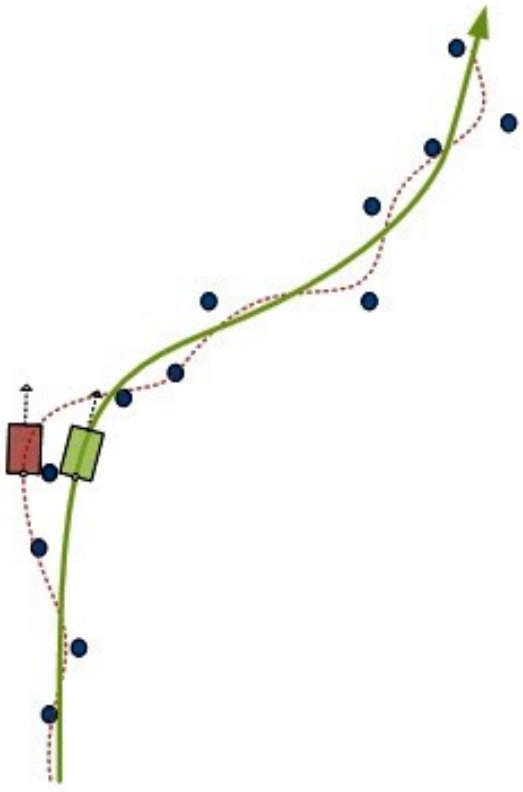
CHALMERS
UNIVERSITY OF TECHNOLOGY



Vehicle Trajectory Optimisation

Employing Extended Target Tracking, Guardrail Information and Smoothing Methods on Radar-only Detections

Master's thesis in Systems, Control and Mechatronics

SINDRI ÓLAFSSON
SRINANDAN KRISHNAMOORTHY

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:EX021

Vehicle Trajectory Optimisation

Employing Extended Target Tracking, Guardrail Information and Smoothing Methods on Radar-only Detections

SINDRI ÓLAFSSON
SRINANDAN KRISHNAMOORTHY

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Automatic Control, Automation and Mechatronics
Sensor Fusion
Chalmers University of Technology
Gothenburg, Sweden 2017

Vehicle Trajectory Optimisation
Employing Extended Target Tracking, Guardrail Information and Smoothing Methods on Radar-only Detections
SINDRI ÓLAFSSON, SRINANDAN KRISHNAMOORTHY

Supervisor & Examiner: Karl Granström, Department of Electrical Engineering

Master's Thesis 2017:EX021
Department of Electrical Engineering
Automatic Control, Automation and Mechatronics
Sensor Fusion
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Figure showing the optimal trajectory that needs to be achieved using an extended target tracking method (green) against the one that would be the case while using a point filter (red).

Typeset in LaTeX
Gothenburg, Sweden 2017

Vehicle Trajectory Optimisation
Employing Extended Target Tracking, Guardrail Information and Smoothing Methods on Radar-only Detections
SINDRI ÓLAFSSON
SRINANDAN KRISHNAMOORTHY
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Accurate knowledge of the surroundings and the trajectories that other vehicles take is an important factor that decides the autonomous performance of a host vehicle. There is continuous research and development being made to accomplish this task of obtaining information about the environment as close as possible to the ground truth. The main objective of this thesis is to optimise the accuracy of a vehicle trajectory using radar-only detections with the focus on developing offline, acausal, filtering and smoothing algorithms based on a suitable motion model.

Radar detections usually arise from a variety of objects in the *Field of View (FOV)* of the sensor. This follows that the detections from the object of interest, which is the target vehicle, first need to be filtered from the other detections, called *clutter*. The radar sensor may even give rise to multiple detections from the same target in the same instant of time. Thus, these filtered detections need to be further resolved using *Extended Target Tracking* techniques to give a better estimate on the target state. The detections arising from other objects like guardrails can also be useful while correcting for heading estimates under the assumption that the target follows the contour of the guardrails.

In order to develop some prior knowledge of the target state, a birthing algorithm is incorporated such that the entire FOV of the radar is scanned to first create the object that needs to be tracked. The target may also go out of view from the radar, and hence needs to be killed under such circumstances. Causality, real-time requirements and low latency were not the main focus of this thesis project, thereby allowing the possibility of using smoothing solutions for estimating the optimal trajectory taken by the target.

Keywords: Motion model, Radar, Extended target tracking, Clutter, Guardrails, Birthing, Killing, Causality, Vehicle trajectory, Smoothing.

Acknowledgements

We would like to extend our heartfelt gratitude to everyone who has supported our thesis work. Most of all, our supervisor from Delphi Automotive, Kenny Karlsson and also our supervisor and examiner from the Department of Electrical Engineering at Chalmers University of Technology, Dr. Karl Granström. Without the support of the two, from both the academic and industrial side of things, this thesis would not have been possible to do. We would also like to thank Dr. Lennart Svensson for his suggestions and support, Dr. Lars Hammarstrand for his guidance on following the paper that he authored and finally, Alexander Lyckell who helped us with several tasks regarding radar-related modelling.

Sindri Ólafsson and Srinandan Krishnamoorthy, Gothenburg, June 2017

Contents

List of Figures

List of Acronyms

FOV	Field of View
ACC	Adaptive Cruise Control
GPS	Global Positioning System
RMS	Root Mean Squared
RCS	Radar Cross Section
CT	Coordinated Turn
CKF	Cubature Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
CV	Constant Velocity
CA	Constant Acceleration
IMM	Interactive Multiple Model
JMS	Jump Markov System
BGSF	Bernoulli Gaussian Sum Filter
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
RTS	Rauch-Tung-Striebel
TPM	Transition Probability Matrix
RMA	Random Matrix Approach
MTT	Multi Target Tracking
JPDA	Joint Probabilistic Data Association
GNSS	Global Navigation Satellite System

1
Introduction

This thesis was done in collaboration with Delphi Automotive Systems, an automotive company that develops sensors used by several vehicle manufacturers worldwide. A part of what Delphi works on, is active safety systems within the automotive industry, where sensor accuracy is vital. These sensors are primarily focused on positioning the objects that lie in the immediate surroundings, relative to the host vehicle that equipped these sensors. In order to detect these objects, typically, the sensors used in these systems include a radar, which will be the main sensor used in the thesis.

From an industry point of view, the applications of the algorithms developed and the solutions to the problems considered, contribute to the autonomous driving sector. The outputs of these solutions are used for perception. Perception forms an important part of the autonomous driving field, since the knowledge of a vehicles surroundings needs to be accurate in order to do planning followed by the execution of the planned actions. In that sense, the outputs provided by the sensors forms the first step to perform any activity within the task of automating an automobile. In particular, an easy example to explain this would be the Adaptive Cruise Control (ACC) which has become a vital feature in many high-end vehicles. It makes the vehicle to be self-driven by controlling the velocity to follow a preceding vehicle, while adapting the speed to maintain a particular distance from it. Therefore, accurate knowledge about the state of this preceding vehicle needs to be estimated, in order to perform these tasks while simultaneously satisfying safety protocols.

For these reasons, and placing focus on increasing the accuracy of tracking this vehicle, the application considered for this thesis is the trajectory optimisation of that target from the perspective of the host. The focus is on the radar readings from that of a moving target vehicle as well as other stationary objects around the host vehicle. All these radar detections are then classified based on their utility and source of origin, in order to determine their usefulness for the task at hand. The ones that help to improve the estimates of the target are deemed useful and are processed for extracting the required information, whereas the ones that are not useful are ignored and undergo no further processing.

A good example to illustrate the differences between useful and useless radar detections is the presence of guardrails on the sides of the road as shown in Figure 1.1. If an assumption is made that the target vehicle follows the contour of the road, then it might be possible to use these stationary detections from the guardrails to map the contour of the road. These estimates could then be used to further improve the vehicle trajectory, which is the case in Figure 1.1a. Other uninteresting detections that arise from the radar may be ignored as clutter. On the other hand, in Figure

(a) Target vehicle following guardrail trajectory.

(b) Target moving away from guardrail trajectory.

Figure 1.1: Detections arising from two radar sensors, mid-range (orange) and long-range (blue) from the target vehicle at two different instances to illustrate the problem formulation.

1.1b, the vehicle moves away from the guardrails, this means that there should be enough attention being paid to view this change in the preceding vehicle in order to adapt the trajectory of the host. Thus, even interesting detections like the guardrails may sometimes become counterproductive because of the reason that if we use them, the overall estimate of the target becomes wrong since these detections are irrelevant after a particular instance.

Radar sensors are excellent at estimating range and range rate, but automotive radars are generally not capable of estimating azimuth accurately, sometimes making it challenging to minimise uncertainty in estimating a target vehicle's position. The aim of this thesis is to develop an algorithm for resolving these detections from the radar, such that the error to the ground truth is minimised, and to get the best trajectory estimate possible. This algorithm will run offline. Therefore, causality, real-time requirements and low latency will not be the main focus of this thesis project. Acausality then allows the possibility of computing the best estimate of the trajectory by applying smoothing methods. This is because the smoothing methods make use of the motion model of the vehicle and apply it in the backward direction (in terms of time), and thus smooth out the entire trajectory, which in the forward filter might have been more erratic. Smoothing solutions reduce the uncertainty in the filtered estimates, and since the final focus is to get the best trajectory estimate, they fall well within the scope of the thesis though they cannot be employed on a real-time system due to loss of causality.

1.1 Ethical and Sustainability Aspects

Active Safety is an indispensable aspect in automotive applications with respect to the ethical performance of an autonomous system. Safety is the first and foremost concern when it comes to autonomous driving, for example, where several arguments

regarding a machine's decision making skills come into play. All these are based on a machine's ability to understand its environment, and this thesis is focused exactly on that, a branch of automotive engineering called *Perception*. This thesis aims to further improve the accuracy of the information from the sensors used in active safety systems and with that make the driving experience safer. Having accurate sensor systems not only makes it possible to avoid serious accidents, but it also provides the possibility for autonomous functionality that can optimise vehicle fuel-consumption, for example in adaptive cruise-control.

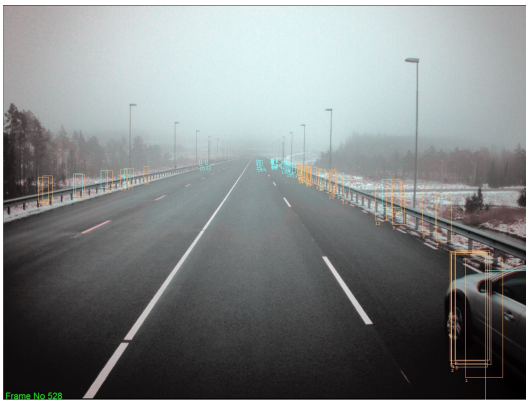
1.2 Objective

The main objective with this thesis is to optimise the accuracy of a vehicle trajectory using radar detections with the focus on developing offline, acausal filtering and smoothing algorithms based on a motion model. There is not much focus on optimising the code for running in real-time and all implementations are done in MATLAB, with the sole focus on tracking a target by resolving radar detections over a period of an entire run for a particular scenario. The tracking estimate needs to be as close as possible to the ground truth, with the detections being categorised into actual readings from the vehicle, clutter measurements, or measurements that can further improve our estimates. As shown in Figure 1.1, the same target vehicle can give rise to multiple detections. Hence, the readings originating from the vehicle need to be associated as close as possible to their actual place of origin from the target vehicle. The rough dimensions/shape of the target vehicle is assumed to be known and therefore can be used to resolve the multiple detections arising from it. The road often has guardrails on the side, which have a well-defined structure and follow the road (usually), as explained earlier. These readings may be used to fine-tune the estimate of the road curvature, instead of rejecting it as an outlier at the first instance of the readings. Investigations also needs to be made to include cases such as illustrated in Figure 1.1b, so that the algorithm becomes generic to account for the target following or not following the guardrail trajectory. Finally, to get the best estimate of the vehicle trajectory, smoothing solutions are investigated.

1.3 Problem Formulation

With the basic idea presented in the previous sections, it is quite straightforward to split the problem formulation into four broad categories. The first, is to be able to resolve multiple detections arising from the target, which is referred to as *Extended Target Tracking*. There are several methods to do this, and the most suitable method is chosen for this thesis and explained. The second task is to investigate the road curvature from the guardrail information to get better heading estimates. This must involve an algorithm that can determine if the target is following the guardrails, so that that information is only used when the criteria holds.

