architecture uses a single state estimator [91] instead of separate state predictors and updates the prediction using the measurements from sensors separately, and then fuses the updated states as shown in Figure 1.9.

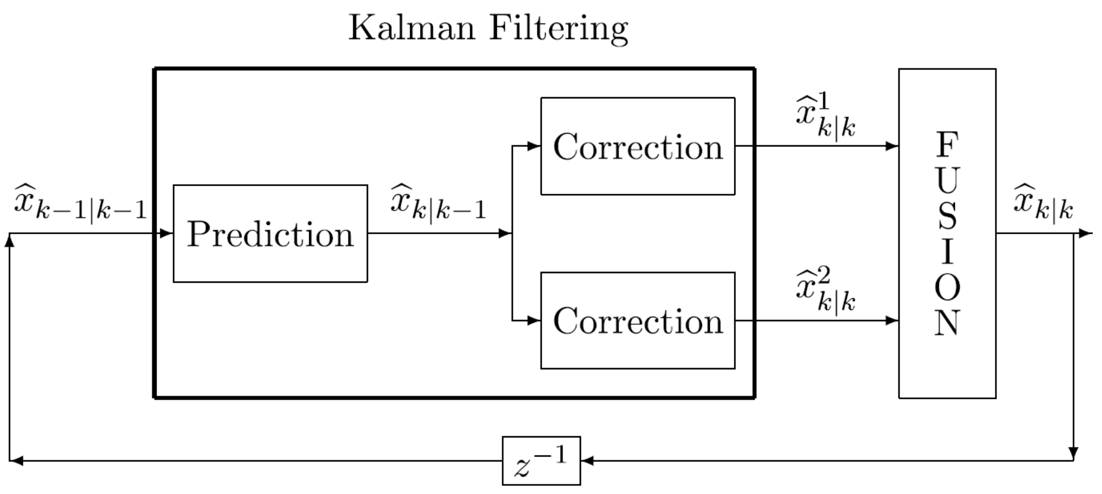


Figure 1.9: Modified Track-to-Track fusion [91]

Track-to-Track fusion with fused prediction [91] fuses the predictions of separate state predictors and updates this fused prediction using the measurements from sensors separately, and then fuses the updated states as shown in Figure 1.10. This architecture is found to perform the best among the above mentioned three architectures in terms of accuracy [105], but it is computationally more expensive since fusion has to be performed twice.

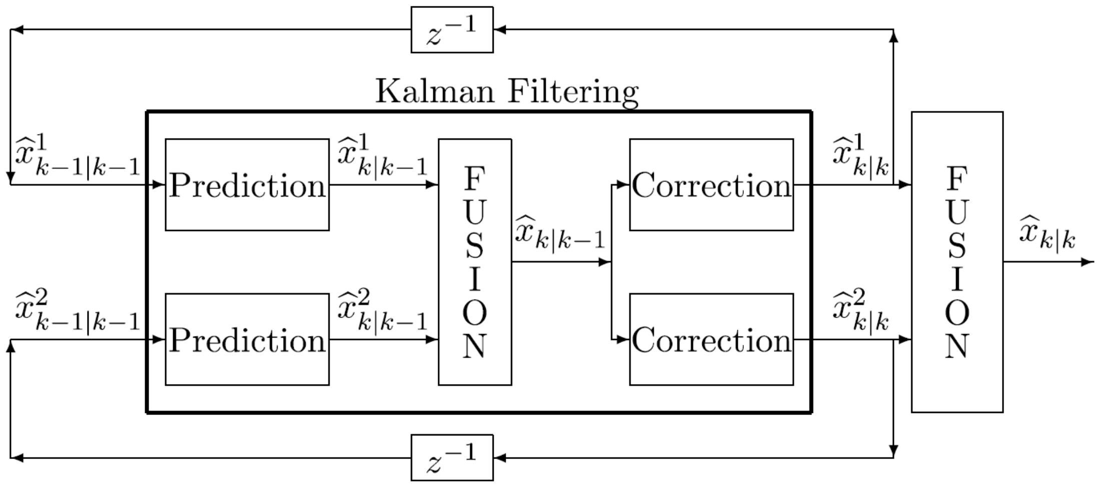


Figure 1.10: Track-to-Track fusion with fused prediction [91]

1.3 Approach to the problem

The focus of this PDEng project is practical implementation of an existing method rather than development of a new method. Also, the implemented method has to achieve a good balance of computational speed and accuracy. Keeping these points in mind, the approach can be summarized in the following steps:

1. With respect to the number of hypotheses tracked, a single hypothesis tracker is chosen since a multi-hypothesis tracker, although more accurate, is much more computationally expensive. In the chosen approach, the detections from the radars are clustered in only one way i.e. using a single threshold value to cluster detections together e.g. a fixed value depending on the dimensions of the platoon vehicle in front. The detections from the stereo camera don't need to be clustered as the camera already places a bounding box on the objects and measures the location of a fixed point on the bounding box e.g. the mid point of the bottom side of the bounding box. The resulting set of detections after this step is the only set i.e. the only hypothesis which is input to the tracker at a time and tracked by it.

2. For data association, the simplest method among the methods shown in the related work section has been used for this project. The GNN (Global Nearest Neighbour) data association method is a heuristic (non-probabilistic) method that associates measurements to tracks based on the magnitude of the normalized distance or *Mahalanobis* distance [106]. Only one measurement is assigned to one track and vice versa.

3. For sensor fusion, low-level sensor data fusion shown in Figure 1.7 is done on the radar data as explained above. Tentative global tracks are initialized from initial measurements. For each sensor, the measurements are associated to these global tracks. From unassigned detections,

global tracks are initiated as tentative tracks. If A out of the next B detections are assigned to a tentative track, it is upgraded to a confirmed track. If no detection is assigned to a track, the track is coasted. If it is coasted for longer than a threshold time without being updated, it is deleted.

4. For tracking the relative yaw rate and relative yaw angle of the target, the tracked path of the target is fitted onto a third order polynomial. The slope and curvature of the polynomial is used to calculate the relative yaw angle and relative yaw rate respectively.

5. The above mentioned methods are first implemented in three simulation scenarios in MATLAB and then validated using offline reconstruction from real-world data and online real-time performance in a real-world scenario.

1.4 Report outline

Chapter 2 explains the hardware setup on Twizy 2. Chapter 3 explains the principles of object tracking and sensor fusion. Chapter 4 shows the results of different simulation scenarios in MATLAB. Chapter 5 shows the results obtained offline using real-world measurements from 1 radar, GNSS and the IMU. Chapter 6 explains the conclusions and future work.

2 Hardware setup

Before this PDEng project, Twizy 2 was fitted with additional hardware to convert it into an autonomous vehicle. The following requirements had to be kept in mind while deciding the additional hardware setup [107]:

1. The original architecture must be available in case of fallback from automated driving to manual driving. Thus the original hardware setup i.e. CAN bus, steering, brakes and accelerator must not be invaded but only used as a base for additional functions e.g. drive, steer and brake by wire.
2. The added hardware must have self diagnosis capability e.g. range check or redundancy.
3. Additional sensors and actuators must be connected through Controller Area Network (CAN) bus or automotive Ethernet.
4. The vehicle must have two people inside it at all times. One person sits just behind the steering wheel and monitors the environment at all times to be ready in case of a fallback to manual driving. The second person monitors the automated driving system at all times.
5. The additional components must be accessible for updates and/or repair. This is important in the context of Twizy because limited space is available for additional components and the back seat cannot be used for hardware since 2 people have to be seated in the Twizy at all times.

The hardware setup of Twizy 2 before this PDEng project is shown in Figure 2.1.

2.1 Real Time Target (RTT) machine

The RTT machine [108] receives all information from the sensors, processes it in the algorithms uploaded in it, and makes and executes decisions through the actuators. The Advantech ARK-3520P RTT machine was chosen with Simulink Real Time as the computation software. It is mounted behind the passenger seat in a waterproof box. The real time software runs at 100 Hz. The machine has 2 CAN boards with 2 CAN channels each, 3 Ethernet ports, 4 serial RS-232 ports, and 4 configurable RS-232/422/485 ports. The 4 CAN ports connect to 4 CAN buses:

1. V-CAN: The original vehicle CAN bus.
2. C-CAN: The controller CAN bus which connects to the steer motor and brake motor controllers.
3. P-CAN: The CAN bus which is connected to the CAN based front facing Bosch mid-range radar (MRR) [109] used for Hardware-in-Loop testing for safety analysis [107].

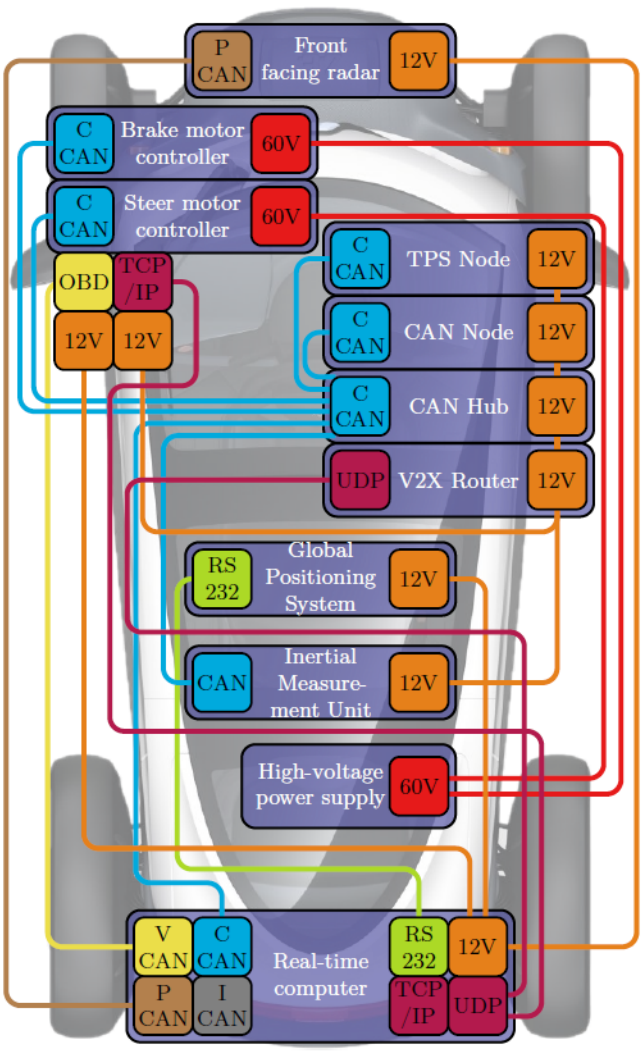


Figure 2.1: Twizy 2 hardware setup before this PDEng project [107]

4. I-CAN: The instrumentation CAN bus which is connected to the Nvidia DrivePX2.

For this PDEng project, 1 Ethernet port of the RTT machine was also used. It connects to an Ethernet port of the NXP Bluebox, which is connected to two front facing Ethernet based NXP Cocoon radars. The Bluebox is described in the last section of this chapter.

2.2 Drive By Wire (DBW) hardware

A take-over module i.e. TPS (Throttle Position Sensor) node which can emulate the accelerator pedal position sensor signal was mounted behind the right head lamp and is connected to the RTT machine via the controller CAN bus. It contains two take over relays to switch between manual driving when the accelerator pedal is connected to the Power Electronics Block (PEB) and automated driving when the control signal is connected to the PEB. In case of a power outage/fault the take over relays default to the accelerator pedal for manual driving.

2.3 Steer By Wire (SBW) hardware

The Twizy does not have power steering, so there is no power steering motor that can be electronically controlled. A Commercial Off The Shelf (COTS) solution in the form of the power-assisted steering column of Renault Clio 2 was fitted in the Twizy with minor modifications. The electric motor is connected to the steering column via a worm gear. A controller connected to the 60V power supply of the Twizy is used to drive the electric motor. The controller is connected to the RTT machine through the controller CAN bus.

2.4 Brake By Wire (SBW) hardware

The Twizy does not have Electronic Stability Control (ESC) or Electronic Hydaulic Braking (EHB), so additional hardware in the form of a brake actuator is added to the brake master cylinder. The actuator is a brake motor connected to a cam follower mechanism via a gearbox. This mechanism is not coupled directly to the brake pedal system and does not change the original braking system. The electric motor is controlled through a controller and is connected to the RTT machine through the controller CAN bus.

2.5 CAN hub and CAN node

Many components have to be connected to the controller CAN bus. Also the number of components are not fixed. Thus a CAN hub is developed to connect all C-CAN devices. Other CAN buses are point to point connections and do not need additional connections. The CAN hub is mounted in the right hand storage compartment and can connect upto 11 devices in star connection. A CAN node is developed to act as CAN to I/O interface for the RTT machine because the RTT machine only has serial communication ports as I/O interfaces. It is mounted behind the right headlamp next to the CAN hub. It controls the status LEDs for the safety driver, high voltage power supply relay, brake light relay and also reads and analyses analog sensor signals e.g. steering torque and brake pressure signals.

2.6 Vehicle-to-Vehicle (V2V) communication modem

An ETSI ITS-G5 [110] compliant IEEE 802.11p wireless communication router (a PC Engines APU2 computer [111] running the open source software GeoNetworking) was chosen for V2V communication. The router supports Cooperative Awareness Messages (CAM), Decentralized Environmental Notification Messages (DENM), and custom messages. CAM messages containing vehicle speed, position and acceleration can be communicated to other vehicles in the platoon. The DENM message is used to alert road users and is triggered in case of an emergency event e.g. an emergency vehicle passing by.

2.7 Global Navigation Satellite System (GNSS) receiver

A u-blox EVK M8T GNSS receiver [112] is mounted under the roof to receive global coordinates, speed and heading. The update frequency is 5 Hz and the communication to the RTT machine takes place via a serial RS-232 port. The precision of the module was improved through an improved antenna and the European Geostationary Navigation Overlay Service (EGNOS). The improve the accuracy further, sensor fusion of the GNSS receiver and the inertial sensors was also done (not part of this PDEng project). To synchronize the local clock on the RTT machine with the absolute time, the GNSS receiver provides a digital time pulse every second. The local clock is used to determine the age of the received wireless messages and the GNSS time delay due to data transmission and internal processing.

2.8 Inertial Measurement Unit (IMU)

A Bosch MM5.10 Inertial Measurement Unit (IMU) [113] was chosen. It is mounted beneath the driver seat near the center of gravity. It measures the yaw rate, roll rate, longitudinal, lateral and vertical acceleration. The signals are low pass filtered at 15 Hz and are communicated via the controller CAN bus.

2.9 Power supply and operator panel

The low voltage power supply i.e. 12 V supply of the Twizy is not sufficient for the added steering and brake motors. Thus these motors are powered by the high voltage i.e. 60V supply. For this purpose, a fused relay box is mounted near the PEB (Power Electronics Block) and is connected to the PEB motor output. Outputs from the relay box lead to the braking and steering motors. These outputs are mounted on the right side of the vehicle to prevent interference with the communication lines mounted on the left side of the vehicle. The CAN hub distributes power to the RTT machine and other additional components. The CAN hub is powered by the fused and ignition switched original accessories supply wire. An operator panel is placed on top of the left hand storage compartment in view of the safety driver as shown in Figure 2.2.

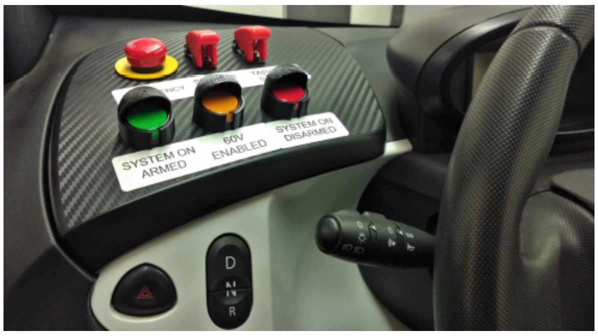


Figure 2.2: Twizy 2 operator panel [107]

A switch for powering off the additional hardware in case of manual driving is placed on the operator panel. This switch is in series with the original supply wire. A similar switch is added on the panel to control power to the RTT machine. A red emergency switch is also mounted on the panel in series with the CAN hub. When this button is pressed an emergency situation, a reset signal is sent to the TPS (Throttle Position Sensor) and CAN nodes that disables their output. As the CAN node controls the high voltage power supply relay, the latter is disconnected. There are 3 LEDs to convey the system status to the emergency driver. The green LED goes on when autonomous driving is enabled. The orange LED goes on when the 60 V power supply is enabled. The red LED turns on automatically when the car is started. It turns off when autonomous driving is enabled.

2.10 NXP Cocoon radar

2 Ethernet based NXP Cocoon radars [114] were supplied by NXP, which is a partner in i-CAVE. The NXP Cocoon radar is a Frequency Modulated Continuous Wave (FMCW) radar.

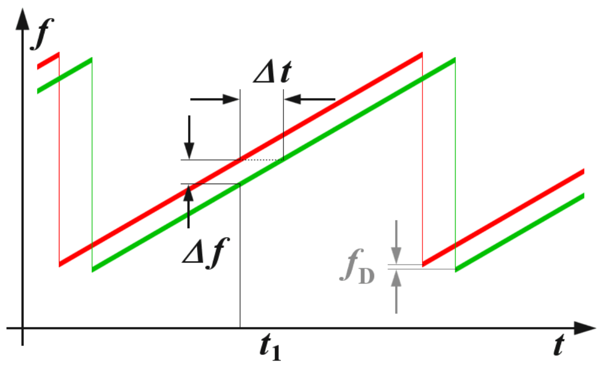


Figure 2.3: FMCW radar frequency time plot [115]

In the Radio Frequency (RF) generator, an electromagnetic wave is generated with increasing frequency. This creates a ramp as shown in Figure 2.3 with the red line. This signal is transmitted with the transmitting antenna. The wave is reflected back from a surface and is received with the receiving antenna which is shown in Figure 2.3 with the green line. Using the difference in frequency Δf between the transmitted signal and received signal, the time of flight Δt of the radar signal is calculated. Using the time of flight, the distance to the target is calculated. The radial velocity is calculated using the Doppler effect. When the moving electromagnetic wave hits a moving object, the frequency of the wave shifts by f_D as shown in Figure 2.3. This shift depends on the radial velocity of the object relative to the radar. The phase difference between two consecutive ramps is used to calculate the radial velocity.

If two measurements have the same distance and radial velocity, they are likely to belong to the same object. Based on this assumption, the signal processing inside the radar calculates the average of the angle of the two measurements and creates one measurement with the average of the angle of the two measurements. Therefore the radar does not have any angular resolution [105]. The radar measures the range (r), azimuth (φ) , elevation (θ), Signal-to-Noise Ratio (SNR), the number of objects detected, and the radial velocity/range rate ($\dot{r}$). The measurement vector can be written as

$$Z_r = \begin{bmatrix} r & \theta & \varphi & \dot{r} \end{bmatrix}^\top \tag{2.1}$$

The measurements are obtained in spherical coordinates as shown in Figure 2.4.

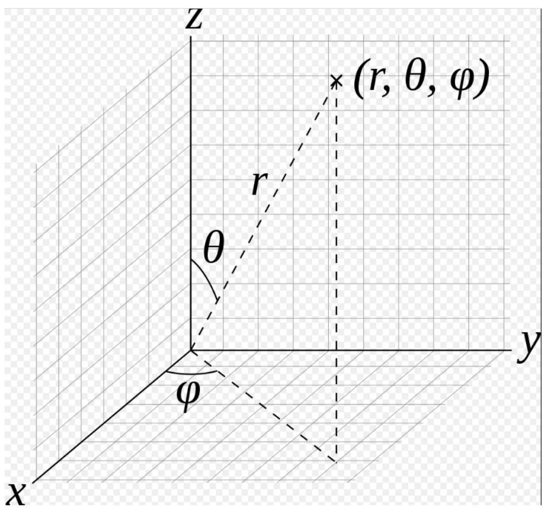


Figure 2.4: Spherical coordinates

These measurements can be converted to Cartesian coordinates:

$$\begin{aligned} x &= r\sin\theta\cos\varphi &&\implies \dot{x} = \boldsymbol{\dot{r}\sin\theta\cos\varphi} - r\dot{\varphi}\sin\theta\sin\varphi + r\dot{\theta}\cos\theta\cos\varphi \\ y &= r\sin\theta\sin\varphi &&\implies \dot{y} = \boldsymbol{\dot{r}\sin\theta\sin\varphi} + r\dot{\varphi}\sin\theta\cos\varphi + r\dot{\theta}\cos\theta\sin\varphi \\ z &= r\cos\theta &&\implies \dot{z} = \boldsymbol{\dot{r}\cos\theta} - r\dot{\theta}\sin\theta \end{aligned} \tag{2.2}$$

Since the rate of change of azimuth and elevation are unknown, MATLAB takes the terms in bold as estimates of the speed measurements in Cartesian coordinates for Bayesian filtering. The specifications of the Cocoon radar are shown in Table 2.1.

Table 2.1: NXP Cocoon radar specifications

Parameter	Long range configuration	Short range configuration
Effective bandwidth	275 MHz	1500 MHz
Effective center frequency	78.5 GHz	79 GHz
Measurement distance	69.8 m	12.8 m
Minimum distance	0.75 m	0.1 m
Distance resolution	0.55 m	0.1 m
Maximum relative approaching velocity	250 km/hr	250 km/hr
Maximum relative receding velocity	150 km/hr	150 km/hr
Velocity resolution	1 km/hr	1 km/hr
Azimuth field of view at 1 m	±60°	±60°
Elevation field of view at 1 m	±10°	±10°
Angular resolution	N/A	N/A
System cycle	70 ms	70 ms

2.11 Sekonix AR0231 GMSL Camera

The stereo camera is made up by two Sekonix AR0231 GMSL mono cameras [116]. Gigabit Multimedia Serial Link (GMSL) cameras use serializer and deserializer (SerDes) high speed communication Integrated Chips (ICs). They support high bandwidths, complex inter-connectivity and data integrity requirements of ADAS [117].

Stereo cameras use binocular stereopsis i.e. the principle of using 2 cameras separated horizontally to perceive depth based on the disparity obtained after superimposing the images obtained from the cameras on top of each other as shown in Figure 2.5.

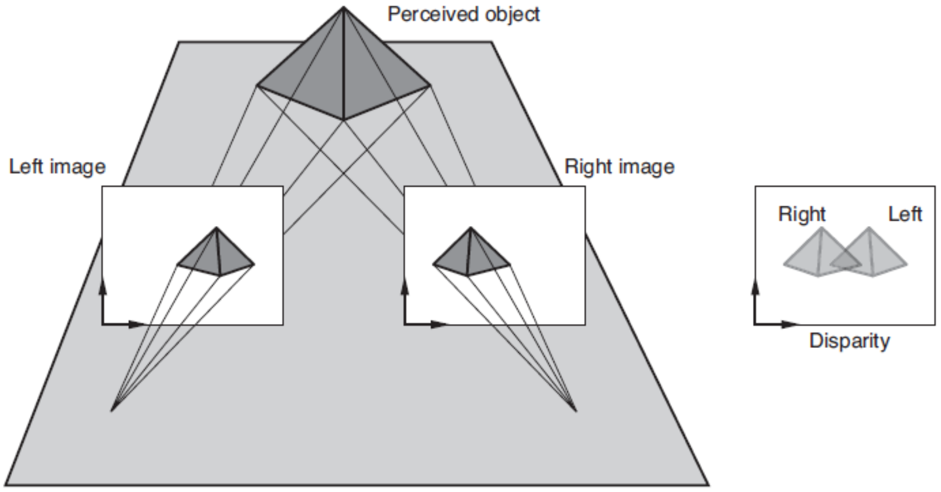


Figure 2.5: Stereo vision [118]

Using disparity and geometry from Figure 2.6, the depth can be calculated using:

$$\tan\psi = \frac{b/2}{Z} \qquad \text{and} \qquad \tan(\psi - \frac{\delta\psi}{2}) = \frac{b/2}{Z+\delta Z} \tag{2.3}$$

where b is the baseline and Z is the depth.

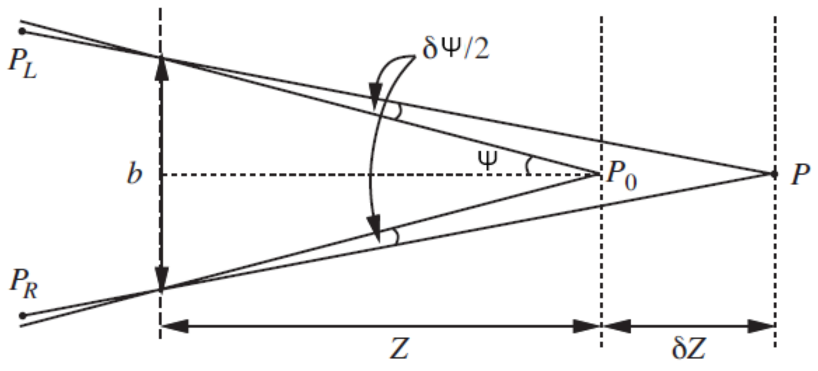


Figure 2.6: Geometry of depth and disparity [118]

For small angles, $\tan\psi \approx \psi$, so the disparity $\delta\psi$ is

$$\begin{aligned} \frac{\delta\psi}{2} &= \frac{b/2}{Z} - \frac{b/2}{Z+\delta Z} \approx \frac{b\delta Z}{2Z^2} \\ \implies \delta\psi &\approx \frac{b\delta Z}{Z^2} \end{aligned} \tag{2.4}$$

The camera measures the range (r), azimuth (φ) , elevation (θ), object class (car, pedestrian, bicycle, road sign, or truck), the number of objects detected, and the radial velocity/range rate ($\dot{r}$). The measurement vector can be written as

$$Z_c = \begin{bmatrix} r & \theta & \varphi & \dot{r} \end{bmatrix}^\top \tag{2.5}$$

Measurement in spherical coordinates are converted into Cartesian coordinates similar to that of the radar. The specifications of the cameras are given in Table 2.2.

Table 2.2: Sekonix mono camera specifications

Model name	Serializer	Type	Color filter	Image size	Effective pixels	Azimuth FOV	Elevation FOV
SF3325-100	MAX96705	GMSL	RCCB	1/2.7-inch	1928x1208	$\pm 30°$	$\pm 15°$

2.12 NXP Bluebox

Two autonomous driving platforms have been installed on Twizy 2:

1. Nvidia Drive PX2 [10]: For camera data processing
2. NXP Bluebox [119]: For radar data processing and sensor fusion

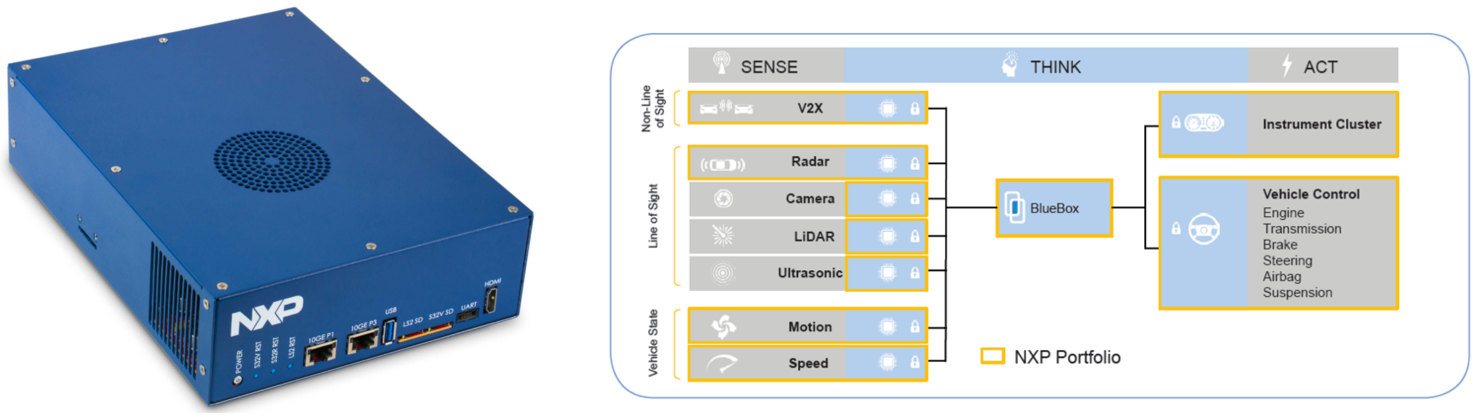


Figure 2.7: NXP Bluebox [119]

This PDEng project is focused on implementation of NXP Bluebox to attain the project goal. The Nvidia Drive PX2 is the focus of another PDEng project done in parallel with this PDEng project. The Bluebox is Linux based with Ubuntu 18.04 and ROS (Robot Operating System) Melodic installed. Sensors act as ROS nodes and can communicate with the Bluebox using automotive Ethernet or CAN/CAN-FD or Flexray [120]. As mentioned in Section 2.1, 2 Ethernet based NXP Cocoon radars are connected to the Bluebox. The Bluebox is connected to the RTT machine through an Ethernet port. The Bluebox is equipped with the following primary hardware:

1. **LS2084A embedded computing processor**: This processor has eight ARM A72 cores [121] and 15 GB DDR4 RAM. It is suitable for high single-threaded and multi-threaded performance with low latency. It is intended to be used for decision making e.g. path planning, which requires heavy computation.
2. **S32V234 vision and sensor fusion processor**: This processor has four ARM A53 cores and 2GB DDR3 RAM. It uses the APEX2 engine for vision processing and basic neural network inference. It is intended to be used for sensor fusion and thus obtaining a clear picture of the environment around the vehicle.
3. **S32R27 radar microcontroller**: It is an ASIL-D [122] automotive Micro-Controller Unit (MCU) with accelerators for radar signal processing. It is intended to be used primarily as an optional safety supervisor to monitor external processor faults, system temperature and voltage conditions.

2.13 Current setup

The old setup had 1 front facing CAN based NXP Cocoon radar and 1 front facing stereo camera. The sensor coverage area with 1 radar and 1 stereo camera is shown in Figure 2.8.

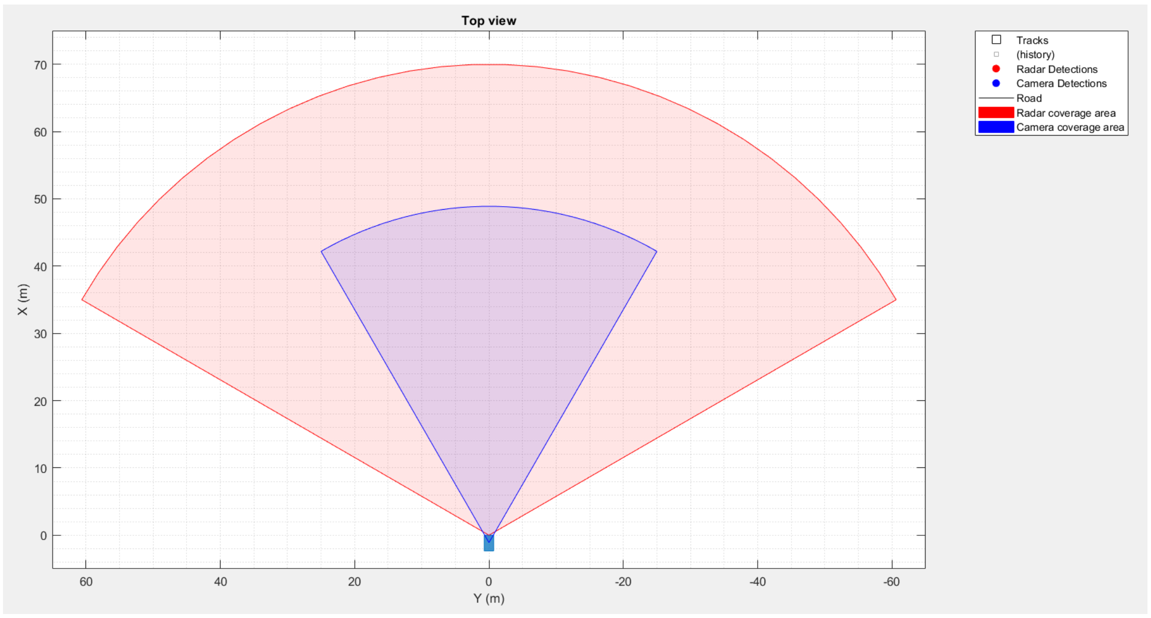


Figure 2.8: Coverage area with old setup

The current setup has 2 Ethernet based NXP Cocoon radars aligned at 30° and −30° to the ego Twizy frame respectively, and one front facing stereo camera. The sensor coverage area with 2 radars and 1 stereo camera is shown in Figure 2.9. The current setup gives more coverage area than the old setup and has a lesser chance of losing track of the leading vehicle during a sharp turn.

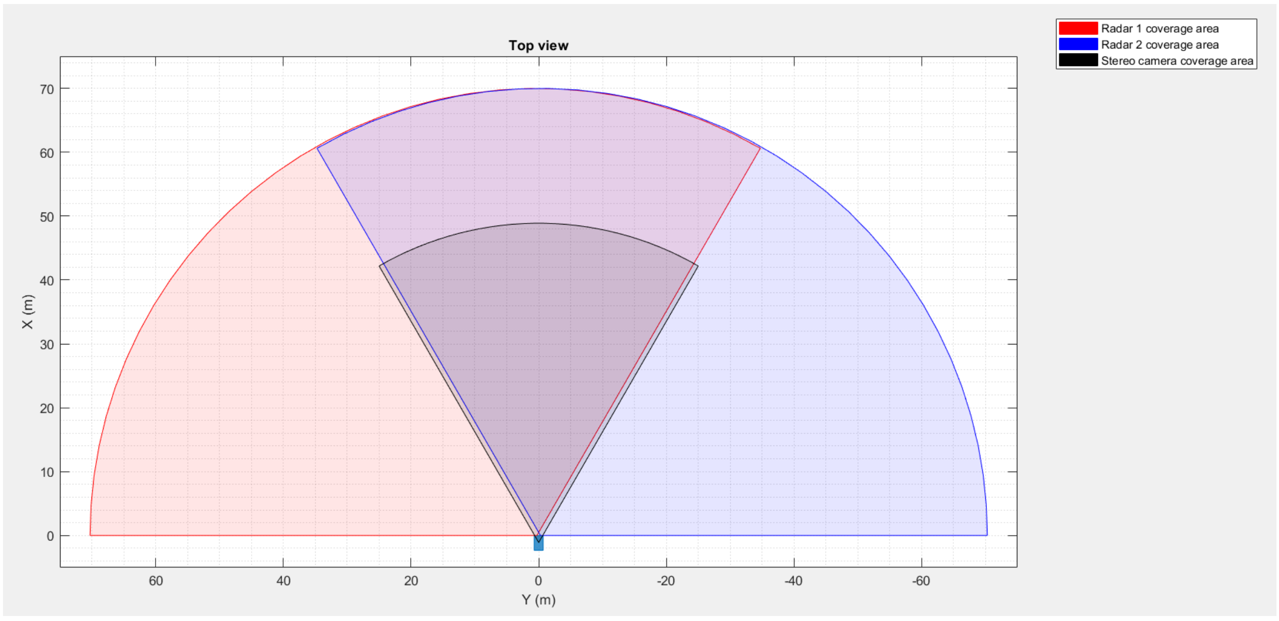


Figure 2.9: Coverage area with new setup

The radars are placed at 0.375 m on either side of the origin because the minimum measurable distance from the radar is 0.75 m. The area of the blind spot is 0.4871 square meters as shown in Figure 2.10.

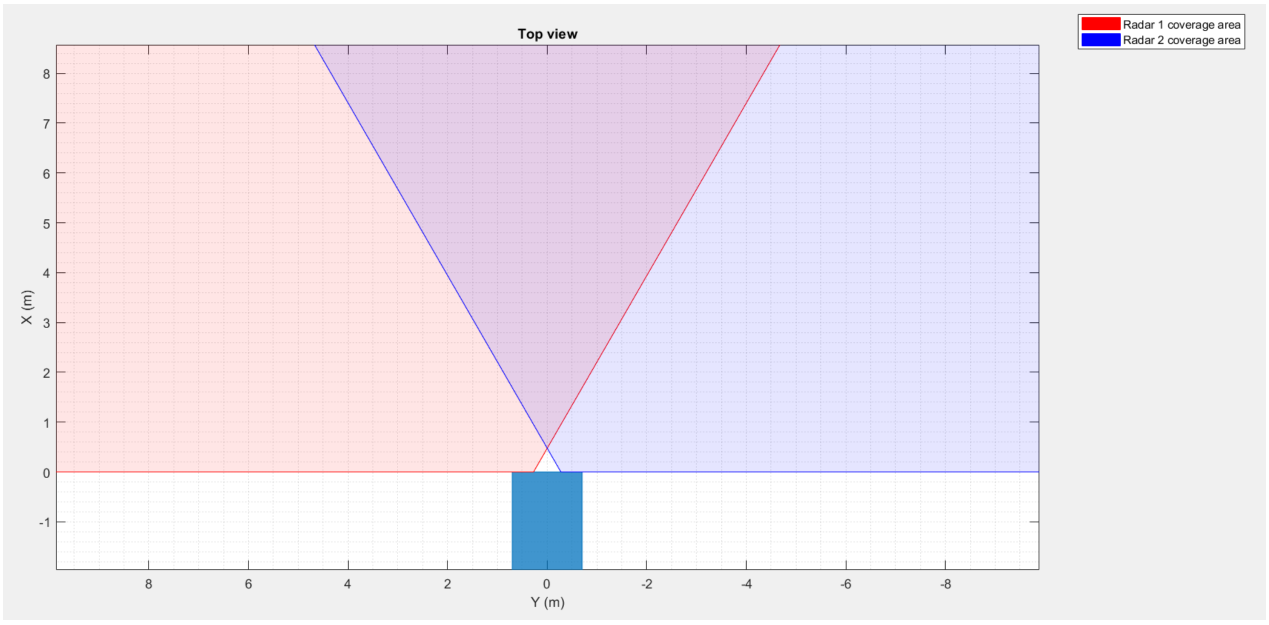


Figure 2.10: Blind spot

3 Principles of object tracking and sensor fusion

For the first task of autonomous driving i.e. sensing, a combination of 3 types of sensors are primarily used: radar, lidar and camera [6]. Multiple sensors are used because each sensor has its own strengths and weaknesses as shown in Table 3.1, and the goal is to maximize the accuracy of detection. The areas in which one sensor is lacking can be taken care of by another sensor e.g. angular resolution of radar is not good but is taken care of by the camera, the camera does not work well in bad weather but the radar does etc.

Table 3.1: Comparison of radar, lidar and camera [123]

Parameter	Radar	Lidar	Camera
Range	Very good	Good	Very good
Range resolution	Good	Very good	Average
Angular resolution	Average	Very good	Good
Performance in rain/fog/snow	Very good	Average	Poor
Performance in dark	Very good	Very poor	Very poor
Performance in bright light	Very good	Good	Good
Color/Contrast	Very poor	Very good	Very good
Radial velocity	Very good	Average	Poor

Actual measurements from sensors can not be used as states of targets because sensors have a measurement noise. Therefore measurements have to be filtered. A Bayesian filter uses Bayes theorem to filter the measurements. Given some prior distribution i.e. measurements, the posterior distribution i.e. the state of the target is calculated using an assumed motion model and a measurement model. As explained in Chapter 1, three types of Bayesian filters can be used to track the state of the target: Kalman filters, Grid Based Filters, and Particle Filters. An Unscented Kalman Filter (UKF) is chosen for this project since the state i.e. posterior distribution for one target can be assumed to be Gaussian as the priors i.e. the measurements are Gaussian with a single mean vector and measurement noise covariance matrix, and the measurement function is non-linear.

The mathematical concepts used for estimating the state of the target from measurements are shown in Figure 3.1. Inside the UKF (Unscented Kalman Filter) block, the current state estimate and the measurement received from the measurement processing block at the next time step is used to estimate the state at the next time step. The state estimated at the next time step is used as a waypoint for relative yaw angle and yaw rate tracking using polynomial path fitting, and for relative sideslip estimation using the tracked relative velocities and accelerations. The following sections explain each major block and corresponding sub-blocks in Figure 3.1.