The third task is added to the preexisitng algorithm developed from the first two steps, so that a prior may be defined to begin the search for the target in the entire Field Of View (FOV) of the radar. This is because, in the real-world scenario,



(a) Target entering from the right.

(b) Target entering from the left.

Figure 1.2: Target vehicle entering the Field of View of the radar from either direction, whose prior is not known and needs to be defined.

the prior is not known to the sensor since it may occur at any place irrespective of the instance at which the particular scenario begins (the target might/might not exist at the first instance and may gradually come into the FOV). This can be illustrated as in Figures 1.2a and 1.2b, from which it can also be seen that the place of origin might change even with respect to the side from which the target enters the FOV. The solution could be provided by creating a birthing function that can create a prior for the other algorithms to work within this outer algorithm so that targets may be birthed or killed during the entire run-time. The killing should be triggered by the target ceasing to exist in the FOV. Also, since the vehicle has been modelled using extended target tracking, this knowledge may also be used to detect the place of origin of the measurements, which is then used to map the place of origin of the entire vehicle in the observation space of the radar. The final challenge is to select a suitable smoothing method to get the best trajectory estimate. As much information regarding the features of the vehicle, assumed to be known during filtering as well as the ones obtained through filtering (like the likelihood of guardrails being followed or not) need to be used for the smoothing and thus, some solutions are investigated and presented.

1.4 Data Used and Scenarios Considered

All the measurements that arise from the radar at every instance are passed into the algorithm for the processing. There are two radar modes, a mid-range radar scan and a long-range radar scan. They operate alternatively and the sampling time is fixed and known. The sampling frequency is 30 Hz, giving the time between samples to be $1/30 = 0.033$ s. The radar is placed on the host vehicle, facing forward, placed more or less at the centre of its front bonnet with a known elevation. Further radar-related modelling parameters are known and the procedure of modelling them in the required manner for the applications considered are explained further in Section 2.2.2.

1.4.1 Sensor Data and Specifications

The radar sensor used in this project is a solid-state automotive mid-range radar developed by Delphi automotive. It generates detections based on energy reflected from different ranges and Doppler bins. That means that it is able to separate detections based on the range and relative velocity of the surface that is reflecting the radar beam. The sensor is also capable of separating two detections that fall within the same range and Doppler bin if the azimuth angle between them is sufficient. Thus, the sensor generates detections at instant k described by the measurement vector:

$$y_k^i = [r_k^i \quad \dot{r}_k^i \quad \phi_k^i]^T, i = 1, 2, ..., n_k^y$$

where n_k^y refers to the number of detections at that instant, r_k^i is the range to the detection, $\dot{r}_k^i$ is the range rate and ϕ_k^i is the azimuth.

1.4.2 Validation Data

The validation data is not generated in the form of an assumed ground truth, but instead obtained from two separate sources. One is from a commercial system which uses a fused output from the radar measurements and forward looking camera. The second source is data from a GPS system with high accuracy, which gives the relative positions, velocities and heading between host and target. This provides a means of comparison for all error calculations and for validating the performance of the algorithms presented in the thesis. There are several other factors and performance parameters that were defined by Delphi, and the ones that are relevant and provide insight into the validation of the algorithms are further explained in Chapter 6.

1.4.3 Test Scenarios

The number of scenarios that can be evaluated is restricted by the availability of data, so the following set of scenarios were chosen to validate the algorithms. To evaluate the performance of the extended target tracking methods, three cases are considered as follows:

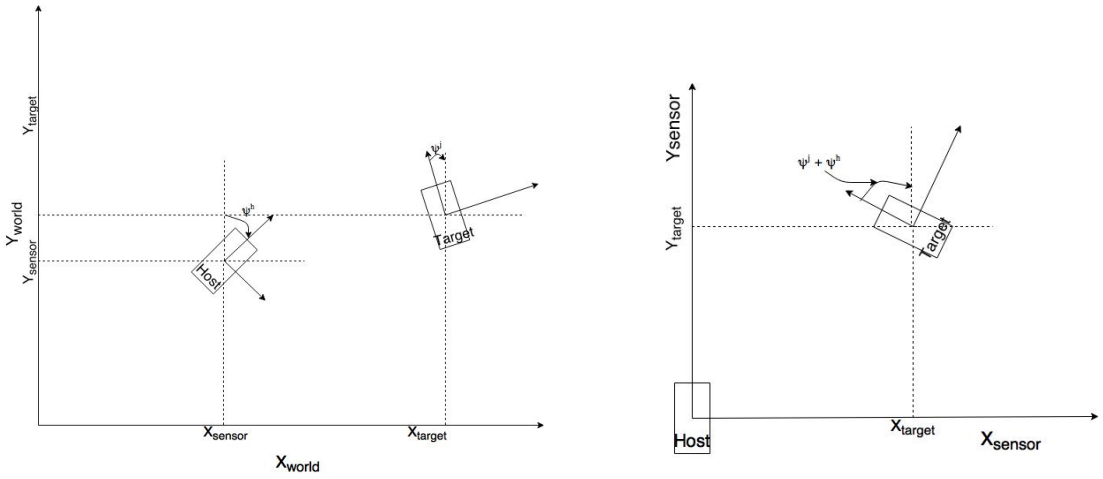
1. A case where the host follows the target, presented in Section 6.4.1,
2. A case where the target turns in the FOV of the radar mounted on the host, presented in Section 6.4.2,
3. A case where the target overtakes the host from the right, presented in Section 6.4.3.

These tests can sufficiently validate the extended target tracking methods and models. For the validation of a scenario which has guardrails on the side, the target following the contour of guardrails gradually moving away from it is presented in Section 6.5.

1.4.4 Frames Used

The following frames are defined in order to make the reading of this thesis easier:

1. *The world frame:* this frame is used for plotting the entire trajectory of the host and the target at the end of a scenario running, so as to provide a means of comparison for validating the algorithm. The origin of this frame is at the origin of the host frame at the first instance of starting that particular scenario. Thus, the host and the target frames move in this world frame. The coordinates are represented as (X_{world}, Y_{world}). This frame is fixed and the heading angles of the other two frames are with respect to this frame.
2. *The sensor frame:* coordinates are denoted as (X_{sensor}, Y_{sensor}). This is the frame of the sensor, and for reducing the complexity of frame definitions, it is assumed that this is the host frame too, i.e., the sensor lies at the origin of the host at every instant even as the host moves. Thus, in cases where the host ground truth and the sensor position on this host are known, there needs to be a rotation matrix mapping between the host and the sensor frames in order to do the tracking. The heading of this host/sensor at any instant k is denoted as ψ_k^h.
3. *The target frame coordinates:* this is, the targets frame coordinates that need to be tracked. They are denoted as (X_{target}, Y_{target}), while the heading at instant k is ψ_k^j.



(a) Target, host/sensor frames in the world frame.

(b) Target frame as seen by the host.

Figure 1.3: Different coordinate frames used for the thesis and their definitions illustrated.

Figure 1.3a illustrates the different frames. The host and the target can move in the world frame and naturally, the target can also be interpreted in terms of position in the host frame. These transformations permit visualisations in different frames and coordinate systems which sometimes simplifies tasks, since any local frame can be transformed to the *world frame* using appropriate transformations, for overall comparisons of performance, and trajectory optimisation.

1.5 Limitations

In this thesis only scenarios with a single moving target will be considered. Therefore, it is assumed that every moving detection above a certain threshold velocity is that of the moving target. This, however, is not true for all instances due to the discrepancies in radar measurements. The range rate measurements are measured in the direction of the azimuth angle of those detections, hence any movement perpendicular to the line of vision of the radar is resolved to be stationary. Also, radar measurements sometimes are caused by energy bouncing off from some stationary object, but wrongly measured to be moving, thus giving these stationary detections a velocity. When prior knowledge is known about the targets whereabouts, the measurements that fall far away from the target can be ignored, but this assumption for an object to have a minimum velocity is especially important when the sensor searches for the target and hence all measurements above a minimum value for range rate are considered to be from the target. Though this is not true, it is better to assume them to be from the target and then eventually kill them as possibilities for the target. Data and scenarios are thus chosen with only a single target to track, in order to limit the scope of the problem.

With respect to the target, there is another underlying assumption which should be made to limit the scope of the thesis. This is the dimensional approximation that is done for the target model, which then becomes a limitation for the type of targets considered. In reality, even in scenarios where there is just one target that is of interest, this target can be anything between a small sized vehicle like a bicycle to a huge truck. But in order to the extended target tracking, a predefined target model is required and thus, it is assumed that the target of interest is a small-to-medium sized car. The sensitivity analyses for the dimensional dependencies of the algorithms presented have not been performed. However, it was observed that this sensitivity to target dimensions is marginal. Steps to be taken to make real-time realisations of the algorithms to have adaptable dimensional parameter calculations have also not been presented since this is out of scope of the thesis.

Since the sensor keeps moving with the host, there is a need for knowing the position of host with as much accuracy as possible. The thesis does not cover positioning the host, since the interest is in tracking the target. Therefore, in the ideal case, there would be access to the ground truth of these host positions, from which the sensor could be placed at every instance in the local frame of the host, thereby making the target tracking the sole source of errors. But this is not the case due to the unavailability of this absolute ground truth for the host. Instead, the host positions are obtained from a variety of sensors mounted on it. In particular, there are two ways of obtaining this host data: either through fusing the various sensors placed on it, or its GPS data.

Accurate validation data was not available for all scenarios considered. So instead of validating the absolute tracking accuracy for those scenarios an improvement is validated by comparing the results from two different implementations to the results from a commercial tracker that is on board the host vehicle. This commercial tracker fuses measurements from multiple sensors on board the host vehicle and is therefore a valid comparison value but has an unknown Root Mean Square

(RMS) error.

Another important limitation, as stated earlier, is with respect to the real-time performance of the algorithm. Such an implementation on board an actual host vehicle would not be possible with the developed solutions since the causality requirements are not satisfied in certain parts. In specific, for tracking a single target using radar, the solutions may be used up to the point where they have been recognised as a valid object and use the extended target tracking method proposed. Beyond this, the part of the algorithm which deals with using guardrail information and smoothing methods for optimal trajectory estimates, the causality does not hold, and therefore cannot be realised as a real-time implementation.

1.6 Structure of the Thesis

The tasks are split as follows: in Chapter 2, a technique for resolving the multiple target measurements is discussed and presented using a technique suitable to the scope of this thesis; in Chapter 3, a measurement model for using guardrail detections is proposed, and a solution to use these measurements in multiple scenarios is explained; in Chapter 4, methods to introduce a birthing function in order to create a prior for the target are presented; in Chapter 5, solutions to smooth the final trajectories are briefed; and finally, in Chapter 6, the results of the various algorithms in different scenarios are illustrated with graphs, comparisons and discussions.

2
Extended Target Tracking

There are several detections at any instant in the radar. In such a sensor, discriminating between detections is vital. For the needs of this thesis, the detections are classified as shown in Figure 2.1.

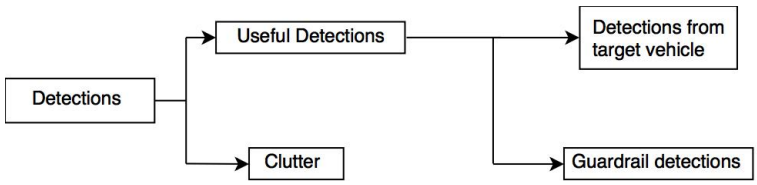


Figure 2.1: Classification of detections from the radar based on utility and place of origin.

This chapter deals with the detections that arise from the target. If there is only one detection from the target at every instance, then the target may be assumed as a point target and the basic filters may be used to track it. These basic filtering techniques, when working as a standalone algorithm, cannot accurately handle the cases when there are multiple detections from the target. Thus, it would be beneficial if the target is to be "extended" in order to include all detections in an efficient manner.

There are several techniques to achieve this goal, and this chapter provides an appropriate background study which includes the motivation for using the chosen method, which is the *Radar resolution model*. The chapter goes on to further explain this method in detail, along with the assumptions and simplifications adopted to make it suitable for the task at hand.

2.1 Choice of Method and Background Study

An extended object is by definition an object that can result in multiple spatially distributed measurements at each time step. As a consequence, a target becomes an extended object when the sensor resolution is high enough such that the object occupies more than one resolution cell. This may result in the target giving rise to multiple measurements at each time instance [20].

There are multiple methods that have been developed and studied on extended target tracking. During the thesis work, several of these methods where studied and validated. In [20], a number of these methods are briefly explained and there are also discussions on appropriate applications. A suggestion, which was based on modeling the target as a spatial model, from this paper was also used in this

thesis, which is explained in Section 4.2.3. This method deals with tracking a single extended object, which is exactly what is required, but a close look revealed that the extent of the object is not fixed and is adaptive based on the measurements. But it is known that this is not true in the case of a target vehicle, since the object size is fixed and there are only a finite number of sources around the vehicle that can give rise to radar detections.

This means that it should be possible to fix a number of predefined points around the vehicle, which then becomes an extended object, due to its behaviour of being composed of several reflecting surfaces (reflectors), and then associate detections with them. This method demands modeling the sensor as well as the target in a manner which is based on the resolution of the sensor used, and the way it *perceives* the object. This is exactly what the *Radar resolution model* proposes. As an extended target tracking technique, it is more advanced and takes the thesis closer to its expected outcome. Further motivation for choosing this method is provided in the Section 2.1.1.

2.1.1 Motivation for the selected method

A car has some resemblance with a *group object* as defined in [20] from the sensors point of view. That is, the likelihood of getting a radar detection from the surface of a regular car is not uniformly distributed. This is because of how the different parts of the car reflect the radar beam. For example, the wheelhouses tend to reflect more radar energy than the bonnet and are therefore more likely to give rise to radar detections. This information can be used to model the target car as a group of smaller extended targets with the mean of each extended sub-target placed where it is most likely to reflect the most radar energy. This radar energy that is reflected back is explained in the Section 2.2.3, where a mathematical interpretation of this expected return amplitude is provided.

It is not possible to be sure that all sub-targets will be visible by the sensor at all times, so the relation between the sub-targets needs to be known in order to keep track of all of them even when some of them are not visible. A visibility calculation could be introduced, as explained in 2.2.2, which determines the sub-groups that can actually form a part of the measurements that arise from the radar by mapping the visible reflectors to the measurement space. Thus, this fixed frame relation between the sub-targets results in that the target is an extended target by definition, since the reflectors cannot have a motion of their own as in a group target, due to the fact that the reflectors are bound to match these criteria of visibility and expected return amplitude.

It is then also possible to resolve multiple radar detections into an improved orientation estimate for the target vehicle since the measurements are rightly matched to their actual place of origin(from the nearest reflector). This can only be done if the relation between the different sub-target is known and given that the detections are associated with different sub-targets. The algorithm would still perform as well as any other extended target tracking (or even better) when there are detections from the same reflector or there is only one detection from the target overall. The drawback is though that the rough dimensions of the target vehicle need to be known

in order to place them as close as possible to their actual true positions.

2.2 Radar Resolution Model

The algorithm that has been explained, implemented and reported here follows from [10]. The implementation was made using the radar-only detections data provided by Delphi. The main idea behind this implementation is to be able to resolve multiple measurements that arise from a single target vehicle which leads to ambiguity in the actual position and orientation of the vehicle. This is made possible by assuming that the target is not a point target but a combination of several reflectors that are fixed around it in the target frame. Using this mapping and depending on the state prediction of the target vehicle in that instance (as shown in Algorithm 7), the reflectors are positioned in the host frame. The reflectors are then grouped based on the resolution of the radar sensor and thus, the expected positions (mean of reflector group) of origin of the measurements from the extended target can be estimated, along with a *gate* (covariance of reflector group).

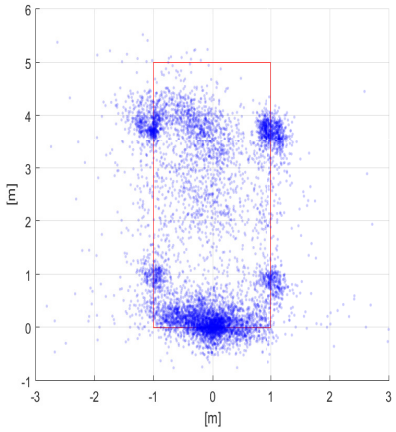
Then, based on a mapping from the state space to measurement space, the expected measurements from each reflector group is predicted. The reason the expected measurement prediction is made based on the group (which occupies one unit of the resolution of the radar) and not the reflectors themselves is because the radar can give rise to detections only once inside one resolution unit. These predictions of measurements are then compared with the actual measurements for associating the appropriate data to its corresponding place of origin. This enables the formulation of hypotheses for the measurements belonging to the various groups, based on which the update step is done as shown in Algorithm 8.

This can be shortly explained by considering the target detections in Figure 1.1b, where there are two detections on the back and one from the right side of the front wheel of the vehicle. If these detections are matched correctly with their place of origin on the vehicle, then there is more information regarding the state of the target and the heading estimates are also much better. From the sensor point of view, this demands a model for the radar in a way which allows for this computation to be possible. The radar resolution model proposed in this paper [10] has this approach to model the measurements based on the physics behind the working of a radar sensor.

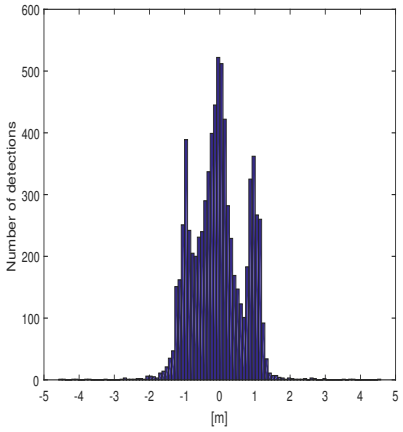
2.2.1 Target Model

The vehicle being tracked is composed of several reflectors. Each reflector is associated with a specific field of vision, which defines the visibility of that particular reflector and a strength which defines the expected return signal amplitude from that reflector on the radar, explained in 2.2.3. For defining the positions, the visibility regions and the expected signal strength of these reflectors, out of some data collected for some scenarios using the radar, the Figure 2.2a shows the detections resolved in the local target frame. The Figures 2.2b and 2.2c show the histogram of distributions of the detections all around the vehicle from the X_{target} and Y_{target}

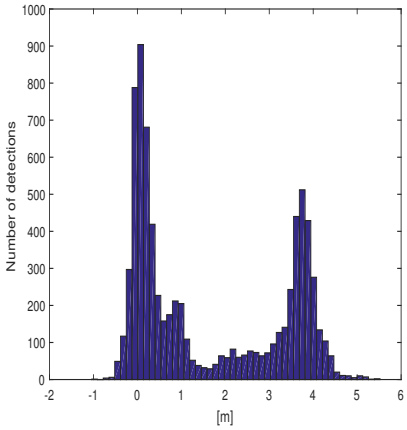
axes respectively. These illustrations are more or less enough to define the strategic reflector points around the target.



(a) Detections distribution all around the vehicle.



(b) Histogram of detections from the rear of the vehicle.



(c) Histogram of detections from the side of the vehicle.

Figure 2.2: Detections distribution all around the vehicle, along with the histograms showing the motivation for choosing the reflectors.

From the Figure 2.2a, it is already obvious that most of the detections come from the rear of the vehicle and this is due to the host being behind the target. Since there is only one forward facing radar, most of the measurements bounce from the rear. The next place giving rise to most detections is the front wheel. As stated earlier, the wheelhouse is a sound reflector of radar energy. Along with this reason, there is also the fact that the front wheels have a steeper angle while turning than the actual vehicle and sometimes give detections that are head-on with the radar. Rear wheels are also a strong source of detections, but not as much as the front wheels and this helps the definition of the return signal strength. The detections that fall within the vehicle do not, of course, come from within it but are reflected off the road, and this is not considered to be a reflector within the vehicle.

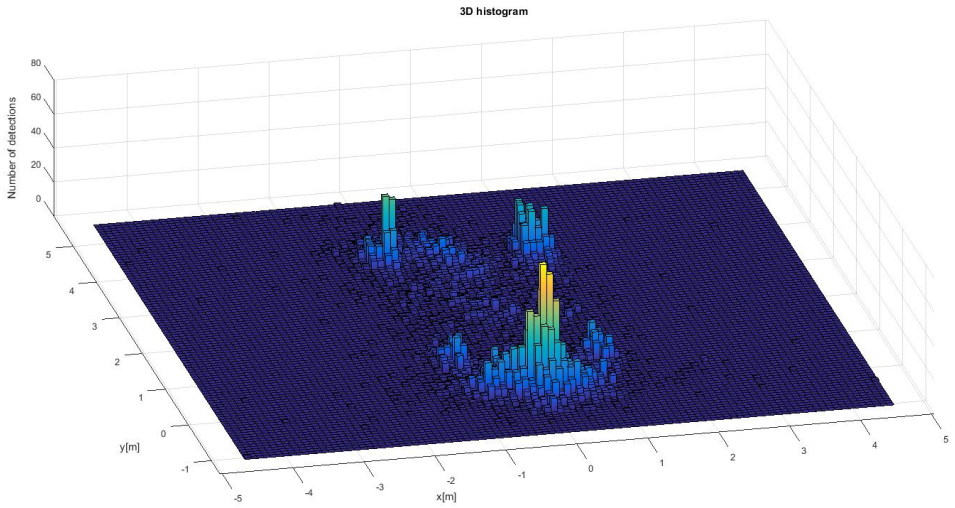
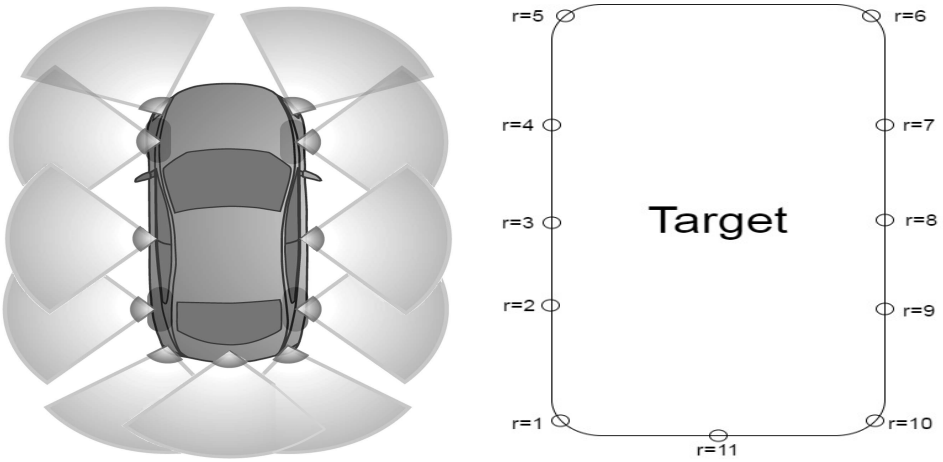


Figure 2.3: 3D view of the histogram of detections shown in Figure 2.2a.

Figure 2.2b is a histogram of the detections on the rear of the vehicle, and 2.2c is a histogram of the detections on the side of the vehicle. The choice of reflectors and their parameters is made from these Figures. Figure 2.2b suggests that there should be three reflectors on the rear, and the Figure 2.2c can be mirrored to choose reflectors on either side of the target, and it suggests that there should be three reflectors on the side. But this mapping might lead to some bias if the detections bounce off some other part on the side, and they are associated with another reflector. Hence more reflectors need to be chosen on the sides since it is the longer side and this is done heuristically through a trial and error approach. Finally, 11 reflectors are chosen around the target, strategically placed so as to be able to get maximum performance while not blowing up the computational complexity. Figure 2.4b shows the IDs that the reflectors use and their arrangement around the target. A basic sketch of these reflectors is given in Figure 2.4a.