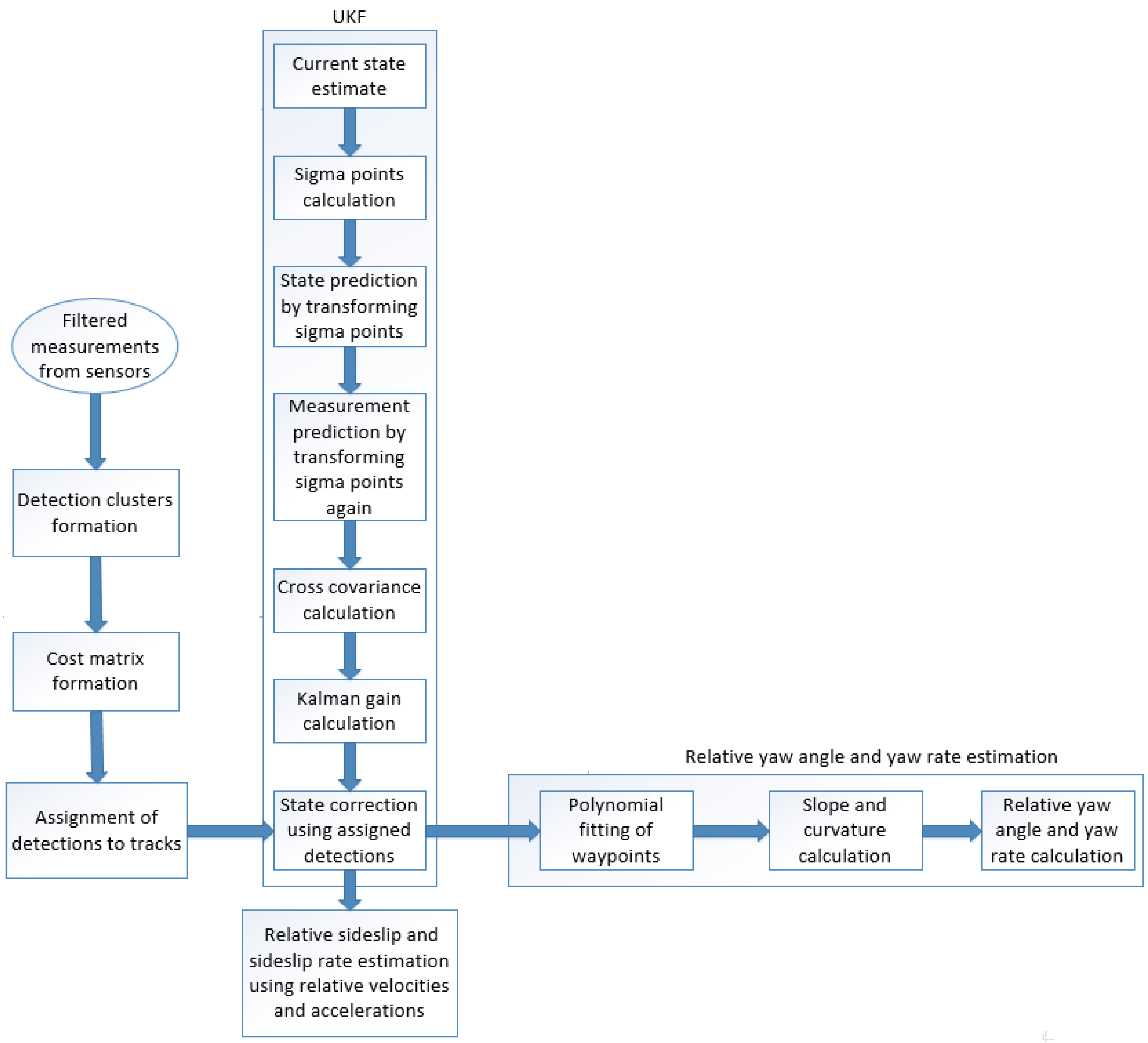

UKF
Current state estimate
Sigma points calculation
State prediction by transforming sigma points
Measurement prediction by transforming sigma points again
Cross covariance calculation
Kalman gain calculation
State correction using assigned detections
Filtered measurements from sensors
Detection clusters formation
Cost matrix formation
Assignment of detections to tracks
Relative sideslip and sideslip rate estimation using relative velocities and accelerations
Relative yaw angle and yaw rate estimation
Polynomial fitting of waypoints
Slope and curvature calculation
Relative yaw angle and yaw rate calculation

Figure 3.1: Flowchart of methods used for object tracking

3.1 Unscented Kalman Filter (UKF)

The UKF [46] is used for non-linear state transition and/or non-linear measurement models that can be assumed to have Gaussian distributions. As seen in the previous section, since the measurements are converted from spherical coordinates to Cartesian coordinates, the measurement model is non-linear. The Extended Kalman Filter (EKF) can also be used for non-linear state transition and/or non-linear measurement models. The EKF involves a linearization step that approximates the output Gaussian distribution using the first order Taylor series expansion of the non-linear state transition and/or measurement models. Thus the Jacobian of the state transition and/or measurement matrices has to be calculated about only one point i.e. the mean of the original distribution. In contrast, the

UKF can accurately calculate the mean and covariance of the posterior upto the third order of Taylor series expansion of any non-linear function. Therefore, the disadvantage of the EKF is that it is less accurate than the UKF but has the same computational cost [124]. The UKF uses the concept of Unscented Transform i.e. a non-linear transformation applied to a probability distribution to obtain another distribution represented by a finite set of parameters e.g. mean and covariance for Gaussian distributions. There are 3 steps: computing sigma points, predicting the next state, and updating the predicted state with the measurement.

3.1.1 Computing a set of sigma points/vectors

Ideally, to predict the state or measurement at the next time step through a non-linear state transition function or measurement function, if each point on the prior Gaussian distribution is transformed through the given non-linear function, the output distribution will not be Gaussian and also, the computation time will increase. To avoid these problems, the EKF calculates the entire output PDF (Probability Distribution Function) using linearization about one point i.e. the mean of the original PDF. To improve the accuracy while keeping the computational load the same, the UKF chooses more points in addition to the mean of the original PDF to calculate the output PDF. These points are chosen in a minimal way such that they are enough to accurately calculate the output PDF to the third degree of the Taylor series expansion. This minimal set of points (including the mean) are called sigma points/vectors as shown in Figure 3.2. Sigma vectors are chosen for the state distribution, process noise, and the measurement noise.

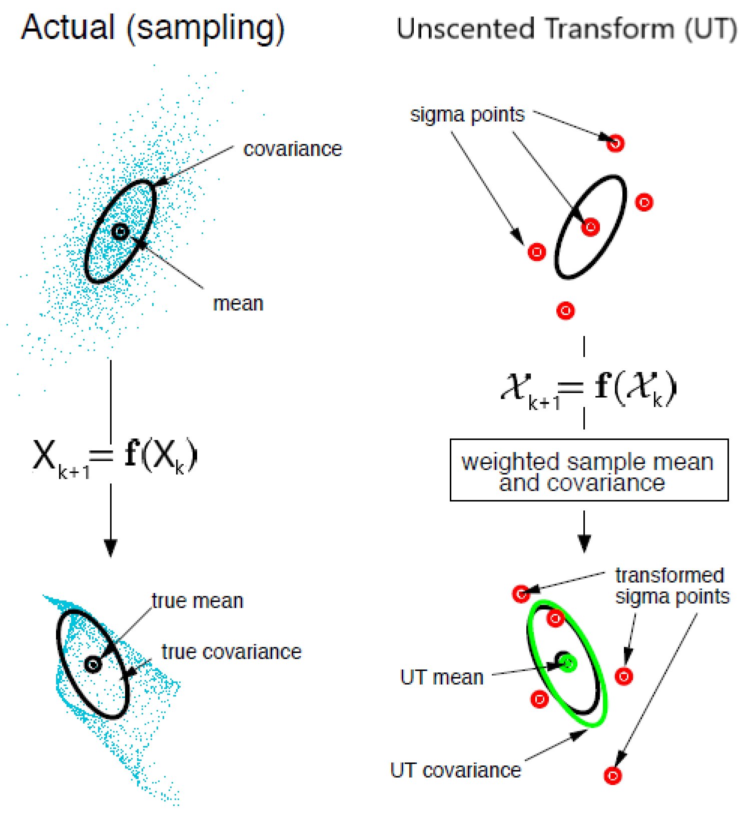


Figure 3.2: Unscented Transform (UT) [124]

The number of sigma points/vectors is chosen to be $2N+1$, where N is the dimension of the state vector [124]. The sigma vectors are given by:

$$\begin{aligned}
\mathcal{X}_{0,k} &= X_k \\
\mathcal{X}_{i,k} &= X_k + \left(\sqrt{(N+\lambda_N)\varSigma_{X,k}}\right)_i \quad \text{for} \quad i = 1,2,3\ldots N-1,N \\
\mathcal{X}_{i,k} &= X_k - \left(\sqrt{(N+\lambda_N)\varSigma_{X,k}}\right)_{i-N} \quad \text{for} \quad i = N+1,N+2,N+3\ldots 2N-1,2N
\end{aligned} \tag{3.1}$$

where k is the current time step, X_k is the current state, $\varSigma_{X,k}$ is the current state covariance matrix, $\left(\sqrt{(N+\lambda_N)\varSigma_{X,k}}\right)_i$ is the i^{th} row of the square root of the matrix $(N+\lambda_N)\varSigma_{X,k}$, and λ_N is a scaling parameter given by:

$$\lambda_N = \alpha^2(N+\kappa) - N \tag{3.2}$$

where α determines the spread of sigma points around the mean and is usually taken as a small positive value e.g. $1e-3$ [124], and κ is a secondary scaling parameter usually set to 0. From Equation 3.1, it can be seen that sigma points are chosen in such a way that they are spread around mean evenly in order to avoid a bias in the calculated parameters of the output PDF. Sigma points are also calculated for the process noise and measurement noise distributions as they act as inputs for the state transition function and measurement function respectively. Sigma points for the process noise are calculated as:

$$\begin{aligned}
\mathcal{W}_{0,k} &= W_k \\
\mathcal{W}_{i,k} &= W_k + \left(\sqrt{(N+\lambda_N)Q_k}\right)_i \quad \text{for} \quad i = 1,2,3\ldots N-1,N \\
\mathcal{W}_{i,k} &= W_k - \left(\sqrt{(N+\lambda_N)Q_k}\right)_{i-N} \quad \text{for} \quad i = N+1,N+2,N+3\ldots 2N-1,2N
\end{aligned} \tag{3.3}$$

where W_k is the process noise at the current time step, Q_k is the process noise covariance matrix at the current time step, and N, λ_N, α and κ are same as in Equation 3.2. Sigma points for the measurement noise are calculated as:

$$\begin{aligned}
\mathcal{V}_{0,k} &= V_k \\
\mathcal{V}_{i,k} &= V_k + \left(\sqrt{(M+\lambda_M)R_k}\right)_i \quad \text{for} \quad i = 1,2,3\ldots M-1,M \\
\mathcal{V}_{i,k} &= V_k - \left(\sqrt{(M+\lambda_M)R_k}\right)_{i-M} \quad \text{for} \quad i = M+1,M+2,M+3\ldots 2M-1,2M
\end{aligned} \tag{3.4}$$

where V_k is the measurement noise at the current time step, R_k is the measurement noise covariance matrix at the current time step, M is the dimension of the measurement vector, $\left(\sqrt{(M+\lambda_M)R_k}\right)_i$ is the i^{th} row of the square root of the matrix $(M+\lambda_M)R_k$, and λ_M is given by:

$$\lambda_M = \alpha^2(M+\kappa) - M \tag{3.5}$$

where α and κ are same as in Equation 3.2.

3.1.2 Predicting the next state

As shown in figure 3.2, to calculate the distribution of the predicted state at the next time step, each sigma vector $\mathcal{X}_{i,k}$ and $\mathcal{W}_{i,k}$ is transformed through the given non-linear state transition function:

$$\mathcal{X}_{i,k+1} = f(\mathcal{X}_{i,k}, \mathcal{W}_{i,k}) \tag{3.6}$$

The mean and covariance of the transformed sigma points are calculated to represent the distribution of the predicted state:

$$\begin{aligned}\hat{X}_{k+1} &= \sum_{i=0}^{2N} w_{N,i}^{(m)} \mathcal{X}_{i,k+1} \\ \hat{\Sigma}_{X,k+1} &= \sum_{i=0}^{2N} w_{N,i}^{(c)} (\mathcal{X}_{i,k+1} - \hat{X}_{k+1})(\mathcal{X}_{i,k+1} - \hat{X}_{k+1})^\top\end{aligned} \tag{3.7}$$

where the weights $w_{N,i}^{(m)}$ and $w_{N,i}^{(c)}$ are given by:

$$\begin{aligned}w_{N,0}^{(m)} &= \frac{\lambda_N}{N+\lambda_N} \\ w_{N,0}^{(c)} &= \frac{\lambda_N}{N+\lambda_N} + (1-\alpha^2+\beta) \\ w_{N,i}^{(m)} &= w_{N,i}^{(c)} = \frac{1}{2(N+\lambda_N)} \quad \text{for} \quad i = 1,2,3\ldots 2N-1, 2N\end{aligned} \tag{3.8}$$

where β is used to incorporate prior knowledge of the distribution e.g. for Gaussian distributions, $\beta = 2$ is ideal [124].

3.1.3 Predicting the next measurement

For the next step, the predicted measurement has to be calculated from the predicted state so that the predicted state can be updated/corrected using the actual measurement and the Kalman gain. To calculate the distribution of the predicted measurement at the next time step, the already transformed sigma points in the previous step and the measurement noise sigma points are transformed through the given non-linear measurement function:

$$\mathcal{Z}_{i,k+1} = h(\mathcal{X}_{i,k+1}, \mathcal{V}_{i,k+1}) \tag{3.9}$$

The mean and covariance of the further transformed sigma points is calculated to represent the distribution of the predicted measurement:

$$\begin{aligned}\hat{Z}_{k+1} &= \sum_{i=0}^{2N} w_{M,i}^{(m)} \mathcal{Z}_{i,k+1} \\ \hat{\Sigma}_{Z,k+1} &= \sum_{i=0}^{2N} w_{M,i}^{(c)} (\mathcal{Z}_{i,k+1} - \hat{Z}_{k+1})(\mathcal{Z}_{i,k+1} - \hat{Z}_{k+1})^\top\end{aligned} \tag{3.10}$$

where the weights $w_{M,i}^{(m)}$ and $w_{M,i}^{(c)}$ are given by:

$$\begin{aligned} w_{M,0}^{(m)} &= \frac{\lambda_M}{M+\lambda_M} \\ w_{M,0}^{(c)} &= \frac{\lambda_M}{M+\lambda_M} + (1-\alpha^2+\beta) \\ w_{M,i}^{(m)} &= w_{M,i}^{(c)} = \frac{1}{2(M+\lambda_M)} \quad \text{for} \quad i = 1, 2, 3 \ldots 2M-1, 2M \end{aligned} \tag{3.11}$$

where α and β are same as in Equation 3.8.

3.1.4 Calculating the cross co-relation matrix and the Kalman gain

To calculate the Kalman gain, the cross correlation matrix between the predicted state and the predicted measurement is calculated:

$$\hat{\Sigma}_{XZ,k+1} = \sum_{i=0}^{2N} w_{N,i}^{(c)} (\mathcal{X}_{i,k+1} - \hat{X}_{k+1})(\mathcal{Z}_{i,k+1} - \hat{Z}_{k+1})^\top \tag{3.12}$$

The cross correlation matrix is used to calculate the Kalman gain:

$$\mathcal{K} = \hat{\Sigma}_{XZ,k+1} \hat{\Sigma}_{Z,k+1}^{-1} \tag{3.13}$$

3.1.5 Updating the predicted state

Using the Kalman gain, and the mean and covariance of the predicted state distribution obtained from Equation 3.10, the mean and covariance of the distribution of the predicted state obtained from Equation 3.7 are corrected:

$$\begin{aligned} X_{k+1} &= \hat{X}_{k+1} + \mathcal{K}(Z_{k+1} - \hat{Z}_{k+1}) \\ \Sigma_{X,k+1} &= \hat{\Sigma}_{X,k+1} - \mathcal{K}\hat{\Sigma}_{Z,k+1}\mathcal{K}^\top \end{aligned} \tag{3.14}$$

where Z_{k+1} is the actual measurement obtained from the sensors at the next time step.

3.1.6 Motion modelling

To predict the next state of a target and the next measurement from the target, the motions of the target in the x, y, and z coordinates have to be modelled. Many dynamic models can be used for this purpose [36]. Since the UKF works in discrete time as every sensor has a sampling time i.e. the measurements are in discrete time, discrete time state space models are widely used to model the motion:

$$\begin{aligned} X_{k+1} &= f(X_k, U_k, W_k) \\ Z_k &= h(X_k) + V_k \end{aligned} \tag{3.15}$$

where X_k is the current state vector with dimension N having the positions, velocities and accelerations in the x, y and z axes, U_k is the current control input, W_k is the process noise, V_k is the measurement noise, Z_k is the measurement vector with dimension M, $f : \mathbb{R}^N \to \mathbb{R}^N$ is the state transition function, and $h : \mathbb{R}^N \to \mathbb{R}^M$ is the measurement function. In most motions in the real world, the coordinates are coupled. For simplicity, maneuvering target motion models assume the coordinates to be uncoupled [36]. Maneuvering target motion models can be divided into three categories:

1. **White noise models**: The model input U_k is modelled as a white noise process e.g. constant velocity and constant acceleration models.

2. **Markov process models**: The model input U_k is modelled as a Markov process e.g. Singer acceleration model with zero mean.

3. **Semi-Markov jump process models**: The model input U_k is modelled as a semi-Markov jump process e.g. Singer acceleration model with non-zero mean.

White noise models i.e. the constant velocity and constant acceleration models have been used for this PDEng project because they are reasonably accurate as shown in Chapter 4, and simple i.e. they assume uncoupled motion in the x, y, and z coordinates and therefore have lesser number of parameters for tuning than motion models which assume coupled coordinates.

3.1.6.1 Constant velocity (CV) model

The CV model is the simplest model among the models used for maneuvering target tracking because it assumes that the velocity of the target is constant and the acceleration is a white noise process. If the state vector is given by $X_k = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k & z_k & \dot{z}_k \end{bmatrix}^\top$ and the sampling time is T_s, then the state space equation for the motion model is [36]:

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \\ z_{k+1} \\ \dot{z}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T_s & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T_s & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \\ z_k \\ \dot{z}_k \end{bmatrix} + \begin{bmatrix} \frac{T_s^2}{2} & 0 & 0 \\ T_s & 0 & 0 \\ 0 & \frac{T_s^2}{2} & 0 \\ 0 & T_s & 0 \\ 0 & 0 & \frac{T_s^2}{2} \\ 0 & 0 & T_s \end{bmatrix} \begin{bmatrix} q_{x,k} & 0 & 0 \\ 0 & q_{y,k} & 0 \\ 0 & 0 & q_{z,k} \end{bmatrix} \begin{bmatrix} \frac{T_s^2}{2} & 0 & 0 \\ T_s & 0 & 0 \\ 0 & \frac{T_s^2}{2} & 0 \\ 0 & T_s & 0 \\ 0 & 0 & \frac{T_s^2}{2} \\ 0 & 0 & T_s \end{bmatrix}^\top \tag{3.16}$$

where q_x, q_y and q_z are the zero mean white noise accelerations in the x, y and z directions respectively.

3.1.6.2 Constant acceleration model (CA) model

The CA model assumes that the acceleration of the target is constant and the rate of change of acceleration i.e. jerk is a white noise process. If the state vector is given by $X_k = \begin{bmatrix} x_k & \dot{x}_k & \ddot{x}_k & y_k & \dot{y}_k & \ddot{y}_k & z_k & \dot{z}_k & \ddot{z}_k \end{bmatrix}^\top$ and the sampling time is T_s, then the state space equation for the motion model is [36]:

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ \ddot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \\ \ddot{y}_{k+1} \\ z_{k+1} \\ \dot{z}_{k+1} \\ \ddot{z}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T_s & \frac{T_s^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & T_s & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T_s & \frac{T_s^2}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T_s & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & T_s & \frac{T_s^2}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \\ y_k \\ \dot{y}_k \\ \ddot{y}_k \\ z_k \\ \dot{z}_k \\ \ddot{z}_k \end{bmatrix} + \begin{bmatrix} \frac{T_s^3}{6} & 0 & 0 \\ \frac{T_s^2}{2} & 0 & 0 \\ T_s & 0 & 0 \\ 0 & \frac{T_s^3}{6} & 0 \\ 0 & \frac{T_s^2}{2} & 0 \\ 0 & T_s & 0 \\ 0 & 0 & \frac{T_s^3}{6} \\ 0 & 0 & \frac{T_s^2}{2} \\ 0 & 0 & T_s \end{bmatrix} \begin{bmatrix} q_{x,k} & 0 & 0 \\ 0 & q_{y,k} & 0 \\ 0 & 0 & q_{z,k} \end{bmatrix} \begin{bmatrix} \frac{T_s^3}{6} & 0 & 0 \\ \frac{T_s^2}{2} & 0 & 0 \\ T_s & 0 & 0 \\ 0 & \frac{T_s^3}{6} & 0 \\ 0 & \frac{T_s^2}{2} & 0 \\ 0 & T_s & 0 \\ 0 & 0 & \frac{T_s^3}{6} \\ 0 & 0 & \frac{T_s^2}{2} \\ 0 & 0 & T_s \end{bmatrix}^\top \tag{3.17}$$

where q_x, q_y and q_z are the zero mean white noise jerks in the x, y and z directions respectively.

3.2 Measurement processing

This section explains the measurement processing block shown in Figure 3.1.

3.2.1 Measurement filtering and clustering

After measurements have been received from the sensors, they are first filtered e.g. if the SNR (Signal-to-Noise) value of a measurement from the radar is below 15 dB, it is not taken into account. The values of other thresholds are described in Chapter 5. Filtered measurements from the radar have to be clustered together because one object can reflect multiple beams from the radar as shown by the red dots in Figure 3.3. The measurements from the camera are not clustered because the camera already provides measurements based on a bounding box placed on the object image.

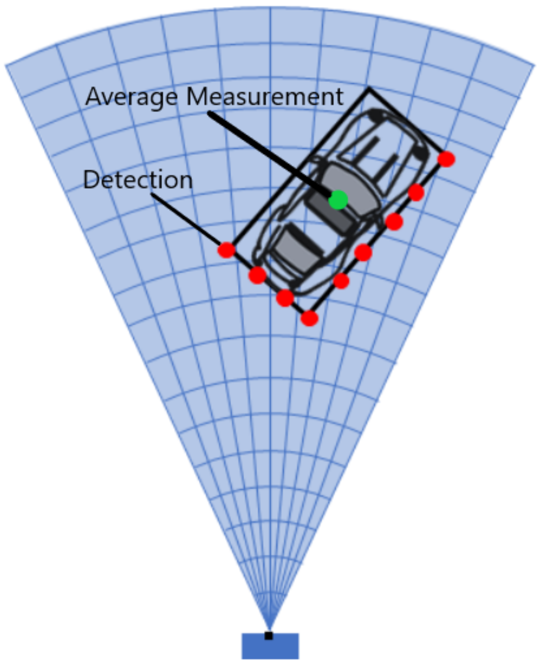


Figure 3.3: Multiple radar detections from a single object [125]

Points within a user defined threshold distance of each other are clustered together by taking a simple average of the measured Cartesian coordinates as shown by the green dot in Figure 3.3. For simulations, the clustering threshold is calculated using dimensions of the target in Chapter 4 and for real life scenarios, it is tuned using Latin Hypercube Sampling (LHS) as explained later in this chapter.

3.2.2 Cost matrix formation and assignment of detections to tracks

The cost matrix for each sensor is a matrix with current tracks as rows and current detections from the sensor as columns. Each element e_{ij} of the matrix is a metric e.g. the probability of the j^{th} detection belonging to the i^{th} track or the normalized distance of the j^{th} detection from the i^{th} track. This metric is used to decide whether to assign a detection to the track or not. For this PDEng project, a heuristic (non-probabilistic) tracker has been used. Therefore the normalized distance is chosen as the metric. The normalized distance d_n [106] at the current time step k is given by:

$$d_{n,k} = \sqrt{Z_{res,k}^{\top} \Sigma_{Z,k}^{-1} Z_{res,k} + \log(|\Sigma_{Z,k}|)} \tag{3.18}$$

where $\Sigma_{Z,k}$ is the residual covariance matrix and $Z_{res,k}$ is the residual vector given by:

$$Z_{res,k} = Z_k - \hat{Z}_k \tag{3.19}$$

where Z_k is the actual measurement vector and $\hat{Z}_k$ is the predicted measurement vector. To decide whether it is possible to assign a detection to a track and vice-versa or not i.e. the assignment threshold acts as a gating value. For example, if the detection's normalized distance from the track is less than the assignment threshold, then it is possible to assign the detection to the track. This means that a measurement will be taken into account if its normalized distance to a track is less than the threshold, but it may or may not be assigned to the track. For simulations, the assignment threshold is manually tuned as shown in Chapter 4 and for real life scenarios, it is tuned using Latin Hypercube Sampling (LHS) as explained later in this chapter.

After gating has been done using the assignment threshold, the Munkres algorithm/ Hungarian method [126] is used to assign detections to tracks and vice-versa. This method is a solution to the assignment problem. Only one detection can be assigned to each track and only one track can be assigned to each detection i.e. only one assignment is possible in each row and column. This is done in a way such that the overall cost of assignment is minimized. Unassigned tracks are coasted or deleted, and unassigned detections are used to initialize new tracks as shown in Chapter 4.

3.3 Motion tracking of an extended body

In the motion models used in the UKF block, the target is assumed to be a point mass. In real life, the target is always an extended body. This section explains the methods to track the motions of the extended body inside the the relative sideslip and relative sideslip rate estimation block, and the relative yaw angle and relative yaw rate estimation block in Figure 3.1.

3.3.1 Relative sideslip and sideslip rate calculation

The bicycle model [127] is widely used as a representation of a four-wheeled vehicle. It has 3 degrees of freedom i.e. velocities along the x and y axes, and rotation about the z axis as shown in Figure 3.4.

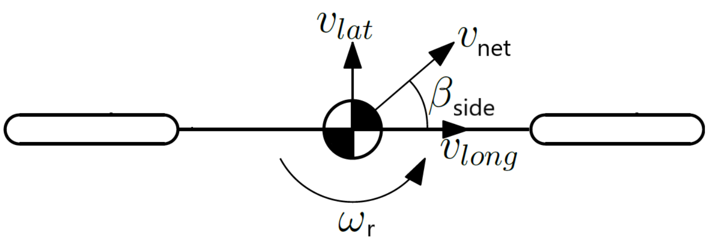


Figure 3.4: 3 DOF bicycle model

v_{long} and v_{lat} are the velocities in the global X and Y directions respectively, v_{net} is the net velocity in the global frame, $\beta_{side} = \tan^{-1}\frac{v_{lat}}{v_{long}}$ is the sideslip in the global frame, and ω_r is the yaw rate in the global frame. The equations of motion can be written as:

$$\begin{aligned} a_{long} &= \frac{F_{long}}{m_v} \\ a_{lat} &= \frac{F_{lat}}{m_v} \\ \alpha_r &= \frac{M_{rot}}{I_{rot}} \end{aligned} \tag{3.20}$$

where m_v is the mass of the vehicle, F_{long} and F_{lat} are forces in the global X and Y directions respectively, a_{long} and a_{lat} are accelerations in the global X and Y directions respectively, M_{rot} and I_{rot} are the rotational moment and mass moment of inertia about the global Z axis respectively, and α_r is the rotational acceleration about the global Z axis. The leading vehicle being followed by the ego vehicle are represented by the bicycle model as shown in Figure 3.5. The net tracked relative velocity of the target vehicle in the coordinate frame of the ego vehicle is v_{rel}. This relative velocity can be resolved into its components v_x and v_y along the x and y axes of the ego vehicle frame.

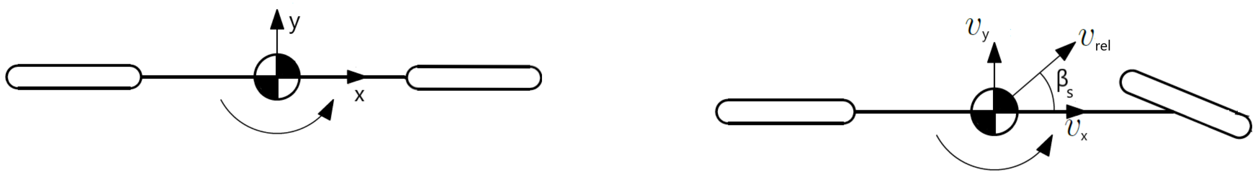


Figure 3.5: The lead vehicle being followed by the ego vehicle

The relative sideslip β_s of the target vehicle with respect to the ego vehicle can be calculated as:

$$\beta_{s,k} = \tan^{-1}\frac{v_{y,k}}{v_{x,k}} \tag{3.21}$$

where k is the current time step. If the tracked relative accelerations of the target vehicle along the x and y axes of the ego vehicle frame are a_x and a_y respectively, and T_s is the sampling time for the measurements being obtained in discrete time, then the relative sideslip of the target vehicle with respect to the ego vehicle at the next time step is:

$$\beta_{s,k+1} = \tan^{-1}\frac{v_y + a_y T_s}{v_x + a_x T_s} \tag{3.22}$$

The rate of change of the relative sideslip is calculated using the difference between the relative sideslip at the next time step and the relative sideslip at the current time step:

$$\gamma_{s,k} = \frac{\tan^{-1}\frac{v_y + a_y T_s}{v_x + a_x T_s} - \tan^{-1}\frac{v_y}{v_x}}{T_s} \tag{3.23}$$

For tracking the relative sideslip rate, only the constant acceleration model can be used because the tracked relative acceleration of the target is required.

3.3.2 Relative yaw angle and yaw rate calculation

The path of the leader vehicle can be traced by least squares fitting of its tracked positions with respect to the ego vehicle on a third degree polynomial. A third degree polynomial is the simplest polynomial with continuous curvature [128]. Highway corners are also designed with gradually changing curvatures to ensure continuity [129]. The cubic polynomial is of the form:

$$y_k = c_1 x_k^3 + c_2 x_k^2 + c_3 x_k + c_4 \tag{3.24}$$

where k is the current time step, c_1, c_2, c_3, and c_4 are the polynomial coefficients, and x_k and y_k are the relative coordinates of the lead vehicle in the ego vehicle frame at the current time step k as shown in Figure 3.6.

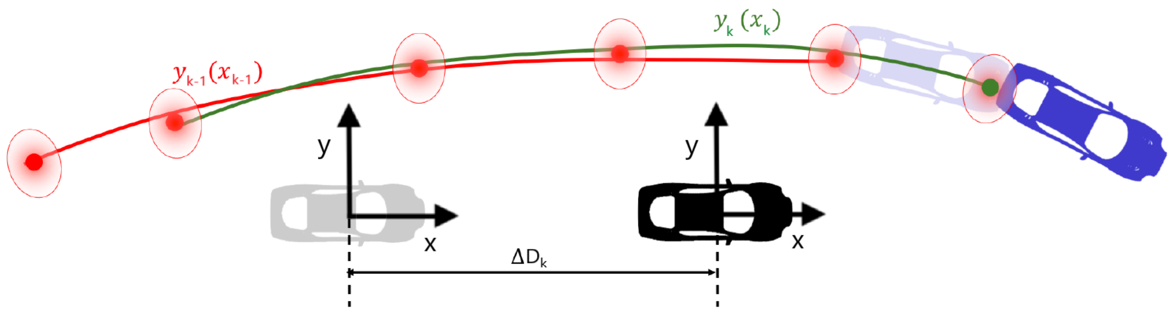


Figure 3.6: Polynomial paths at two consecutive time instances [128]

ΔD_k is the difference between the global X coordinate of the ego vehicle frame at the current time step and the global X coordinate of the ego vehicle frame at the previous time step. The red path is the path of the leading vehicle traced by the ego vehicle at the previous time step, and the green path is the path of the leading vehicle traced by the ego vehicle at the current time step. The idea is to fit both the red path and green path on their respective third degree polynomials by minimizing the respective mean squared errors. For each polynomial, the points used for polynomial formation i.e. the previous and current relative positions of the leading vehicle, need to be transformed to the frame of the ego vehicle which is moving most of the time (except e.g. stopping at traffic lights).

Since the ego vehicle frame is moving, the slope and curvature of the polynomial obtained using the green path in Figure 3.6 at its origin is not equal to the slope and curvature of the polynomial obtained using the red path at the same point. This results in discontinuous path tracing. To ensure continuity of the polynomial obtained at the current time step at its origin, constraints can be placed on the slope and the curvature of the current polynomial at its origin and the slope and curvature at the same point of the polynomial obtained at the previous time step i.e. the curvature and slope at the first point of the green path, and the slope and curvature at the second point of the red path can be constrained to be equal. The constraint on the heading angle ϕ can be written as:

$$\begin{aligned}
\phi_{k-1}|_{x=-\Delta D_k} &= \phi_k|_{x=0} \\
\implies \tan^{-1}\left(\frac{dy_{k-1}}{dx_{k-1}}\right)_{x=-\Delta D_k} &= \tan^{-1}\left(\frac{dy_k}{dx_k}\right)_{x=0} \\
\implies \left(\frac{dy_{k-1}}{dx_{k-1}}\right)_{x=-\Delta D_k} &= \left(\frac{dy_k}{dx_k}\right)_{x=0} \\
\implies \left(3c_1x_{k-1}^2 + 2c_2x_{k-1} + c_3\right)_{x=-\Delta D_k} &= \left(3c_1x_k^2 + 2c_2x_k + c_3\right)_{x=0}
\end{aligned} \tag{3.25}$$

The constraint on the curvature κ_c can be written as:

$$\begin{aligned}
\kappa_{c,k-1}|_{x=-\Delta D_k} &= \kappa_{c,k}|_{x=0} \\
\implies \frac{\left(\frac{d^2y_{k-1}}{dx_{k-1}^2}\right)_{x=-\Delta D_k}}{\left(1+\left(\frac{dy_{k-1}}{dx_{k-1}}\right)^2\right)^{\frac{3}{2}}_{x=-\Delta D_k}} &= \frac{\left(\frac{d^2y_k}{d^2x_k}\right)_{x=0}}{\left(1+\left(\frac{dy_k}{dx_k}\right)^2\right)^{\frac{3}{2}}_{x=0}} \\
\implies \frac{\left(6c_1x_{k-1}+2c_2\right)_{x=-\Delta D_k}}{\left(1+\left(3c_1x_{k-1}^2+2c_2x_{k-1}+c_3\right)^2\right)^{\frac{3}{2}}_{x=-\Delta D_k}} &= \frac{\left(6c_1x_k+2c_2\right)_{x=0}}{\left(1+\left(3c_1x_k^2+2c_2x_k+c_3\right)^2\right)^{\frac{3}{2}}_{x=0}}
\end{aligned} \tag{3.26}$$

In addition to these constraints, a point can be weighted with weight based on its distance from the ego vehicle at time step k to give preference to the points that are closer since the curvature of the fitted polynomial at any point is affected more by points that are in immediate vicinity of the point than those points that are further away from the point. The weight w_i for a point at a distance D_i from the ego vehicle can be calculated for the current time step k as:

$$w_{i,k} = \frac{1}{D_{i,k}} \tag{3.27}$$

If P points are fitted on to the polynomial, then the coefficients c_1, c_2, c_3, and c_4 of the polynomial can be calculated by solving a system of P equations through formation of a Van der Monde matrix [130] with constraints and weights given by Equation 3.25, Equation 3.26, and Equation 3.27. After estimating the curvature of the polynomial at a point, the yaw angle and yaw rate for steady state cornering can be estimated as:

$$\begin{aligned}
\phi_k &= \tan^{-1}\left(\frac{dy_k}{dx_k}\right)_{x=x_l} \\
\implies \dot{\phi}_k &= \frac{\left(\frac{d^2y_k}{dx_k^2}\right)_{x=x_l}}{\left(1+\left(\frac{dy_k}{dx_k}\right)^2\right)_{x=x_l}} \times \left(\frac{dx_k}{dt}\right)_{x=x_l}
\end{aligned} \tag{3.28}$$

where x_l is the last known relative x position of the target vehicle. Assuming that $\frac{dx_k}{dt} = v_{x,k}$ is the tracked relative velocity of the target vehicle in the x direction, the relative yaw rate is:

$$\dot{\phi}_k = \frac{\left(\frac{d^2 y_k}{dx_k^2}\right)_{x=x_l}}{\left(1+\left(\frac{dy_k}{dx_k}\right)^2\right)_{x=x_l}} \times \left(v_{x,k}\right)_{x=x_l} \tag{3.29}$$

3.4 Latin Hypercube Sampling (LHS)

In real life scenarios, the number of targets is not fixed and can vary sharply. For example, during daytime on weekdays, the amount of traffic in the Netherlands is usually high. In contrast, on weekends, the roads are quite empty. For such scenarios, manual tuning of tracker parameters like the clustering threshold or assignment threshold is difficult. The difficulty increases when the number of parameters increases e.g. tuning of measurement noise covariance matrix. Latin Hypercube Sampling (LHS) [131] can be used to tune the values of multiple parameters as shown in Chapter 5. LHS is a method of generating near random samples of parameter values in a multi-dimensional sample space. It used as a substitute to normal random sampling in Monte Carlo methods to speed up simulations and obtain a reasonably accurate result. A Latin square is a 2D square having one sample in each row and column e.g. a Latin square for 2 parameters and 5 samples is shown in Figure 3.7.

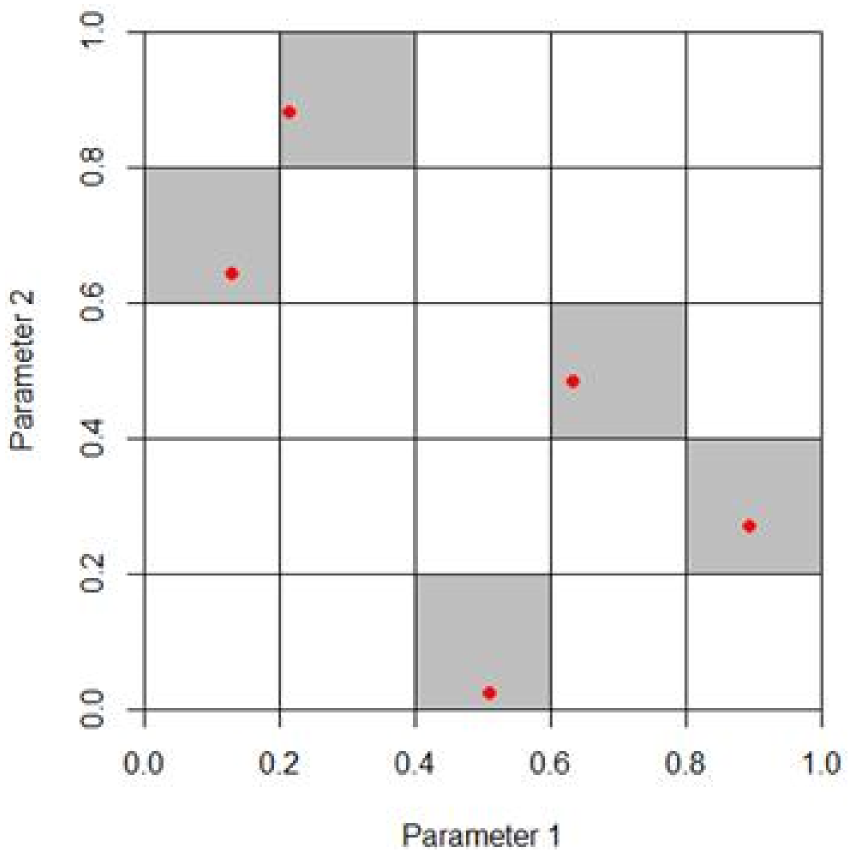


Figure 3.7: A Latin square for 2 parameters and 5 samples (figure provided by Peter Heuberger [132])

LHS extends the use of the Latin Square to a hypercube i.e. a cube having more than 3 dimensions.

For example, if there are 5 parameters and 100 samples need to be generated, a five dimensional hypercube will be formed where each dimension of the hypercube is represented by one parameter. Each parameter range will be normalized and split into 100 divisions. Only one sample will be selected in each combination of the respective divisions of the 5 dimensions. The benefit of LHS, especially in higher dimensions, is that the number of samples to obtain a reasonable covering of the sample space is much lower than in normal random sampling.

4 Simulation of object tracking with one radar, one camera, GNSS and IMU

In this chapter, the developed algorithm has been implemented in a simulation environment. The limitation of the RTT (Real Time Target) machine is that it supports up to Simulink Real Time R2017b. MATLAB R2017b was used to develop the tracker and run simulations. A simple combination of sensors has been simulated i.e. the sensor fusion of 1 radar, 1 camera, GNSS and IMU. Three simulation scenarios have been tested. Each scenario has a lead Twizy being followed by the ego Twizy and the lead Twizy being overtaken by a third Twizy that starts between the lead Twizy and the ego Twizy. The following sections describe the design and working of the tracker, different simulation scenarios for testing it, and the performance of the tracker in the tested scenarios.

4.1 Tracker development in MATLAB

The automated driving toolbox in MATLAB R2017b has a built-in *multiObjectTracker* system object [133]. This system object has been used as the tracker for this PDEng project. MATLAB system objects [134] are built for simulating dynamical systems whose input changes with time and whose output depends on the previous states of the system. System objects store previous states and use them in the next computational step i.e. they are optimized for iterative computations that process streams of data in segments e.g. sensor data. The advantage is that large amounts of data do not have to be stored in memory. The choices made for components of the tracker are:

1. **Filter**: The UKF (Unscented Kalman Filter) was chosen as described in Chapter 3. The choices for UKF parameters are:
    - **Alpha** (α): It is taken to be $1e-3$ [124].
    - **Beta** (β): It is taken to be 2 for Gaussian distributions [124].
    - **Kappa** (κ): It is taken to be 0 [124].
    - **State**: As shown in Chapter 3, it is a 6×1 vector i.e. $X_k = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k & z_k & \dot{z}_k \end{bmatrix}^\top$ for the constant velocity model and a 9×1 vector i.e. $X_k = \begin{bmatrix} x_k & \dot{x}_k & \ddot{x}_k & y_k & \dot{y}_k & \ddot{y}_k & z_k & \dot{z}_k & \ddot{z}_k \end{bmatrix}^\top$ for the constant acceleration model.
    - **State covariance matrix**: The covariance matrix of the state is a 6×6 matrix i.e. $\Sigma_{X,k} = diag(\begin{bmatrix} \sigma^2_{x,k} & \sigma^2_{\dot{x},k} & \sigma^2_{y,k} & \sigma^2_{\dot{y},k} & \sigma^2_{z,k} & \sigma^2_{\dot{z},k} \end{bmatrix})$ for the constant velocity model and a 9×9 matrix i.e. $\Sigma_{X,k} = diag(\begin{bmatrix} \sigma^2_{x,k} & \sigma^2_{\dot{x},k} & \sigma^2_{\ddot{x},k} & \sigma^2_{y,k} & \sigma^2_{\dot{y},k} & \sigma^2_{\ddot{y},k} & \sigma^2_{z,k} & \sigma^2_{\dot{z},k} & \sigma^2_{\ddot{z},k} \end{bmatrix})$ for the constant acceleration model. $\sigma_{var,k}$ denotes the variance of a variable var at the k^{th} time step.

- **State transition function**: The constant velocity model or constant acceleration model was chosen as the state transition model at a time.
- **Process noise**: The process noise is assumed to be additive for both the CV and CA models because it is simple yet accurate [36]. The process noise covariance matrix is the same size as the state covariance matrix.
- **Measurement function**: The constant velocity or constant acceleration measurement function was used to calculate the predicted measurement from the predicted state.
- **Measurement noise**: The measurement noise is assumed to be additive because it is simple yet accurate [36]. The measurement noise covariance matrix is the same size as the state covariance matrix except that the order of the elements is different i.e. for the constant velocity model, $\Sigma_{Z,k} = diag(\begin{bmatrix}\sigma^2_{x,k} & \sigma^2_{y,k} & \sigma^2_{z,k} & \sigma^2_{\dot{x},k} & \sigma^2_{\dot{y},k} & \sigma^2_{\dot{z},k}\end{bmatrix})$ and for the constant acceleration model, $\Sigma_{Z,k} = diag(\begin{bmatrix}\sigma^2_{x,k} & \sigma^2_{y,k} & \sigma^2_{z,k} & \sigma^2_{\dot{x},k} & \sigma^2_{\dot{y},k} & \sigma^2_{\dot{z},k} & \sigma^2_{\ddot{x},k} & \sigma^2_{\ddot{y},k} & \sigma^2_{\ddot{z},k}\end{bmatrix})$.

2. **Clustering threshold**: The clustering threshold is defined based on the dimensions of the Twizy which are given in Table 4.1.

Table 4.1: Dimensions of Renault Twizy

Dimension	Value [m]
Length	2.34
Width	1.405
Height	1.42

If the value of the clustering threshold is too low, then multiple clusters will be obtained from the same object. Similarly, if the value is too high, then one cluster will be obtained from multiple objects. If the Twizy is assumed as a cuboid, then the the clustering threshold can be a value approximately around the length of the diagonal of the cuboid i.e. $\sqrt{2.34^2 + 1.405^2 + 1.42^2} = 3.0767$ m since radar reflections can be spread across the three dimensions of the Twizy. In simulation, the value was taken to be 3.5 m. For real world scenarios, the value was found out using Latin Hypercube Sampling (LHS) as shown in Chapter 5.

3. **Assignment threshold**: For the simulated scenarios, the value of this property was manually tuned because each scenario only has 2 targets. The value was chosen as 40. If the value was too low e.g. 10 or 20, multiple tracks were formed out of the same target because measurements from the same target are not assigned to the existing track for the object due to the low threshold, and these unassigned measurements form a new track. There is no upper limit on this value because for larger values, measurements from both targets will always be taken into account to decide whether to associate them to existing tracks or not. For real world scenarios, the value was tuned using Latin Hypercube Sampling (LHS) as explained in Chapter 3.

4. **Confirmation parameters**: The status of a track is changed from tentative to confirmed when at least A detections are assigned to the track during the first B updates after the track is initialized as a tentative track. A must be less than or equal to B. To ensure that false positive detections do not lead to confirmed tracks, A is chosen to be equal to B, and the value is chosen to be 5 since 5 sampling intervals are sufficient to differentiate between false positive detections and true positive detections.

5. **Number of coasting updates**: A track is coasted when no detections are assigned to the track after its next state has been predicted once or more. If the number of coasting steps exceeds this threshold, the track is deleted. The value of this property cannot be too high because if a track is coasted for too long, then measurements not belonging to the same object can be assigned to the track e.g. even if the object is no longer in the FOV of the sensors, the track is still being coasted and measurements from a new object can be assigned to the coasted track. Thus, instead of initializing a new track and deleting the old track, the old track is continued. Similarly, the value cannot be too low because if measurements are not assigned to a track when they should have been, the track will be deleted too soon. Keeping these points in mind, the value was chosen as 5.

4.1.1 Inputs to the tracker

The inputs to the tracker are objects from the built-in *objectDetection* class in MATLAB. These objects are formed using the measurements from the radar or the camera. The object detections record the time, the measurement, the measurement noise, object class and the parameters of the sensor making the measurement as shown in Table 4.2. There are 5 identifiers of the class of an object i.e. 0 for unknown, 1 for car, 2 for bicycle, 3 for pedestrian, 4 for road sign, and 5 for truck. Since the radar cannot detect the class of an object, it always assigns 0 to the value of the object class. The radar and the camera are aligned parallel to the ego vehicle frame and are fixed to it i.e. have zero velocity with respect to the ego vehicle frame.

Table 4.2: Sensor parameters in simulation

Parameter	Radar	Camera
Sensor index	1	2
Location	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$	$\begin{bmatrix}-1.11 & 0 & 0.9\end{bmatrix}^\top$
Orientation	$diag(\begin{bmatrix}1 & 1 & 1\end{bmatrix})$	$diag(\begin{bmatrix}1 & 1 & 1\end{bmatrix})$
Velocity	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
Update frequency	14 Hz	7 Hz
Maximum detections	25	25

The measurement is initialized as a 6×1 vector i.e. $Z_k = \begin{bmatrix}x_k & y_k & z_k & \dot{x}_k & \dot{y}_k & \dot{z}_k\end{bmatrix}^\top$ for the CV model and CA model. The CA model sets the accelerations measurements to 0 because neither the radar nor the camera measure the radial accelerations. If d_{th} is the clustering threshold, the value of the measurement noise covariance matrix for the constant velocity model is taken as $\Sigma_{Z,k} = diag(\begin{bmatrix}d_{th}^2 & d_{th}^2 & d_{th}^2 & 5 \times d_{th}^2 & 5 \times d_{th}^2 & 5 \times d_{th}^2\end{bmatrix})$, and for the constant acceleration model as $\Sigma_{Z,k} = diag(\begin{bmatrix}d_{th}^2 & d_{th}^2 & d_{th}^2 & 5 \times d_{th}^2 & 5 \times d_{th}^2 & 5 \times d_{th}^2 & 10 \times d_{th}^2 & 10 \times d_{th}^2 & 10 \times d_{th}^2\end{bmatrix})$. The variance in the position measurements is taken to be equal to the clustering threshold because the measurements are spread across the dimensions of the Twizy. The variances in velocity and acceleration measurements are taken to be higher because velocities and accelerations are first order and second order derivatives of the positions respectively. For real world scenarios, the measurement noise covariance matrix is tuned using Latin Hypercube Sampling (LHS) as explained in Chapter 3.

4.1.2 Outputs of the tracker

The outputs of the tracker are tentative tracks and confirmed tracks. Both tentative and confirmed tracks record the time of the track output, a unique track ID, the state of the target, the state covariance matrix, the status of the track i.e. coasted and/or confirmed, and the class of the target.

4.1.3 Working procedure of the tracking algorithm

This sub-section describes the step-by-step working procedure of the tracking algorithm.

4.1.3.1 Pre-processing measurements

After a measurement has been received from a sensor, it is used to make an *objectDetection* object in accordance with the parameters of the sensor as shown in Table 4.2. Measurements are not filtered in simulations because there are only 2 targets. There are a maximum of 25 detections from the radar at any update point e.g. at $t = 0$ s, there are 25 *objectDetection* objects in total due to multiple reflections from the lead Twizy and third Twizy in any of the three tested scenarios. Since there are 2 vehicles which act as radar reflectors, two detection clusters are obtained from the radar after measurement clustering. Also, two detection clusters are obtained from the camera. So at $t = 0$ s, 4 detection clusters are input into the tracker.

4.1.3.2 Track initialization

At $t = 0$ s, there are no existing tracks, so a track is initialized from the first detection cluster that is obtained from the radar in this case. To see if any of the three other detection clusters should also be assigned to the same track, a 1×3 cost vector is calculated where each element of the vector is the normalized distance between the track and the detection cluster. If the value of this element is lower than the assignment threshold, then that detection cluster is assigned to the track. In this case, as shown in Table 4.3, the first camera detection cluster that comes from the same vehicle as detected by the radar will be assigned to the initialized track and the other two detection clusters will not be assigned to it.

Table 4.3: Cost vector at t = 0 s

Track ID	Detection cluster 2	Detection cluster 3	Detection cluster 4
1	Unassigned	Assigned	Unassigned

From the two remaining unassigned detection clusters, a new track is initialized from detection cluster 2 i.e. the remaining radar detection cluster. The last remaining camera detection cluster is assigned to the initialized track after checking its normalized distance to it. In this way, two tracks are initialized for the two targets.

4.1.3.3 Track updating and/or coasting

The track formation and updating procedure is shown in Figure 4.1.

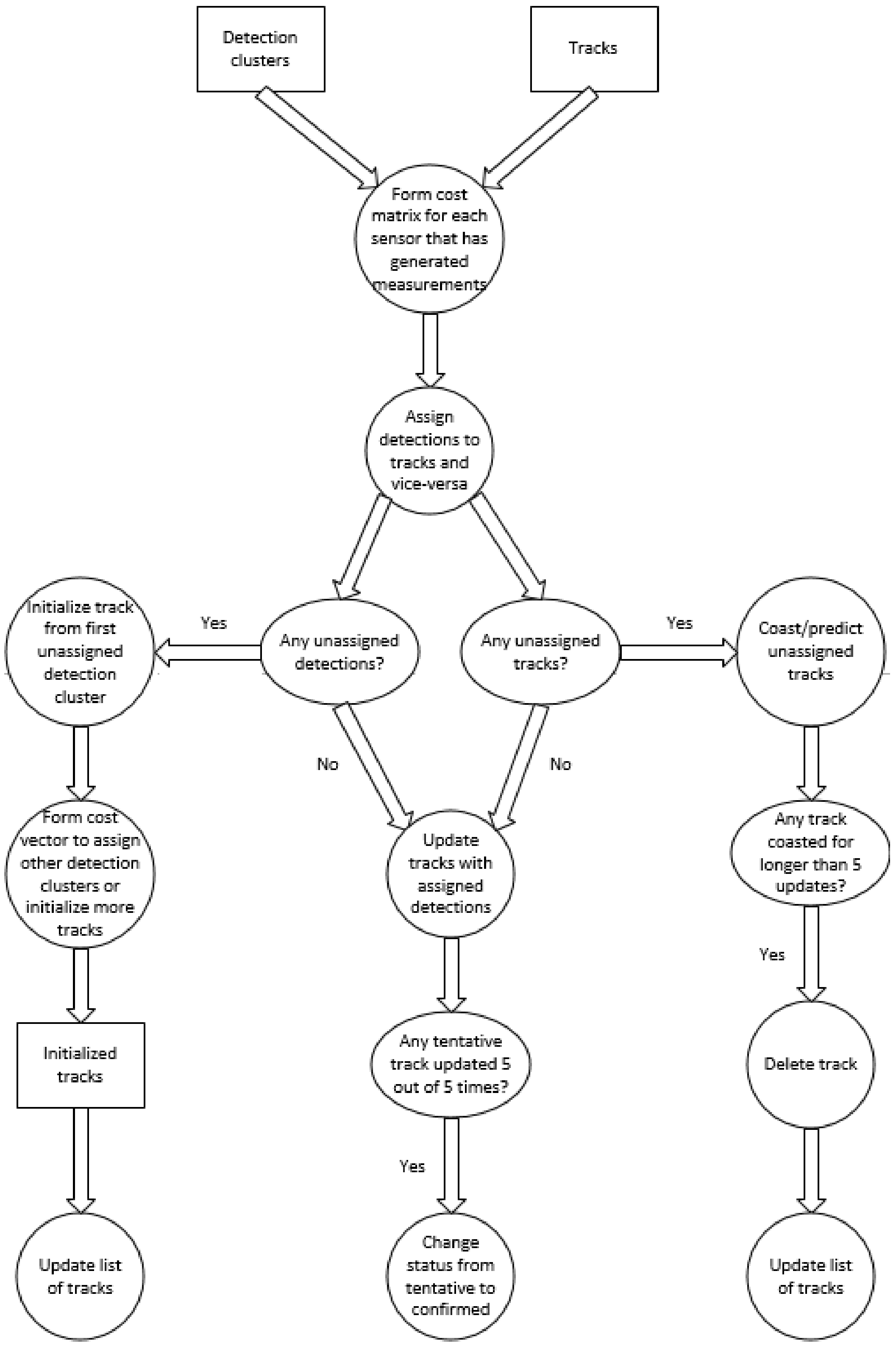

Detection clusters
Tracks
Form cost matrix for each sensor that has generated measurements
Assign detections to tracks and vice-versa
Initialize track from first unassigned detection cluster
Yes
Any unassigned detections?
Any unassigned tracks?
Yes
Coast/predict unassigned tracks
No
No
Form cost vector to assign other detection clusters or initialize more tracks
Update tracks with assigned detections
Any track coasted for longer than 5 updates?
Yes
Initialized tracks
Any tentative track updated 5 out of 5 times?
Delete track
Yes
Update list of tracks
Change status from tentative to confirmed
Update list of tracks

Figure 4.1: Track updating process [133]

At time $t = 0.07$ s, there are measurements only from the radar. Similar to step 1, two detection clusters are obtained from 25 measurements. A 2×2 cost matrix is formed, where each element is the normalized distance between the track and the detection. The detections are assigned to their respective tracks and vice-versa. If a tentative track is updated 5 out of 5 times, it becomes a confirmed track. Otherwise, it is deleted. If a confirmed track is not updated, it is coasted. If it is coasted for longer than 5 update intervals without being updated, it is deleted. At time $t = 0.14$ s, there are measurements from both the radar and the camera. Two cost matrices are formed, one for detection clusters from the radar and one for detection clusters from the camera. So the tracks are updated twice at the same time step.

4.1.4 Identification of the MIO (Most Important Object)

The MIO is the lead Twizy i.e. Twizy 1. It can be assumed that the lead Twizy is almost always in the front of Twizy 2 i.e. within a short azimuth range measured by the ego Twizy. The measured azimuth of Twizy 1 as visualized in Chapter 2 is close to zero on a straight path and changes slightly while overtaking, changing lanes or turning if both Twizys approximately have the same speed. Also, the object class identifier of the MIO should be always 1 i.e. vehicle. The relative radial velocity of the MIO should also be close to zero because the two Twizys move at nearly the same speed. Using the a combination of these assumptions, the MIO can be identified.

The above mentioned assumptions will be used to identify the MIO during real time tracking in the additional chapter that will be added later. In simulations, the main aim is to check the accuracy of the tracker. Also, there are only two targets. Therefore, instead of using the above mentioned assumptions to identify the MIO, the target index has been used to identify the MIO. The lead Twizy has target index 2 and the third Twizy has target index 3. After the MIO i.e. the lead Twizy has been identified, its relative yaw angle and yaw rate with respect to the ego Twizy is tracked by fitting the tracked points on to a third order polynomial using a least squares approach as explained in Chapter 3.

4.2 Simulation scenarios in MATLAB

Three simulation scenarios were tested for both the constant velocity and constant acceleration models: straight road driving, straight road driving with added sinusoidal oscillations of ego Twizy, and steady state cornering. Each scenario starts with a third Twizy to the left of and between the lead Twizy i.e. Twizy 1 and the ego Twizy i.e. Twizy 2. The third Twizy overtakes the lead Twizy after some time.

For tracking the relative yaw angle and relative yaw rate, polynomial fitting of the lead vehicle's path was used as described in Chapter 3. 4 cases of polynomial fitting were tested:

1. Without heading and curvature constraints
2. With heading constraint only
3. With curvature constraint only
4. With both heading and curvature constraints

Along with the relative yaw angle and yaw rate, the waypoint error and polynomial fitting error were also calculated. The waypoint error is defined as the error between the actual position of the target and the tracked position of the target. The polynomial error is defined as the error between actual waypoints and the calculated polynomial points. The results shown here are for a third degree polynomial that has been calculated by fitting 31 points using the least squares method. To decide the number of points and the degree of the polynomial, 4 combinations were tried:

1. 16 points with a third degree polynomial
2. 16 points with a fifth degree polynomial
3. 31 points with a third degree polynomial
4. 31 points with a fifth degree polynomial

The third combination was found to perform the best. The fifth order polynomial led to overfitting because the degree is too high. 16 points did not give as much accuracy as 31 points.

4.2.1 Straight road driving

The lead Twizy and ego Twizy both have a speed of 10 m/s. The third Twizy has a speed of 15 m/s. The paths taken by the vehicles is shown in Figure 4.2.

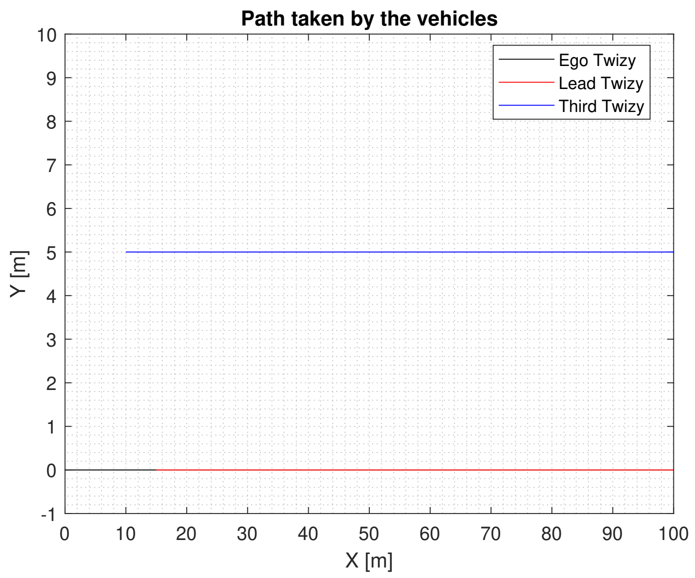
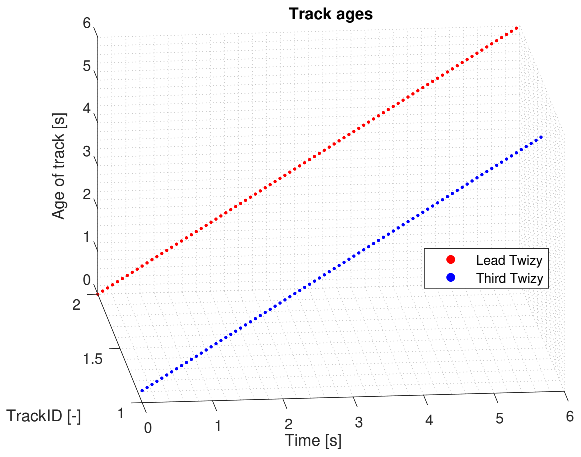


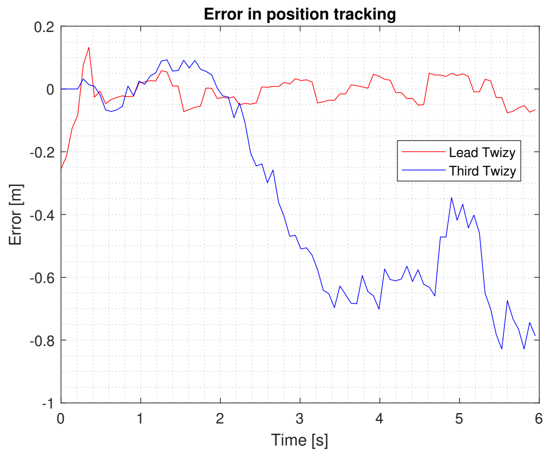
Figure 4.2: Paths taken by the vehicles

4.2.1.1 Constant velocity model

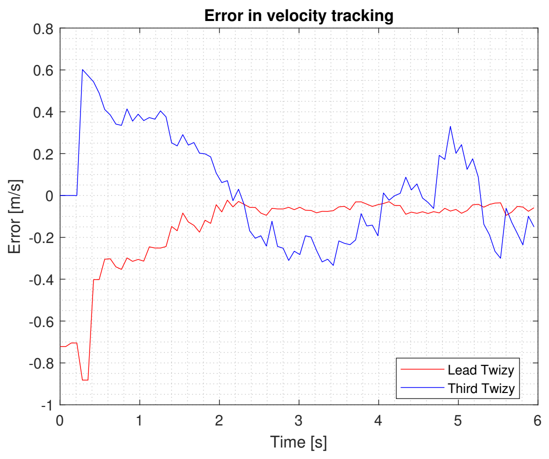
The ages of tracks, and errors in position tracking and velocity tracking are shown in Figure 4.3.



(a) Ages of tracks



(b) Error in position tracking



(c) Error in velocity tracking

Figure 4.3: Straight road driving with the CV model

The RMS (Root Mean Square) errors in position and velocity tracking are given in Table 4.4.

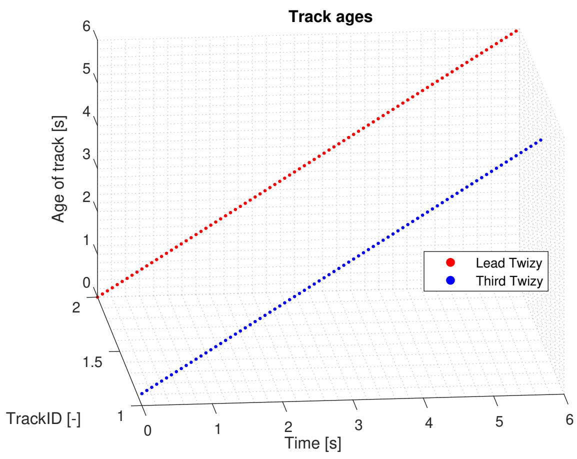
Table 4.4: RMS errors for the CV model for driving on a straight path

Target	RMS error position tracking	RMS error velocity tracking
Lead Twizy	0.0561	0.2496
Third Twizy	0.4552	0.25

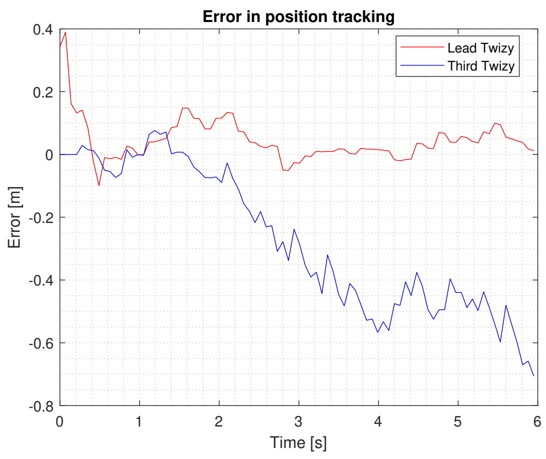
The tracker does not lose track of the targets at any point of time. Even though both targets move at a constant speed, the errors in position and velocity tracking are higher for the third Twizy because it moves away from the ego Twizy i.e. the measurements become more noisy as the target moves away from the sensors. The results for relative yaw angle and yaw rate tracking using polynomial fitting of the tracked path are shown in Appendix C. The third case i.e. with the curvature constraint only, gives the best results in terms of oscillations and error magnitudes.

4.2.1.2 Constant acceleration model

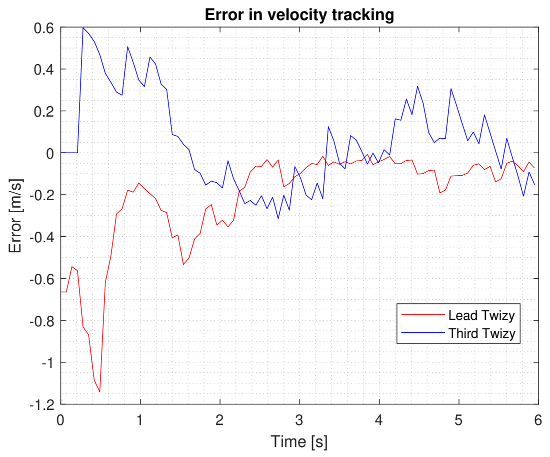
The ages of tracks, and errors in position tracking and velocity tracking are shown in Figure 4.4.



(a) Ages of tracks



(b) Errors in position tracking



(c) Errors in velocity tracking

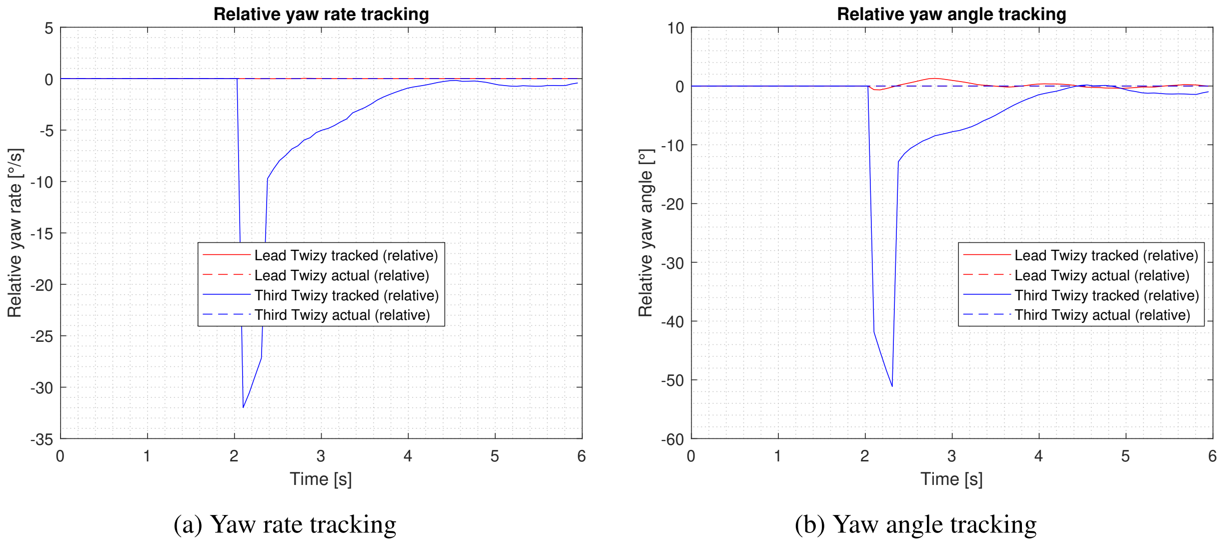
Figure 4.4: Straight road driving with the CA model

The RMS (Root Mean Square) errors in position and velocity tracking are given in Table 4.5.

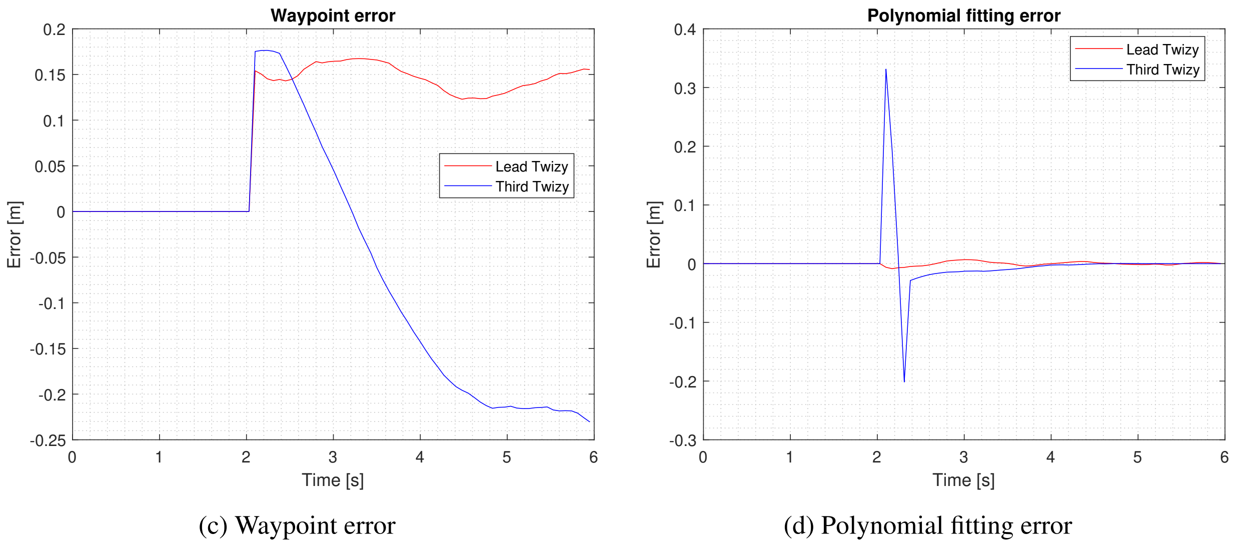
Table 4.5: RMS errors for the CA model for driving on a straight path

Target	RMS error position tracking	RMS error velocity tracking
Lead Twizy	0.0842	0.3187
Third Twizy	0.3548	0.2271

The tracker does not lose track of the targets at any point of time. For the lead Twizy, the tracking errors of the CA model are higher than those of the CV model but for the third Twizy, the opposite is true. The reason is that the CA model has higher measurement noise than the CV model. For targets moving at the same speed as the ego vehicle, the errors are higher due to higher measurement noise but for targets moving away from the ego Twizy, the increased measurement noise helps in tracking the target better than the CV model. The tracked relative yaw angle and yaw rate using polynomial fitting of the tracked path for the third case i.e. with the curvature constraint only, are shown in Figure 4.5. Results for the other three cases are shown in Appendix C. The third case gives the best results in terms of oscillations and error magnitude.



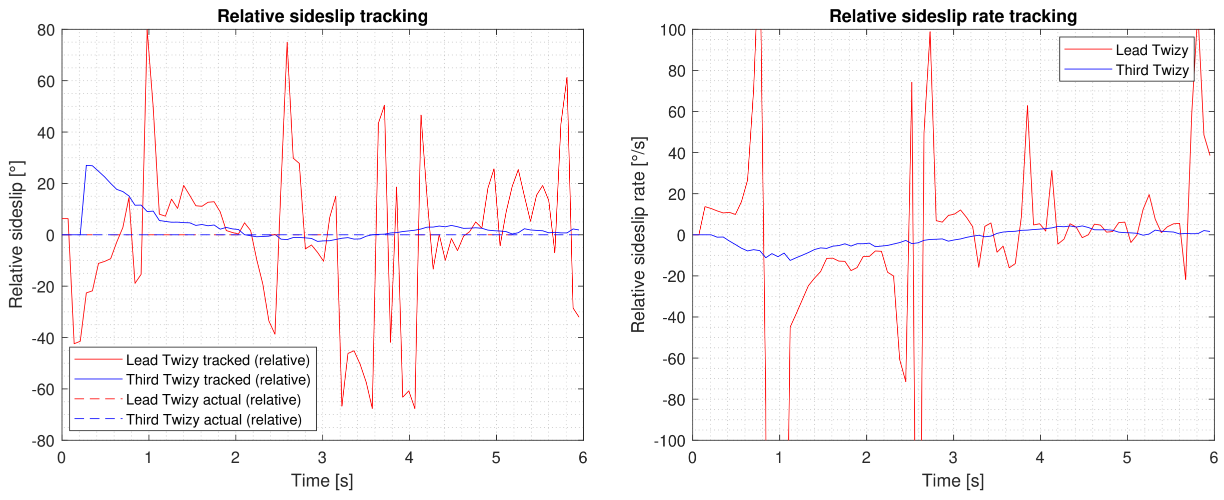
(a) Yaw rate tracking (b) Yaw angle tracking



(c) Waypoint error (d) Polynomial fitting error

Figure 4.5: Relative yaw angle and yaw rate tracking for case 3: curvature constraint only

The tracked relative sideslip and sideslip rate are shown in Figure 4.6. Since the tracked relative velocity of the lead Twizy in the x direction is very low, the relative sideslip and sideslip rate for the lead Twizy spike to high values i.e. $80°$ and $-900°/s$ respectively. This does not happen for the third Twizy since it is moving away from the ego Twizy at a constant speed.



(a) Relative sideslip tracking

(b) Relative sideslip rate tracking

Figure 4.6: Relative sideslip and sideslip rate tracking using tracked velocities and accelerations

4.2.2 Straight road driving with added sinusoidal oscillations of the ego Twizy

The lead Twizy and ego Twizy both have a speed of 10 m/s. The third Twizy has a speed of 15 m/s. The ego Twizy has a path in form of a sine wave with amplitude 0.5 m and frequency 0.1 Hz i.e. $\omega = \pi/5$ rad. The paths taken by the vehicles is shown in Figure 4.7.

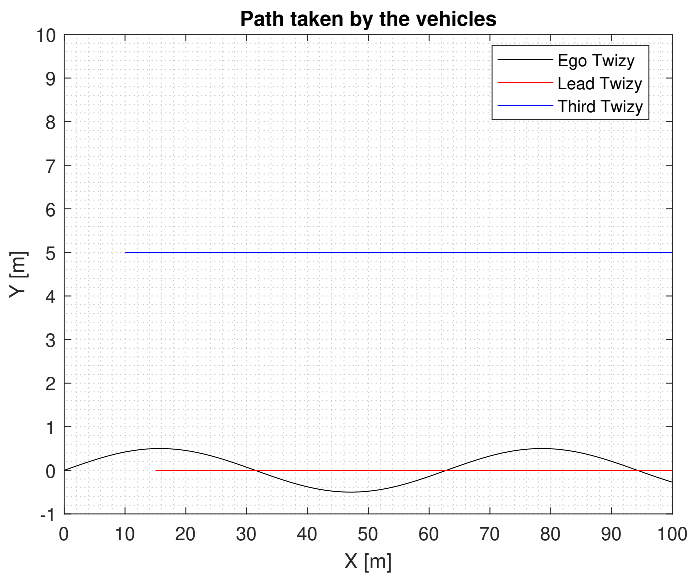
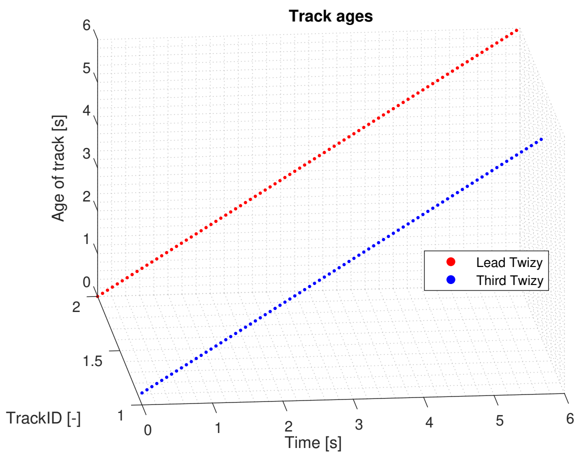


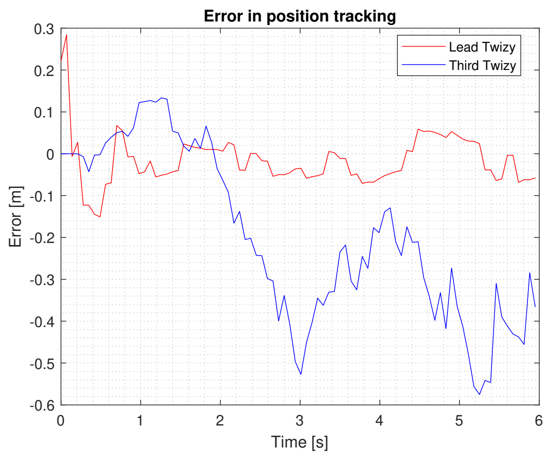
Figure 4.7: Paths taken by the vehicles

4.2.2.1 Constant velocity model

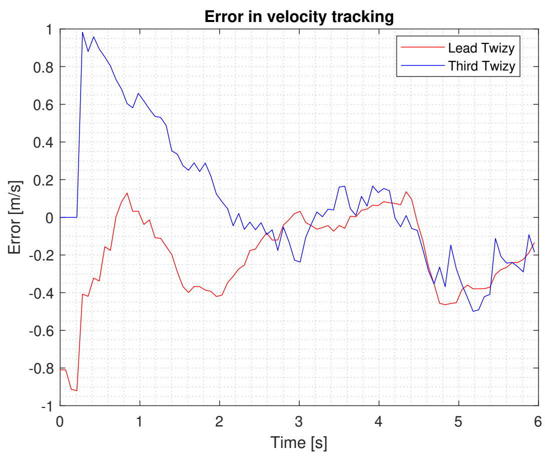
The ages of tracks, and errors in position tracking and velocity tracking are shown in Figure 4.8.



(a) Ages of tracks



(b) Error in position tracking



(c) Ages of tracks

Figure 4.8: Straight road driving with added sinusoidal oscillations of ego Twizy with CV model

The RMS (Root Mean Square) errors in position and velocity tracking are given in Table 4.6.

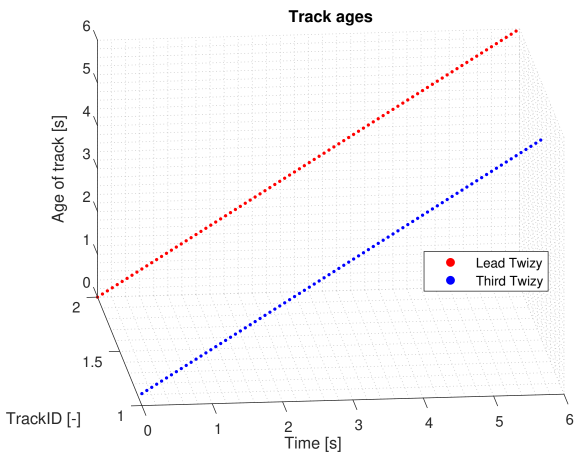
Table 4.6: RMS errors for the CV model for driving on a straight path with sinusoidal oscillations of ego Twizy

Target	RMS error position tracking	RMS error velocity tracking
Lead Twizy	0.0638	0.3036
Third Twizy	0.2817	0.3668

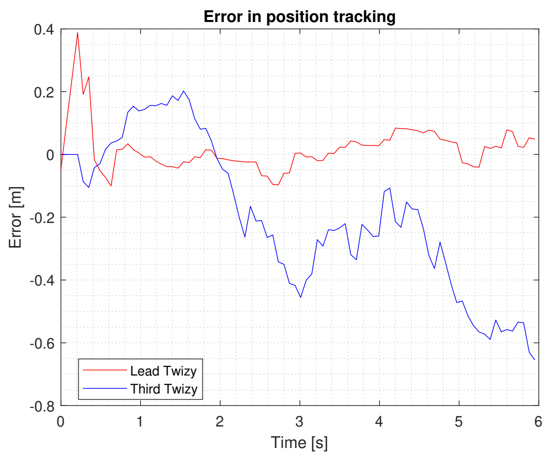
The tracker does not lose track of the targets at any point of time. The errors in position and velocity tracking are higher for the third Twizy because the third Twizy moves away from the ego Twizy i.e. the measurements become more noisy as the target moves away from the senors. The results for relative yaw angle and yaw rate tracking using polynomial fitting of the tracked path are shown in Appendix C. As with straight road driving, the third case i.e. with the curvature constraint only, gives the best results in terms of oscillations and error magnitudes.

4.2.2.2 Constant acceleration model

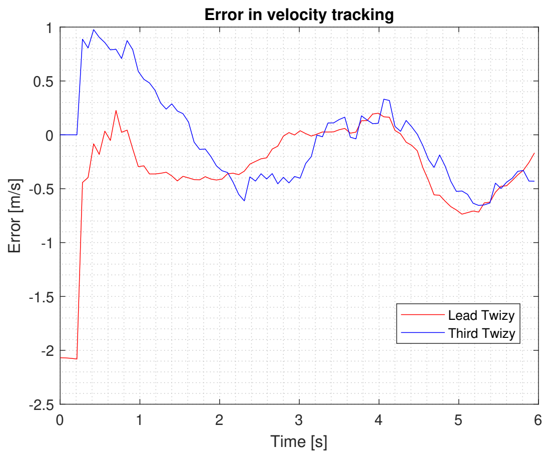
The ages of tracks, and errors in position tracking and velocity tracking are shown in Figure 4.9.



(a) Ages of tracks



(b) Error in position tracking



(c) Error in velocity tracking

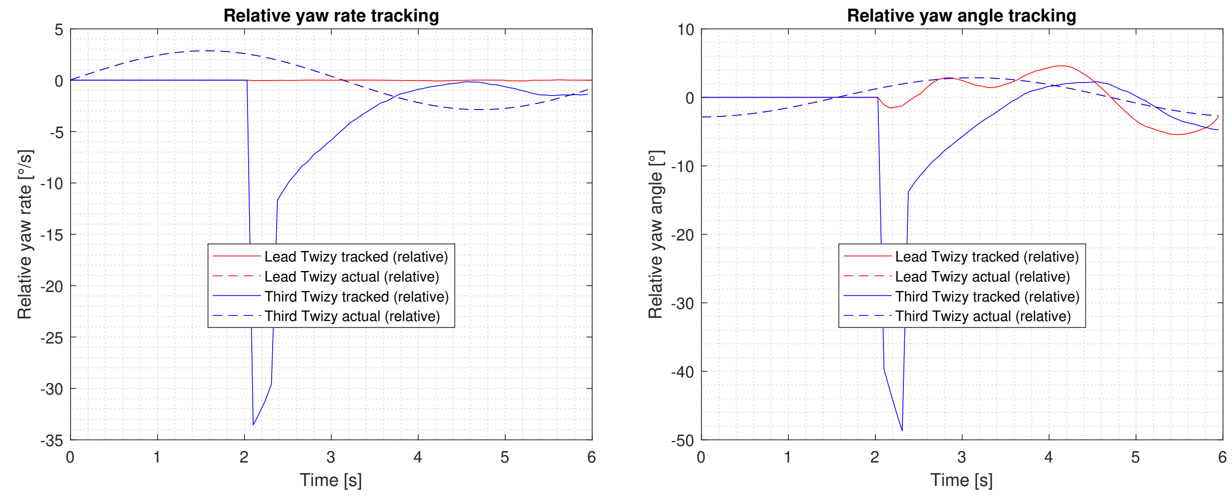
Figure 4.9: Straight road driving with added sinusoidal oscillations of ego Twizy with CA model

The RMS (Root Mean Square) errors in position and velocity tracking are given in Table 4.7.