As for the defining parameters of the reflectors, the visibility region of a reflector defines the region within which the sensor, if placed in the line of vision, would be able to get a detection. Of course, just because a reflector is visible doesn't mean that it could give rise to a detection. But, at this point, the interest is only to look at all the reflectors and define a good visible region for each of them. The other factor that is very important to define the reflectors is the Radar Cross Section (RCS) which is an intensity parameter describing the amplitude of the return signal expected from an object on the radar. This is chosen by taking appropriate ratios from the histograms presented above. The values for the positions, visibility regions and RCS of each reflector in the target frame are given in Table 2.1. The higher the RCS value, the higher strength that the reflector is expected to reflect the radar energy, and is therefore, more likely to give rise to a detection.

The coordinates of the predicted sigma points for each reflector in the global frame should be computed in order to keep in mind that the sensor frame is in motion and be able to visualise the entire trajectory in the world frame. The first transformation is to convert the reflector coordinates in the target frame to the



(a) Position of reflectors around the target vehicle, shown along with their visibility region. (Not drawn to scale). Figure from [10].

(b) Reflectors chosen and theirs IDs.

Figure 2.4: Mapping of the chosen reflectors (not to scale) and their IDs.

Table 2.1: Reflector positions from the target centre,visibility regions and relative RCS values in the target frame.

Reflector (j)	Position (x_j, y_j)	Visibility region	RCS
1	(-0.95,-2.3) $[m]$	$\pm 45\,[deg]$	1
2	(-0.95,-1.4) $[m]$	$\pm 70\,[deg]$	2
3	(-0.95,0.0) $[m]$	$\pm 70\,[deg]$	0.5
4	(-0.95,1.4) $[m]$	$\pm 70\,[deg]$	2
5	(-0.95,2.3) $[m]$	$\pm 45\,[deg]$	1
6	(0.95,-2.3) $[m]$	$\pm 45\,[deg]$	1
7	(0.95,1.4) $[m]$	$\pm 70\,[deg]$	2
8	(0.95,0.0) $[m]$	$\pm 70\,[deg]$	0.5
9	(0.95,-1.4) $[m]$	$\pm 70\,[deg]$	2
10	(0.95,-2.3) $[m]$	$\pm 45\,[deg]$	1
11	(0.0,-2.3) $[m]$	$\pm 70\,[deg]$	2

sensor frame

$$r^{(i,j)}_{x(k|k-1)} = \hat{\mathcal{Z}}^{(i)}_{x(k|k-1)} + x_j \cos(\hat{\mathcal{Z}}^{(i)}_{\psi(k|k-1)}) - y_j \sin(\hat{\mathcal{Z}}^{(i)}_{\psi(k|k-1)}), \tag{2.1}$$

$$r^{(i,j)}_{y(k|k-1)} = \hat{\mathcal{Z}}^{(i)}_{y(k|k-1)} + x_j \sin(\hat{\mathcal{Z}}^{(i)}_{\psi(k|k-1)}) - y_j \cos(\hat{\mathcal{Z}}^{(i)}_{\psi(k|k-1)}), \tag{2.2}$$

given target's predicted sigma points $\hat{\mathcal{Z}}^{(i)}_{(k|k-1)}$ with coordinates $(\hat{\mathcal{Z}}^{(i)}_{x(k|k-1)}, \hat{\mathcal{Z}}^{(i)}_{y(k|k-1)})$. These coordinates are the ones that define the centre of the target vehicle in the sensor frame. The coordinates (x_j, y_j) are the fixed positions of each reflector in target frame given in Table 2.1. These coordinates are then used to form a set of sigma points for each reflector $R^{(i,j)}_k$ in the measurement space. Since these

computations are made in the world frame, relative angle between host and target in the world frame is computed as follows

$$\alpha^{i,j} = \arctan\left(\frac{r^{(i,j)}_{y(k|k-1)} - s_y}{r^{(i,j)}_{x(k|k-1)} - s_x}\right). \tag{2.3}$$

In the measurement space, the second argument is the range rate, which means that the velocity of the target needs to be calculated relative to this angle, at each reflector point.

$$\begin{aligned} r^{(i,j)}_{v(k|k-1)} =& \hat{\mathcal{Z}}^{(i)}_{v(k|k-1)} \cos(\hat{\mathcal{Z}}^{(i)}_{\psi(k|k-1)} - \alpha^{i,j}) + \\ & \hat{\mathcal{Z}}^{(i)}_{\psi(k|k-1)} ||r^{(i,j)}_{k|k-1}|| \cos\left(\frac{\pi}{2} + arg(r^{(i,j)}_{k|k-1}) - \alpha^{i,j}\right) \end{aligned} \tag{2.4}$$

The final mapping of each reflector to the measurement space is computed as follows

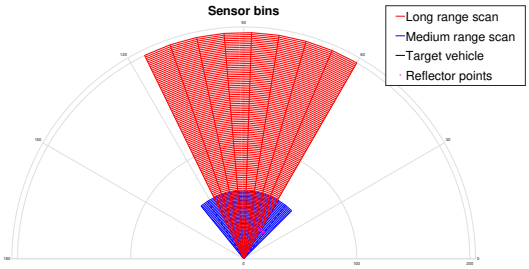
$$R^{(i,j)}_k = \begin{bmatrix} \sqrt{(r^{(i,j)}_{x(k|k-1)} - s_x)^2 + (r^{(i,j)}_{y(k|k-1)} - s_y)^2} \\ r^{(i,j)}_{v(k|k-1)} - s_v \cos(s_\psi - \alpha^{i,j}) \\ \alpha^{i,j} - s_\psi \end{bmatrix}. \tag{2.5}$$

where $i = 1, 2, ..., 2n_z$ is the total number of sigma points and $j = 1, 2, ..., 11$ is the reflector number. With these steps, the transformations between target space and measurement space, represented as $z_k \xrightarrow{R_i(\cdot)} R^{i,j}_k$ is done. This mapping is important, especially in the steps involving checking for visibility of reflectors and for data association.

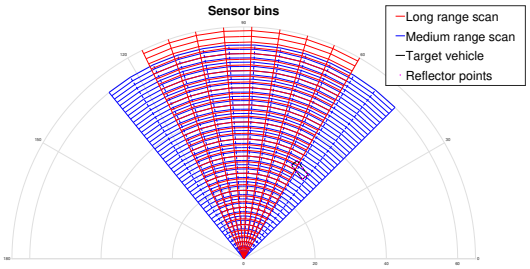
2.2.2 Sensor Model

As explained earlier, the radar can discriminate between detections that are divided by range and range rate. This gives rise to a feature of the sensor called *resolution bin*. The size of these *resolution bin* is such that a large target can occupy multiple bins and thus generate multiple detentions from the sensor. On top of these bins the radar sensor is also able to separate detections if the azimuth angle between them is greater than a certain threshold value. An illustration of these bins is provided in Figures 2.5 and 2.6.

There are no fixed angular bins for the radar, like the range and Doppler bins. The detections are generated based on energy from different range and Doppler (not represented in these figures) bins. The detections can then further be separated by a difference in azimuth angle which is represented here as fixed angular bins. The angular bins for the actual sensor are more dynamic in the sense that any two detections that have been separated based on their range and range rate are checked for the difference in their azimuth angles. If this value is more than a threshold value, the two detections are processed to be two separate measurements, and if the angular difference value is less than the threshold, it is considered to be one detection. But for the modeling, the angular bins need to be fixed in order to classify them based on the predictions. This is a simplification that needs to be adopted in order to

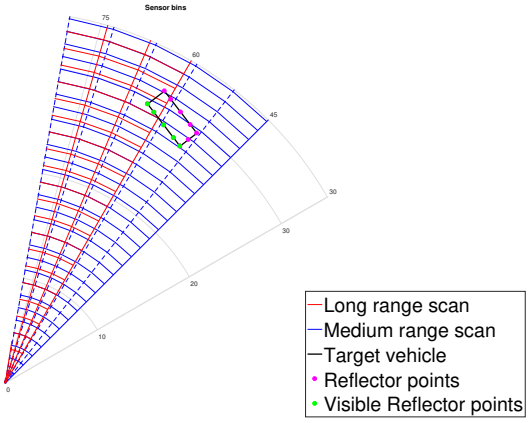


(a) The full field of view of the sensor showing both the long range ans medium range scans.

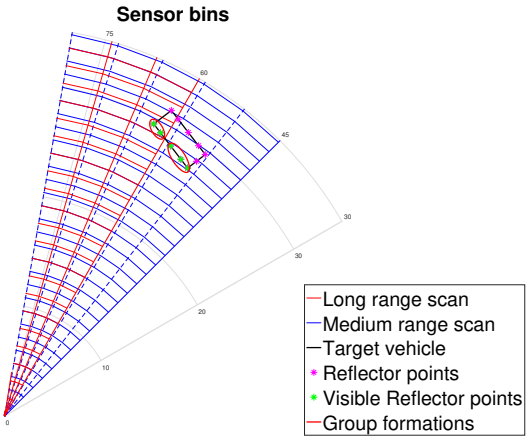


(b) Zoomed in showing the medium range view.

Figure 2.5: Illustration of the radar sensor with a medium range and a long range scan. Note that these are not drawn to scale, i.e., the values of the actual resolutions were not used to generate this Figure.



(a) Zoomed in on the target. Here the reflector point start to be visible



(b) An example of how the reflectors are grouped together based on range and azimuth angle.

Figure 2.6: The grouping of reflector is done based on the range, angular and Doppler bins (the Doppler bins are not represented on this Figure) as well as visibility.

perform the dynamic group allocation as shown in Section 2.2.4.2. This may cause the wrong detections being associated to the wrong groups, but this problem can be alleviated to an extent by considering cross correlations between groups, as shown in Equation 2.21.

2.2.3 Expected Return Signal Amplitude

The expected return signal amplitude is used to determine the weights of each visible reflector at the current time instance. That is, the higher the return signal amplitude from a specific reflector the more likely it is that it will give rise to a detection. The base of the return signal amplitude for each visible reflector is the fixed RCS given

in Table 2.1. The fixed RCS values are indications on how much energy a reflector can reflect relative to all visible reflectors. The expected return signal amplitude then be computed as [10]

$$\Sigma_i = A_a(\theta_i)A_r(r_i)v_\sigma^i(\alpha_{i,s}, \psi_k^j), \tag{2.6}$$

for each visible reflector i. $A_a(\theta_i)$ is the antenna gain pattern, $A_r(r_i)$ is the signal attenuation and $v_\sigma^i(\alpha_{i,s}, \psi_k^j)$ is the visibility functioned that was modelled as a square function spanning the visibility region of each reflector with the fixed RCS value for each visible reflector. The signal attenuation is related to the range r_i to each reflector [3]

$$A_r(r_i) \propto \frac{1}{r_i^4}. \tag{2.7}$$

Due to lack of information on the antenna gain pattern for the mid-range radar sensor $A_a(\theta_i)$ was modelled as one for $\theta = 0$ (directly in front of the sensor) and then with a linear decay to the edges of the sensors FOV.

$$A_a(\theta_i) = \frac{\theta_{max} - |\theta_i|}{\theta_{max}} \tag{2.8}$$

2.2.4 Algorithm

With the basic modeling of all systems done, and the extended target tracking scenario set up, the algorithm that follows aims to track the target vehicle based on the assumed target spacial dynamics as well as the sensor dynamics that have been explained above. It follows many of the key steps described in [10], especially in sections 2.2.4.3 and 2.2.4.5.

2.2.4.1 State Prediction

The model used for this algorithm is the Coordinated turn (CT) model, described in Appendix A.1, along with the nonlinear transition matrix $f(z_{k-1|k-1})$ which predicts the motion of the state model from time step $k-1$ to k. This prediction of motion model involves the use of a sigma-point method called the Cubature Kalman Filter(CKF), also stated in the Appendix A.3 [15]. The CT model and the CKF were determined to be the best method to handle a highly nonlinear case such as that of vehicle dynamics. This test was done by employing different nonlinear estimators like the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), along with different motion models like the Constant Velocity (CV) and Constant Acceleration (CA) models. The ones with the best performance were chosen, which were the CT model working alongside the CKF.

2.2.4.2 Dynamic Group Allocation

In order to accurately describe the extended object the reflectors need to be grouped together based on whether the radar sensor is able to separate them as well as if they are visible to the sensor, based on the prediction. This is done by looking both

at the target predicted states as well as the sensor properties, that is, looking at how the radar sensor can be expected to generate detections on the predicted target. These are dealt with in Sections 2.2.1 and 2.2.2. The logic followed for the dynamic group allocation process is:

1. Determine which reflectors are expected to be visible to the sensor.
2. Separate the reflectors that are visible based on the relative velocity (reflectors that fall in different Doppler bins get different IDs).
3. Separate reflectors based on the relative range (reflectors that fall in different range bins get different IDs).
4. Separate the reflectors based on the azimuth angle between them, that is, if the azimuth angle difference between reflectors that fall in the same range and Doppler bins is greater than the threshold provided by the resolution bin, they are given different IDs.

These IDs that are formed refer to the group ID, and thus, all reflectors with the same ID belong to the same group. After forming the groups, the clusters and cluster constellations are formed based on the relative positions of the reflectors within each group. Groups break down into cluster constellations, which in turn contain different clusters of the reflectors. For the case shown in Figure 2.6b, there are two groups being formed since they lie in different resolution bins and are visible to the sensor. The groups contain 3 and 2 reflectors respectively. The Figure 2.7 illustrates the formation of the different cluster and cluster constellations.

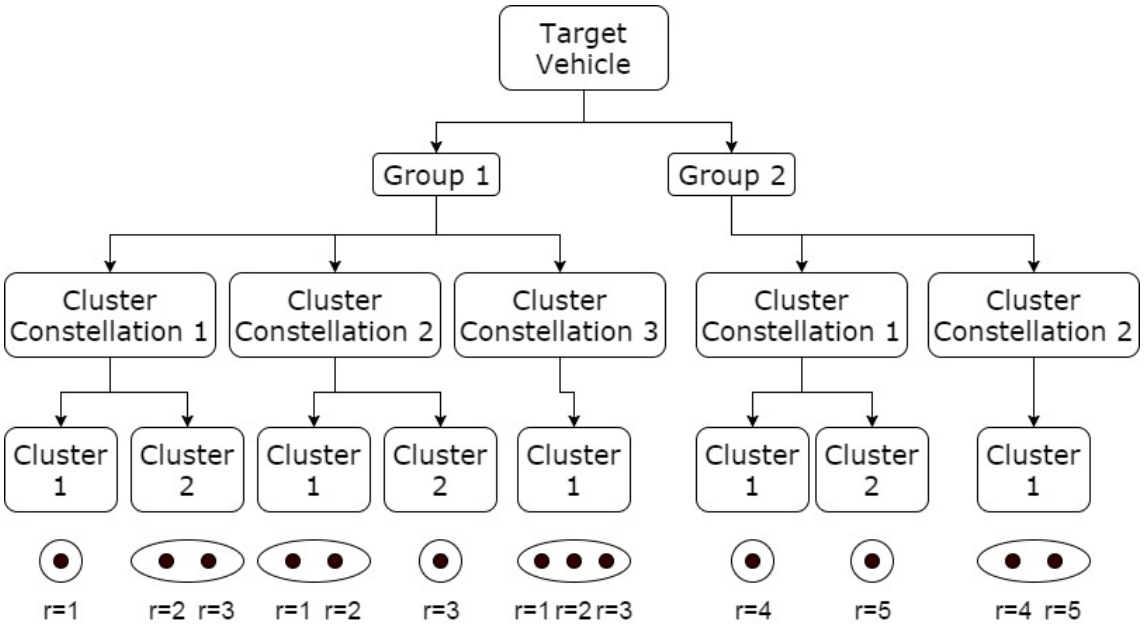


Figure 2.7: Group allocation as done by the algorithm for the scenario shown in Figure 2.6.

2.2.4.3 Measurement Prediction

This step deals with performing the measurement prediction for the mean and covariance of each group. The predicted sigma points $\hat{\mathcal{Z}}^{(i)}_{k|k-1}$ are then propagated through the steps given in Section 2.2.1 and 2.2.3 to convert the target estimates to the reflector positions and signal power in the measurement space. This propagation yields two sets of values: $\{R^i_k, u^i_k\}^{2n_z}_{i=1}$, all the sigma points of the reflector position estimates in the measurement space and $\{\Sigma^i_k, u^i_k\}^{2n_z}_{i=1}$, the signal power of all the sigma points of the reflectors. From $\{R^i_k, u^i_k\}^{2n_z}_{i=1}$, it is straightforward to calculate the first two moments (mean and covariance) of the reflector positions in the measurement space as follows:

$$\hat{r}_{k|k-1} = \sum_{i=1}^{2n_z} R^i_k u^i_k, \tag{2.9}$$

$$P_{rr} = \sum_{i=1}^{2n_z} (R^i_k - \hat{r}_{k|k-1})(R^i_k - \hat{r}_{k|k-1})^T u^i_k \tag{2.10}$$

where $u^i_k = \frac{1}{2n_z}$ gives all sigma points equal weights.

The expected signal power from $\{\Sigma^i_k, u^i_k\}^{2n_z}_{i=1}$ is also computed in order to form the probability of detections of each reflector, group, cluster and so on. This is calculated using

$$\hat{\sigma}_{k|k-1} = \sum_{i=1}^{2n_z} \Sigma^i_k u^i_k. \tag{2.11}$$

From $\hat{\sigma}_{k|k-1}$, an array of signal strengths of all reflectors is created such that $\hat{\sigma}_{l_i}$ ranges from $i = 1, 2, ..., J$ where J denotes the total number of reflectors on the entire vehicle. Since the grouping which is done at the beginning of this iteration or time instant is known, the detection probability of each cluster is calculated, which is denoted as $P^{cc}_n(l)$, where l is an identifier of the cluster being referring to. For this, the assumption that the signal power is Rayleigh distributed in taken into account such that

$$\sigma_l = \sqrt{\sum_{i=m}^{J^{cc}_{n,l}+m} \hat{\sigma}^2_{l_i}}. \tag{2.12}$$

Here, σ_l is the expected received amplitude of the cluster l, and thus ranges from $l = 1, 2, ..., L$ where L denotes total number of clusters across all cluster constellations and groups. $J^{cc}_{n,l}$ refers to the total number of reflectors in the cluster l in constellation cc in group n at the current time instance. m is an identifier that takes the value of the starting reflector identifier that that cluster l has.

From this expected signal amplitude for each cluster, it is possible to compute the detection probability of each cluster as

$$P^{cc}_n(l) = \frac{\sigma_l}{\sum_{l=1}^{L} \sigma_l}, \quad l = 1, 2, ..., L. \tag{2.13}$$

$P_n^{cc}(l)$ is used to compute the weights $q_{n,l}^{cc}$ that indicate the detection probability of each cluster within its own cluster constellation using

$$q_{n,l}^{cc} = \frac{P_n^{cc}(l)}{\sum_{i=p}^{L_n^{cc}+p} P_n^{cc}(i)}, \quad l = 1, 2, ..., L. \tag{2.14}$$

where p is the identifier that takes the value of the identifier that points to the starting index of the clusters within that cluster constellation. The mean of each cluster is $\hat{c}_{n,l}^{cc} = E\{\mathbf{c}_{n,l}|cc, \mathbf{Y}_{k-1}\}$ and can be approximated as in equation (2.15). The weights $\bar{w}_{l_i}$ are derived from the ratio of the expected signal amplitude for the reflectors within the clusters,

$$\hat{c}_{n,l}^{cc} \approx \sum_{i=m}^{J_{n,l}^{cc}+m} \hat{r}_{k|k-1}^{l_i} \bar{w}_{l_i}, \tag{2.15}$$

$$\bar{w}_{l_i} = \frac{\hat{\sigma}_{l_i}}{\sum_{l=1}^{J_{n,l}^{cc}} \hat{\sigma}_{l_i}}. \tag{2.16}$$

The cluster covariance $\mathbf{P}_{\mathbf{c}_l\mathbf{c}_l}^{cc,n} = Cov\{\mathbf{c}_{n,l}|cc, \mathbf{Y}_{k-1}\}$ can be approximated as

$$\mathbf{P}_{\mathbf{c}_l\mathbf{c}_l}^{cc,n} \approx \sum_{i,j=1}^{J_{n,l}^{cc}} (\mathbf{P}_{\mathbf{rr}}^{l_j,l_i} + (\hat{\mathbf{r}}_{k|k-1}^{l_i} - \hat{c}_{n,l}^{cc})(\hat{\mathbf{r}}_{k|k-1}^{l_j} - \hat{c}_{n,l}^{cc})^T). \tag{2.17}$$

The mean and covariance of each group can then be derived from the clusters forming them as well as the detection probability $q_{n,l}^{cc}$ of each cluster.

$$\hat{\mathbf{g}}_n = \frac{1}{N_n^{cc}} \sum_{cc=1}^{N_n^{cc}} \sum_{l=1}^{L_n^{cc}} q_{n,l}^{cc} \hat{c}_{n,l}^{cc}, \tag{2.18}$$

$$\mathbf{P}_{\mathbf{gg}}^{n} = \sum_{cc=1}^{N_n^{cc}} \sum_{l=1}^{L_n^{cc}} \frac{q_{n,l}^{cc}}{N_n^{cc}} (P_{\mathbf{c}_l\mathbf{c}_l}^{cc,n} + (\hat{\mathbf{g}}_n - \hat{c}_{n,l}^{cc})(\hat{\mathbf{g}}_n - \hat{c}_{n,l}^{cc})^T). \tag{2.19}$$

The covariance between detections from different groups needs to be computed in order to perform the state update for all groups at the same step. Since every group can give rise to a maximum of one detection(due to the behaviour of the radar sensor giving a maximum of one measurement per resolution bin), the detections are assumed to be uncorrelated (being conditioned on z_k) and therefore the only correlation comes from the uncertainty in the detections themselves. Therefore we can approximate the group cross covariance as:

$$\bar{w}_{n_i} = \frac{\hat{\sigma}_{n_i}}{\sum_j^{J_n^{cc}} \hat{\sigma}_{n_j}}, \tag{2.20}$$

$$\mathbf{P}_{\mathbf{gg}}^{nm} = \sum_{i}^{J_n^{cc}} \sum_{j}^{J_m^{cc}} \bar{w}_{n_i} \bar{w}_{n_j} \mathbf{P}_{\mathbf{rr}}^{n_i,m_j}. \tag{2.21}$$

This allows for the radar modeling to hold even while there is only one measurement prediction mean from a group but the group actually has more than one measurement in that instant (due to the inconsistency between radar modeling and actual

radar working in the azimuth discrimination). Thus, more than one measurement can still be included in the group due to the cross covariance between groups being able to span beyond the resolution bin. This is not utilised in the data association problem, explained in Section 2.2.4.4, but it is in the measurement update step, explained in Section 2.2.4.5 in the form of covariance between measurements between groups, represented as P_{yy} computed in Equation 2.24. This means that the detections are not directly mapped to the correct groups, but mapped as per the predictions that are made, but compensated to a certain degree by considering the cross correlations between groups.

2.2.4.4 Data Association

The data association is based on a gating method. Each reflector group is viewed as an extended target, where there can be more than one detection associated with each group but a detection can only belong to one group. At each time instance the radar detections coming in from the sensor are sorted into target measurements and clutter measurements by forming gates around each group. The position and size of each gate is determined by the corresponding group mean $\hat{\mathbf{g}}_n$ and group covariance matrix P^n_{gg} of each group respectively. To determine if a detection belongs to a particular group the Mahalanobis distance between the detection and the group mean is computed as follows

$$D = \sqrt{(y - \hat{g}_n)^T (P^n_{gg})^{-1} (y - \hat{g}_n)}. \tag{2.22}$$

Detections with $D < 3$ are then associated with the corresponding group. This is the 3σ distance of the Gaussian distribution within which 99.7% of the values are expected to fall. Figure 2.8 shows an example for the gates formed and the measurements associated with each group.