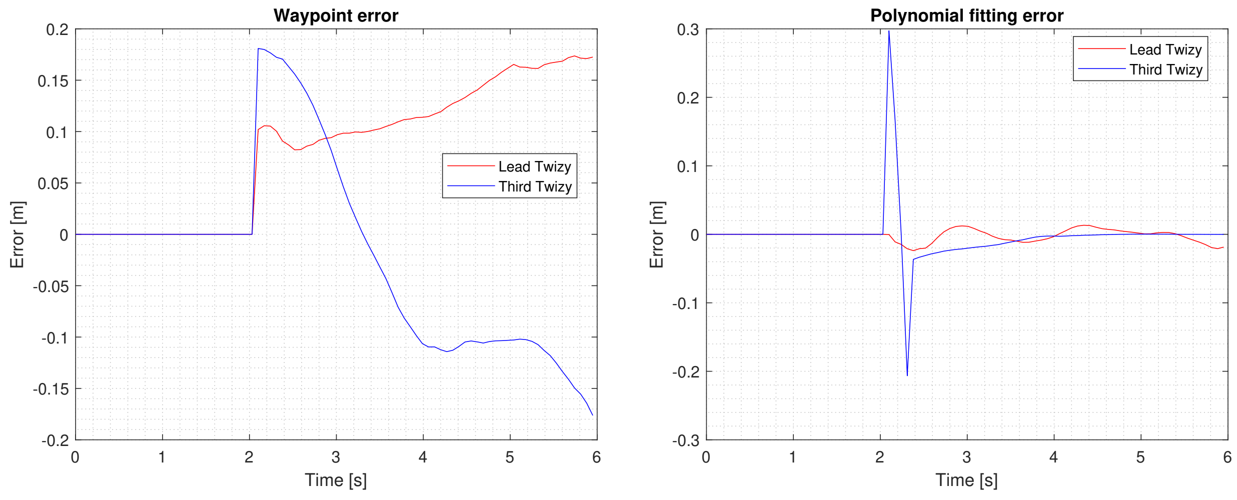
Table 4.7: RMS errors for the CA model for driving on a straight path with sinusoidal oscillations of the ego Twizy

Target	RMS error position tracking	RMS error velocity tracking
Lead Twizy	0.0747	0.5635
Third Twizy	0.3084	0.4341

The tracker does not lose track of the targets at any point of time. For the lead Twizy, the result is similar to the previous scenario i.e. the tracking errors of the CA model are higher than the tracking errors of the CV model. But for the third Twizy, unlike the previous scenario, the tracking errors of the CA model are higher than the tracking errors of the CV model. A possible reason is that due to the continuous sinusoidal movements of the ego Twizy, the large measurement noise of the CA model is not able to compensate for the measurements becoming more noisy as the target moves away from the sensors. For yaw angle and yaw rate tracking, the third case i.e. curvature constraint only, is found to perform the best as shown in Figure 4.10. The results for the other three cases are shown in Appendix C.



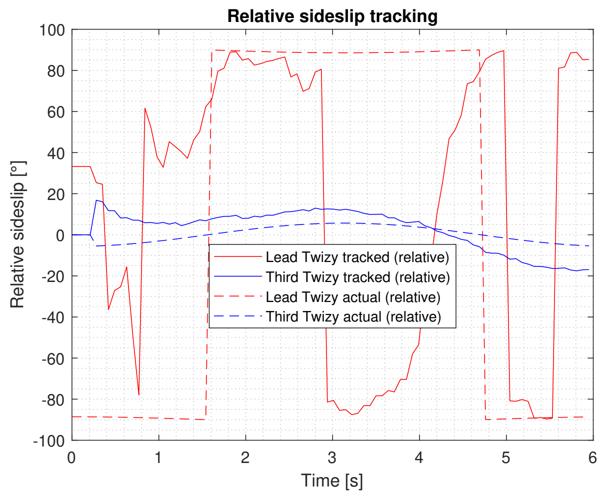
(a) Yaw rate tracking (b) Yaw angle tracking



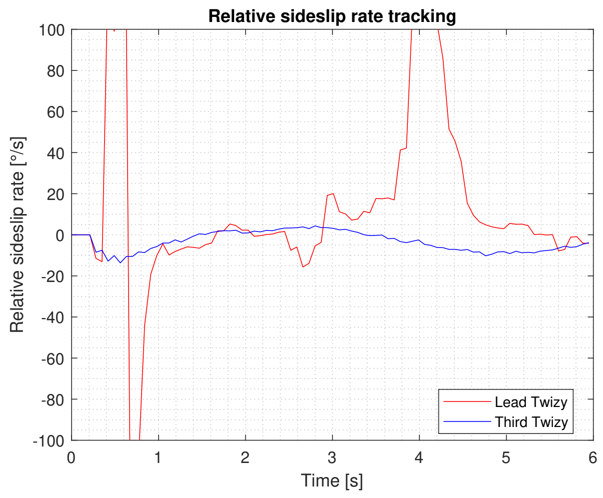
(c) Waypoint error (d) Polynomial fitting error

Figure 4.10: Relative yaw angle and yaw rate tracking for case 3: curvature constraint only

The tracked relative sideslip and sideslip rate are shown in Figure 4.11. Since the tracked relative velocity of the lead Twizy in the x direction is very low, the relative sideslip and sideslip rate for the lead Twizy spike to extreme values i.e. 90° and $-260^\circ/s$ respectively. This does not happen for the third Twizy since it is moving away from the ego Twizy at a constant speed.



(a) Relative sideslip tracking



(b) Relative sideslip rate tracking

Figure 4.11: Relative sideslip and sideslip rate tracking using tracked velocities and accelerations

4.2.3 Steady state cornering

The paths shown in Figure 4.12 are from top to bottom (towards negative X and towards positive Y). The lead Twizy has a speed of 8 m/s and another Twizy has a speed of 10 m/s, while the ego Twizy has a speed of 5 m/s. The lead Twizy is given a higher speed to simulate the situation in which the ego Twizy is not able to keep up with the lead Twizy.

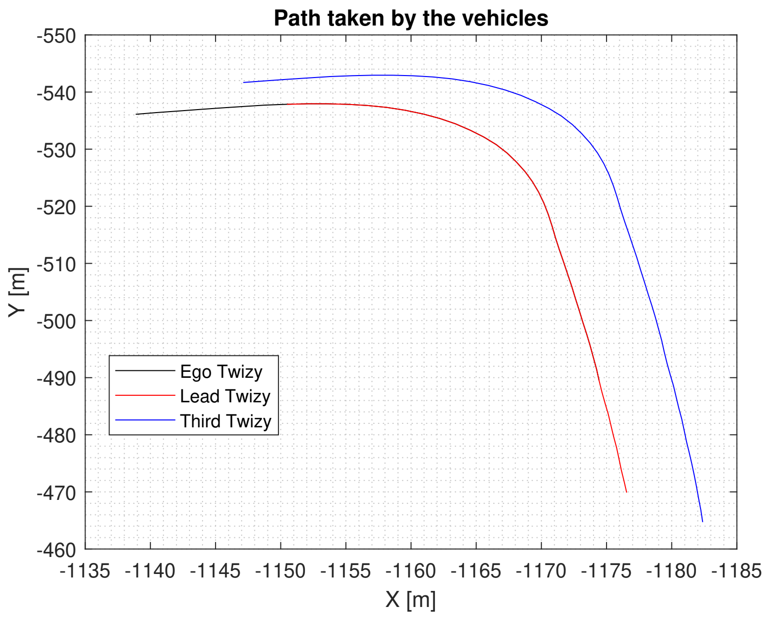
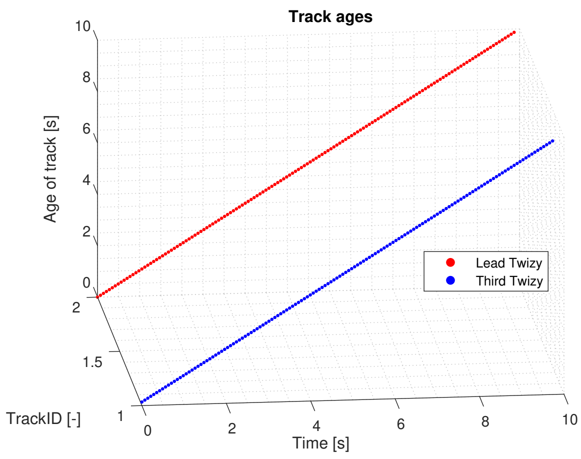


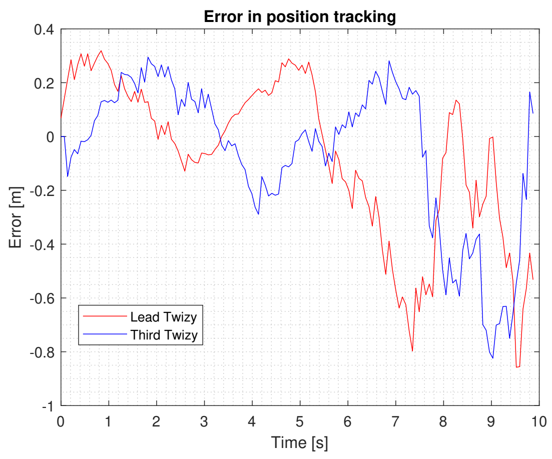
Figure 4.12: Paths taken by the vehicles

4.2.3.1 Constant velocity model

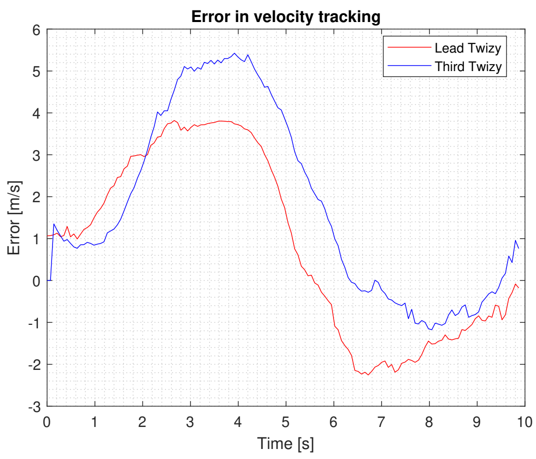
The ages of tracks, and errors in position tracking and velocity tracking are shown in Figure 4.13.



(a) Ages of tracks



(b) Error in position tracking



(c) Error in velocity tracking

Figure 4.13: Steady state cornering with CV model

The RMS (Root Mean Square) errors in position and velocity tracking are given in Table 4.8.

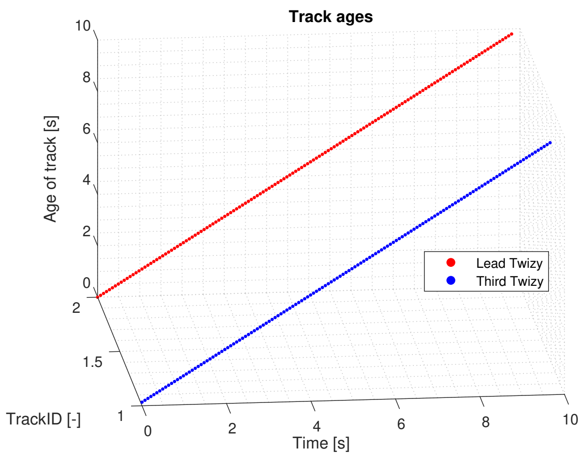
Table 4.8: RMS errors for the CV model for steady state cornering

Target	RMS error position tracking	RMS error velocity tracking
Lead Twizy	0.3019	2.3076
Third Twizy	0.2827	2.8186

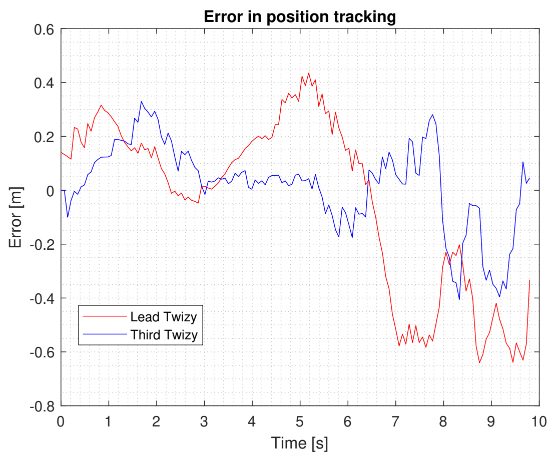
The tracker does not lose track of the targets at any point of time. The velocity tracking error for the third Twizy is higher than the velocity tracking error for the lead Twizy because the third Twizy moves away from the ego Twizy. Accordingly, the position tracking error of the third Twizy should be higher than the position tracking error of the lead Twizy, but the opposite is observed. A possible reason is that the lead Twizy goes out of the FOV (Field of View) of the camera earlier than the third Twizy because it takes the inside curve. During this time, it is detected only by the radar. The radar measures the radial velocity well, but its angular resolution is not as good as the camera. Therefore the position tracking error increases, but the velocity tracking error does not increase. For relative yaw angle and yaw rate tracking, the third case i.e. curvature constraint only, gives the best result. The results for this part are shown in Appendix C.

4.2.3.2 Constant acceleration model

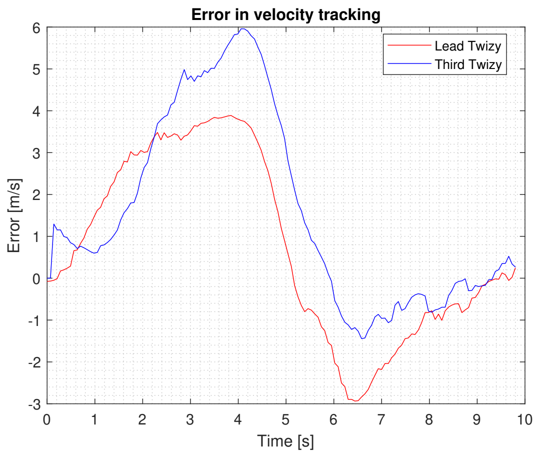
The ages of tracks, and errors in position tracking and velocity tracking are shown in Figure 4.14.



(a) Ages of tracks



(b) Error in position tracking



(c) Error in velocity tracking

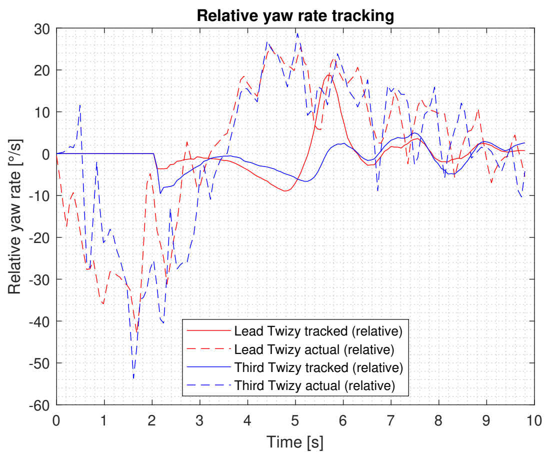
Figure 4.14: Steady state cornering with CA model

The RMS (Root Mean Square) errors in position and velocity tracking are given in Table 4.9.

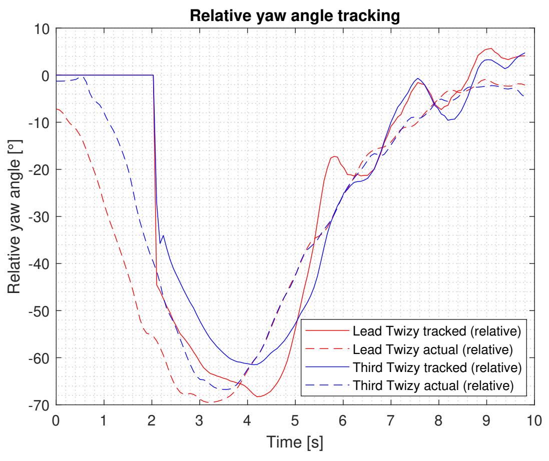
Table 4.9: RMS errors for the CA model for steady state cornering

Target	RMS error position tracking	RMS error velocity tracking
Lead Twizy	0.3221	2.2748
Third Twizy	0.1582	2.7662

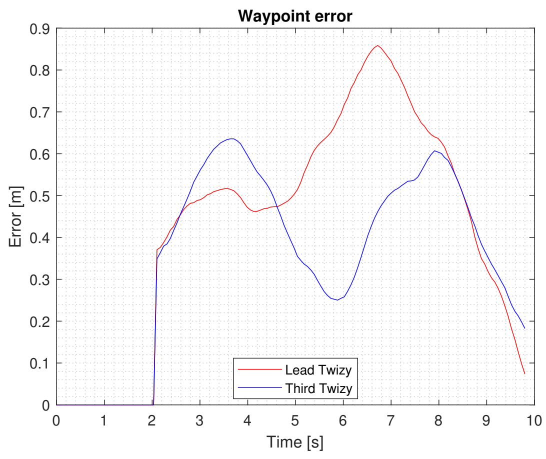
The tracker does not lose track of the targets at any point of time. For the lead Twizy, the velocity tracking error of the CA model is lower than the velocity tracking error of the CV model because although the targets move at a constant speed, they have a constant lateral acceleration while turning. The CA model tracks this acceleration and estimates the velocity better. For the third Twizy, the tracking errors of the CA model are lower than the tracking errors of the CV model. The reason is that the CA model has higher measurement noise than the CV model. For targets moving away from the ego Twizy, the higher measurement noise of the CA model helps in tracking the target better than the CV model. For yaw angle and yaw rate tracking, the third case i.e. curvature constraint only, is found to perform the best as shown in Figure 4.15. The results for the other three cases are shown in Appendix C.



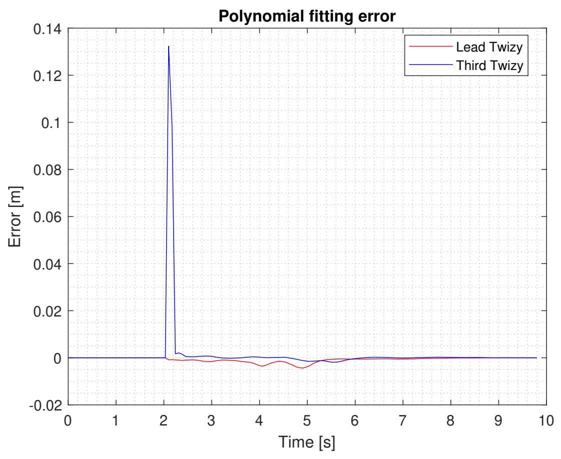
(a) Yaw rate tracking



(b) Yaw angle tracking



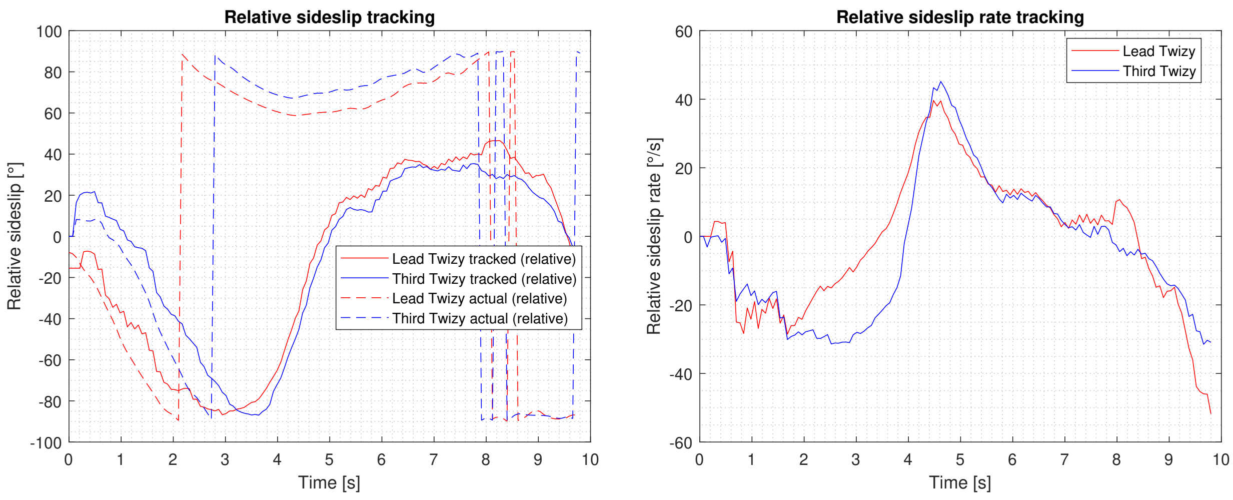
(c) Waypoint error



(d) Polynomial fitting error

Figure 4.15: Relative yaw angle and yaw rate tracking for case 3: curvature constraint only

The tracked relative sideslip and sideslip rate are shown in Figure 4.16. Unlike the first two scenarios, the relative sideslip and sideslip rate for the targets do not spike to very high values since both the targets are moving away from the ego Twizy and therefore do not have a small relative velocity in the x direction of the ego Twizy frame.



(a) Relative sideslip tracking (b) Relative sideslip rate tracking

Figure 4.16: Relative sideslip and sideslip rate tracking using tracked velocities and accelerations

The conclusions of this chapter are given along with the conclusions of Chapter 5 in Chapter 6.

5 Offline tracking with real world data from 1 radar, GNSS and IMU

In November 2019, 425.84 seconds of real world testing data was obtained using one CAN based NXP Cocoon radar, the stereo camera, GNSS and the IMU. Twizy 1 and Twizy 2 were driven on a path around TU/e from point A to point B as shown in Figure 5.1. There are 5 straights and 4 turns. The test was done during the afternoon, so there was a decent amount of traffic on the roads. The Twizys stopped at traffic lights on the first two turns. During the second straight, the lead Twizy changed lanes twice. The lead Twizy was mostly in front of the ego Twizy except at the turns.

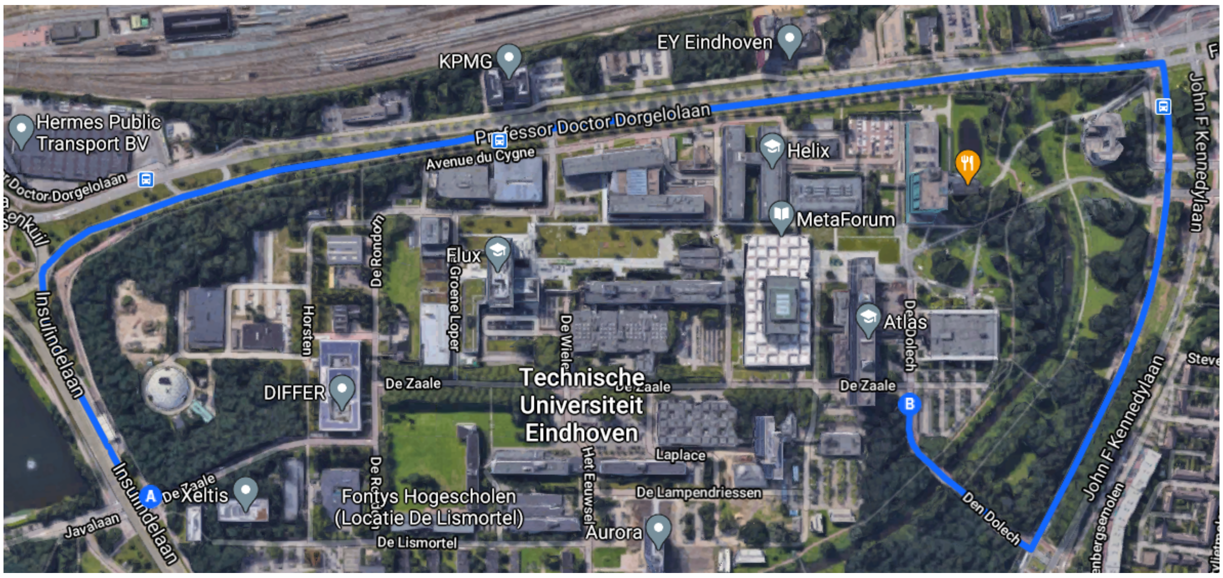


Figure 5.1: Path on Google maps

To see the effect of sensor fusion, track reconstruction was done using measurements from the radar, GNSS and the IMU i.e. the recorded measurements from the radar, GNSS, and the IMU were input to the tracker using the methods explained in Chapter 3 and the track of the lead Twizy was extracted from the confirmed tracks received as output from the tracker. The track of the lead Twizy was matched with the recorded measurements to check the performance of the tracker.

5.1 Filtered measurements from the radar

The measurements from the radar were filtered by applying thresholds based on the specifications of the radar. The thresholds for the measured variables are given in Table 5.1. Measurements not within the range of the minimum and maximum thresholds were not used for tracking. The minimum threshold value for the SNR (Signal to Noise) ratio was chosen as 15 because CFAR (Constant False Alarm Rate) detections [135] generally have an SNR in the range of 15-20.

Table 5.1: Threshold values for radar measurements

Variable [Unit]	Minimum threshold	Maximum threshold
Range [m]	0.75	70
Azimuth [°]	-60	60
Elevation [°]	-10	10
SNR [dB]	15	N/A
Radial speed [m/s]	-69.45	41.67

After applying these thresholds, detection clusters are made from radar measurements as explained in Chapter 3. These detection clusters are then fed into the tracker. The values of tracker parameters are tuned by Latin Hypercube Sampling (LHS) as described Chapter 3.

5.2 Application of LHS (Latin Hypercube Sampling)

In addition to the assignment threshold and the clustering threshold, the position measurement noise covariance matrix i.e. $diag(\begin{bmatrix}\sigma^2_{x,k} & \sigma^2_{y,k} & \sigma^2_{z,k}\end{bmatrix})$, velocity measurement noise covariance matrix i.e. $diag(\begin{bmatrix}\sigma^2_{\dot{x},k} & \sigma^2_{\dot{y},k} & \sigma^2_{\dot{z},k}\end{bmatrix})$, and the acceleration measurement noise covariance matrix (in case of the CA model) i.e. $diag(\begin{bmatrix}\sigma^2_{\ddot{x},k} & \sigma^2_{\ddot{y},k} & \sigma^2_{\ddot{z},k}\end{bmatrix})$ were also tuned using LHS. These measurement noise covariance matrices were also weighted differently for the x, y, and z values because the radar has better depth i.e. longitudinal measurements than lateral and vertical measurements. Therefore, the weights of the position, velocity, and acceleration measurement covariance matrices were also tuned using LHS. If the assignment threshold and clustering threshold are the first two parameters given by p_1 and p_2 respectively, then the measurement noise covariance matrix R for the constant velocity model is given by:

$$R = \begin{bmatrix} p_3 \times p_5 & 0 & 0 & 0 & 0 & 0 \\ 0 & p_3 \times p_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & p_3 \times p_6 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_4 \times p_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & p_4 \times p_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & p_4 \times p_6 \end{bmatrix} \tag{5.1}$$

where $diag(\begin{bmatrix}p_3 & p_3 & p_3\end{bmatrix})$ is the position measurement covariance matrix, $diag(\begin{bmatrix}p_4 & p_4 & p_4\end{bmatrix})$ is the velocity measurement covariance matrix, p_5 is the weight for the x axis, and p_6 is the weight for y and z axes. Similarly, for the constant acceleration model, the measurement covariance matrix R is given by:

$$R = \begin{bmatrix} p_3 \times p_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p_3 \times p_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p_3 \times p_7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_4 \times p_6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p_4 \times p_7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p_4 \times p_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_5 \times p_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_5 \times p_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_5 \times p_7 \end{bmatrix} \tag{5.2}$$

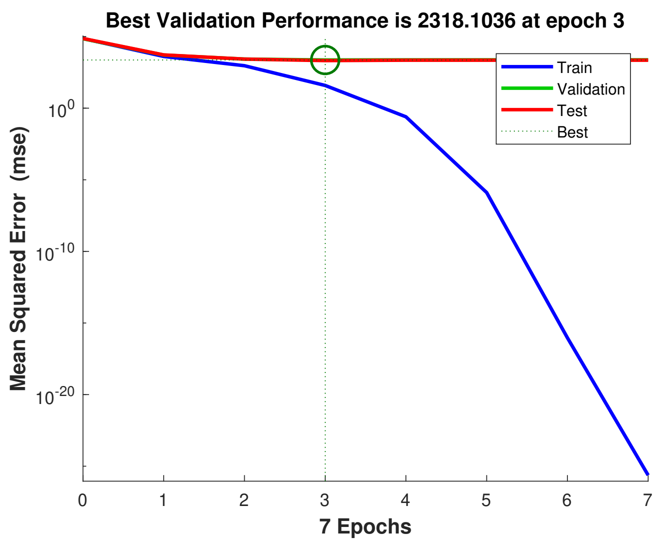
where $diag(\begin{bmatrix} p_3 & p_3 & p_3 \end{bmatrix})$ is the position measurement covariance matrix, $diag(\begin{bmatrix} p_4 & p_4 & p_4 \end{bmatrix})$ is the velocity measurement covariance matrix, $diag(\begin{bmatrix} p_5 & p_5 & p_5 \end{bmatrix})$ is the acceleration measurement covariance matrix, p_6 is the weight for the x axis, and p_7 is the weight for y and z axes.

There are 6 parameters for the CV model and 7 parameters for the CA model. LHS was used to create 500 samples for a six-dimensional hypercube for the CV model, and 500 samples for a seven-dimensional hypercube for the CA model. Since Twizy 1 was always being followed, the track having the longest age should belong to Twizy 1 if the tracker is working correctly. Therefore for LHS, the parameters were being optimized on the basis of the age of the track having the maximum age. The values of the parameters which give the best maximum ages for the constant velocity model and constant acceleration model are given in Table 5.2.

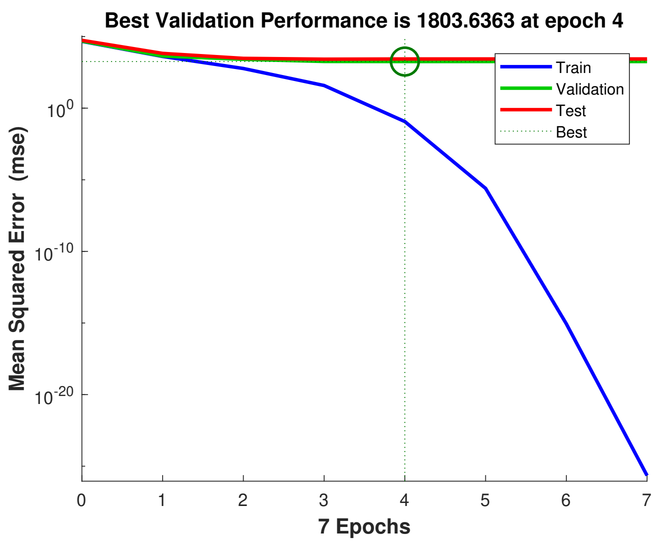
Table 5.2: Parameters giving the best age for the CV and CA models

Model	Constant velocity	Constant acceleration
Maximum track age [s]	292.04	280.49
p_1	20.84	26.28
p_2	2.613	2.103
p_3	9.756	17.712
p_4	9.7767	19.1608
p_5	0.729	73.875
p_6	0.561	0.655
p_7	N/A	0.583

Out of the 500 tested samples, 90 samples were found to have a maximum track age less than 0.5 seconds for the constant velocity model, and 84 samples were found to have a maximum track age lesser than 0.5 seconds for the constant acceleration model. To find out the co-relation between parameters and the maximum track age, a feedforward neural network having three hidden layers each having 50 neurons was trained for both the constant velocity and the constant acceleration models as shown in Figure 5.2. The mean squared error for the training data shown by the blue line for both the CV and CA models reaches very low values i.e. 10^{-25} after 7 episodes, but the least error on the validation data shown by the green line for both the motion models remains very high i.e. 2318 for the CV model and 1803 for the CA model. Therefore, there is no clear correlation between the parameters and the maximum track age i.e. the maximum track age for a given set of parameters can not be predicted. The Levenberg-Marquardt method was used for training the neural network because this method was found to have the best performance in terms of the mean squared error for linear and non-linear function approximation in [136]. The neural network was also trained with 100 neurons in each layer but that gave a worse result due to overfitting.



(a) Constant velocity model

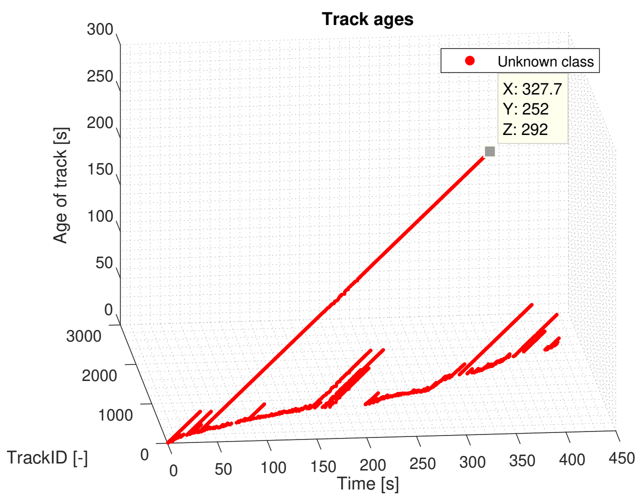


(b) Constant acceleration model

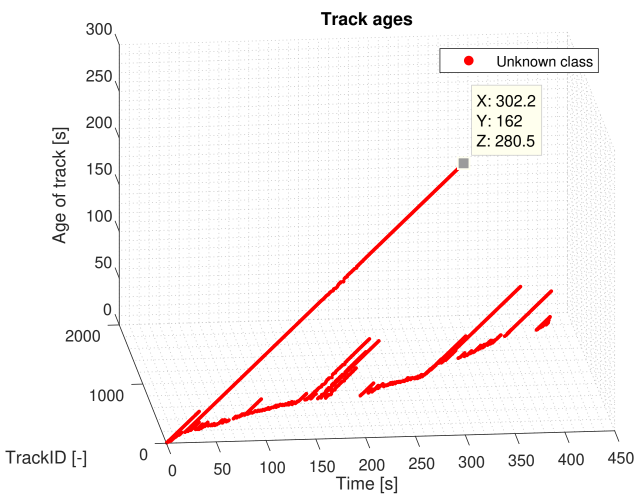
Figure 5.2: Performance of the neural network

5.3 Tracking results

The ages of all tracks for the CV and CA models are shown in Figure 5.3. The longest age of the track belonging to the leader Twizy is 292.04 seconds for the CV model and 280.49 seconds for the CA model.



(a) Constant velocity model



(b) Constant acceleration model

Figure 5.3: Ages of all tracks

From Figure 5.3, it can be seen that the longest track starts at $t = 35.7$ s for the CV model and at $t = 21.7$ s for the CA model. It is known from a video recording of the experiment that on the path taken during the experiment in Figure 5.1, the Twizys stopped at a traffic light at the first turn at about $t = 30$ s. At about $t = 160$ s, the Twizys stopped at the traffic light at the second turn, and at $t = 280$ s the Twizys took the third turn. The Twizys also switched lanes twice between $t = 100$ s and $t = 150$ s. This means that the longest tracks for the CV model and the CA model are maintained during the first three turns and during the lane changes. Ideally the tracks should have been maintained for the whole time of testing. Performance evaluation of the tracker is shown in the last section of this chapter.

5.3.1 Matching tracks with measurements

To verify that the track having the maximum age belongs to the leader Twizy, the tracked parameters have to be matched against the measurements from the radar i.e. azimuth, elevation, range and radial speed. The closest 4 azimuth, elevation, range and radial speed measurements corresponding to the times when the track with the maximum age was active for the constant acceleration model are shown in Figure 5.4, Figure 5.5, Figure 5.6, and Figure 5.7 respectively. Measurement matching results for the constant velocity model have been shown in Appendix D.

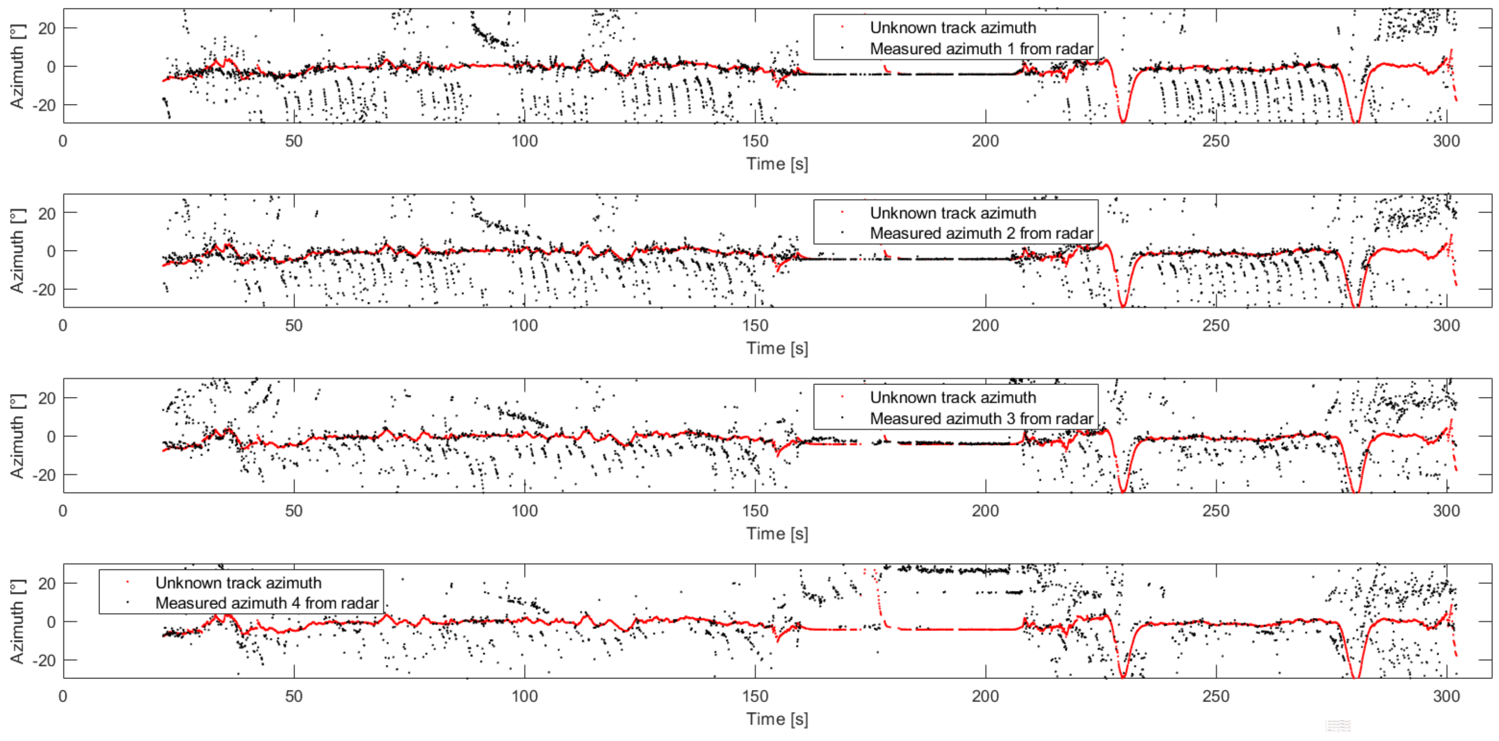


Figure 5.4: Matched azimuths for the constant acceleration model

As seen in the graphs, most of the tracked values overlap with the first 3 measurements. The black points represent the recorded measurements and the red points represent the tracked values from the longest track. The class of the longest track is unknown as the radar does not detect the class of the target. Therefore the track is marked as an unknown track on the graphs. The ground truth was not available in this real world scenario but it is known that the lead Twizy was mostly in front of the ego

Twizy, therefore the leader Twizy can be assumed to be within the 3 closest detections from the radar. Based on this assumption, it can be said the tracked object is the leader Twizy.

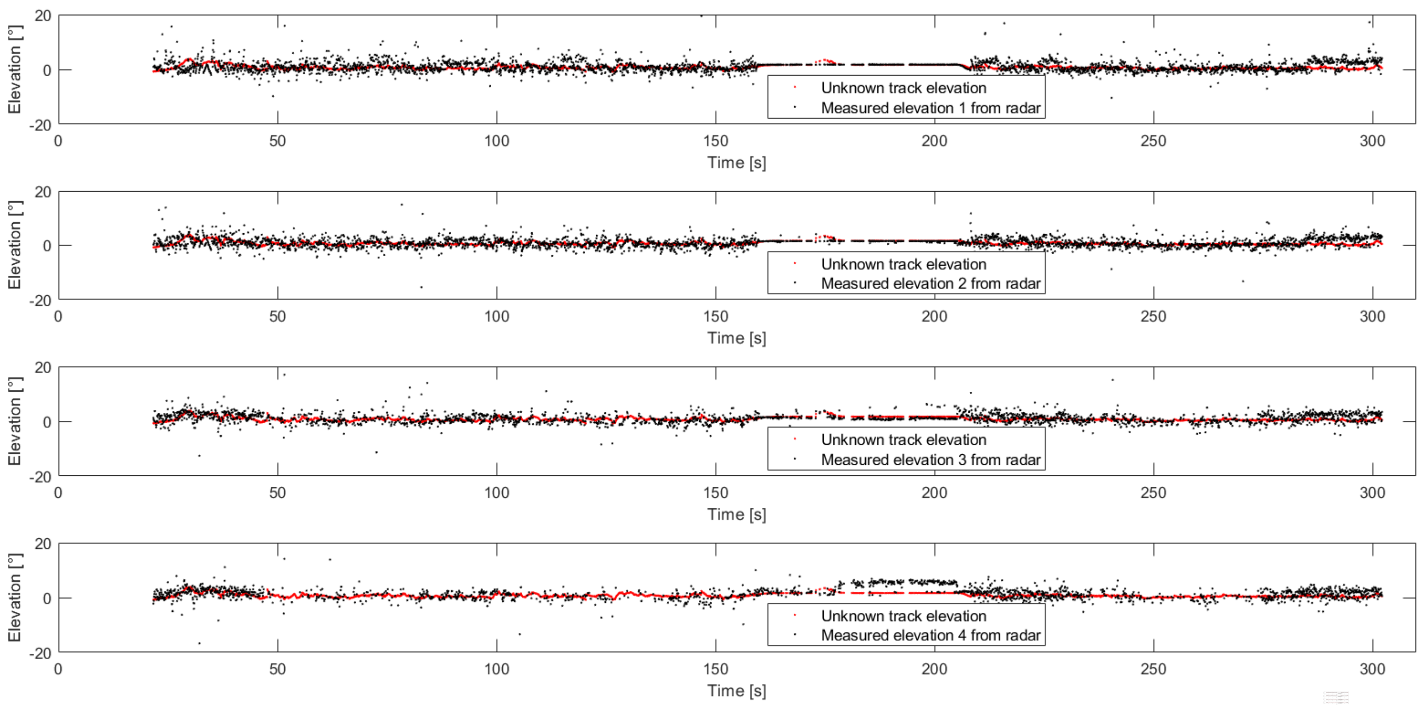


Figure 5.5: Matched elevations for the constant acceleration model

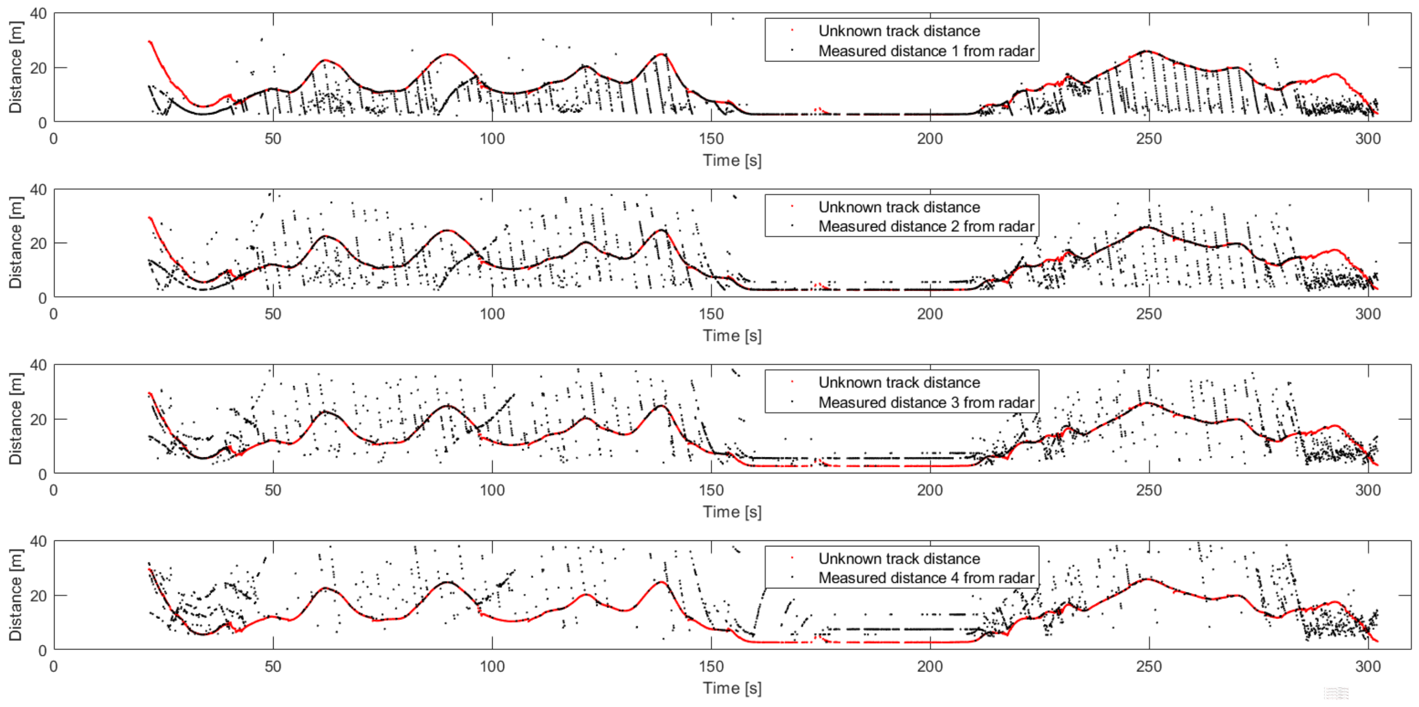


Figure 5.6: Matched distances for the constant acceleration model

Since the lead Twizy was mostly in front of the ego Twizy and the road was flat, the tracked azimuth and tracked elevation shown by the red points in Figure 5.4 and Figure 5.5 are close to zero. The tracked azimuth drops sharply to negative values at $t = 225$ s when the lead Twizy takes a sharp right turn after the traffic light at turn 2. This is seen in Figure 5.6 as an increasing distance as the lead Twizy pulls away from the ego Twizy. The tracked azimuth in Figure 5.4 drops sharply again to negative values at $t = 275$ s when the lead Twizy takes a sharp right turn at turn 3. In Figure 5.6, the tracked distance drops to a low value from $t = 155$ s to $t = 210$ s as the Twizys stopped at the traffic light on turn 2 during this time interval. In the first sub-plot, the tracked distance does not match with the closest measured distance from $t = 90$ s to $t = 96$ s because the closest measurement comes from a metal road sign on the right side of the road. Initially at $t = 21.7$s, the tracked distances match with the third and fourth closest measurements as other vehicles are closer to the ego Twizy than the lead Twizy when they overtake the ego Twizy.

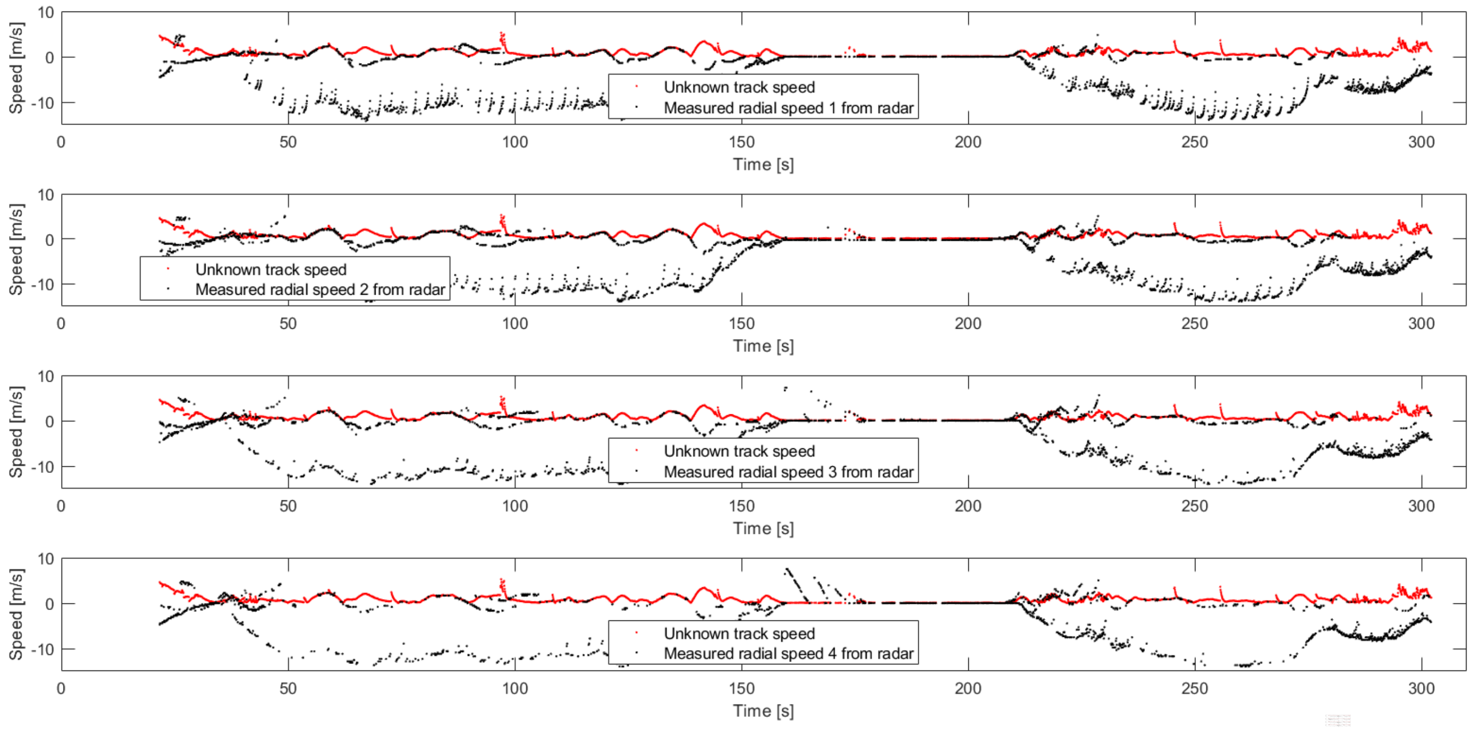
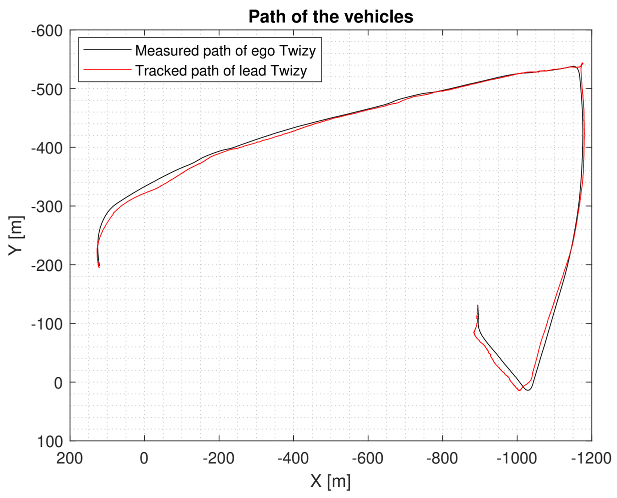


Figure 5.7: Matched speeds for the constant acceleration model

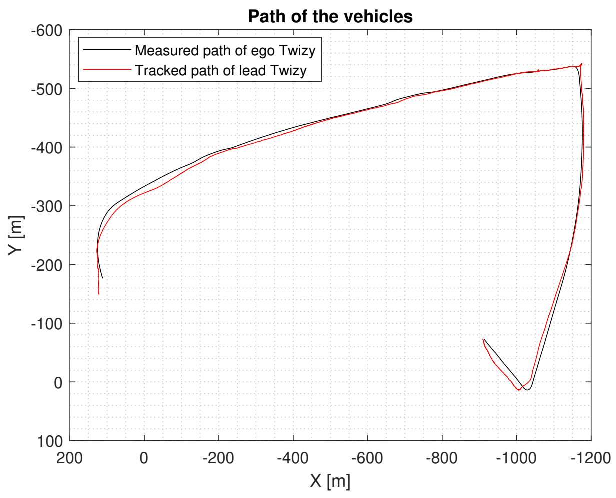
The tracked velocity in Figure 5.7 does not match with the radial velocity measurements as much as the tracked azimuth, tracked elevation and tracked distance match with their corresponding measurements because in real world scenarios, the velocity tracking of the tracker is not as accurate as its position tracking. The reason is that when velocity measurements are resolved into their x, y and z components, the rate of change of azimuth and elevation is neglected by MATLAB as they are not known.

5.3.2 Global path and speed reconstruction

The path of the lead Twizy in the global X and Y coordinates can be reconstructed using the tracked relative coordinates of the lead Twizy, and the recorded global coordinates and heading angle of the ego Twizy. If the heading angle of the ego Twizy is known, the tracked relative coordinates of the lead Twizy with respect to the frame of the ego Twizy can be transformed to the global frame to obtain the global coordinates of the lead Twizy. The reconstructed global path of the lead Twizy and the recorded global path of the ego Twizy for the CV model and the CA model are shown in Figure 5.8. The reconstructed path of the lead Twizy is similar to that of the ego Twizy. This is as expected because the ego Twizy was following the leader Twizy at all times.



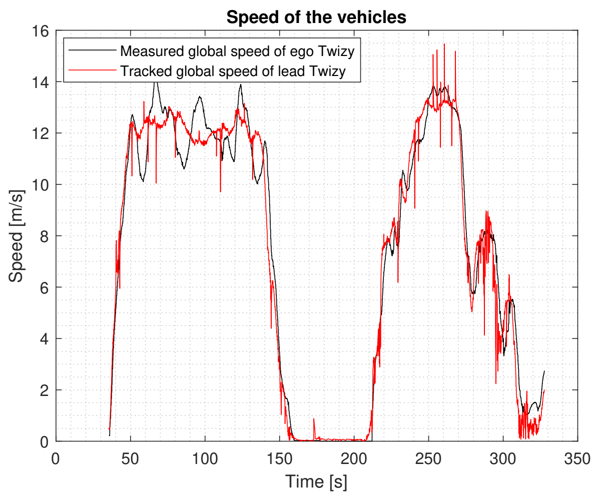
(a) Constant velocity model



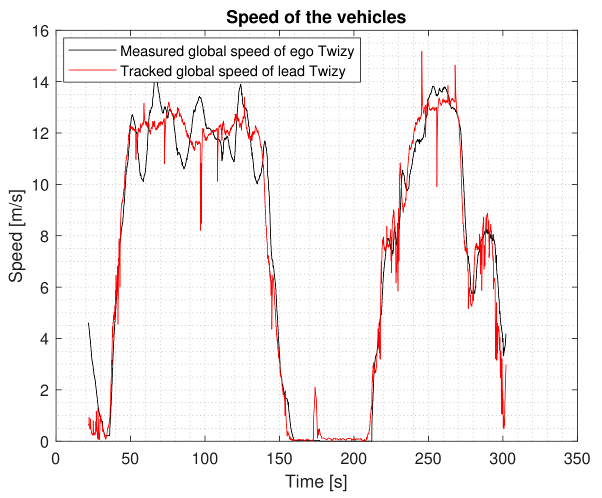
(b) Constant acceleration model

Figure 5.8: Global paths of the Twizys

The global speed of the lead Twizy was also reconstructed using vector addition of the tracked relative velocities of the lead Twizy and the recorded global speed of the ego Twizy. The recorded global speed of the ego Twizy and reconstructed global speed of the leader Twizy for the constant velocity and constant acceleration model are shown in Figure 5.9.



(a) Constant velocity model



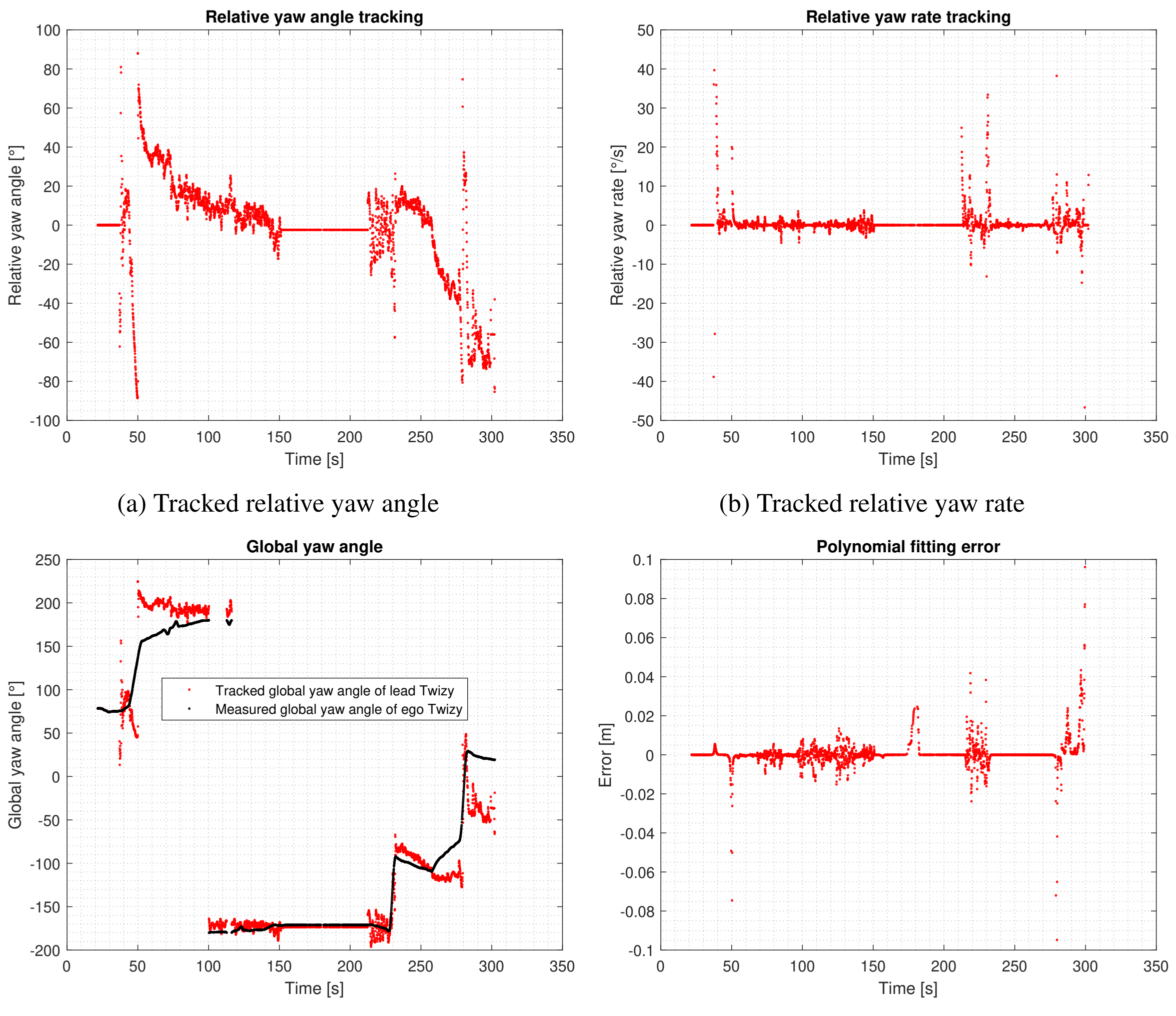
(b) Constant acceleration model

Figure 5.9: Global speeds of the Twizys

The reconstructed speed of the lead Twizy is similar that of the ego Twizy which is as expected because the Twizys travelled at nearly the same speeds. The reconstructed speed for the constant acceleration model has higher noise than the constant velocity model because the measurement noise for the acceleration is much higher as shown in Table 5.2.

5.3.3 Relative yaw angle, relative yaw rate, and global yaw angle tracking

As shown in Chapter 4, polynomial path fitting with only curvature constraint gives the best performance. The results for the constant acceleration model and constant velocity model are shown in Figure 5.10 and Appendix D respectively. The global yaw angle of the lead Twizy was reconstructed using the sum of the tracked relative yaw angle of the ego Twizy and the recorded global yaw angle of the ego Twizy.



(a) Tracked relative yaw angle

(b) Tracked relative yaw rate

(c) Global yaw angle

(d) Polynomial fitting error

Figure 5.10: Relative yaw angle, relative yaw rate, and global yaw angle for the CA model

The reconstructed global yaw angle of the lead Twizy is similar to the global yaw angle of the ego Twizy especially from $t = 100$ s to $t = 260$ s, but is noisy since the tracked positions for polynomial path fitting are noisy due to high measurement noise of the CA model. The sharp rises and drops correspond to the turns made by the Twizys except at $t = 100$ s. The sharp drop at $t = 100$ s is due to the global yaw exceeding 180° and reaching a large negative value close to $-180°$ as the measured yaw range is between $-180°$ and 180°. At at $t = 60$ s, the first turn starts. The measured global yaw angle of the ego Twizy increases sharply and then gradually as the ego Twizy completes the turn and comes on to the straight part of the road. The reconstructed global yaw angle of the lead Twizy follows the same trend but increases first at a faster rate than that of the ego Twizy and then drops gradually. At $t = 225$ s, the second turn is made and the global yaw angle of both Twizys increases sharply, before dropping gradually as they complete the turn and come on to the straight part of the road. A similar trend is seen at $t = 280$ s when the third turn starts.

The minimum thresholds on the global speed of the target vehicle for tracking relative yaw rate and yaw angle for the CV and CA models are taken as 1 m/s and 2.5 m/s respectively. The reason is that if the speed is too low or zero, then points used to calculate the polynomial curvature and slope lie too close or on top of each other respectively. This leads to a spike in curvature values that leads to inaccuracy and discontinuity. The corresponding thresholds for the gap between consecutive waypoints for the CV and CA models are 0.6311 m and 1.3699 m respectively.

5.3.4 Tracker performance evaluation

This subsection describes the criteria used to evaluate the performance of the tracker.

5.3.4.1 Track age and consistency

The data was recorded for 425.84 seconds i.e. the lead Twizy was in front of the ego Twizy for 425.84 seconds. For the constant velocity model, the age of the longest track is 292.04 seconds i.e. Twizy 1 is tracked for $292.04/425.84 = 68.58\%$ of the total time at one stretch. For the constant acceleration model, the age of the longest track is 280.49 seconds i.e. Twizy 1 is tracked for $280.49/425.84 = 65.86\%$ of the total time at one stretch. In terms of the age of the longest track, the CV model performs better than the CA model.

In addition to track age, track consistency is another measure of tracking performance. The longest track for the constant velocity model has a total of 3744 possible update instances. Out of these 3744 instances, the track is actually updated 3429 times and coasted (not updated) 315 times. The track consistency is calculated as $3429/3744 = 91.59\%$. For the constant acceleration model, the longest track has a total of 3583 update instances. Out of these 3583 update instances, the track is actually updated 3327 times and coasted 256 times i.e. the track consistency is $3327/3583 = 92.86\%$. In terms of track consistency, the CA model performs better than the CV model.

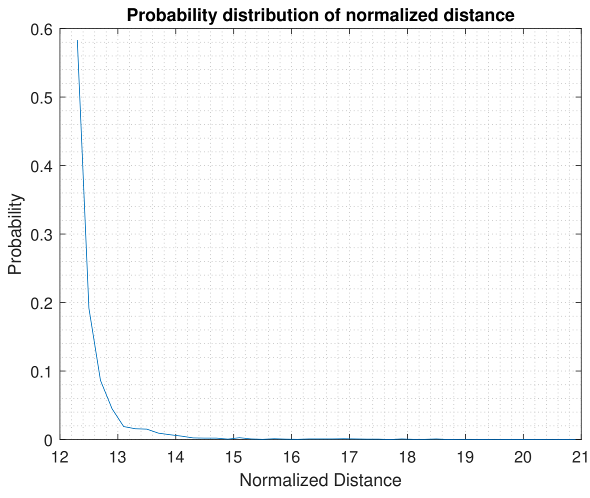
5.3.4.2 Filter performance

The NIS (Normalized Innovation Squared) is used to assess the performance of a Kalman filter when the ground truth is not available [37], [137]. For a single run, the NIS is given by:

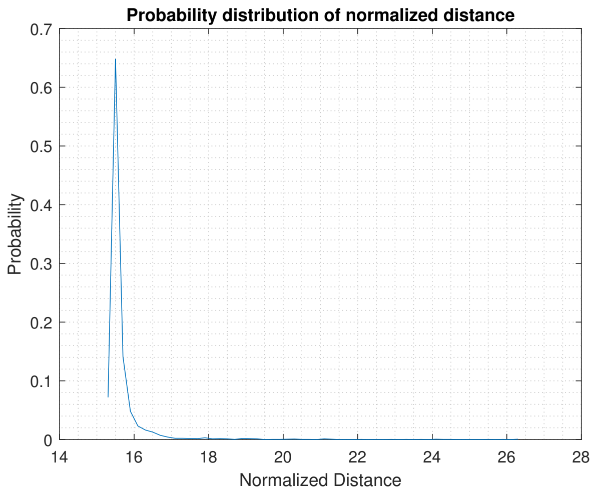
$$\bar{\epsilon} = \frac{1}{K}\sum_{k=1}^{K}\left(Z_{res,k}^{\top}\Sigma_{Z,k}^{-1}Z_{res,k} + \log(|\Sigma_{Z,k}|)\right) \tag{5.3}$$

where $Z_{res,k}$ is the innovation vector / residual vector, $\Sigma_{Z,k}$ is the innovation covariance matrix, and K is the total number of time steps. The dimension of the measurement vector is 6. The value of NIS over 16 time steps for the CV model is 12.7613 , and for the CA model is 16.7513. The NIS value is higher for the CA model because it has a higher measurement noise. Ideally, the obtained NIS values should lie within the range specified by the Chi-square table for given degrees of freedom and a given confidence value to pass the Chi-square goodness of fit test [37]. The two sided 95% confidence region for $16 \times 6 = 96$ degrees of freedom is given by $\begin{bmatrix}70.783 & 125\end{bmatrix}$ [138]. Dividing this range by the number of time steps i.e. 16 gives $\begin{bmatrix}4.4239 & 7.8125\end{bmatrix}$. The NIS values obtained for the CV model and the CA model do not lie in this interval and thus do not pass the Chi square goodness of fit test. The NIS values are higher than the upper limit of the interval. Therefore the filters for the CV model and the CA model are optimistic [37].

When the track having the longest age is updated with a new measurement using the cost matrix as shown in Chapter 3, the assigned measurement has a normalized distance to the track. The probability distributions of the normalized distances of the assigned measurements to the tracks having the longest age for the CV model and the CA model over the whole age of the tracks are shown in Figure 5.11.



(a) Probability distribution for the CV model



(b) Probability distribution for the CA model

Figure 5.11: Distribution of normalized distances

The CV model gives a distribution which is only to the right side of the mean value. The CA model gives a better distribution of the normalized distance in terms of being distributed evenly about the mean value because it is also tracks the acceleration of the targets and therefore is better in tracking the state of the lead Twizy. Ideally, both the distributions should be normal, but this is not the case here since the filters for the two models do not pass the Chi-square test. The reason is that both the constant velocity and constant acceleration models assume no co-relation between motions in x, y, and z coordinates. Markov process motion models e.g. the Singer acceleration model might give better results because it assumes co-relation between motions in x, y, and z coordinates.

The conclusions of this chapter are given along with the conclusions of Chapter 4 in Chapter 6.

6 Conclusions and Future work

This chapter explains the conclusions and future work.

6.1 Conclusions

The conclusions derived from Chapter 4 are:

1. In terms of track ages, both the CV and CA models perform well. This is as expected because there are only 2 targets and both are well separated.

2. For position and velocity tracking of targets that move at the same speed as the ego vehicle, the CV model is better than the CA model. For targets that move away from the ego vehicle, the CA model is more accurate than the CV model. In most real world scenarios except highways, the targets don't move at a constant speed i.e. they have accelerations and also don't move at the same speed as the ego vehicle. For such real world scenarios, the CA model is more suitable for tracking than the CV model because it estimates the accelerations and is thus able to estimate the velocities better. This is verified in the steady state cornering scenario where the CA model is found to be more accurate than the CV model because the targets have a lateral acceleration while turning. The larger measurement noise of the CA model helps in tracking targets moving away from the ego vehicle better than the CV model. This is verified in the straight road driving scenario.

3. Objects moving at the same speed as the ego vehicle are tracked better than objects moving away from the ego vehicle as the measurements become more noisy as the target moves away from the sensors.

4. For the steady state cornering case, the error in velocity tracking is much higher than the error in position tracking for both the CV and CA models because the targets move away from the ego vehicle and also move laterally. Since the targets start turning before the ego vehicle, the initial lateral movement of the targets is perpendicular to the orientation of the radar on the ego vehicle. The speed of this lateral movement is not measured by the radar as it measures the radial speed. The camera alone is not accurate in measuring the lateral speed.

5. For relative yaw angle and yaw rate tracking using polynomial path fitting, the third constraint i.e. curvature constraint only, gives the best performance in terms of errors and oscillations for both the CV and CA models.

6. Relative sideslip and sideslip rate tracking using tracked velocities and accelerations is suitable for targets that do not have a low relative velocity in the x direction of the ego vehicle frame.

This is verified in the steady state cornering scenario. This method can not be used with the constant velocity model because it does not track the accelerations.

The following conclusions can be drawn from Chapter 5:

1. As seen from the performance of the neural networks on being trained with parameter values and maximum track ages obtained from LHS (Latin Hypercube Sampling), there is no co-relation between the parameters and the age of the longest track i.e. the age of the longest track cannot be predicted for a given set of parameters and recorded measurements.

2. The CV model tracks the lead Twizy for a longer time at one stretch than the CA model but has lower track consistency than the CA model. The CA model has a better distribution of normalized distances for the longest track than the CV model as it also tracks the accelerations of the targets.

3. A single radar along with the IMU and GNSS is able to track the lead Twizy through sharp turns as seen from the age of the longest tracks for the CV model and the CA model. The tracked values from the CV model and the CA model match mostly with the three closest measurements from the radar as the lead Twizy is always in front of the ego Twizy except for the turns.

4. Based on the results of simulations in Chapter 4 and the results of track reconstruction in Chapter 5, the CA model is a better choice for the tracker since the CA model can track both velocities and accelerations, and can track targets moving away from the ego vehicle better than the CV model.

6.2 Future work

The possible additions to this project are:

1. In addition to the CV model and CA model, Markov process models e.g. Singer acceleration model or Semi-Markov process models e.g. Singer acceleration model with non-zero mean can be used. These models assume coupled motions in the x, y, and z directions and can track the motions of targets in real world scenarios better because moving objects in the real world have coupled motions.

2. The GNN tracker uses the normalized distance as a metric in the cost matrix. In place of NN (Nearest Neighbour) data association, probabilistic data association methods like OSPDA (Order Statistics Probabilistic Data Association) which uses association probabilities as a metric inside the cost matrix can be used since it performs better than the NN method and other PDA (Probabilistic Data Association) methods [32], [28] .

3. The GNN tracker only tracks a point target. To track an extended object, methods mentioned in Chapter 1 can be implemented. For example, GGIW (Gamma Gaussian Inverse Wishart)-PHD (Probability Hypothesis Density) tracking [125] uses an elliptical shape to track the extent of an object, GM (Gaussian Mixture)-PHD tracking [125] uses Gaussian mixtures to define any shape to be tracked.

4. Currently measurements are assigned to one global track for the camera and the radar using the cost matrix. Another possible method is to create separate tracks for the measurements from the radar and the camera, and fuse the tracks using T2TFP (Track to Track fusion with Fused Prediction).

Bibliography

[1] Bruce Brown. Evidence stacks up in favor of self-driving cars in 2016 nhtsa fatality report. https://www.digitaltrends.com/cars/2016-nhtsa-fatality-report. Accessed: 2020-07-16.

[2] Intelligent transport systems, road. https://ec.europa.eu/transport/themes/its/road_it. Accessed: 2020-07-16.

[3] The history and evolution of self-driving cars. https://kambria.io/blog/the-history-and-evolution-of-self-driving-cars. Accessed: 2020-07-16.

[4] Come for a drive with us! https://zoox.com/journal/. Accessed: 2020-07-16.

[5] Taxonomy SAE. Definitions for terms related to driving automation systems for on-road motor vehicles. *J3016, SAE International Standard*, 2018.

[6] Victor Talpaert, Ibrahim Sobh, B Ravi Kiran, Patrick Mannion, Senthil Yogamani, Ahmad El-Sallab, and Patrick Perez. Exploring applications of deep reinforcement learning for real-world autonomous driving systems. *arXiv preprint arXiv:1901.01536*, 2019.

[7] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[8] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006.

[9] Integrated cooperative automated vehicles. https://i-cave.nl. Accessed: 2020-07-16.

[10] Welcome to drive px2. https://docs.nvidia.com/drive/archive/5.0.5.0bL/nvvib_docs/index.html. Accessed: 2020-07-16.

[11] Merrill Ivan Skolnik. *Radar handbook*. McGraw-Hill Professional, 1990.

[12] Changzhen Qiu, Zhiyong Zhang, Huanzhang Lu, and Huiwu Luo. A survey of motion-based multitarget tracking methods. *Progress In Electromagnetics Research*, 62:195–223, 2015.

[13] GW Pulford. Taxonomy of multiple target tracking methods. *IEE Proceedings-Radar, Sonar and Navigation*, 152(5):291–304, 2005.

[14] Marcelo GS Bruno and Jose MF Moura. Multiframe detector/tracker: Optimal performance. *IEEE Transactions on Aerospace and Electronic Systems*, 37(3):925–945, 2001.

[15] Yair Barniv. Dynamic programming solution for detecting dim moving targets. *IEEE Transactions on Aerospace and Electronic Systems*, (1):144–156, 1985.

[16] Pavlina Konstantinova, Alexander Udvarev, and Tzvetan Semerdjiev. A study of a target tracking algorithm using global nearest neighbor approach. In *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'03)*, pages 290–295, 2003.

[17] Donald Reid. An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*, 24(6):843–854, 1979.

[18] Charles Morefield. Application of 0-1 integer programming to multitarget tracking problems. *IEEE Transactions on Automatic Control*, 22(3):302–312, 1977.

[19] Aubrey B Poore. Multidimensional assignments and multitarget tracking. *Partitioning Data Sets*, 19:169–196, 1993.

[20] Roy Jonker and Anton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[21] Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[22] Somnath Deb, Murali Yeddanapudi, Krishna Pattipati, and Yaakov Bar-Shalom. A generalized sd assignment algorithm for multisensor-multitarget state estimation. *IEEE Transactions on Aerospace and Electronic systems*, 33(2):523–538, 1997.

[23] Aubrey P Poore and Nenad Rijavec. A lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking. *SIAM Journal on Optimization*, 3(3):544–563, 1993.

[24] Ali Onder Bozdogan and Murat Efe. Improved assignment with ant colony optimization for multi-target tracking. *Expert Systems with Applications*, 38(8):9172–9178, 2011.

[25] Yaakov Bar-Shalom and Edison Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, 1975.

[26] Henry Leung, Zhijian Hu, and Martin Blanchette. Evaluation of multiple radar target trackers in stressful environments. *IEEE Transactions on Aerospace and Electronic Systems*, 35(2):663–674, 1999.

[27] Shyam Srinivasan, ND Gangadhar, and Hariharan Ramasangu. Design and development of probabilistic data association filters for complete object refinement. *SASTech-Technical Journal of RUAS*, 11(1):17–24, 2012.

[28] Yaakov Bar-Shalom, Thomas E Fortmann, and Peter G Cable. Tracking and data association, 1990.

[29] Robert J Fitzgerald. Development of practical pda logic for multitarget tracking by microprocessor. In *1986 American Control Conference*, pages 889–898. IEEE, 1986.

[30] JA Roecker and GL Phillis. Suboptimal joint probabilistic data association. *IEEE Transactions on Aerospace and Electronic Systems*, 29(2):510–517, 1993.

[31] James A Roecker. A class of near optimal jpda algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 30(2):504–510, 1994.

[32] Moon-Sik Lee and Yong-Hoon Kim. Automotive radar tracking of multi-target for vehicle cw/ca systems. *Mechatronics*, 14(1):143–151, 2004.

[33] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control*, 54(3):481–497, 2009.

[34] Roy L Streit and Tod E Luginbuhl. Maximum likelihood method for probabilistic multihypothesis tracking. In *Signal and Data Processing of Small Targets 1994*, volume 2235, pages 394–405. International Society for Optics and Photonics, 1994.

[35] Peter Willett, Yanhua Ruan, and R Streit. Pmht: Problems and some solutions. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):738–754, 2002.

[36] X Rong Li and Vesselin P Jilkov. Survey of maneuvering target tracking. part i. dynamic models. *IEEE Transactions on aerospace and electronic systems*, 39(4):1333–1364, 2003.

[37] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[38] Chaw-Bing Chang and Michael Athans. State estimation for discrete systems with switching parameters. *IEEE Transactions on Aerospace and Electronic Systems*, (3):418–425, 1978.

[39] Peter S Maybeck and Brian D Smith. Multiple model tracker based on gaussian mixture reduction for maneuvering targets in clutter. In *2005 7th International Conference on Information Fusion*, volume 1, pages 8–pp. IEEE, 2005.

[40] Henk AP Blom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE transactions on Automatic Control*, 33(8):780–783, 1988.

[41] David J Salmond. Mixture reduction algorithms for target tracking in clutter. In *Signal and Data Processing of Small Targets 1990*, volume 1305, page 434. International Society for Optics and Photonics, 1990.

[42] Darko Musicki and Sofia Suvorova. Tracking in clutter using imm-ipda-based algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 44(1):111–126, 2008.

[43] RJ Dempster, SS Blackman, and TS Nichols. Combining imm filtering and mht data association for multitarget tracking. In *Proceedings The Twenty-Ninth Southeastern Symposium on System Theory*, pages 123–127. IEEE, 1997.

[44] Xiao-Rong Li, Yaakov Bar-Shalom, and William Dale Blair. Engineer's guide to variable-structure multiple-model estimation for tracking. *Multitarget-multisensor tracking: Applications and advances.*, 3:499–567, 2000.

[45] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[46] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics, 1997.

[47] Daniel Alspach and Harold Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE transactions on automatic control*, 17(4):439–448, 1972.

[48] Igal Bilik and Joseph Tabrikian. Mmse-based filtering in presence of non-gaussian system and measurement noise. *IEEE Transactions on Aerospace and Electronic Systems*, 46(3):1153–1170, 2010.

[49] Genshiro Kitagawa. Non-gaussian state—space modeling of nonstationary time series. *Journal of the American statistical association*, 82(400):1032–1041, 1987.

[50] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

[51] Jayesh H Kotecha and Petar M Djuric. Gaussian sum particle filtering. *IEEE Transactions on signal processing*, 51(10):2602–2612, 2003.

[52] Matthew Orton and William Fitzgerald. A bayesian approach to tracking multiple targets using sensor arrays and particle filters. *IEEE Transactions on Signal Processing*, 50(2):216–223, 2002.

[53] Carine Hue, J-P Le Cadre, and Patrick Pérez. Tracking multiple objects with particle filtering. *IEEE transactions on aerospace and electronic systems*, 38(3):791–812, 2002.

[54] Rickard Karlsson and Fredrik Gustafsson. Monte carlo data association for multiple target tracking. *Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE*, 1(13):1–13, 2001.

[55] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Rao-blackwellized particle filter for multiple target tracking. *Information Fusion*, 8(1):2–15, 2007.

[56] Zia Khan, Tucker Balch, and Frank Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE transactions on pattern analysis and machine intelligence*, 27(11):1805–1819, 2005.

[57] Jaco Vermaak, Simon J Godsill, and Patrick Perez. Monte carlo filtering for multi target tracking and data association. *IEEE Transactions on Aerospace and Electronic systems*, 41(1):309–332, 2005.

[58] Jaco Vermaak, Arnaud Doucet, Perez Patrick, et al. Maintaining multi-modality through mixture tracking. In *null*, page 1110. IEEE, 2003.

[59] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *European conference on computer vision*, pages 28–39. Springer, 2004.

[60] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[61] Ronald PS Mahler. " statistics 101" for multisensor, multitarget data fusion. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):53–64, 2004.

[62] Ronald PS Mahler. Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic systems*, 39(4):1152–1178, 2003.

[63] Ronald Mahler. Phd filters of higher order in target number. *IEEE Transactions on Aerospace and Electronic systems*, 43(4):1523–1543, 2007.

[64] B-N Vo and W-K Ma. The gaussian mixture probability hypothesis density filter. *IEEE Transactions on signal processing*, 54(11):4091–4104, 2006.

[65] Hedvig Sidenbladh. Multi-target particle filtering for the probability hypothesis density. *arXiv preprint cs/0303018*, 2003.

[66] Ya-Dong Wang, Jian-Kang Wu, Ashraf A Kassim, and Weimin Huang. Data-driven probability hypothesis density filter for visual tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1085–1095, 2008.

[67] Nick Whiteley, Sumeetpal Singh, and Simon Godsill. Auxiliary particle implementation of probability hypothesis density filter. *IEEE Transactions on Aerospace and Electronic Systems*, 46(3):1437–1454, 2010.

[68] Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

[69] Branko Ristic, Daniel Clark, and Ba-Ngu Vo. Improved smc implementation of the phd filter. In *2010 13th International Conference on Information Fusion*, pages 1–8. IEEE, 2010.

[70] Syed Ahmed Pasha, Ba-Ngu Vo, Hoang Duong Tuan, and Wing-Kin Ma. A gaussian mixture phd filter for jump markov system models. *IEEE Transactions on Aerospace and Electronic systems*, 45(3):919–936, 2009.

[71] K Punithakumar, T Kirubarajan, and A Sinha. Multiple-model probability hypothesis density filter for tracking maneuvering targets. *IEEE Transactions on Aerospace and Electronic Systems*, 44(1):87–98, 2008.

[72] Ba-Ngu Vo, Ahmed Pasha, and Hoang Duong Tuan. A gaussian mixture phd filter for nonlinear jump markov models. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3162–3167. IEEE, 2006.

[73] Trevor M Wood. Interacting methods for manoeuvre handling in the gm-phd filter. *IEEE Transactions on Aerospace and Electronic Systems*, 47(4):3021–3025, 2011.

[74] Ramona Georgescu and Peter Willett. The multiple model cphd tracker. *IEEE Transactions on Signal Processing*, 60(4):1741–1751, 2012.

[75] Ronald PS Mahler, Ba-Tuong Vo, and Ba-Ngu Vo. Cphd filtering with unknown clutter rate and detection profile. *IEEE Transactions on Signal Processing*, 59(8):3497–3513, 2011.