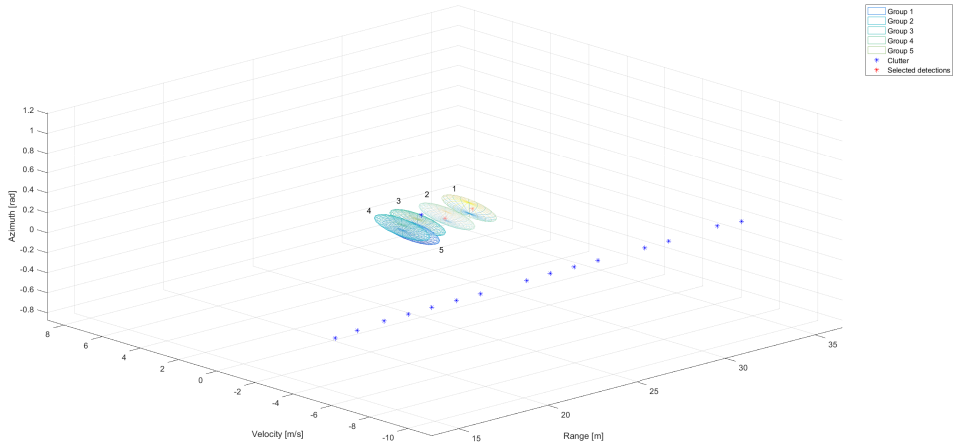


Figure 2.8: Example of the means $\hat{g}_n$ and covariances P^n_{gg} of the groups $n = 1, 2, ..., 5$ at a particular instance, plotted in the measurement space, also illustrating the hard association made with the measurements y_k.

According to the Joint Probabilistic Data Association (JPDA) described in [18] a measurement/detection can only belong to one target which in this case means that it can only belong to one group. But since the object is considered as an extended target, there needs to be another method introduced in this algorithm in order to account for the same target giving rise to multiple detections so that a group may have multiple measurements associated with it.

In the paper [10], for accomplishing data association for such problems, the solution is to form multiple hypotheses such that the detections coming in from the sensor are matched with all the groups. Then, based on a weighing scheme at the end of the step, the most likely hypothesis is given high weight, the most unlikely is almost 0 weight and so on. Such a hypothesis that is formed locally for a group is denoted as $\lambda(j) = n$, where j denotes the detection number in the measurement vector y_k and n is the group number. These local hypotheses are all combined, originating from different groups and proposing different data associations between detections and groups, to form the global hypothesis vector λ. In cases where the clutter measurements need to be ignored or given low weights, the clutter model of the sensor is taken into account, because there needs to be a computation for calculating the probability of the total number of target detections so that these hypotheses can be given corresponding weights. For the clutter detections, the hypothesis is formed by taking $\lambda(j) = 0$, so all hypotheses when $\lambda(j) \neq 0$ belong to a group.

These global hypotheses are formed in such a way that once all the local hypotheses within them have been defined, every measurement has been associated with every group, and thus, there is a large number of hypotheses. However, this scenario was simplified for the thesis due to computational reasons using assumptions explained in Section 2.3.

The resolved detections resulting from the steps followed from Sections 2.3.1 and 2.3.2, are then assembled into a vector that maps the associated group mean and covariance as

$$\hat{y}_{k|k-1} = \begin{bmatrix} \hat{g}_{\lambda(j_1)} \\ \hat{g}_{\lambda(j_2)} \\ . \\ . \\ . \\ \hat{g}_{\lambda(j_n)} \end{bmatrix}, \quad y_k = \begin{bmatrix} y_k^{\lambda(j_1)} \\ y_k^{\lambda(j_2)} \\ . \\ . \\ . \\ y_k^{\lambda(j_n)} \end{bmatrix}, \tag{2.23}$$

$$P_{yy} = \begin{bmatrix} P_{gg}^{\lambda(j_1)} + Y_k^{\lambda(j_1)} + W & \dots & P_{gg}^{\lambda(j_1)\lambda(j_n)} \\ . & . & . \\ . & . & . \\ . & . & . \\ P_{gg}^{\lambda(j_1)\lambda(j_n)} & \dots & P_{gg}^{\lambda(j_n)} + Y_k^{\lambda(j_n)} + W \end{bmatrix}, \tag{2.24}$$

where W is the sensor noise covariance matrix.

2.2.4.5 Measurement Update

In the measurement update step, the relation between the state z_k and the reflectors is computed as in (2.25), where as before all sigma points have equal weights $u_k^i = \frac{1}{2n_z}$.

$$\mathbf{P}_{zr} = \sum_i u_k^i (\mathbf{Z}_k^i - \hat{z}_{k|k-1})(\mathbf{R}_k^i - \hat{r}_{k|k-1}). \tag{2.25}$$

The second step is to approximate the relation between the reflectors and clusters as:

$$\mathbf{P}_{zc_l}^{cc,n} = \sum_i^N \bar{w}_{l_i} \mathbf{P}_{zr}^{l_i}. \tag{2.26}$$

The relation between the clusters and cluster constellations is calculated using:

$$\mathbf{P}_{zg}^{n} = \frac{1}{N_n^{cc}} \sum_{cc=1}^{N_n^{cc}} \sum_{l=1}^{L_n^{cc}} q_{n,l}^{cc} \mathbf{P}_{zc_l}^{cc,n}. \tag{2.27}$$

Using the output from (2.27) the final cross covariance matrix can then be formed as

$$\mathbf{P}_{zy} = [\mathbf{P}_{zg}^{\lambda(j_1)}, ..., \mathbf{P}_{zg}^{\lambda(j_n)}]. \tag{2.28}$$

The final two steps are then the state update and the posterior covariance update:

$$\hat{z}_{k|k} = \hat{z}_{k|k-1} + \mathbf{P}_{zy}(\mathbf{P}_{yy})^{-1}(y_k - \hat{y}_{k|k-1}), \tag{2.29}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{P}_{zy}(\mathbf{P}_{yy})^{-1}(\mathbf{P}_{zy})^T. \tag{2.30}$$

2.3 Assumptions

In order to reduce computational complexity, and also fit the scope of the thesis within the suggested algorithm, some assumptions have been made. These assumptions are made in such a way that the performance is not affected to a high degree with respect to the accuracy of the estimates. The assumptions find their place in the data association scheme presented in [10], where the complexity is high and the performance owing to these complications may be marginally better. This scheme is adapted only to accomplish the following tasks:

- To ignore clutter detections and model the sensor in such a way that these clutter detections get low weights for the hypotheses that match them to any group on the target vehicle.
- To resolve multiple measurements that could arise from the same group, and also these measurements can belong to more than one group due to their place of origin lying in the intersection region of the group covariances.

A simpler approach to solve the multiple hypotheses problem is presented in Section 2.3.1, such that measurements belonging to more than one group may be solved; Section 2.3.2 resolves the cases when one group has multiple measurements; and finally, Section 2.3.3 deals with the clutter model assumptions of the radar, which become superfluous under the other assumptions taken into account.

2.3.1 Hypothesis Formulation

In order to resolve for a detection fulfilling the criteria for multiple groups, only the most likely hypothesis is formed. Thus, the number of hypotheses formed for the data association is equal to the number of measurements that can be associated with distinctly distinguishable groups formed at that instant based on the reflector mapping and sensor model. This is a *hard association* since every measurement can belong only to one group. That means that in the case where a detection can be associated with multiple groups it is only associated with the group that has the smallest Mahalanobis distance.

2.3.2 Moment Matching

The measurement update, as explained in the algorithm, may involve some resolving of measurements in the case where a group has multiple measurements associated with it. This is because according to the algorithm presented, to make a hypothesis, every measurement can only belong to one group, but every group can have multiple measurements associated with it. Thus, there needs to be some logic to resolve these measurements within a single group, this is done by moment matching the measurements using equal weights.

In order to moment match the detections based on equal weights, the detections associated with each group are resolved by computing their mean and covariance,

$$\hat{y}_k^n = \frac{1}{N_n^t} \sum_{i=1}^{N_n^t} y_{i,k}^t, \qquad Y_k^n = \frac{1}{N_n^t} \sum_{i=1}^{N_n^t} (y_{i,k}^t - \hat{y}_k^n)(y_{i,k}^t - \hat{y}_k^n)^T, \tag{2.31}$$

where N_n^t is the number of target detections associated with a group n at the instant k.

2.3.3 Clutter Model

The clutter model of the radar is unnecessary under the assumptions taken into account because of the following reasons:

- The *hard association* associates all the measurements gated through the steps mentioned in 2.2.4.4, and this means that the clutter detections are assumed to be ignored due to their mismatch with this proposed density as long as they are not too close to the mean of the groups.
- The probability of clutter detections within the proposed densities is found to be low, through experiments, and since a scheme is developed to resolve the scenario where multiple measurements belong to a single group, the impact of considering clutter in the circumstances where they are close to the group means is rather minute.
- One of the parameters needed to be defined in the clutter model is the probability of target detection. In this simplified case, the number of global hypotheses formed is 1, with the number of local hypotheses being the total number of groups undergoing a measurement update. Thus, only the most relevant detections are considered as valid detections from the target at every instance,

since only the detections close enough to the target are gated through. This makes the clutter modeling of the sensor an unused parameter, and thus, unnecessary. During instances when there are no detections from the target, the predictions are used anyway for the posterior, and in this case the modeling takes care of ignoring clutter.

3

Using Guardrail Information

The guardrails give rise to multiple detections on the radar. Thus, these detections can be merged in a similar fashion to the target vehicle, as an extended object, in order to use them to further get better estimates on the heading angle of the target since the guardrails tend to follow the road just as a car usually does. This means that the assumption that the target follows a trajectory parallel to the guardrails should hold. In [19] an older version of the guardrail estimator among other sensors is used for road prediction. The guard rail estimator used in this project is an improved algorithm and has shown improvement in performance. The assumption that the target follows a parallel trajectory to the guardrails, however, restricts the scenarios that can be considered for the estimates of the target trajectory since it disallows the situation where the target moves away from the guardrails. This is because not all roads have guardrails on the sides, or it may be that the same guardrails are not being followed anymore as before (when target follows one guardrail and then moves to follow another guardrail), or something as simple as lane-changing where the heading angle provided by the guardrails would not be applicable for a few instances during the manoeuvring of the vehicle. Rather than individually considering all situations, a simpler problem formulation would be to consider two cases: one where the heading is updated based on guardrails and one where it isn't. A likelihood function may then be devised to compute the probabilities of these motions based on the measurements and measurement predictions. The final target state estimate is then done accordingly. In order to do this, an interactive multiple model (IMM) filter [1], [4] is used.

3.1 Assumptions

It is assumed that the estimates of the guardrail positions and its contour are available at the starting of every instant, and this can be interpreted as a prior information, from which a measurement can be generated. This, however, is not true in the case of a real-time implementation since these guardrail estimates are obtained from radar measurements as the posterior of every iteration. It should be noted that the causality requirements that need to be satisfied for a real-time implementation of the algorithms presented below are not considered and therefore, not dealt with.

3.2 Interactive Multiple Model Filter

An Interacting Multiple Model followed from [4] and [11] applies a weighing algorithm based on the proximity of the different updated state estimates to the different measurements models. The different measurement models in this case are not completely independent, since the weighing needs to be done between using barrier estimates and not using them, with both the models simultaneously using the extended target tracking method with radar updates explained in Chapter 2.

The IMM algorithm is based on a Jump Markov System (JMS), where the transition between models is described by a first-order Markov chain [7]. It involves four steps, an extension of the basic steps that make up a normal model-based filter like the Kalman filters. The CT motion model is used, and thus the state space model is a 5×1 vector. Since two different approaches are used to compute the update step, the total estimated states become twice as big, leading to a 10×1 vector for the state space at the end of the update step. From this, using a likelihood function, weights are computed to estimate how much each model is "trusted". The prior from the birthing function is thus initialised to both the state vectors in the first instance and they ideally take the same values until the first measurement from the barrier is available to update one of them. In this case, the top vector is the one that is updated based both on the reflector model and the barrier and the bottom one only gets updated based on the reflector model.

A Transition Probability Matrix (TPM) is defined that numerically represents the probability of switching between the two models during consecutive time instances. This is represented as π_{ji}. This is a fixed number, and thus a tuning parameter that determines the weights that calculate the final state estimates. These weights are represented as μ^j_{k-1}, called the mode probabilities. These values, however, are calculated at each time step based on the updated innovations. Since two models are used, we assign a variable for the total number of state space models $N_r = 2$. π_{ji} is thus a 2×2 matrix and the $\mu^j_{(\cdot)}$ is a 2×1 vector. This is not to be confused with the mixing probabilities $\mu^{ji}_{(\cdot)|(\cdot)}$, which is a 2×2 matrix denoting the probability of both the state estimates mixing with each other.