[76] Feng Lian, Chongzhao Han, and Weifeng Liu. Estimating unknown clutter intensity for phd filter. *IEEE Transactions on Aerospace and Electronic Systems*, 46(4):2066–2078, 2010.

[77] Branko Ristic, D Clark, Ba-Ngu Vo, and Ba-Tuong Vo. Adaptive target birth intensity for phd and cphd filters. *IEEE Transactions on Aerospace and Electronic Systems*, 48(2):1656–1668, 2012.

[78] Emilio Maggio and Andrea Cavallaro. Learning scene context for multiple object tracking. *IEEE Transactions on Image Processing*, 18(8):1873–1884, 2009.

[79] Ju Hong Yoon, Du Yong Kim, Seung Hwan Bae, and Vladimir Shin. Joint initialization and tracking of multiple moving objects using doppler information. *IEEE Transactions on Signal Processing*, 59(7):3447–3452, 2011.

[80] Kusha Panta, Ba-Ngu Vo, and Sumeetpal Singh. Novel data association schemes for the probability hypothesis density filter. *IEEE Transactions on Aerospace and Electronic Systems*, 43(2):556–570, 2007.

[81] Cheng Ouyang, Hong-Bing Ji, and Ye Tian. Improved gaussian mixture cphd tracker for multitarget tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 49(2):1177–1191, 2013.

[82] Lin Lin, Yaakov Bar-Shalom, and Thiagalingam Kirubarajan. Data association combined with the probability hypothesis density filter for multitarget tracking. In *Signal and Data Processing of Small Targets 2004*, volume 5428, pages 464–475. International Society for Optics and Photonics, 2004.

[83] Evangeline Pollard, Benjamin Pannetier, and Michele Rombaut. Hybrid algorithms for multitarget tracking using mht and gm-cphd. *IEEE Transactions on Aerospace and Electronic Systems*, 47(2):832–847, 2011.

[84] Daniel E Clark and Judhith Bell. Data association for the phd filter. In *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 217–222. IEEE, 2005.

[85] Kusha Panta, Daniel E Clark, and Ba-Ngu Vo. Data association and track management for the gaussian mixture probability hypothesis density filter. *IEEE transactions on aerospace and electronic systems*, 45(3):1003–1016, 2009.

[86] Chee-Yee Chong, Shozo Mori, William H Barker, and Kuo-Chu Chang. Architectures and algorithms for track association and fusion. *IEEE Aerospace and Electronic Systems Magazine*, 15(1):5–13, 2000.

[87] Jos Elfring, Rein Appeldoorn, Sjoerd Van den Dries, and Maurice Kwakkernaat. Effective world modeling: Multisensor data fusion methodology for automated driving. *Sensors*, 16(10):1668, 2016.

[88] Ya Xue and Darryl Morrell. Target tracking and data fusion using multiple adaptive foveal sensors. In *Proceedings of the Sixth International Conference of Information Fusion*, volume 1, pages 326–333, 2003.

[89] Kuo Chu Chang, Tian Zhi, and Rajat K Saha. Performance evaluation of track fusion with information matrix filter. *IEEE Transactions on Aerospace and Electronic Systems*, 38(2):455–466, 2002.

[90] Michael Darms and Hermann Winner. A modular system architecture for sensor data processing of adas applications. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 729–734. IEEE, 2005.

[91] JB Gao and Chris J Harris. Some remarks on kalman filters for the multisensor fusion. *Information Fusion*, 3(3):191–201, 2002.

[92] Sofie Nilsson and Axel Klekamp. A comparison of architectures for track fusion. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 517–522. IEEE, 2015.

[93] Alexei Makarenko, Alex Brooks, Tobias Kaupp, Hugh Durrant-Whyte, and Frank Dellaert. Decentralised data fusion: A graphical model approach. In *2009 12th International Conference on Information Fusion*, pages 545–554. IEEE, 2009.

[94] Simon J Julier and Jeffrey K Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, volume 4, pages 2369–2373. IEEE, 1997.

[95] Hassen Fourati. *Multisensor data fusion: from algorithms and architectural design to applications*. CRC press, 2017.

[96] Marc Reinhardt, Benjamin Noack, Pablo O Arambel, and Uwe D Hanebeck. Minimum covariance bounds for the fusion under unknown correlations. *IEEE Signal Processing Letters*, 22(9):1210–1214, 2015.

[97] Wolfgang Niehsen. Information fusion based on fast covariance intersection filtering. In *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002.(IEEE Cat. No. 02EX5997)*, volume 2, pages 901–904. IEEE, 2002.

[98] Dietrich Franken and Andreas Hupper. Improved fast covariance intersection for distributed data fusion. In *2005 7th International Conference on Information Fusion*, volume 1, pages 7–pp. IEEE, 2005.

[99] Abder Rezak Benaskeur. Consistent fusion of correlated data sources. In *IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02*, volume 4, pages 2652–2656. IEEE, 2002.

[100] Yan Zhou and Jianxun Li. Robust decentralized data fusion based on internal ellipsoid approximation. *IFAC Proceedings Volumes*, 41(2):9964–9969, 2008.

[101] Xin Tian and Yaakov Bar-Shalom. On algorithms for asynchronous track-to-track fusion. In *2010 13th International Conference on Information Fusion*, pages 1–8. IEEE, 2010.

[102] Yaakov Bar-Shalom. On the track-to-track correlation problem. *IEEE Transactions on Automatic control*, 26(2):571–572, 1981.

[103] Yaakov Bar-Shalom and Leon Campo. The effect of the common process noise on the two-sensor fused-track covariance. *IEEE Transactions on Aerospace and Electronic Systems*, (6):803–805, 1986.

[104] Hongyan Zhu, Qiaozhu Zhai, Mingwei Yu, and Chongzhao Han. Estimation fusion algorithms in the presence of partially known cross-correlation of local estimation errors. *Information Fusion*, 18:187–196, 2014.

[105] M.A.M. (Maarten) Kloompmakers. “object tracking for autonomous and cooperative driving”. TU Eindhoven, 2020.

[106] Distances between current and predicted measurements of tracking filter. https://www.mathworks.com/help/driving/ref/trackingekf.distance.html. Accessed: 2020-07-16.

[107] Franciscus Nicolaas Hoogeboom. "safety of automated vehicles: design, implementation and analysis". TU Eindhoven, 2020.

[108] Ark-3520p. https://www.advantech.com/products/1-2jkd2d/ark-3520p/mod_6666bf1e-af4f-47b6-8006-1a0a89eb3c93. Accessed: 2020-07-16.

[109] Mid-range radar (mrr) sensor. https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/automatic-emergency-braking/mid-range-radar-sensor-(mrr)/. Accessed: 2020-07-16.

[110] automotive intelligent transport systems. https://www.etsi.org/technologies/automotive-intelligent-transport. Accessed: 2020-07-16.

[111] Pc engines. https://www.pcengines.ch/apu2.htm. Accessed: 2020-07-16.

[112] Evk-8/evk-m8. https://www.u-blox.com/en/product/evk-8evk-m8. Accessed: 2020-07-16.

[113] Acceleration sensor mm5.10. http://www.bosch-motorsport.de/content/downloads/Raceparts/en-GB/51546379119226251.html. Accessed: 2020-07-16.

[114] Nxp cocoon radar demo kit. https://www.nxp.com/video/nxp-cocoon-radar-demo-kit:RADARDEMO-VID. Accessed: 2020-07-16.

[115] Frequency-modulated continuous-wave radar (fmcw radar). https://www.radartutorial.eu/02.basics/Frequency%20Modulated%20Continuous%20Wave%20Radar.en.html. Accessed: 2020-07-16.

[116] Sekonix nvidia drive camera specification. http://sekolab.com/products/camera/. Accessed: 2020-07-16.

[117] Gigabit multimedia serial link (gmsl) serdes ics. https://www.maximintegrated.com/en/products/interface/high-speed-signaling/gmsl-serdes.html. Accessed: 2020-07-16.

[118] P Russel Norvig and S Artificial Intelligence. *A modern approach*. Prentice Hall, 2002.

[119] Nxp bluebox: Autonomous driving development platform. https://www.nxp.com/design/development-boards/automotive-development-platforms/nxp-bluebox-autonomous-driving-development-platform:BLBX. Accessed: 2020-07-16.

[120] Steve C Talbot and Shangping Ren. Comparision of fieldbus systems can, ttcan, flexray and lin in passenger vehicles. In *2009 29th IEEE International Conference on Distributed Computing Systems Workshops*, pages 26–31. IEEE, 2009.

[121] Cortex-a72. https://developer.arm.com/ip-products/processors/cortex-a/cortex-a72. Accessed: 2020-07-16.

[122] What is asil-d? https://www.aptiv.com/newsroom/article/what-is-asil-d. Accessed: 2020-07-16.

[123] The ultimate sensor battle: Lidar vs radar. https://medium.com/@intellias/the-ultimate-sensor-battle-lidar-vs-radar-2ee0fb9de5da. Accessed: 2020-07-16.

[124] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[125] Extended object tracking and performance metrics evaluation. https://www.mathworks.com/help/fusion/examples/extended-object-tracking.html. Accessed: 2020-07-16.

[126] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[127] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099. IEEE, 2015.

[128] Tjalling Talsma. Development of a robust reference path generator for lateral vehicle control. TU Delft, 2018.

[129] Antoine Schmeitz, Jeroen Zegers, Jeroen Ploeg, and Mohsen Alirezaei. Towards a generic lateral control concept for cooperative automated driving theoretical and experimental evaluation. In *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 134–139. IEEE, 2017.

[130] Vandermonde matrix. https://mathworld.wolfram.com/VandermondeMatrix.html. Accessed: 2020-07-16.

[131] Anders Olsson, Göran Sandberg, and Ola Dahlblom. On latin hypercube sampling for structural reliability analysis. *Structural safety*, 25(1):47–68, 2003.

[132] Peter s.c. heuberger. https://research.tue.nl/en/persons/peter-sc-heuberger. Accessed: 2020-09-20.

[133] Multi-object trackers. https://www.mathworks.com/help/fusion/multi-object-trackers.html. Accessed: 2020-07-16.

[134] What are system objects? https://www.mathworks.com/help/matlab/matlab_prog/what-are-system-objects.html. Accessed: 2020-07-16.

[135] Constant false alarm rate (cfar) detection. https://www.mathworks.com/help/phased/examples/constant-false-alarm-rate-cfar-detection.html. Accessed: 2020-07-16.

[136] Choose a multilayer neural network training function. https://www.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html. Accessed: 2020-07-16.

[137] Aliakbar A Gorji, Ratnasingham Tharmarasa, and Thia Kirubarajan. Performance measures for multiple target tracking problems. In *14th International Conference on Information Fusion*, pages 1–8. IEEE, 2011.

[138] Critical values of the chi-square distribution. https://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm. Accessed: 2020-09-20.

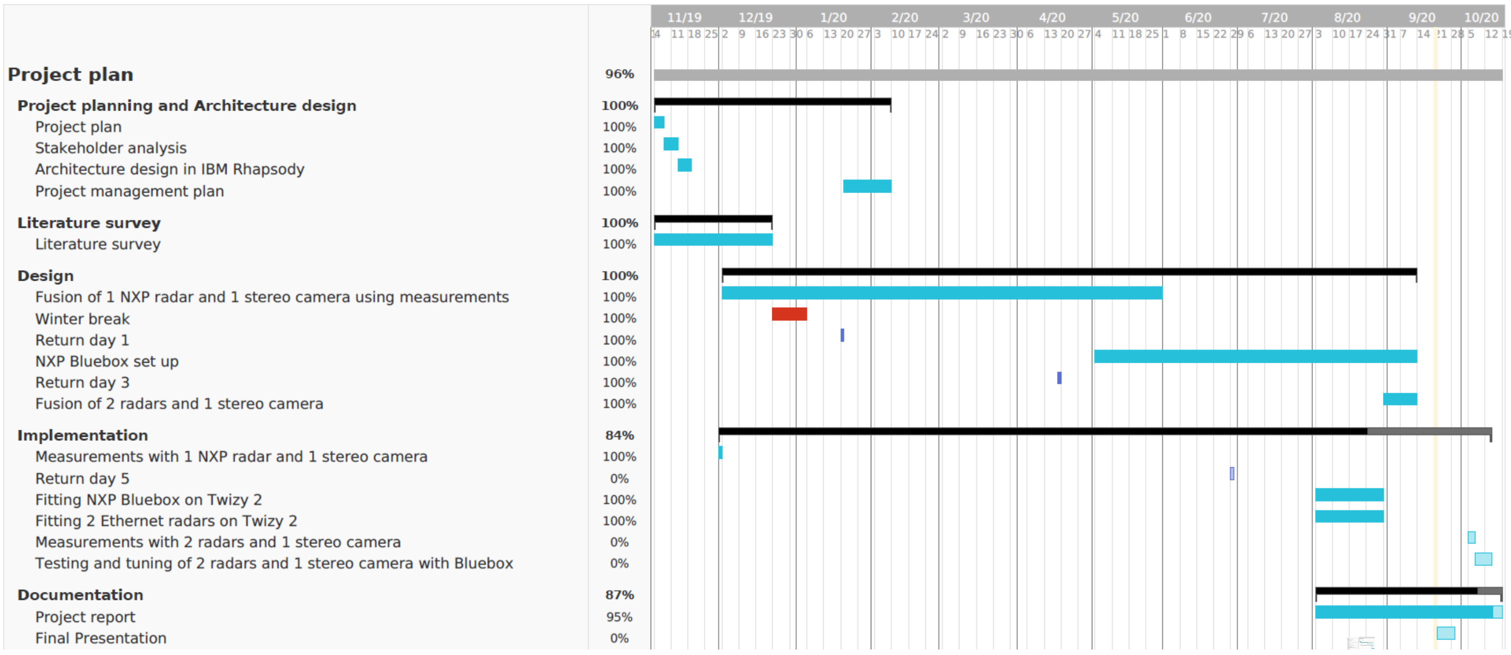
[139] Gerrit Muller. Cafcr: A multi-view method for embedded systems architecting; balancing genericity and specificity. 2004.

[140] Sysml faq: What is sysml?, what is mbse?, who created sysml? https://sysmlforum.com/sysml-faq/. Accessed: 2020-07-16.

A Project management

A.1 Project plan

The project plan is shown in Figure A.1.



Project plan	96%
Project planning and Architecture design	100%
Project plan	100%
Stakeholder analysis	100%
Architecture design in IBM Rhapsody	100%
Project management plan	100%
Literature survey	100%
Literature survey	100%
Design	100%
Fusion of 1 NXP radar and 1 stereo camera using measurements	100%
Winter break	100%
Return day 1	100%
NXP Bluebox set up	100%
Return day 3	100%
Fusion of 2 radars and 1 stereo camera	100%
Implementation	84%
Measurements with 1 NXP radar and 1 stereo camera	100%
Return day 5	0%
Fitting NXP Bluebox on Twizy 2	100%
Fitting 2 Ethernet radars on Twizy 2	100%
Measurements with 2 radars and 1 stereo camera	0%
Testing and tuning of 2 radars and 1 stereo camera with Bluebox	0%
Documentation	87%
Project report	95%
Final Presentation	0%

Figure A.1: Project plan

As seen in Figure A.1, the 2 Ethernet based NXP Cocoon radars and the NXP Bluebox have been fitted on Twizy 2. The design of the Simulink Real-Time model is also complete. Measurements with the 2 radars and one stereo camera will be recorded in October and will be used to tune tracker parameters using LHS (Latin Hypercube Sampling). The results for real time tracking using the 2 radars and the stereo camera will be added in an additional chapter in the project report in the second week of October.

A.2 Sprint planning

During the initial phase of the project from November 2019 to January 2020, progress meetings were held with the customer (i-CAVE) every 2 weeks. Starting from February 2020 progress meetings were held with the PSG (Project Steering Group) every two weeks. The PSG consists of the following members:

1. Tom van der Sande - Customer (i-CAVE)
2. Mohsen Alirezaei - University supervisor (TU/e)
3. Peter Heuberger - ASD/MSD program manager (TU/e)

Since the progress meetings were held every two weeks, the duration of each sprint was 2 weeks. The primary mode of communication with the PSG was e-mail. The V-model approach was used to plan the project as shown in Figure A.2.

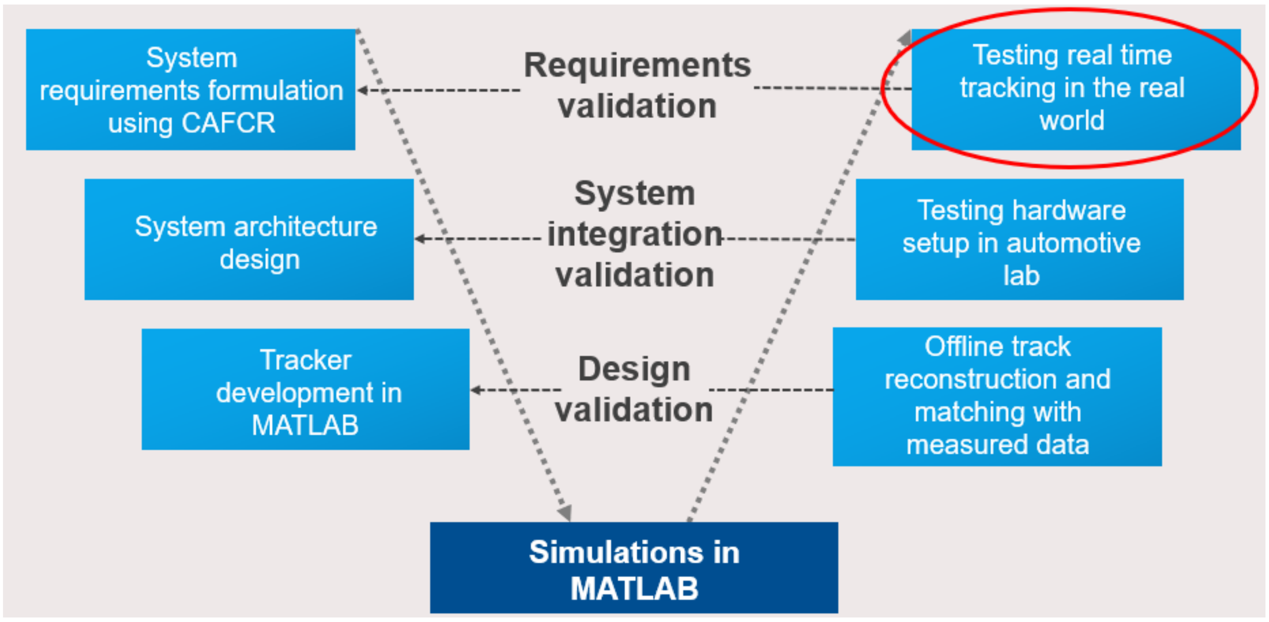

System requirements formulation using CAFCR
Requirements validation
Testing real time tracking in the real world
System architecture design
System integration validation
Testing hardware setup in automotive lab
Tracker development in MATLAB
Design validation
Offline track reconstruction and matching with measured data
Simulations in MATLAB

Figure A.2: The V-model approach

At the time of uploading this report, the last part of this V-model i.e. testing real-time in the real world was still being done. The system architecture design and formulated requirements are explained in Appendix B.

A.3 Project retrospective

The project can be divided into three phases based on the working situation arising from the COVID-19 pandemic:

- **November 2019 to February 2020**: During this phase, the work on the project was done in the university. This period was spent on literature study and the initial exploration of the automated driving toolbox of MATLAB. By the end of January 2020, the major part of the literature study was carried out and a minimum working example of the tracker based on real-world data measured in November 2019 was prepared.

- **March 2020 to June 2020**: During this phase, the university was shut down completely because of the COVID-19 pandemic and the current work-from-home situation started. This period was spent on simulations to verify the design of the tracker, tuning the parameters for reconstruction of tracks obtained from real-world data using Latin Hypercube Sampling, and the initial setup of the NXP Bluebox.
- **July 2020 to October 2020**: In July, access to the automotive lab in the university was obtained for implementing the tracker on Twizy using automotive Ethernet based NXP radars and the NXP Bluebox. Also, the polynomial path tracing method to track the yaw angle and yaw rate of the lead vehicle was verified in simulations. In August, the radars and the Bluebox were fitted on Twizy 2 and the project report was written.

According to the author, the following activities were executed well by the trainee:

- **Understanding and implementation of scientific concepts:** The methods used in the automated driving toolbox in MATLAB, and the methods suggested by the PSG (Project Steering Group) such as the polynomial path tracing method to track the yaw angle and yaw rate of the lead vehicle were understood and implemented well in MATLAB.
- **Adjustment to the work-from-home situation:** Working from home required changes in the working schedule and mindset. The trainee was able to put in consistent effort to adjust to the new situation and achieve good results.

The following things could have been improved by the trainee:

1. **Pro-activeness in the literature survey:** For example, when manual tuning of tracker parameter values produced sub-optimal results, the trainee could have more actively searched for sampling methods from a multi-dimensional parameter space. The Latin Hypercube Sampling (LHS) method was suggested by Peter Heuberger from the PSG.
2. **Speed of execution:** The speed at which tasks were completed could have been improved. This would have left more time for the implementation of the tracker on Twizy 2. At the time of submitting this report, real-time tracking on Twizy 2 was still being implemented.
3. **Risk management:** Risk management could have been improved in view of the work-from-home situation. Since the trainee was less familiar with the operating system of the Bluebox i.e. Linux than Windows, he could have started the Bluebox setup process earlier to ensure that implementation of the tracker using the Bluebox would be completed in the planned time.

B System architecture design

For designing the system architecture, the CAFCR method [139] has been used.

CAFCR

The 5 parts of the CAFCR model are:

1. **C**ustomer Objectives: Customer concerns
2. **A**pplication: Application drivers
3. **F**unction: Technical functions, use cases
4. **C**oncept: Functional decomposition
5. **R**ealization: Design implementation and analysis

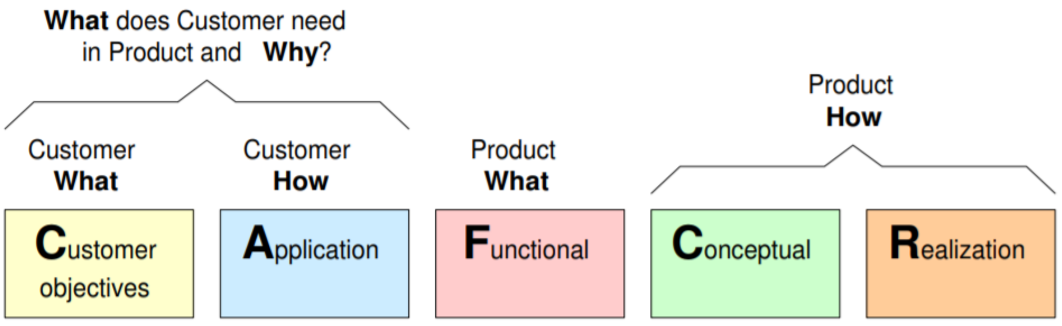


Figure B.1: The CAFCR model [139]

The first step is to write the function part i.e. the primary functions which the product should perform. For this project, the primary function is:

Track all road users in the FOV (Field-of-View) of the sensors till they remain in the FOV of the sensors.

Keeping this primary function in mind, the concerns of the stakeholders and related application drivers are written in detail. Using stakeholder concerns and application drivers, functional and non-functional requirements are formulated for the concept and realization steps. Stakeholder concerns for this project are explained in the next section.

Stakeholder concerns and application drivers

The concerns of the stakeholders of this project and related application drivers are shown in Table B.1.

Table B.1: Concerns and application drivers of stakeholders

Stakeholder	Concerns	Application driver
Project group 2 – Cooperative vehicle control and state estimation	**C1.1** Choosing target from multiple objects around Twizy 2	**A1.1.1** Tracking all road users around Twizy 2
	C1.2 Plan path to follow the selected target	**A1.2.1** Develop and implement path planning methods
	C1.3 Provide control input to Twizy 2 actuators to move on planned path	**A1.3.1** Develop and implement controllers for path following
	C1.4 Achieve above mentioned goals with existing RTT (Real-Time Target) machine	**A1.4.1** Implement methods which are accurate but computationally efficient
Project 7 - Demonstrator	**C2.1** Demonstrating Twizy 2 following Twizy 1 autonomously in the real world	**A2.1.1** Demonstration of project groups 1-6
Project group 1 – Sensing and mapping	**C3.1** Mapping the surroundings of Twizy 2	**A3.1.1** Detection and classification of all objects around Twizy 2
Project 5 – Human factors	**C4.1** Improving the interaction of the driver with the Twizy	**A4.1.1** Designing an interactive and intuitive internal HMI
	C4.2 Making the behavior of Twizy 2 predictable to other road users	**A4.2.1** Designing an interactive and intuitive external HMI
Project 4 - Communication	**C5.1** Using V2V (Vehicle to Vehicle) communication for platooning	**A5.1.1** Using communication with Twizy 1 to track and follow it
i-CAVE (integrated Cooperative Automated Vehicles) consortium	**C6.1** Achieving SAE level 5 autonomy	**A6.1.1** Advancement from SAE level 2 to SAE level 3
	C6.2 Vehicle Platooning	**A6.2.1** Vehicle platooning using in-vehicle and inter-vehicle controllers and communication **A6.2.2** Autonomous platoons should integrate into normal traffic
	C6.3 Safety	**A6.3.1** Automatically maintaining safe speed and time gap with respect to the leading vehicle
Drivers	**C7.1** Comfort	**A7.1.1** Driving with eyes off the road and hands off the steering wheel

	C7.2 Safety	**A7.2.1** Automatically maintaining safe speed and time gap with respect to the leading vehicle
	C7.3 Fuel Efficiency	**A7.3.1** Platooning improves fuel efficiency
PDEng trainee	**C8.1** Completing PDEng graduation project	**A8.1.1** Implementing sensor fusion on Twizy 2
	C8.2 Contributing to improvement in autonomous driving and vehicle platooning	**A8.2.1** Gaining knowledge of sensor fusion methods and hardware used on Twizy 2

Requirements formulated on the basis of concerns and application drivers are explained in the next section.

Requirements formulation

The requirements obtained from Table B.1 can be classified into functional and non-functional requirements. The functional requirements are:

1. **RF1**: The system *must* track all road users in the FOV (Field-of-View) of the radars and cameras on Twizy 2 till the time the road users are being detected by the radars and cameras.
2. **RF2**: The system *must* track the positions, velocities and accelerations of all the road users in the FOV of the radars and cameras.
3. **RF3**: The system *must* select the MIO (Most Important Object) i.e. Twizy 1 from all the tracked road users.
4. **RF4**: The system *must* track the yaw angle and yaw rate of the MIO.

The non-functional requirements are:

1. **RNF1**: The system *must* be compatible with the R2017b version of Simulink Real-Time.
2. **RNF2**: The system *must* use the measurements from the GNSS and IMU along with the measurements from the radars and the cameras to track all road users in the FOV of the radars and cameras on Twizy 2.
3. **RNF3**: The system *must* update at the frequency of the fastest sensor i.e. the radar. This means that the system must process all new measurements in real time within the sampling interval of the fastest sensor i.e. the radar.
4. **RNF4**: The system *must* use Ethernet based NXP Cocoon radars.
5. **RNF5**: The system *must* use the NXP Bluebox to process measurements from the NXP Cocoon radars.

6. **RNF6**: The system *must* use ROS (Robot Operating System) on the NXP Bluebox and the RTT (Real Time Target) machine to receive measurements from the NXP Cocoon radars.

7. **RNF7**: The system *should* use V2V (Vehicle to Vehicle) communication along with measurements from radars, cameras, GNSS, and the IMU to track the MIO.

Based on the formulated requirements, the system structure and behavior can be visualized using SysML (System Modelling Language) [140] in IBM Rhapsody. This visualization is explained in the next section.

System architecture design in IBM Rhapsody

The first step in designing the system architecture is to define the boundary of the system. The system here is assumed to have both hardware and software. It is assumed to only interact with the RTT machine, therefore the RTT machine is the only actor for the system. The system context diagram is shown in Figure B.2.

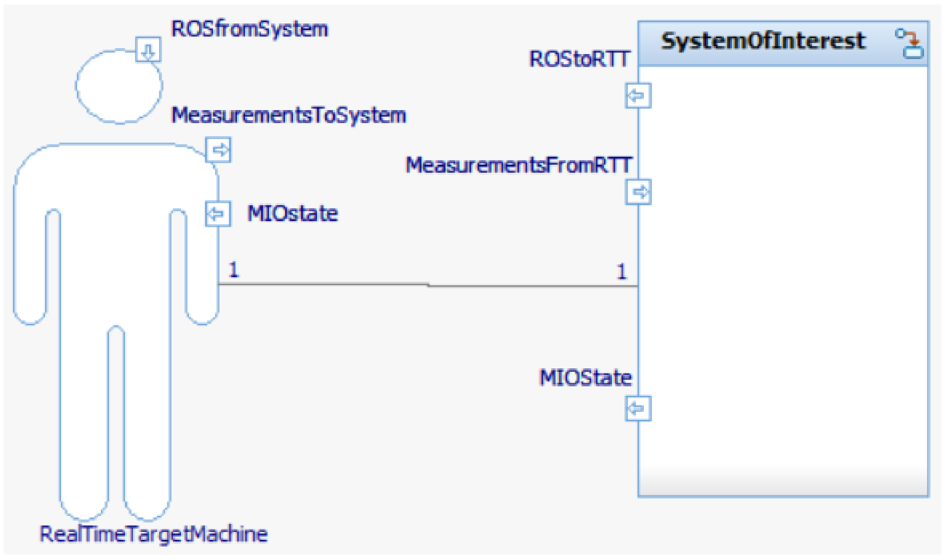


Figure B.2: System context diagram

Since there is only one RTT machine, there is a 1 to 1 connectivity as shown by the line connecting the actor and the system. The internal structure of the system can be visualized in the internal block diagram shown in Figure B.3. The radars in the system send measurements to the Bluebox via automotive Ethernet over a ROS network because the radars are Ethernet based. The Bluebox sends these ROS messages to the RTT machine. The RTT machine receives these messages and sends them to the software part of the system i.e. software modules inside the RTT along with measurements from the cameras, GNSS, and the IMU. The data pre-processing module receives these measurements and sends them to the tracker module after filtering them. The tracker module outputs confirmed tracks of the objects and sends them to the MIO tracking module. The tracked state of the MIO is sent to software modules inside the RTT that are not part of the system i.e. file logging module and controller module.

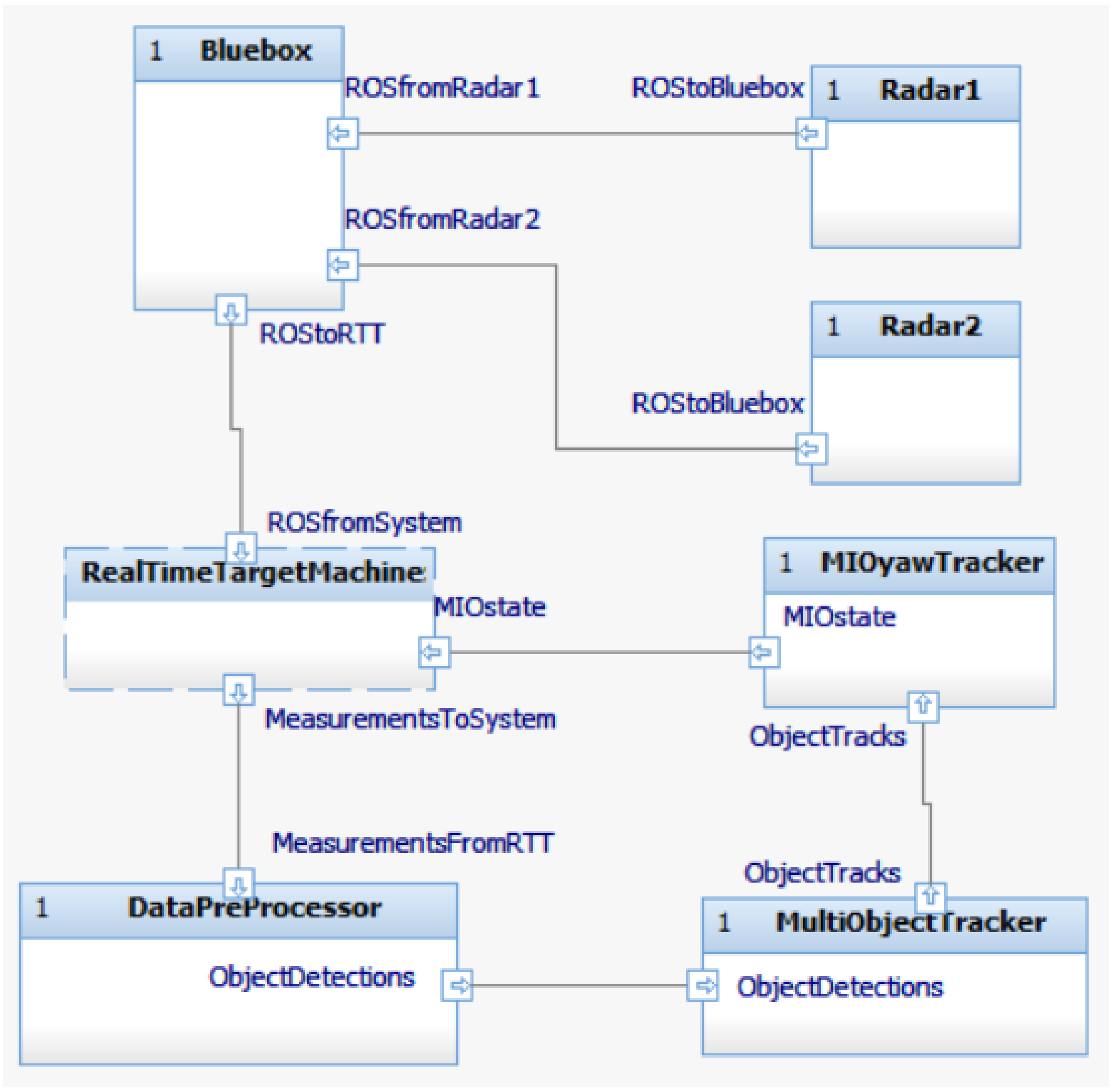


Figure B.3: Internal block diagram

The state chart of the system showing the flow of events is shown in Figure B.4. States are represented by boxes and events are represented by arrows. An event leads to a transition between states. When the reception of ROS messages starts, the system enters the state of receiving ROS messages. When the messages have been received, the system starts sending the messages to the RTT machine. When the messages have been sent, the system starts waiting for receiving measurements from the RTT machine. When the reception of measurements starts, the system enters the state of receiving the measurements. When the measurements have been received, the system starts pre-processing them i.e. filtering and creating *objectDetection* objects from the measurements as shown in Chapter 4. When the objects have been created and sent to the tracker, the tracker starts tracking objects. When confirmed tracks have been created, the system starts identifying the MIO and tracking its yaw angle and yaw rate. After this is complete, the MIO tracks are sent to the different software modules inside the RTT.

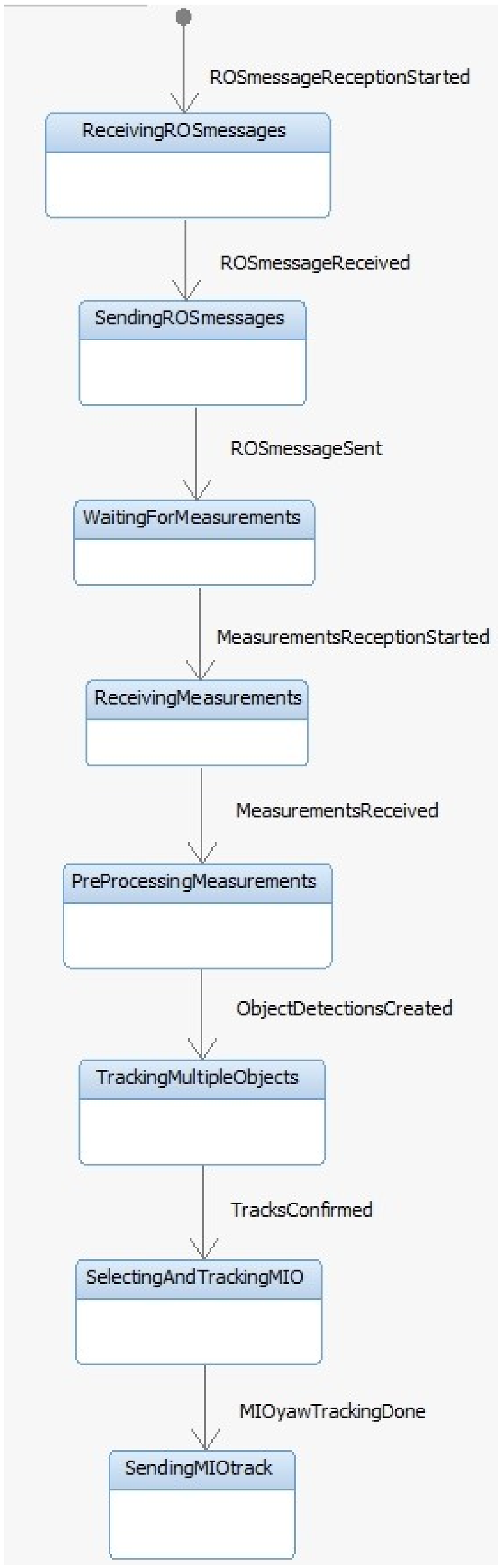


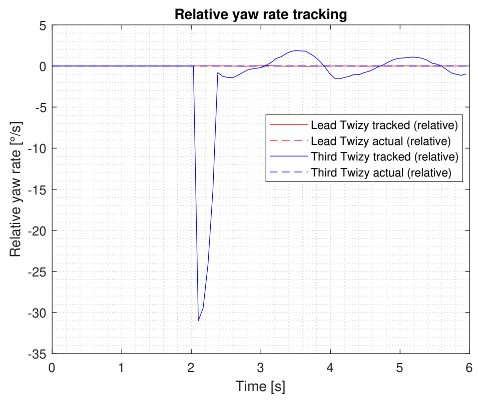
Figure B.4: State chart

C Simulation results for yaw angle and yaw rate tracking

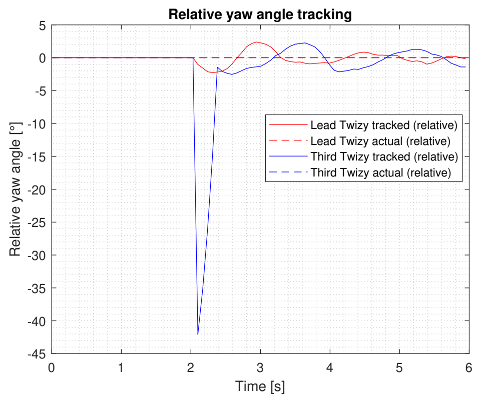
This appendix contains the results of relative yaw angle and yaw rate tracking using polynomial path fitting for the 3 simulation scenarios shown in Chapter 4. The results are for 4 cases of constraints for the constant velocity model, and 3 cases of constraints for the constant acceleration model.

C.1 Driving on a straight path

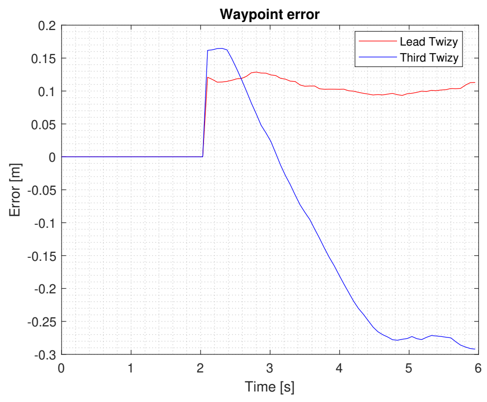
C.1.1 Constant velocity model



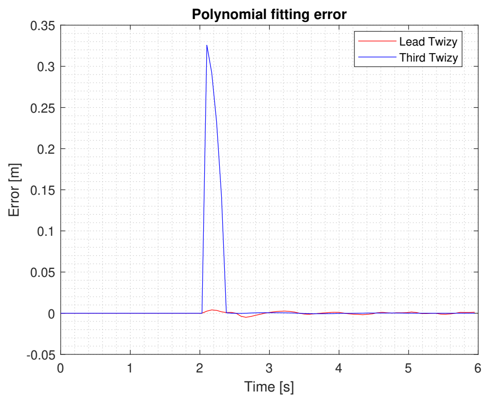
(a) Yaw rate tracking



(b) Yaw angle tracking

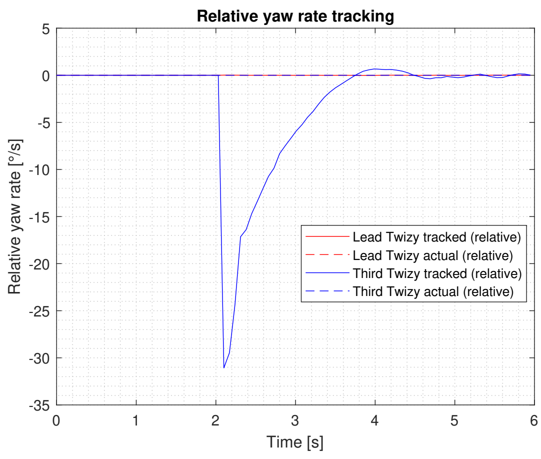


(c) Waypoint error

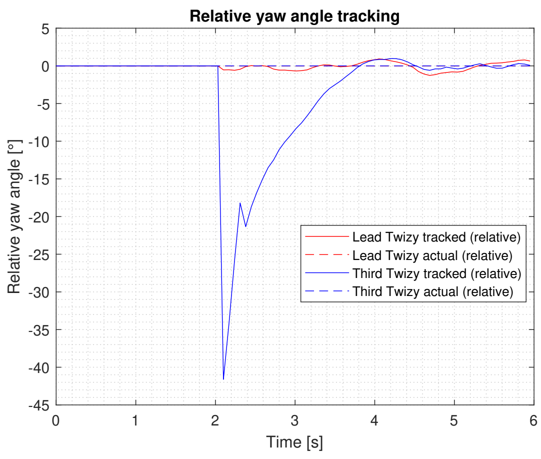


(d) Polynomial fitting error

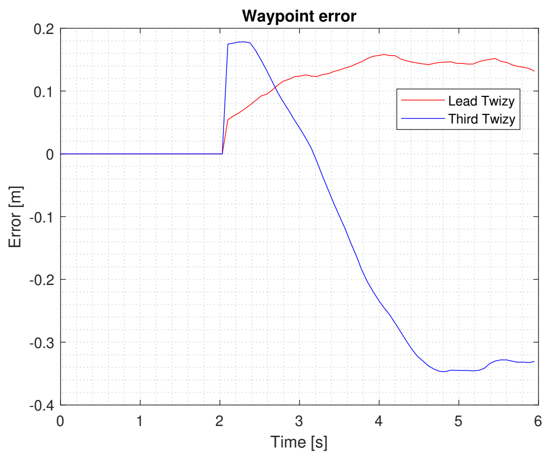
Figure C.1: Results for case 1: No constraints



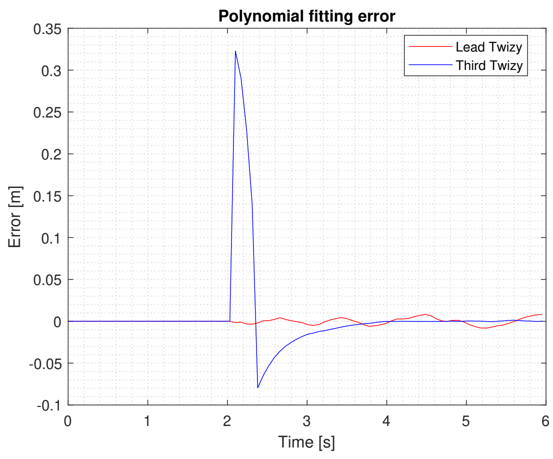
(a) Yaw rate tracking



(b) Yaw angle tracking

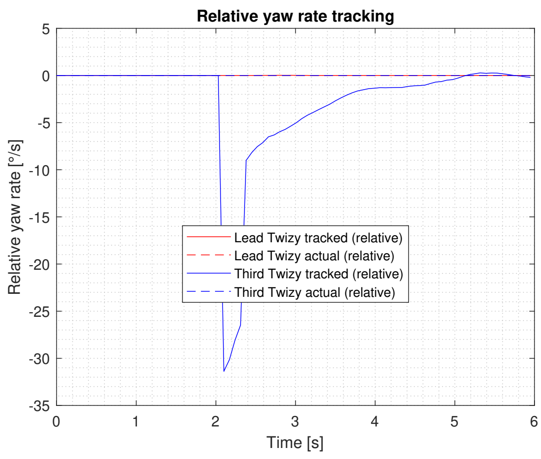


(c) Waypoint error

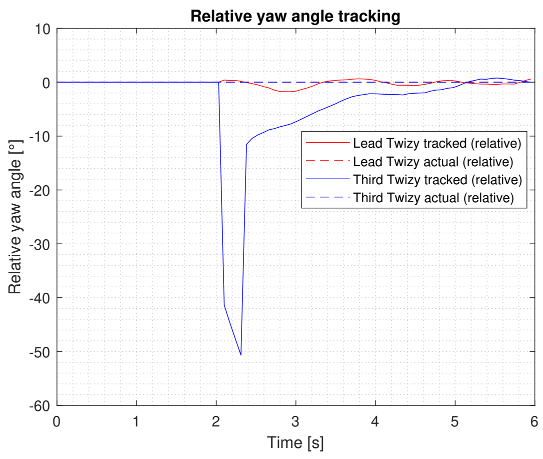


(d) Polynomial fitting error

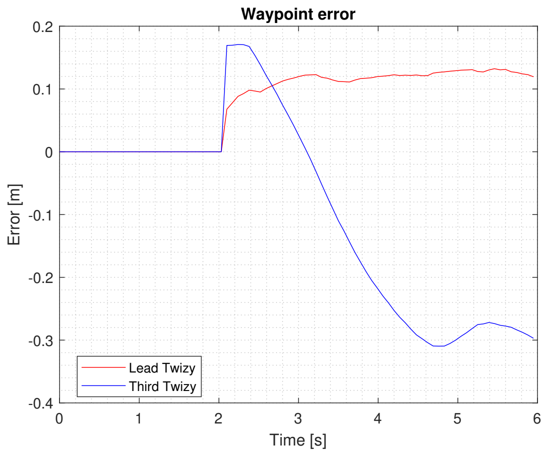
Figure C.2: Results for case 2: Heading constraint only



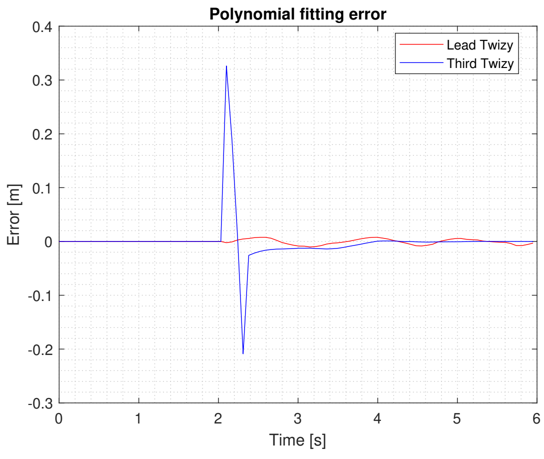
(a) Yaw rate tracking



(b) Yaw angle tracking

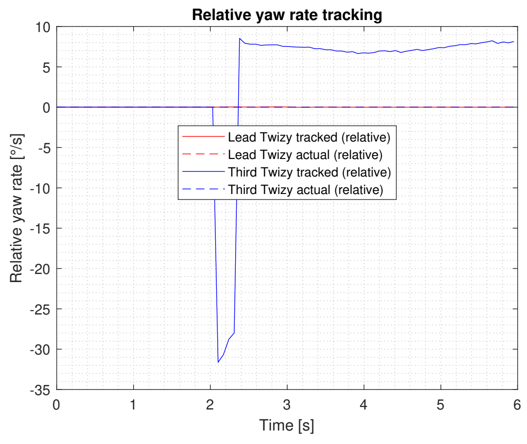


(c) Waypoint error

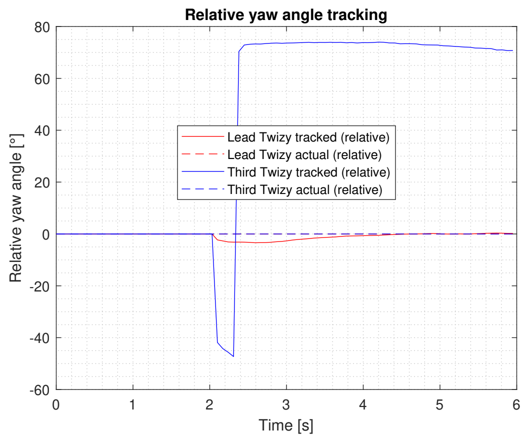


(d) Polynomial fitting error

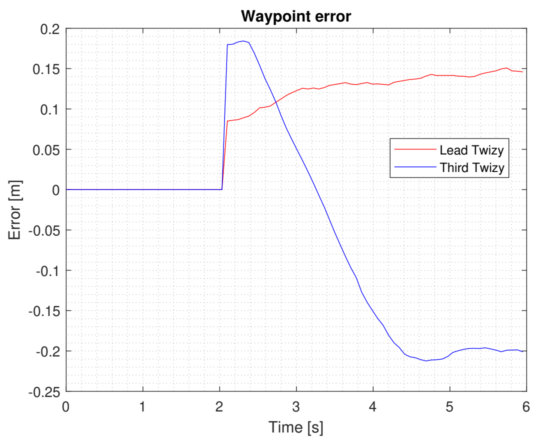
Figure C.3: Results for case 3: Curvature constraint only



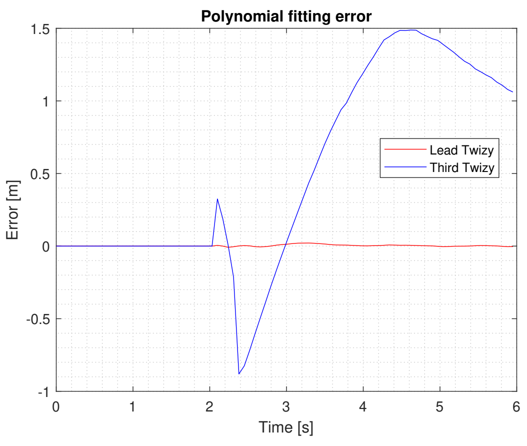
(a) Yaw rate tracking



(b) Yaw angle tracking



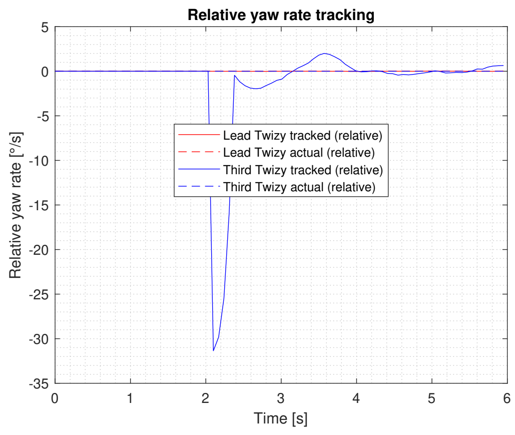
(c) Waypoint error



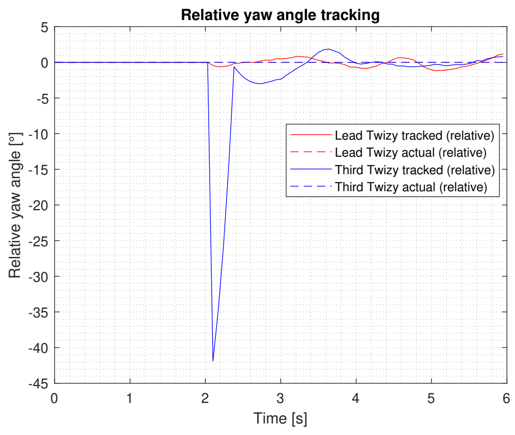
(d) Polynomial fitting error

Figure C.4: Results for case 4: Both heading and curvature constraints

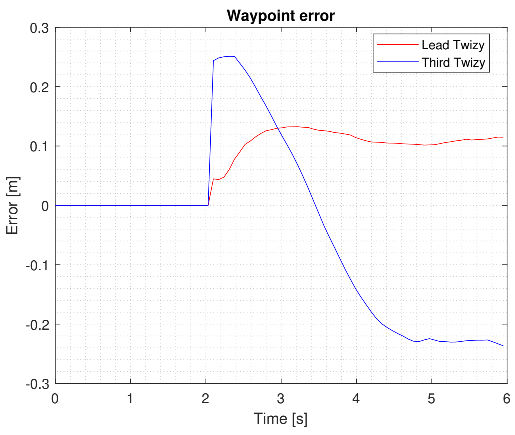
C.1.2 Constant acceleration model



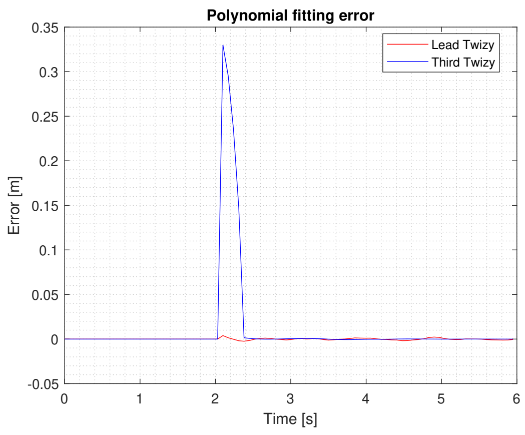
(a) Yaw rate tracking



(b) Yaw angle tracking

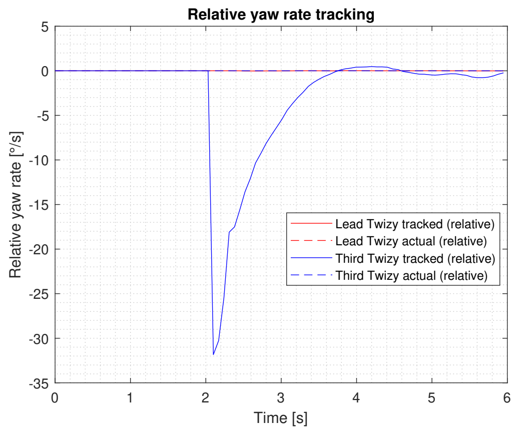


(c) Waypoint error

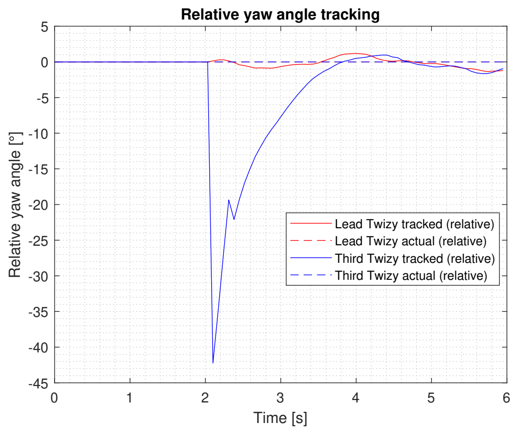


(d) Polynomial fitting error

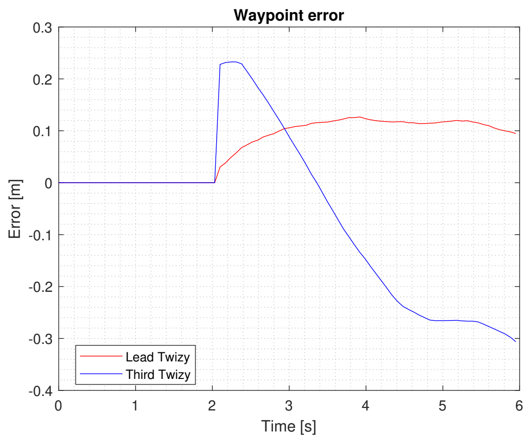
Figure C.5: Results for case 1: No constraints



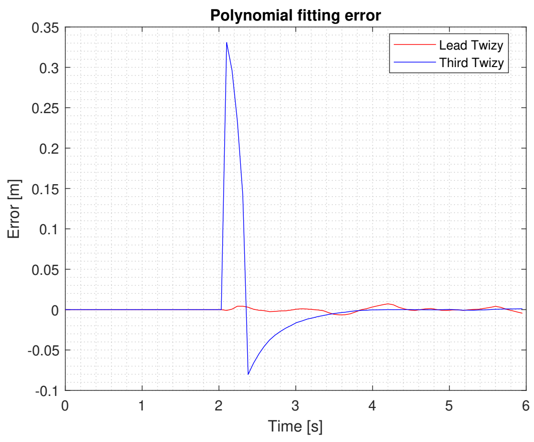
(a) Yaw rate tracking



(b) Yaw angle tracking

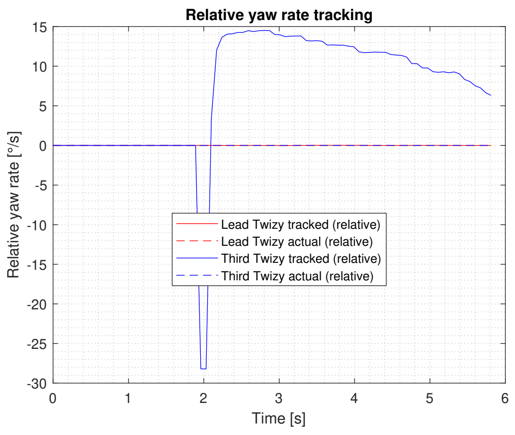


(c) Waypoint error

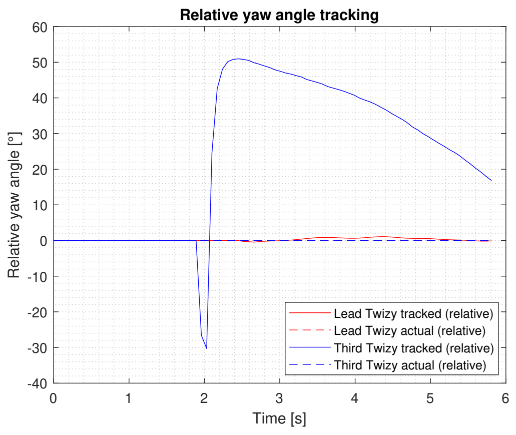


(d) Polynomial fitting error

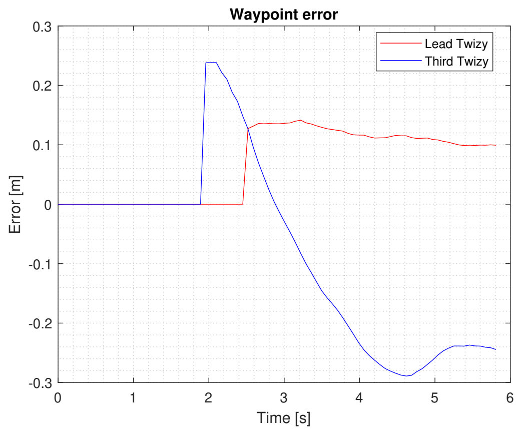
Figure C.6: Results for case 2: Heading constraint only



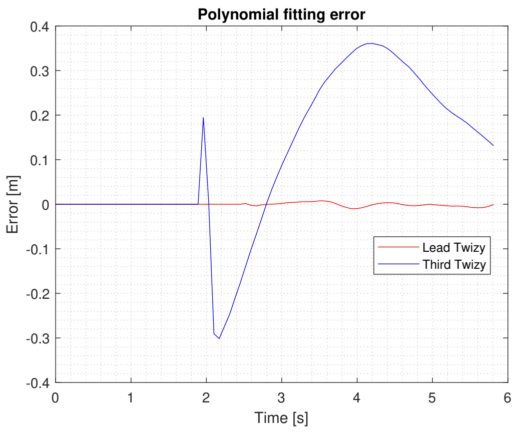
(a) Yaw rate tracking



(b) Yaw angle tracking



(c) Waypoint error

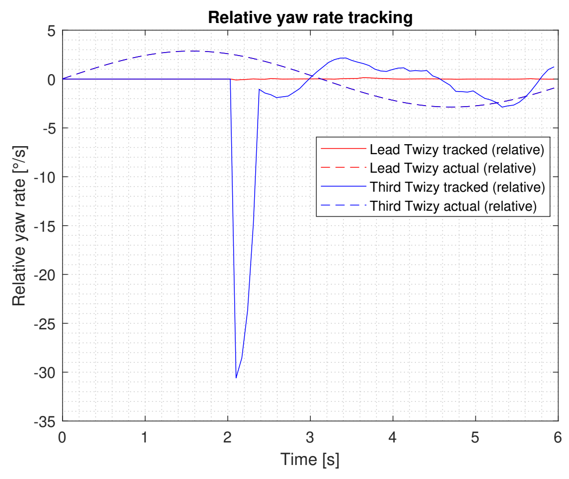


(d) Polynomial fitting error

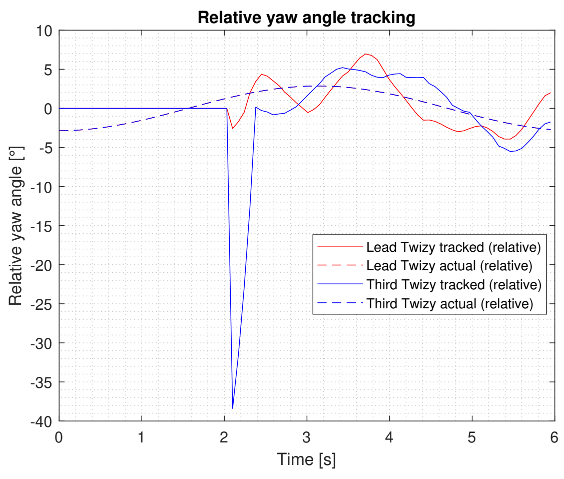
Figure C.7: Results for case 4: Both heading and curvature constraints

C.2 Driving on a straight path with added sinusoidal oscillations of the ego Twizy

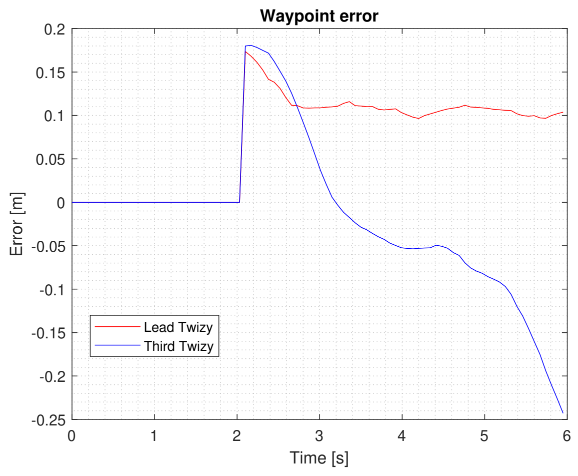
C.2.1 Constant velocity model



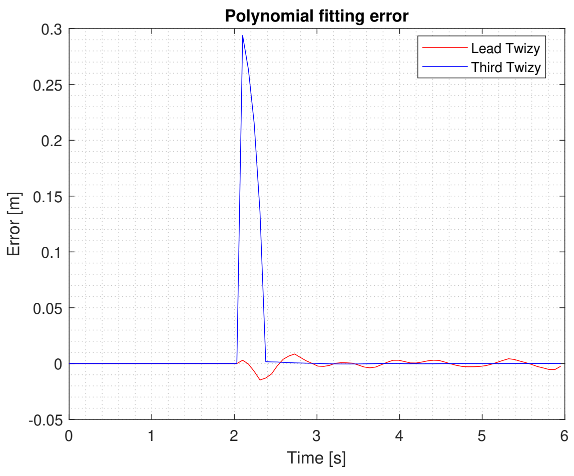
(a) Yaw rate tracking



(b) Yaw angle tracking

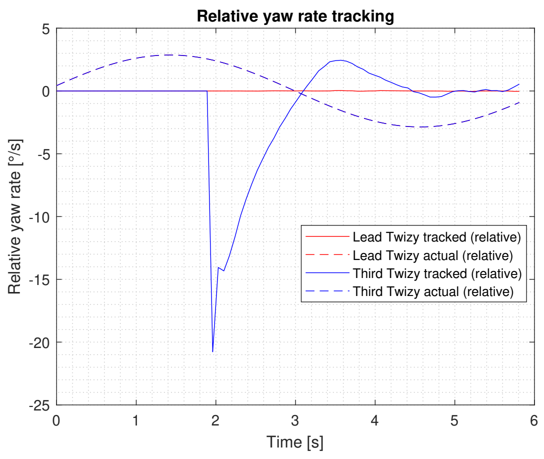


(c) Waypoint error

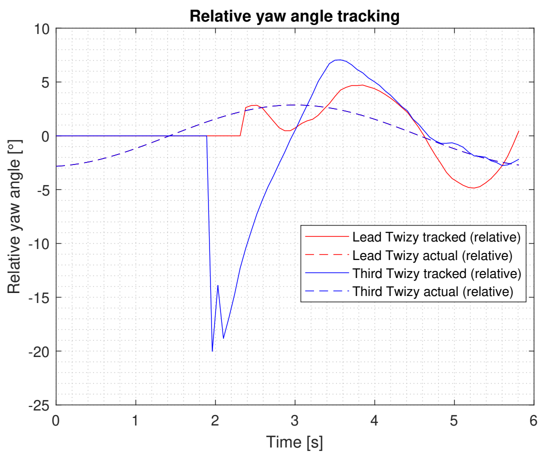


(d) Polynomial fitting error

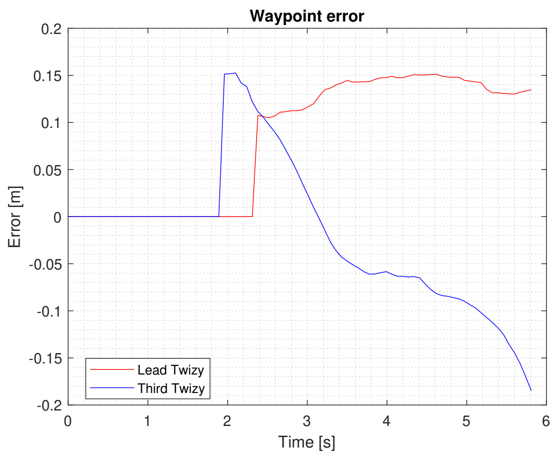
Figure C.8: Results for case 1: No constraints



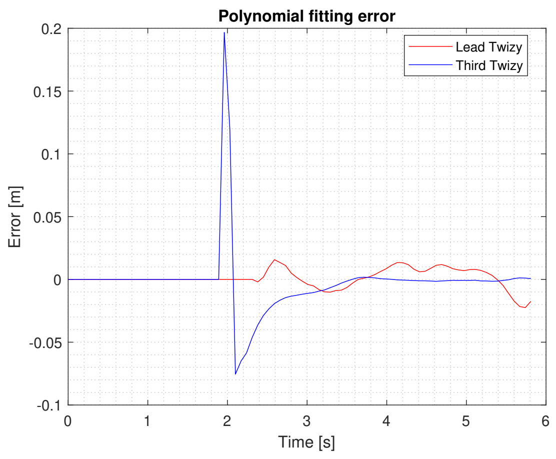
(a) Yaw rate tracking



(b) Yaw angle tracking

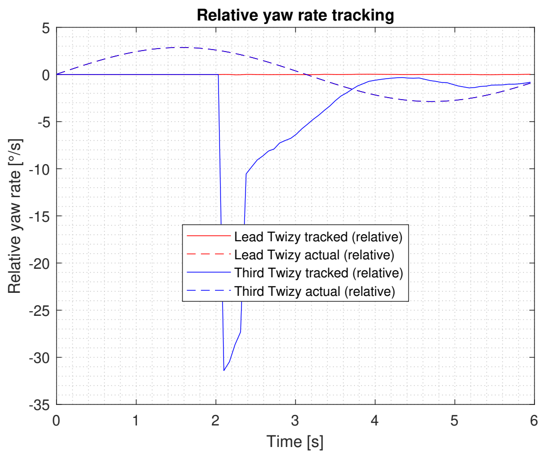


(c) Waypoint error

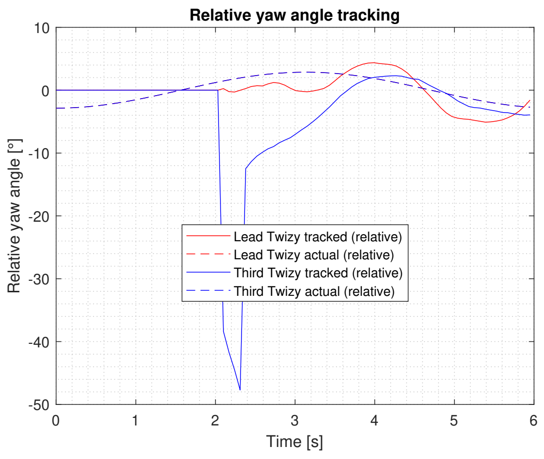


(d) Polynomial fitting error

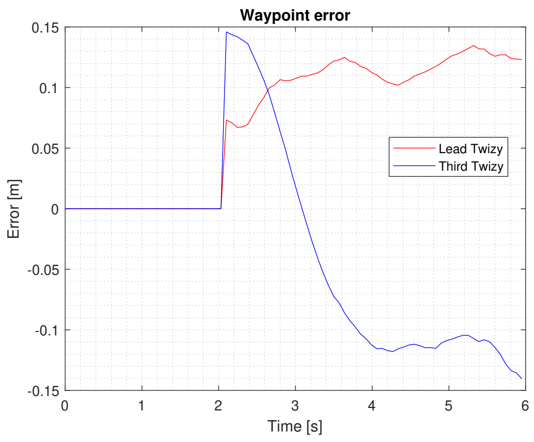
Figure C.9: Results for case 2: Heading constraint only



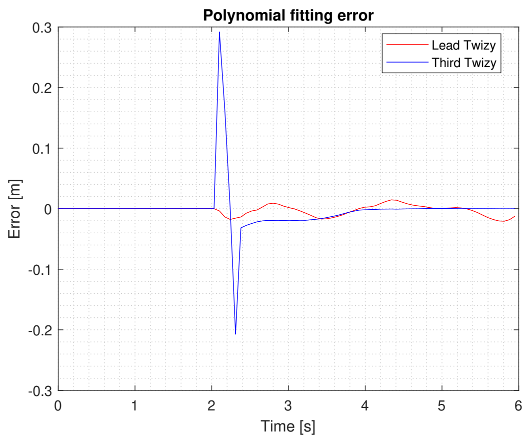
(a) Yaw rate tracking



(b) Yaw angle tracking

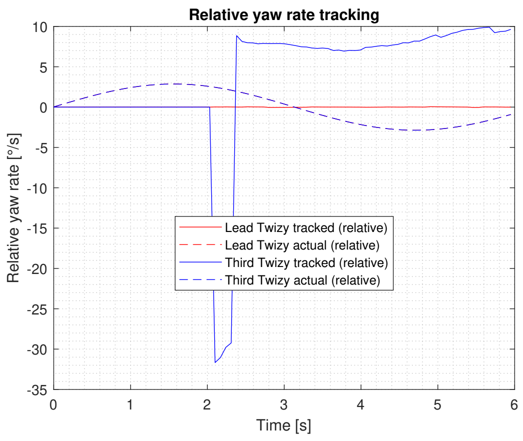


(c) Waypoint error

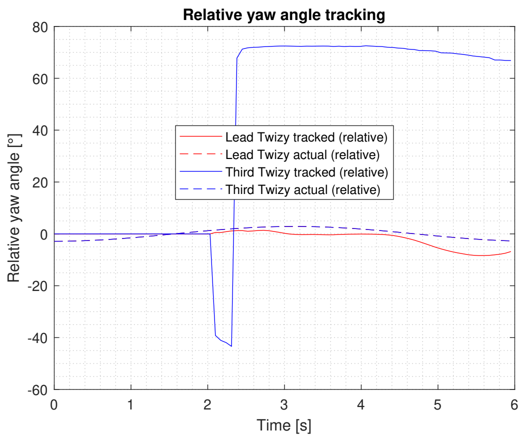


(d) Polynomial fitting error

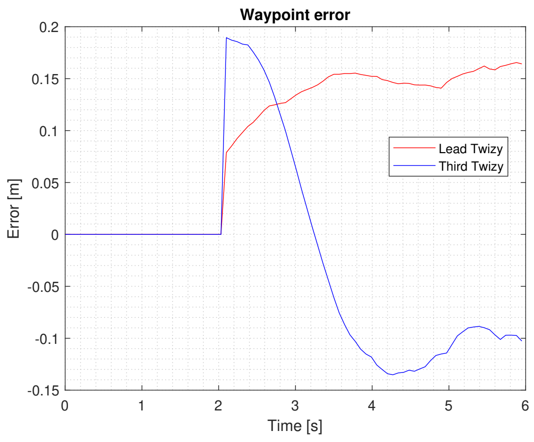
Figure C.10: Results for case 3: Curvature constraint only



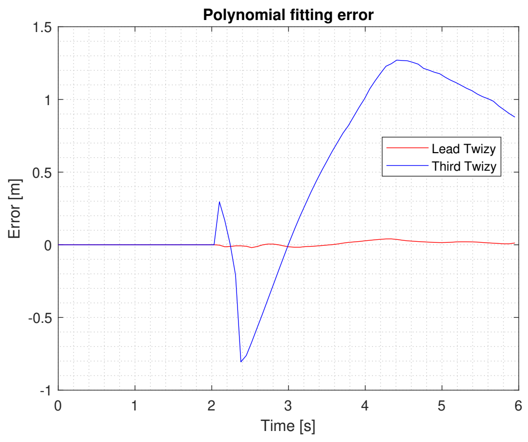
(a) Yaw rate tracking



(b) Yaw angle tracking



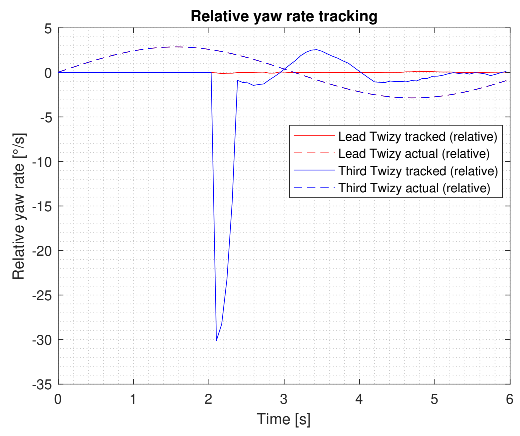
(c) Waypoint error



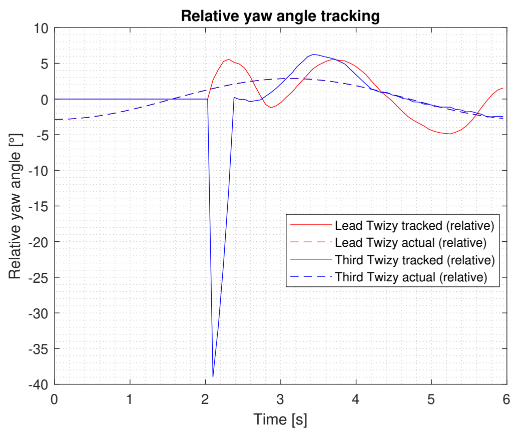
(d) Polynomial fitting error

Figure C.11: Results for case 4: Both heading and curvature constraints

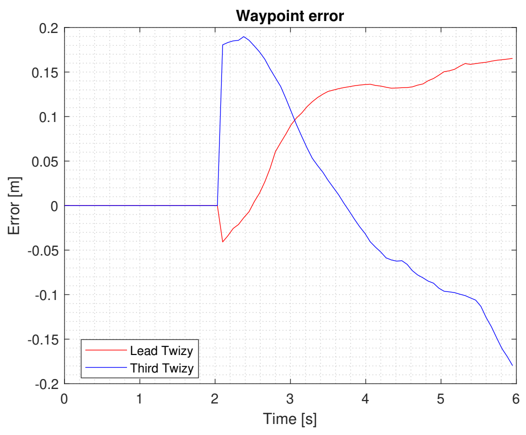
C.2.2 Constant acceleration model



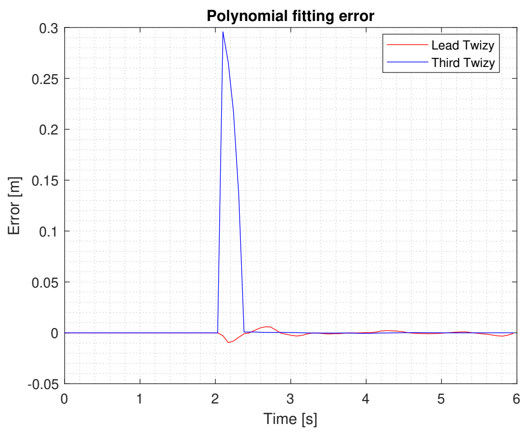
(a) Yaw rate tracking



(b) Yaw angle tracking

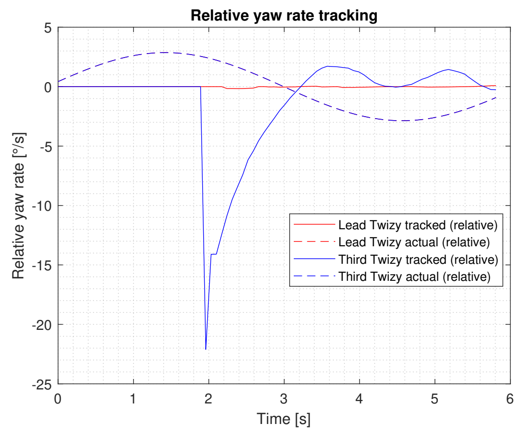


(c) Waypoint error

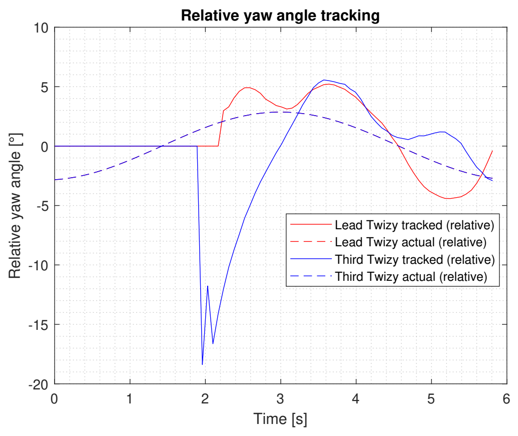


(d) Polynomial fitting error

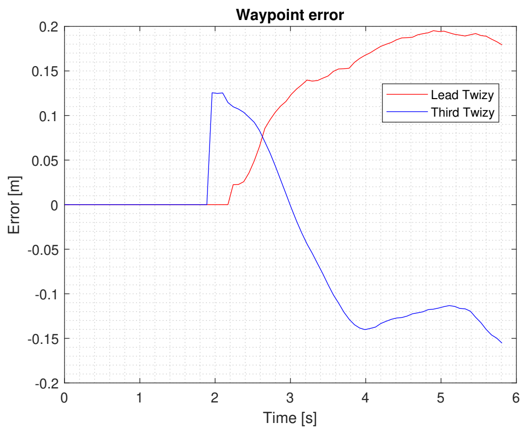
Figure C.12: Results for case 1: No constraints



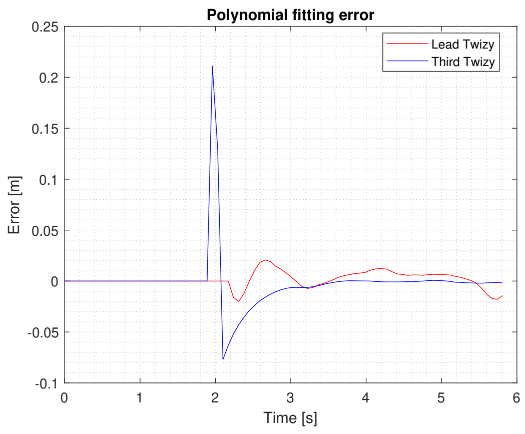
(a) Yaw rate tracking



(b) Yaw angle tracking

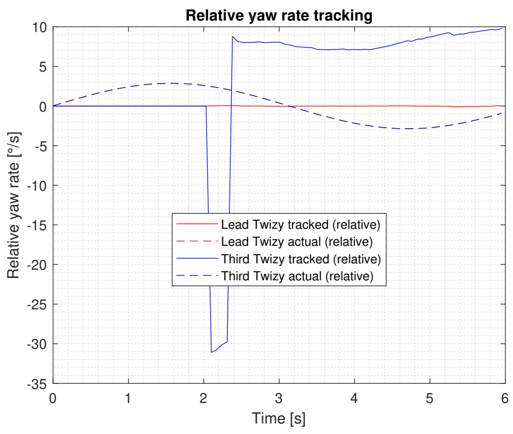


(c) Waypoint error

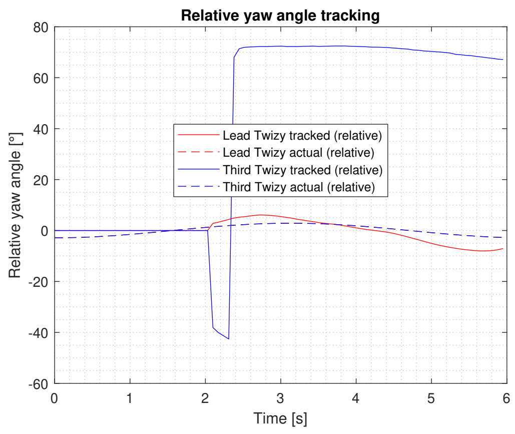


(d) Polynomial fitting error

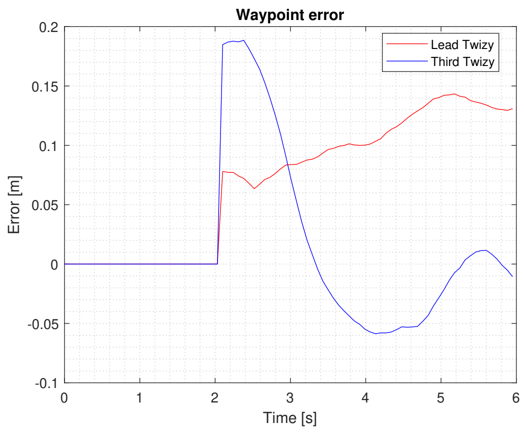
Figure C.13: Results for case 2: Heading constraint only