3.2.1 Mixing

The first step is to obtain the different mixed scenarios from the previous instance. Thus, the probability of mixing, as well as the mixed mean and covariance estimates need to be calculated.

1. *Mixing probabilities:* From the previous mode probabilities μ^j_{k-1}, the mixing probabilities $\{\mu^{ji}_{k-1|k-1}\}^{N_r}_{i,j=1}$ are calculated as:

$$\mu^{ji}_{k-1|k-1} = \frac{\pi_{ji}\mu^j_{k-1}}{\sum_{l=1}^{N_r} \pi_{li}\mu^l_{k-1}}. \tag{3.1}$$

2. *Mixed estimates and covariances:* These mixing probabilities are then used to calculate the mixed estimates of the mean and covariances, also using the previous posteriors of both the state spaces $\{\hat{z}^j_{k-1|k-1}\}^{N_r}_{j=1}$ and $\{P^j_{k-1|k-1}\}^{N_r}_{j=1}$

calculated at the end of the measurement update step.

$$\hat{z}^{0i}_{k-1|k-1} = \sum_{j=1}^{N_r} \mu^{ji}_{k-1|k-1} \hat{z}^{j}_{k-1|k-1} \tag{3.2}$$

$$P^{0i}_{k-1|k-1} = \sum_{j=1}^{N_r} \mu^{ji}_{k-1|k-1} [P^{j}_{k-1|k-1} + (\hat{z}^{j}_{k-1|k-1} - \hat{z}^{0i}_{k-1|k-1})(\hat{z}^{j}_{k-1|k-1} - \hat{z}^{0i}_{k-1|k-1})^T]. \tag{3.3}$$

3.2.2 Radar Resolution Update

Both the state space vectors from the mixed estimates, defined by the two moments $\{\hat{z}^{0i}_{k-1|k-1}\}_{i=1}^{N_r}$ and $\{P^{0i}_{k-1|k-1}\}_{i=1}^{N_r}$ are then passed through the complete radar resolution model algorithm in order to get the posteriors from both the state vectors $[\hat{z}^{i}_{k|k}, P^{i}_{k|k}]$. Note that the measurement update from the radar detections is made for both the state vectors through the radar resolution model, and only then the update for the guardrail heading.

3.2.3 Guardrail Measurement Update

1. For the 1^{st} model, the measurement update using the guardrail information is done as it is in the case of a CKF presented in Algorithm 8. The measurement from the guardrails $y = \Phi$ and its noise covariance W_{barr} are used for this step as shown in Equation 3.4 as computed in Section 3.3.1.

$$[\hat{z}^{1}_{k|k}, P^{1}_{k|k}, s_{barr}] = CKF_{update}(W_{barr}, \Phi, z^{1}_{k|k-1}, P^{1}_{k|k-1}), \tag{3.4}$$

   where s_{barr} is the innovation covariance from performing the barrier update.
2. The mode probability is then calculated based on the log likelihood function using the innovation covariance S^i_k for the i^{th} model computed as shown in Equations 3.8 and 3.9 in Section 3.3.

$$\mu^i_k = \frac{\mathcal{N}(y^i_k; \hat{y}^i_{k|k-1}, S^i_k) \sum_{j=1}^{N_r} \pi_{ji} \mu^j_{k-1}}{\sum_{l=1}^{N_r} \mathcal{N}(y^l_k; \hat{y}^l_{k|k-1}, S^l_k) \sum_{j=1}^{N_r} \pi_{jl} \mu^j_{k-1}}, \tag{3.5}$$

   where $i = 1, 2$, i.e., for all the state vectors which in this case is $N_r = 2$. The measurement vector y^2_k is different from the vector $y^2_k = [r_k \ \dot{r}_k \ \theta_k]$ when there is an update from the guardrails, as explained in Section 3.3.1.

3.2.4 Output Estimate Calculation

The final posterior is calculated as the overall estimate from the individual state space and the mode probabilities.

$$\hat{z}_{k|k} = \sum_{i=1}^{N_r} \mu^i_k \hat{z}^i_{k|k}, \tag{3.6}$$

$$P_{k|k} = \sum_{i=1}^{N_r} \mu^i_k [P^i_{k|k} + (\hat{z}^i_{k|k} - \hat{z}_{k|k})(\hat{z}^i_{k|k} - \hat{z}_{k|k})^T] \tag{3.7}$$

3.3 The Two Measurement Models

The IMM algorithm generally follows the use of multiple motion models, such as combinations of Constant Velocity (CV), Constant Acceleration (CA) and the Coordinated turn (CT) models. However, in this case, the requirement is to have multiple measurement models. The two state vectors are updated such that the top one is updated based on the reflector model algorithm presented previously and the guardrail estimates, while the second is updated only based on the reflector model.

The calculation for the mode probabilities in the IMM algorithm involves the use of the innovation covariance matrices. For the standalone reflector update, this is given by P_{yy} as in Equation 2.24. For the update with the reflector model and the guardrail measurements, this is a combination of P_{yy} and the innovation covariance from the guardrail update s_{barr}, as obtained in Equation 3.4. Thus, when the guardrail update is done along with the reflector model update, the innovation covariances of the two models at an instant k are

$$S_k^1 = \begin{bmatrix} P_{yy(k)} & 0 \\ 0 & s_{barr} \end{bmatrix}, \tag{3.8}$$

$$S_k^2 = P_{yy(k)}. \tag{3.9}$$

It must be noted that the innovation cross-covariance of the two updates are not exactly zero, but can be approximated to a very small value. This was found by doing the measurement update for the 1^{st} model based on the extended measurement vector, which is now 4×1, and observing P_{yy} which would then also comprise of the barrier update innovation covariance. For this, it of course follows that the variance used for the barrier update is W_{barr}. The resulting innovation covariance is found to be almost equal to doing the way it is presented in Equation 3.8. The reason why this way of updating using the extended measurement vector was not followed is because the updates for the reflector model can then be made together using a generic function for both states in the IMM and then separately updating the one with the guardrail measurements. This reduces the computational complexity while keeping the performance of the algorithm intact.

3.3.1 Measurement Generation for the first Measurement Model

The measurements that are used to update the state space in the top 5×1 part of the combined state vector do not directly use any measurement from a sensor since this is done for the heading update from the barrier. The barrier is estimated as a cubic polynomial of the form $a_0 + a_1x + a_2x^2 + a_3x^3 = 0$. This barrier estimation is done with an already available algorithm provided by Delphi Automotive.

This polynomial needs to be converted to a measurement value. This is done by finding the closest coordinate on the polynomial curve to the target vehicle. Once this is done, the immediate next point on the curve is estimated in order to calculate the slope using the classical slope-angle formula: as shown in Equation 3.10

$$m = \frac{y_2 - y_1}{x_2 - x_1}, \Phi = tan^{-1}(m), \tag{3.10}$$

where (x_1, y_1) and (x_2, y_2) are the points on the barrier.

Thus, this makes the measurement from a 3×1 to a 4×1 vector: $y_k^1 = [r_k \ \dot{r}_k \ \theta_k \ \Phi_k]^T$. This is applicable for all reflector points from the reflector model. The mapping from the state space to the measurement space is simple in this case since this is the direct measurement of the target heading which is readily accessible in the state vector of a CT model. Thus, for the barrier measurement, $h(z_k) = z_k(4)$. The update is only made when there are *good* measurements available. This is given by a confidence provided by the guardrail estimator. Therefore, when a heading measurement is flagged to be *bad*, which is given by the confidence flag being set to 0, both the state vectors undergo the radar update but not the heading update.

The resulting estimates are thus a combination of the guardrail heading update and the reflector model update. Naturally, this is subject to two parameters that need to be tuned in order to get desirable behaviour with respect to the updates being made. These are the variance of the barrier values denoted as W_{barr} and the TPM π_{ji}.

4

Target Birthing and Killing

Most sensors, in order to do tracking or state estimation, requires a prior, or in other words some information about the whereabouts of the target at the first instance. A wrong prior mostly leads to the following three issues.

1. The filter takes very long to converge to the suitable filter values.
2. The target is lost and the uncertainty keeps increasing until the tracking has to stop due to the covariance matrix becoming singular.
3. The wrong detections get associated to the target and the target is wrongly tracked.

In these cases, the result is undesirable and thus, giving a good prior is a very important task that lies with the developer of the filtering algorithm. In the real-world scenario, such a prior is never fixed since the tracking takes place online and as shown in Section 1.3, the target could arise anywhere in the FOV. Thus, what is required is that the entire scan-area be searched for invoking the *birth* of an object. This object is the target, the one of interest that needs to be tracked.

Also, targets can move out of the FOV of the radar. This causes the death of an object (the object is no longer visible to the sensor and there for tracking should stop) that has been tracked until that instant. In cases when the target actually moves away, it is right to kill that object, stop tracking it and start searching again. But it is also possible that the target still exists in the FOV, but is killed. This could happen when the target does not give rise to any measurements for some instances continuously, or when the wrong detections get associated to the target and the target is lost. In this chapter, this task of birthing and killing a target is dealt with extensively, and a logic is proposed in order to overcome these difficulties. This logic tries to keep the previously developed algorithms working internally, and invokes them when certain conditions are met.

4.1 Choice of Method

There were investigations made on the different available algorithms for performing this task of creating birth and death. There are several methods to accomplish this, but there was considerable weight placed on keeping the logic complexity to a minimum. The objective is simple: create a birth and feed the prior to the reflector model, and if the target is lost, kill it and begin searching again. Some heuristic methods were initially developed as an attempt to build a simple particle filter that could distribute particles strategically in some resolution bins of the sensor. But this failed due to the randomness of the particles irrespective of the number of particles

chosen. The heuristic particle filter method had some success rate, but it was not robust enough for invoking an extended target tracking method such as the reflector model. This lead to further investigation of more robust methods.

As a trade-off, a Bernoulli Gaussian Sum Filter (BGSF) was implemented, which uses a simple extended tracking method based on a spatial model of an ellipse. The BGSF is responsible for creating the birth and death, and also feeding the prior to the reflector model with the IMM, which would do the actual tracking. The reason for choosing the BGSF was due to its simplicity and making it run alongside the heavy-duty algorithms such as the reflector model would be possible without exploding the computational complexity.

4.2 Bernoulli filter based birthing

The birthing algorithm presented here is based on a Bernoulli Gaussian Sum Filter (BGSF) presented in [14] and [13]. The motion model used in the BGSF was that of a point target based on a linear Kalman filter, both of which are not ideas that are applicable to this thesis. The algorithm proposed in the papers were expanded to include a simple spatial model along with the sigma point method CKF as in the case of the rest of the thesis. Along with these changes, some other ideas and suggestions are also proposed in order to accomplish the task of birthing as effeciently as possible.

4.2.1 Proposed Logic Flow and Algorithm

In order to simplify the birthing for the full reflector model, the birthing and killing are done in a separate filter from the tracking. That means that there are two separate filters running in parallel: one BGSF that runs a much simpler extension model for the target and one that runs the full reflector model. The BGSF is only responsible for birthing the target in order to provide the reflector model with a prior and estimate the probability of existence. Thus, tracking accuracy is not as important. When the target has been found then the state estimation from the BGSF is used to initialise the reflector model which is, from that instant, responsible for tracking the target (or in other words, this is the state estimation that is of interest and the one that is outputted as the posterior). The two filters are then run parallel without any information being fed between them until the existence probability goes under a threshold value. When the estimated existence probability goes under the defined threshold, a signal is sent to the reflector model to stop the tracking until a new target is found. This flow of logic is illustrated using a flowchart in Figure 4.1.