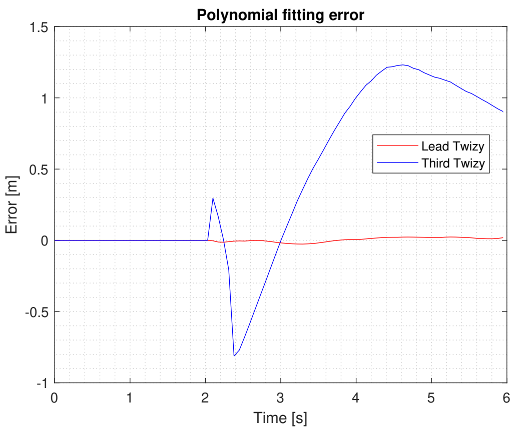
(a) Yaw rate tracking



(b) Yaw angle tracking



(c) Waypoint error

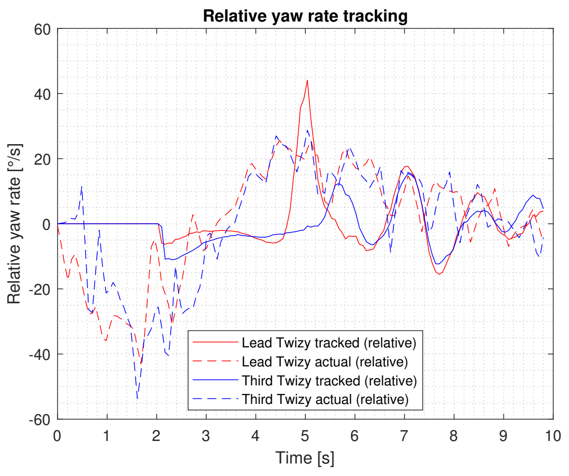


(d) Polynomial fitting error

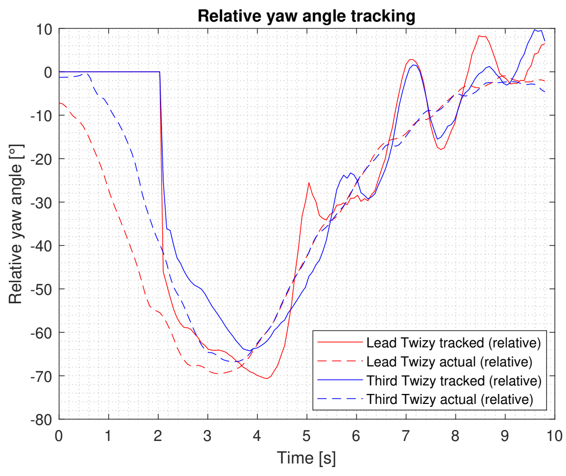
Figure C.14: Results for case 4: Both heading and curvature constraints

C.3 Cornering at a constant speed

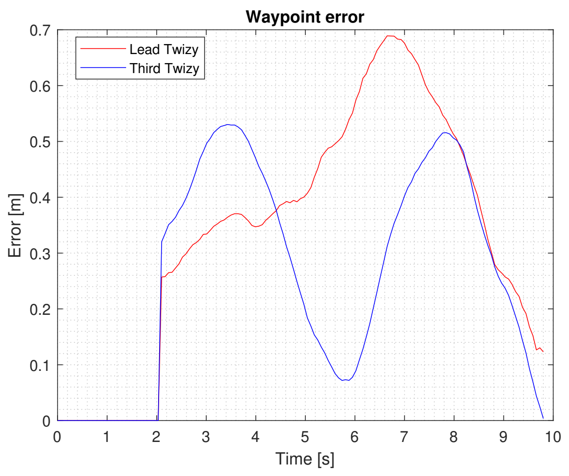
C.3.1 Constant velocity model



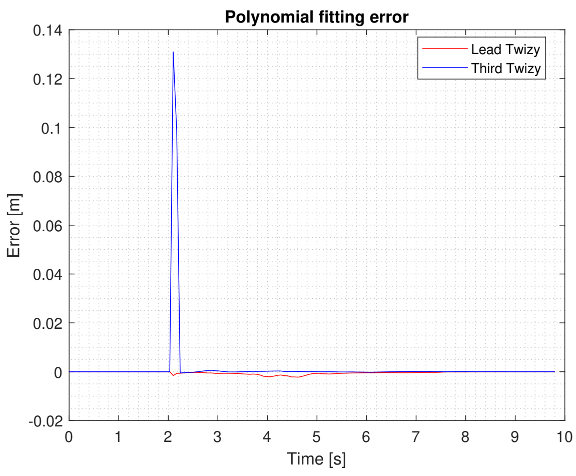
(a) Yaw rate tracking



(b) Yaw angle tracking

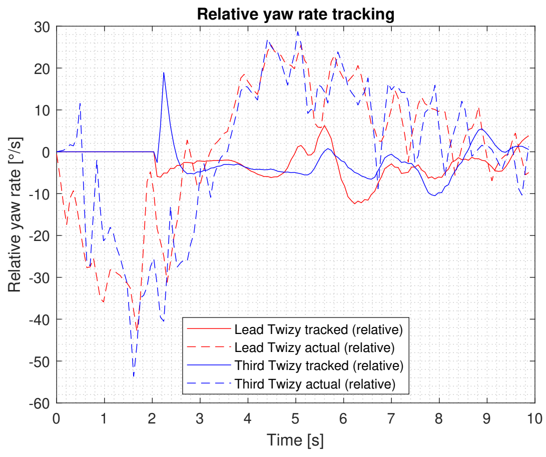


(c) Waypoint error

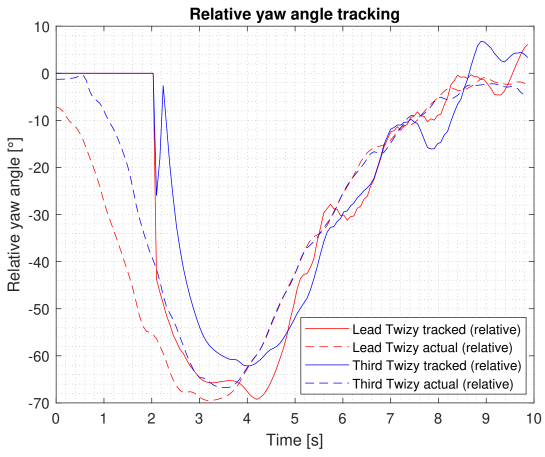


(d) Polynomial fitting error

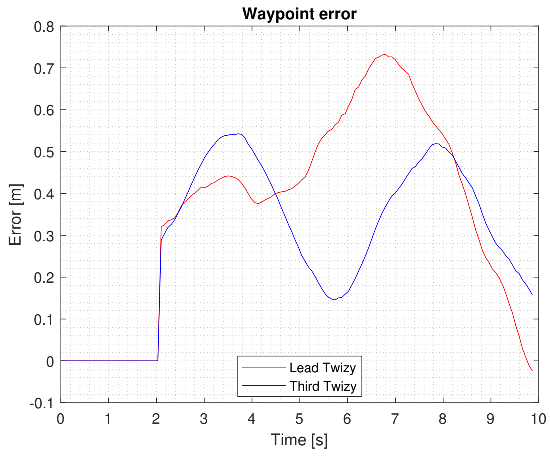
Figure C.15: Results for case 1: No constraints



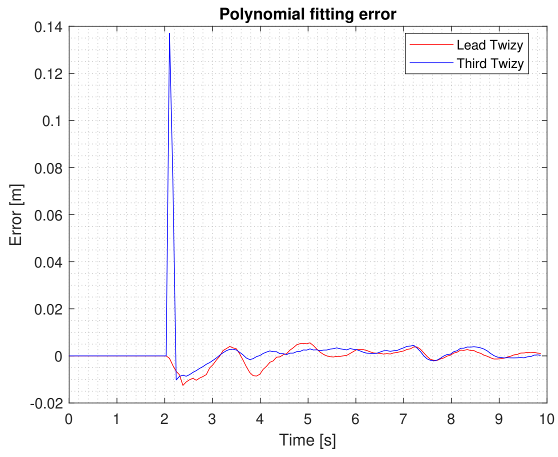
(a) Yaw rate tracking



(b) Yaw angle tracking

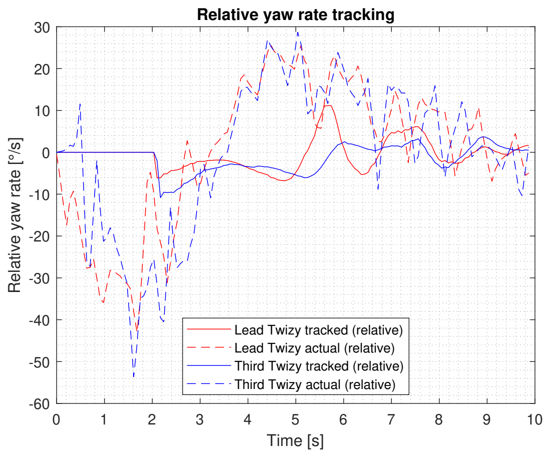


(c) Waypoint error

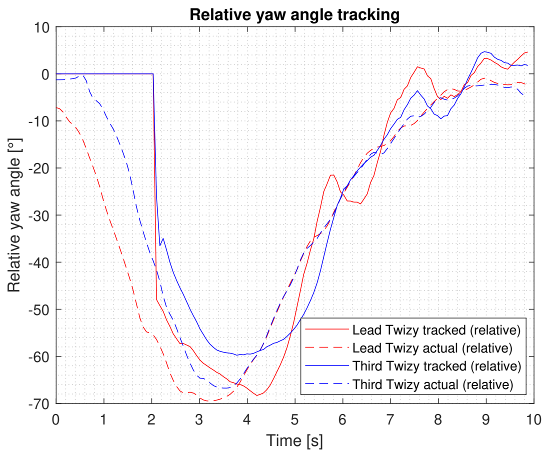


(d) Polynomial fitting error

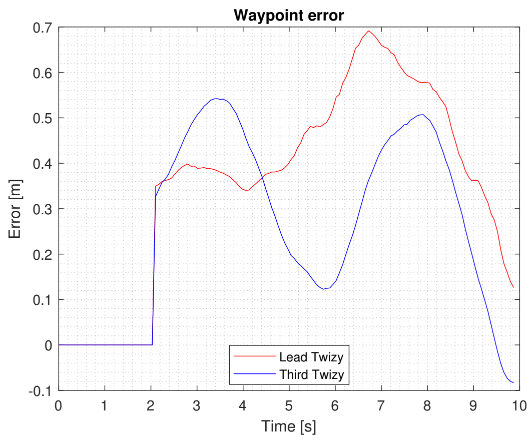
Figure C.16: Results for case 2: Heading constraint only



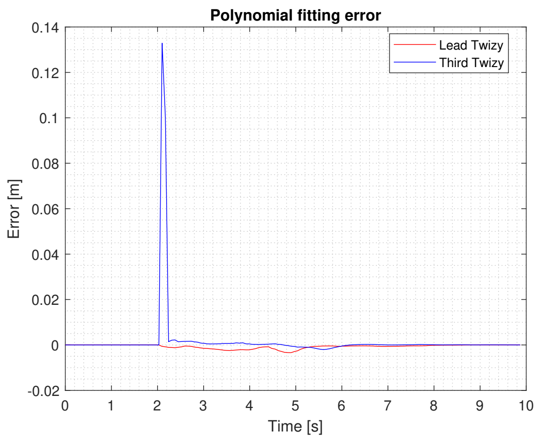
(a) Yaw rate tracking



(b) Yaw angle tracking

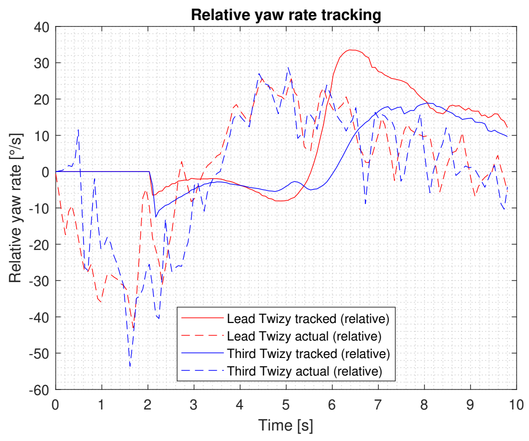


(c) Waypoint error

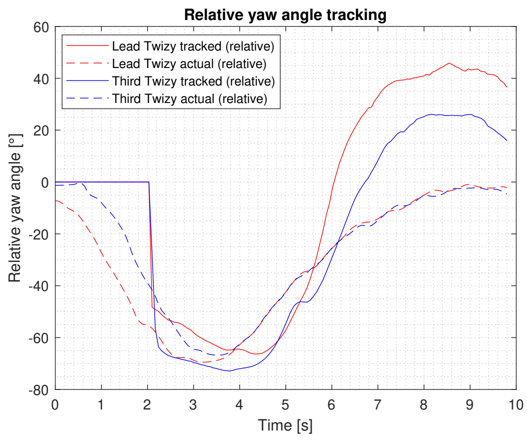


(d) Polynomial fitting error

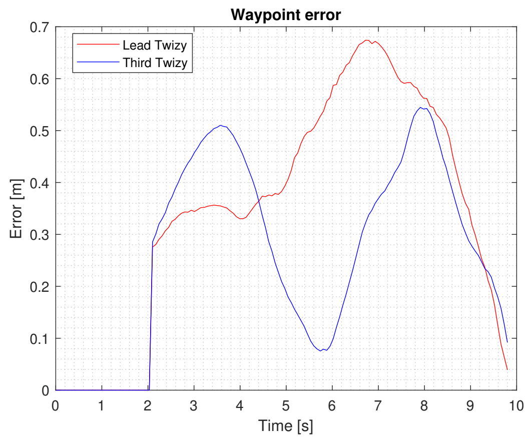
Figure C.17: Results for case 3: Curvature constraint only



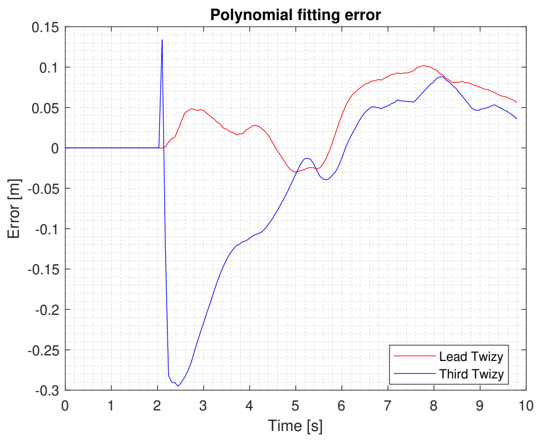
(a) Yaw rate tracking



(b) Yaw angle tracking



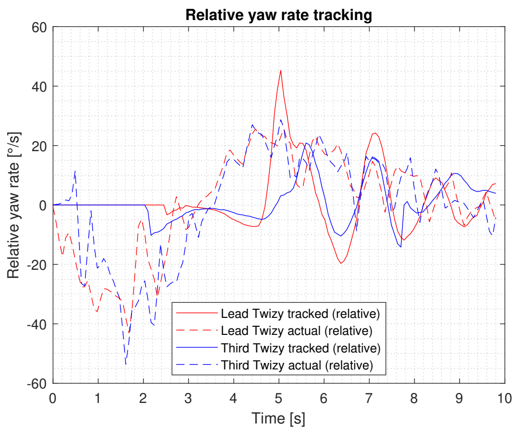
(c) Waypoint error



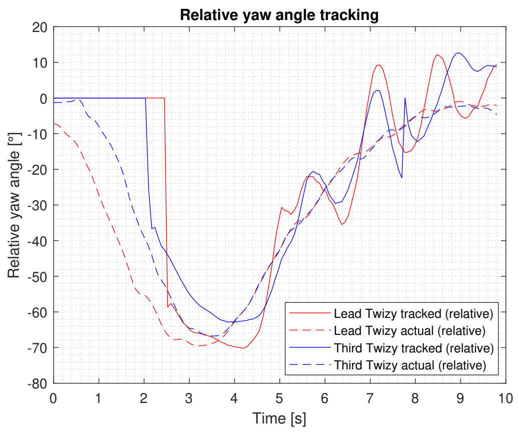
(d) Polynomial fitting error

Figure C.18: Results for case 4: Both heading and curvature constraints

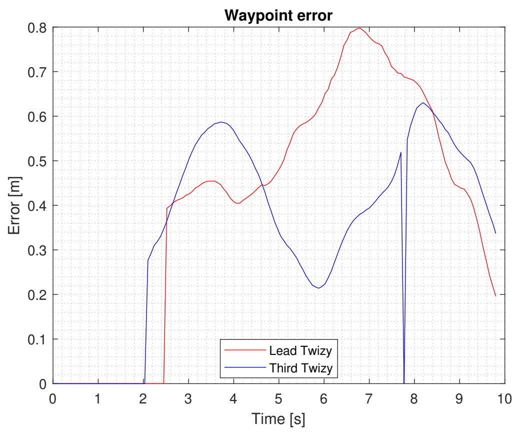
C.3.2 Constant acceleration model



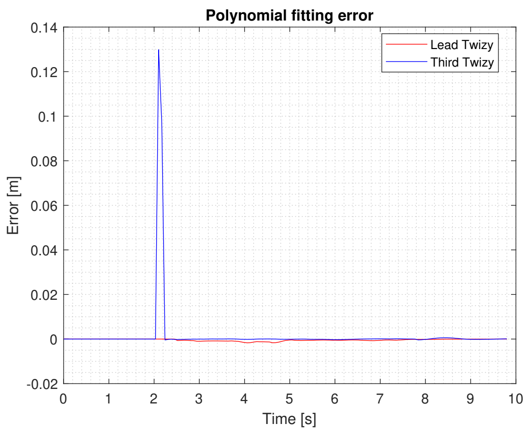
(a) Yaw rate tracking



(b) Yaw angle tracking

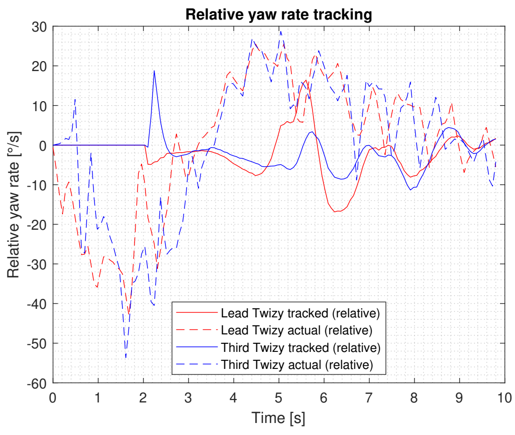


(c) Waypoint error

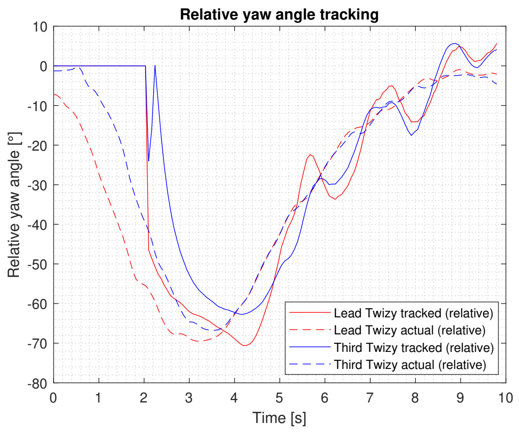


(d) Polynomial fitting error

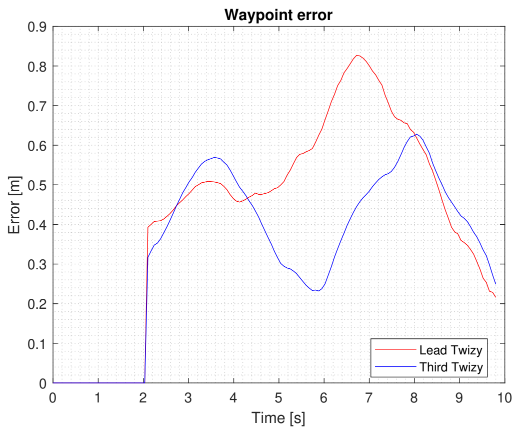
Figure C.19: Results for case 1: No constraints



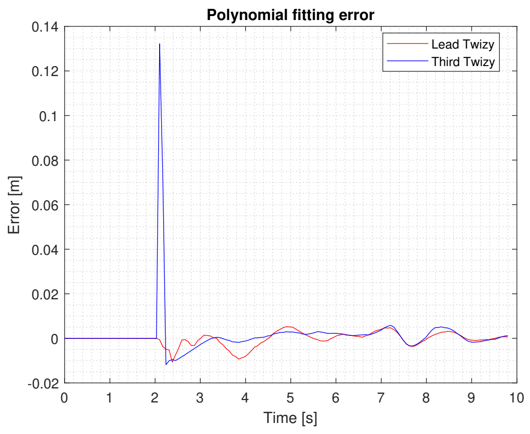
(a) Yaw rate tracking



(b) Yaw angle tracking

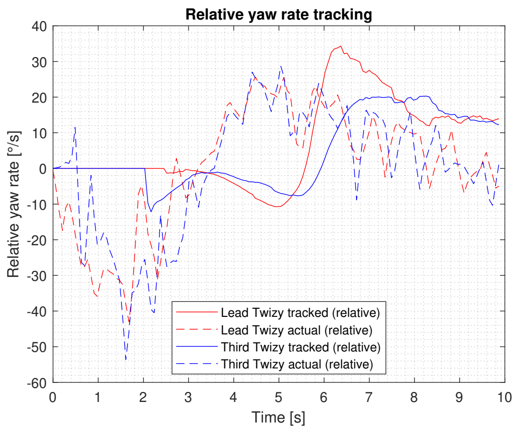


(c) Waypoint error

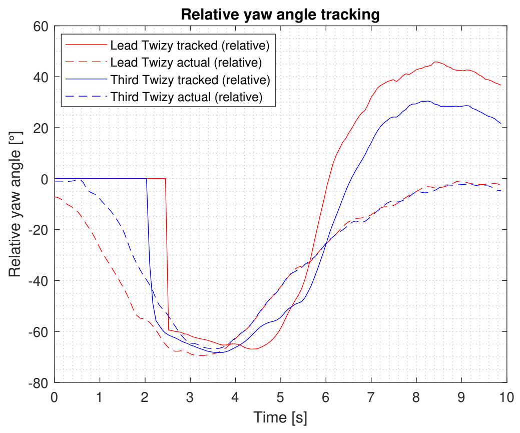


(d) Polynomial fitting error

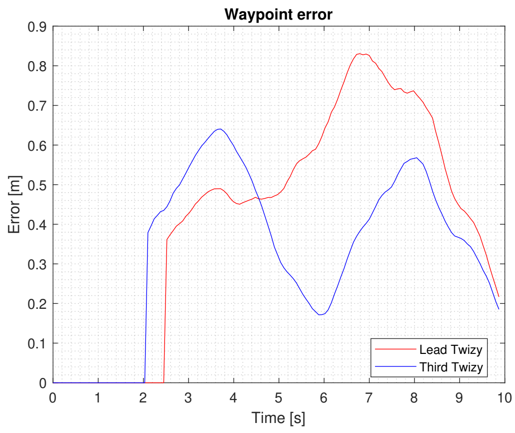
Figure C.20: Results for case 2: Heading constraint only



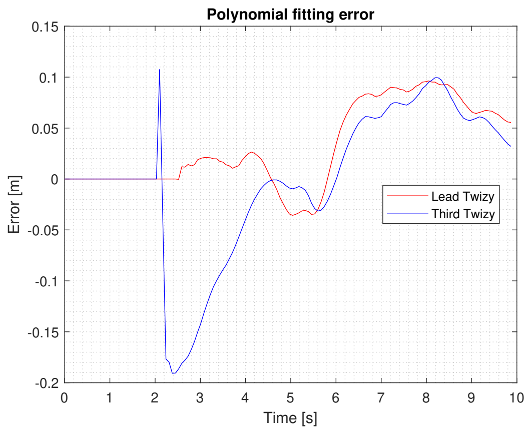
(a) Yaw rate tracking



(b) Yaw angle tracking



(c) Waypoint error



(d) Polynomial fitting error

Figure C.21: Results for case 4: Both heading and curvature constraints

D Track reconstruction from real world data for the CV model

This appendix shows the results of matching the lead Twizy track obtained from offline track reconstruction from measured real world data against the measurements, and the results of tracking the relative yaw angle, relative yaw rate, and global yaw angle of the lead Twizy for the constant velocity model.

D.1 Measurement matching with tracks

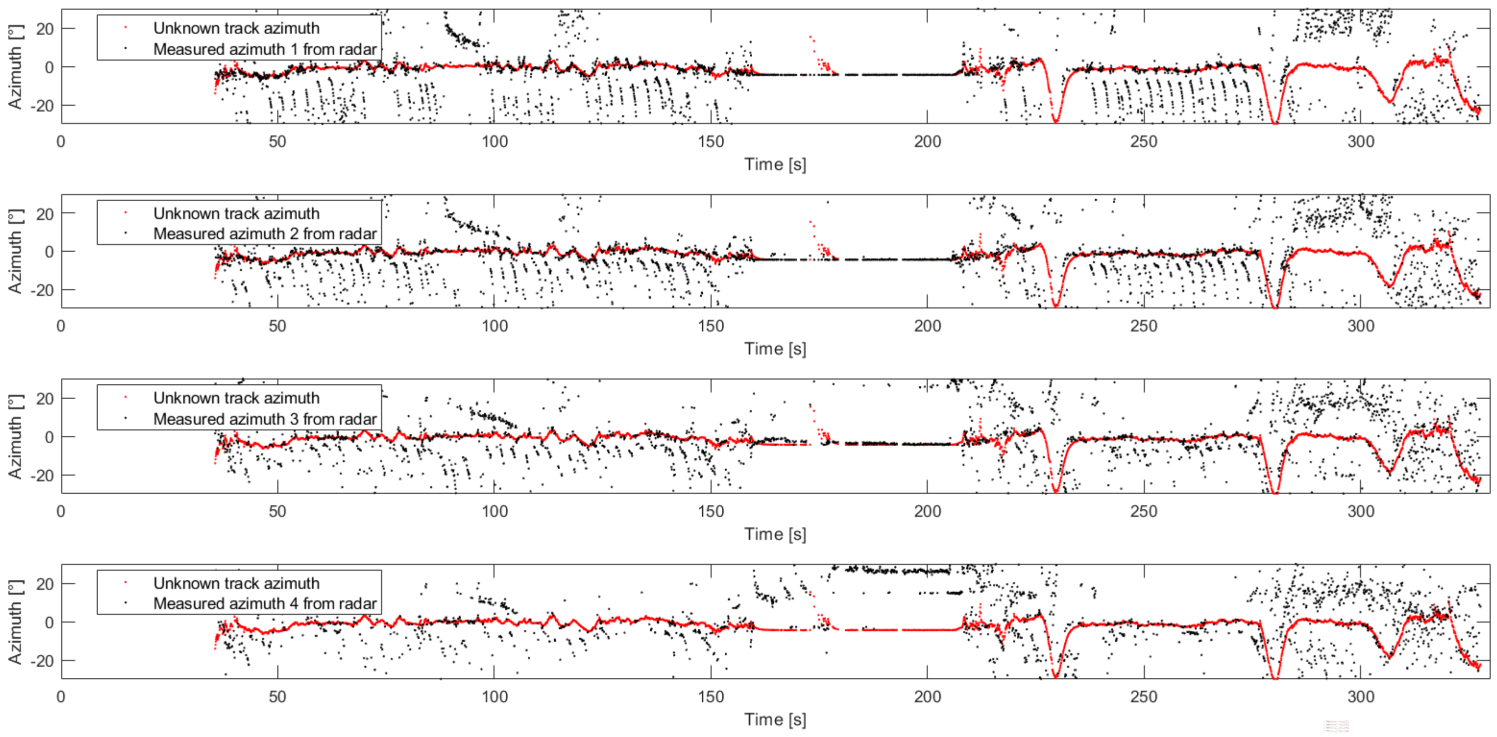


Figure D.1: Matched azimuths for the constant velocity model

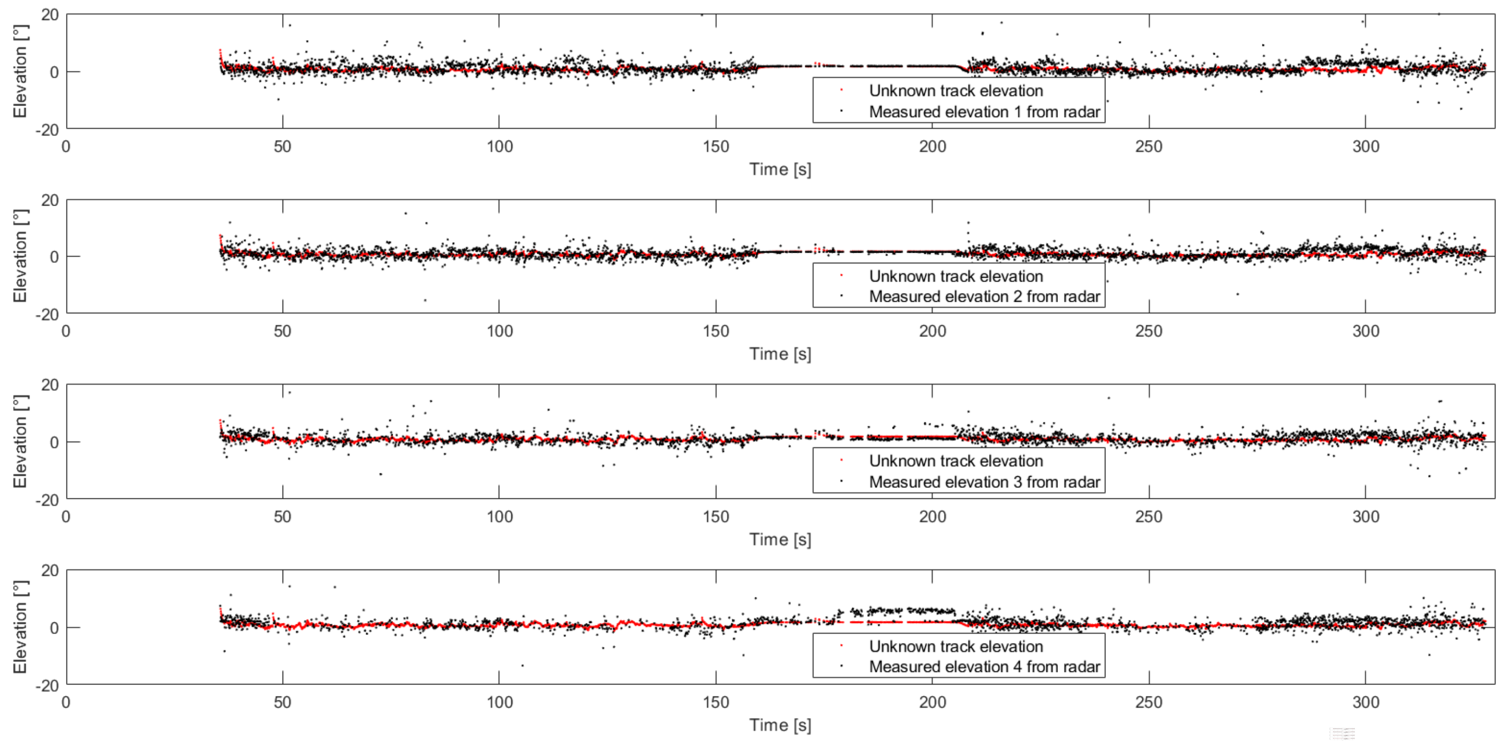


Figure D.2: Matched elevations for the constant velocity model

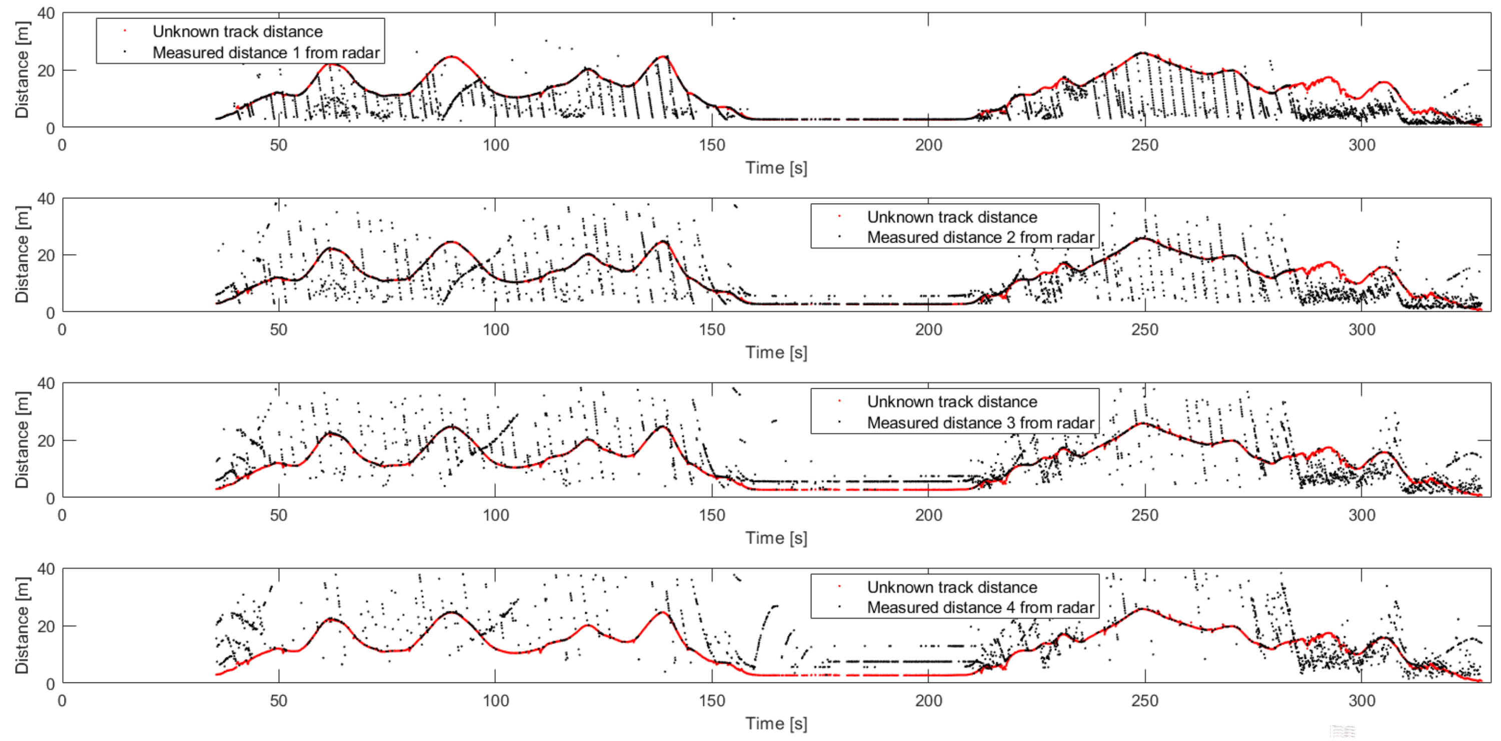


Figure D.3: Matched distances for the constant velocity model

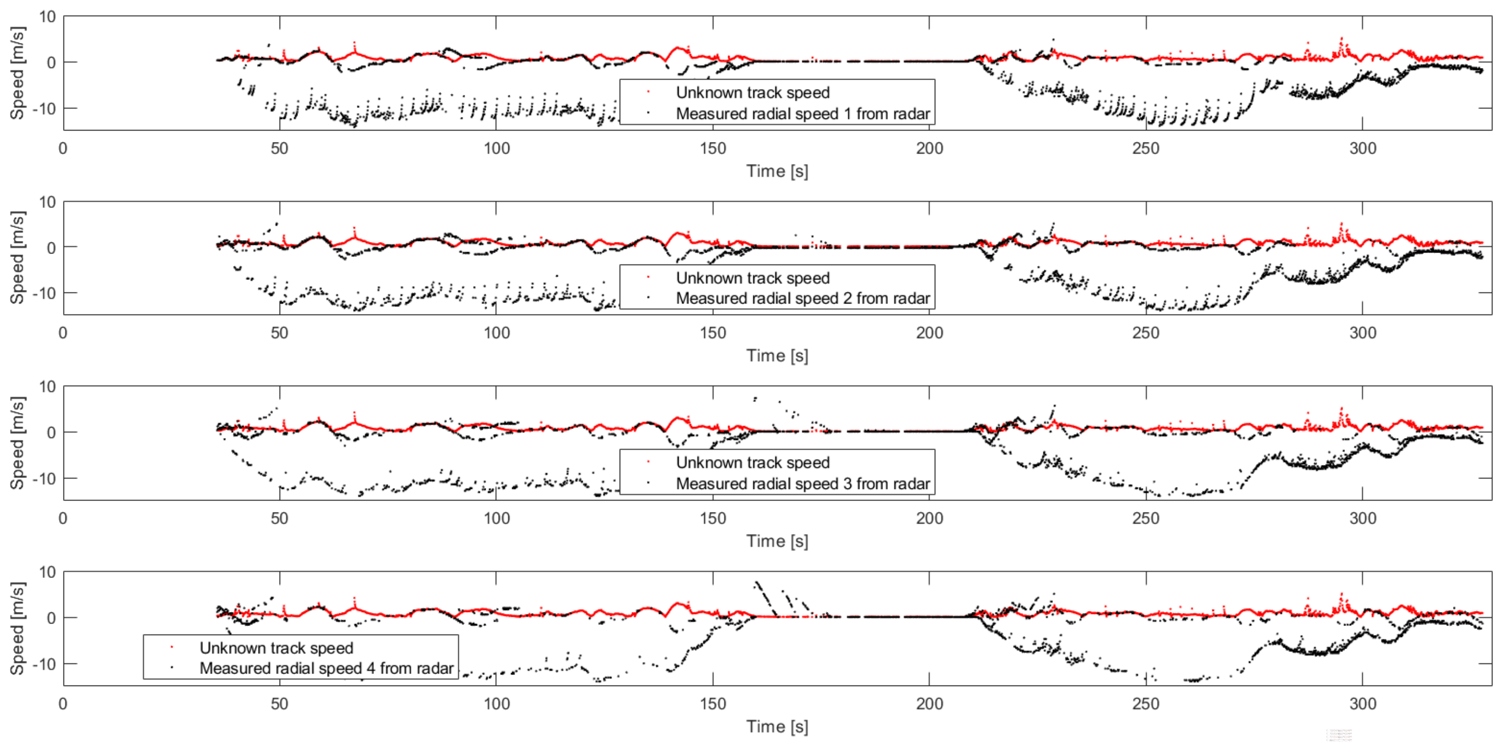
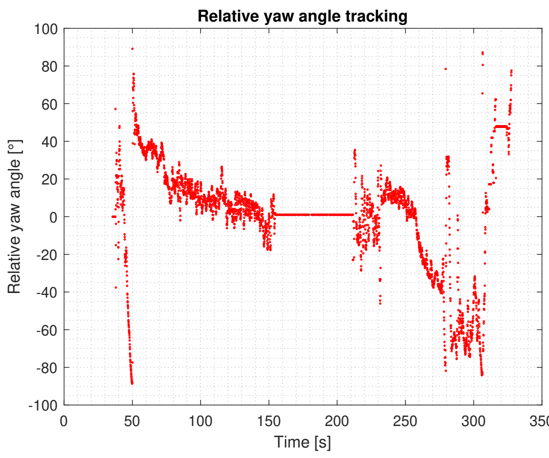


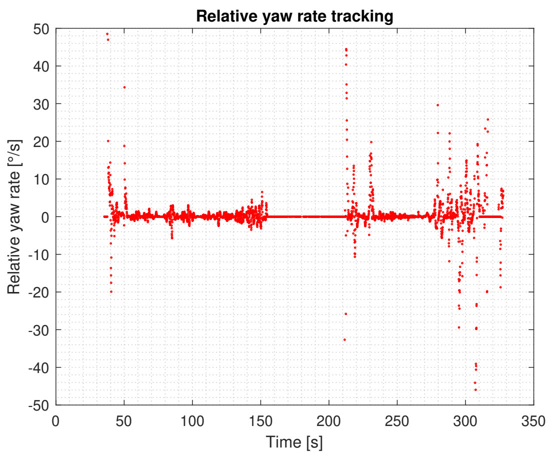
Figure D.4: Matched speeds for the constant velocity model

D.2 Relative yaw angle, relative yaw rate, and global yaw rate tracking

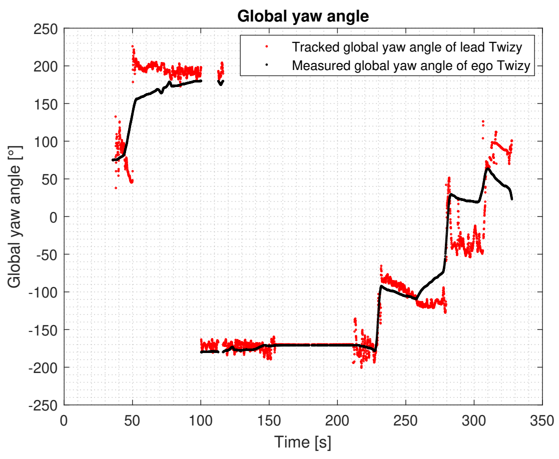
The results for the constant velocity model are shown in Figure D.5.



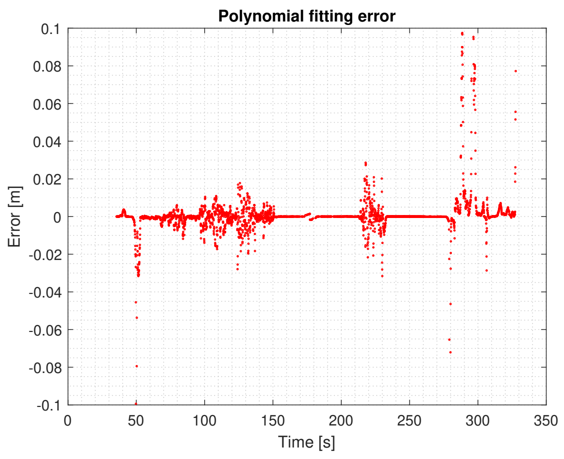
(a) Tracked relative yaw angle



(b) Tracked relative yaw rate



(c) Global yaw angle



(d) Polynomial fitting error

Figure D.5: Relative yaw angle, relative yaw rate, and global yaw angle for the CV model