4.2.2 Density-based spatial clustering of applications with noise

The Density-Based spatial Clustering of Applications with Noise (DBSCAN) method [2], is used for clustering data points together when the data is spread throughout the area of interest. It is one of the simplest methods of doing data clustering, and

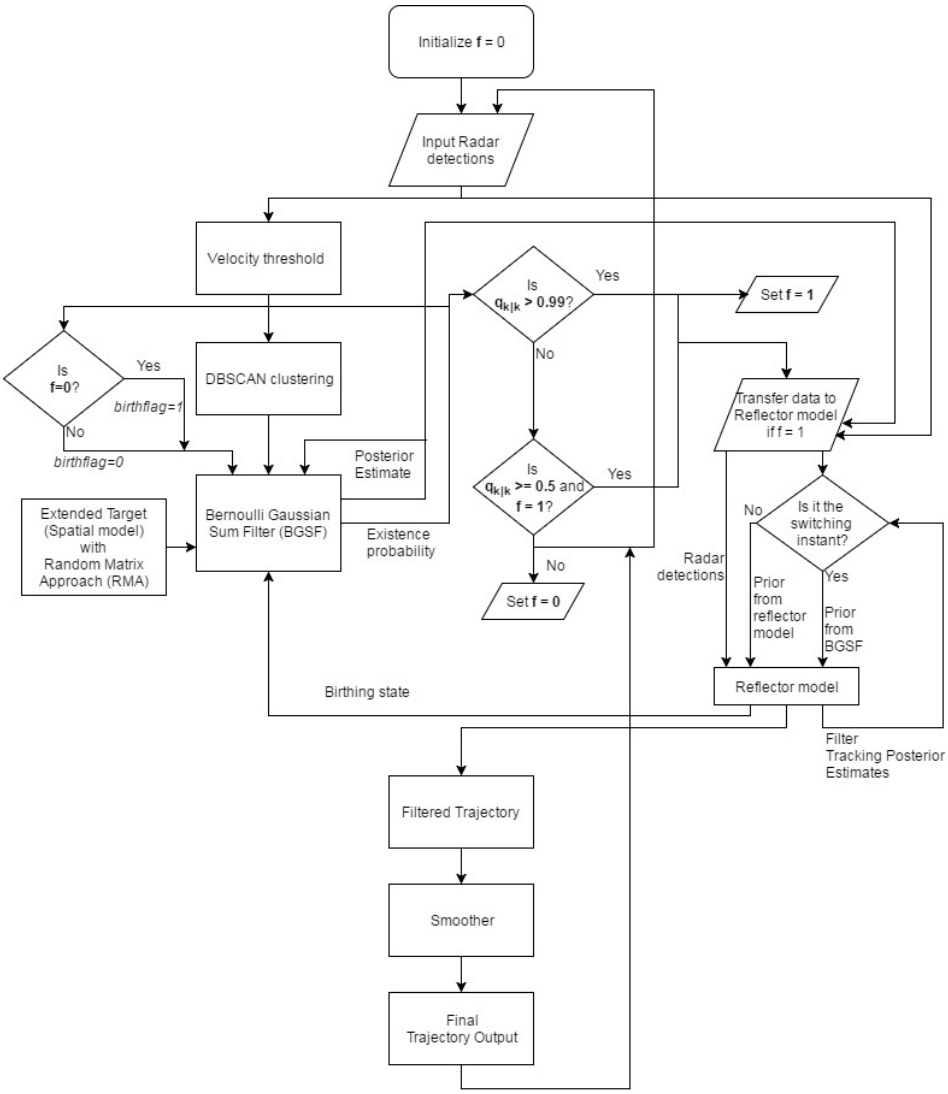


Figure 4.1: A flowchart showing the pipeline of data between algorithms and the flow of logic followed for birthing a target, tracking it using the reflector model and killing it based on the existence probability $q_{k|k}$ calculated by the Bernoulli Gaussian Sum Filter (BGSF).

it is a density-based technique in which there are two parameters that need to be defined. The data is clustered based on these parameters, which are ϵ and $MinPts$. The parameter ϵ defines the neighbourhood in which the clustering takes place and $MinPts$ defines the minimum number of points required to make up a cluster. The source code was obtained from [21].

Only the detections with a minimum threshold value for range rate $\dot{r}_k^i$ are propagated to the DBSCAN logic for clustering. This is defined using a function that thresholds the incoming measurements based on the range rate $\dot{r}_k^i$. Only the measurements that satisfy the velocity threshold, i.e., $\dot{r}_k^i > v_{threshold}$ are selected as valid measurements. This reduces the unnecessary cluster considerations and minimises any clutter detections considered. In further sections, this is addressed

to as a function

$$[y_{thres(k)}] = \textbf{function } velocity\_threshold(y_k), \tag{4.1}$$

where y_k are all measurements at instant k and $y_{thres(k)}$ are all measurements that satisfy the threshold condition.

The parameters are selected as follows:

- The clustering is found to give reasonable values when $\epsilon = 3$, which is selected based on two known scenarios: one when there are many detections from everywhere on the vehicle, and one when there are detections from other objects around the vehicle. This is done so that a trade-off is found to include as much as possible detections from the target, while also ignoring as much possible the detections from other objects around the target.
- The value for $MinPts = 1$ for the simple fact that the target does not always give rise to multiple detections, so it would be better to have these single points clustered in order to keep track of the target than ignore them as noise (which is what is done to data points not satisfying any ϵ-neighbourhood criteria and does not have the minimum number of points to form a cluster).

Before the data points are clustered, the measurements are resolved to coordinates in the host frame as

$$p_k^i = [r_k^i cos(\phi_k^i); r_k^i sin(\phi_k^i)], i = 1, 2, ..., n_y, \tag{4.2}$$

where r is the range of the measurements, ϕ is the azimuth of the measurements and n_y is the total number of measurements at that instant k.

Figure 4.2 illustrates one such instance when the clustering has been done for the radar measurements. The visual data from the camera mounted on the vehicle is shown in Figure 4.2a for comparing how the data clustering should be done. As seen from this figure, there are two distinct data clouds, and the DBSCAN logic works flawlessly for this case. From Figure 4.2b, it can be concluded that the suggested values for ϵ and $MinPts$ handle the scenario well and give the desirable outputs of having formed two clusters.

A simple function is built over the existing one suggested, which is defined as

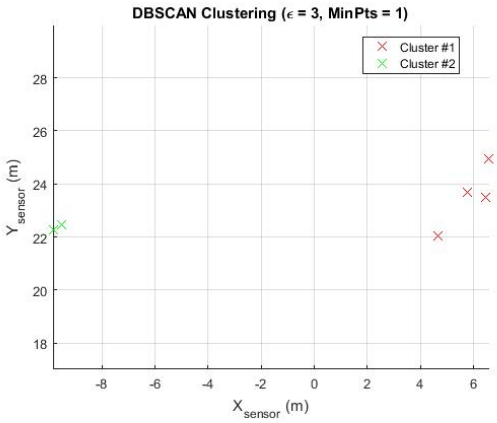
$$[Y_c^{(i)}, n_{clusters}] = \textbf{function } DBSCAN(y_k). \tag{4.3}$$

The function returns $Y_c^{(i)}$, which is an array of clustered measurements of varying size (since the number of measurements in a cluster is not fixed), with i representing the cluster number in $n_{clusters}$ which is the total number of cluster formed.

4.2.3 Spatial Model

It is possible to track a single extended object using the Random Matrix Approach (RMA), originally proposed in [6]. The method involves the use of a spatial model to describe the extended object, defining it using a kinematic state vector z_k and an extent matrix Z_k. The vector z_k, as before, is the state vector of the CT model. The extent matrix is a symmetric, positive definite $d \times d$ matrix, where d is the dimension of the object. The target being tracked is done so without modeling the

(a) Visual comparison for validating the DBSCAN logic.



(b) An example of how the data is clustered, when $\epsilon = 3$, $MinPts = 1$.

Figure 4.2: The clustering done by DBSCAN showing two clusters being formed for this instant. Any mismatch between the detections in the visual comparison and the graph is due to the noise in the radar sensor. The figures are plotted in the host or sensor frame.

elevation, and thus is $2D$, giving $d = 2$. The extended spatial model is an ellipse, which suits the task at hand because of its simplicity and its ability to associate multiple measurements in a more sophisticated way than a single Gaussian. In order to model the elliptical extent model [17] was referred. The transition density for this spatial model was proposed in [9] as:

$$\begin{aligned} p(z_{k+1}, Z_{k+1}|z_k, Z_k) &\approx p(z_{k+1}|z_k)p(Z_{k+1}|Z_k) \\ &= \mathcal{N}(z_{k+1}; f(z_k), Q_k) \times \mathcal{W}_d(Z_{k+1}; n_k, Z_k/n_k), \end{aligned} \tag{4.4}$$

where $\mathcal{W}_d$ is a Wishart distribution describing the dynamic model for the extent matrix Z_k, $n_k > 0$ being the factor that describes the process noise (n_k and process noise are inversely proportional). The parameters that define the Wishart distribution are v, the degrees of freedom and V, which is the scale matrix. This type of modelling is suitable when the rate of turning of the target is not high. Since the radar samples at a high frequency (30 Hz) with respect to the velocity of the target and the initial manoeuvres while the birthing is being done are not erratic, this model would be a good fit for the task. This leads to the following prediction steps, which were obtained from [20] (Algorithm 1). The only changes that need to be made while following [20] were the change to the CKF method of prediction and update, instead of the linear Kalman filtering equations.

Algorithm 1: Random Matrix Prediction

function RMP($V_{k-1|k-1}, v_{k-1|k-1}, z_{k-1|k-1}, P_{k-1|k-1}$)
Data: $V_{k-1|k-1}$ posterior scale matrix from previous instant, $v_{k|k-1}$ posterior degrees of freedom from previous instant, $z_{k-1|k-1}$ posterior state estimate from previous instant, $P_{k|k-1}$ posterior state covariance estimate from previous instant,T_s sampling time, τ temporal decay constant.
Result: $V_{k|k-1}, v_{k|k-1}, z_{k|k-1}, P_{k|k-1}$
begin
  $[z_{k|k-1}, P_{k|k-1}] = CKF_{pred}(z_{k-1|k-1}, P_{k-1|k-1})$
  $v_{k|k-1} = 2d + 2 + e^{-T_s/\tau}(v_{k|k-1} - 2d - 2)$
  $V_{k|k-1} = \frac{v_{k|k-1}-2d-2}{v_{k|k-1}-2d-2} V_{k-1|k-1}$
end

The measurement model chosen is that with improved noise modelling in order to avoid biased estimates. The extended elliptical object, as proposed by [5],[8] and [9], is modelled as the factorised state density

$$p(z_k, Z_k|Y_k) = p(z_k|Y_k)p(Z_k|Y_k) = \mathcal{N}(z_k; z_{k|k}, P_{k|k}) \times \mathcal{IW}_d(Z_k; v_{k|k}, V_{k|k}), \quad (4.5)$$

where $\mathcal{IW}_d$ is the inverse Wishart distribution. The update steps that follow from these approximations are obtained from [20] (see Algorithm 2).

The Gaussians are updated using CKF [15], but the parameter computation is different from the normal point filter update since the model considered now is an extended spatial model. The necessary steps for computing this are provided in Algorithm 2.

The two matrices $\hat{N}$ and $\hat{Y}$ are proportional to the spread around the predicted measurement and the centroid of the elliptical model, thus defining the extent shape matrix. The initial values for the parameters v and V are initialised through a function *Birthing New Gaussians* shown in Algorithm 4, which is set for every Gaussian that needs to be newly created in order to define a spatial model which will search for the target to be birthed.

4.2.4 Bernoulli Gaussian Sum Filter

The Bernoulli Gaussian Sum Filter (BGSF) presented here follows from algorithms presented in [14] and [13]. The actual implementation was then based on source code provided in [22]. The most significant changes to the original algorithm presented here where to track the Gaussisans as an extended spiral object as well as making use of the DBSCAN in order to cluster measurements.

4.2.4.1 BGSF Prediction

The first step is to predict the existence probability $q_{k|k-1}$ based on the predefined probability of birth p_b, probability of survival p_s and the existence probability from the previous time step $q_{k-1|k-1}$

$$q_{k|k-1} = p_b(1 - q_{k-1|k-1}) + p_s\, q_{k-1|k-1}. \quad (4.6)$$

Algorithm 2: Random Matrix Update

function RMU($y^i, V_{k|k-1}, v_{k|k-1}, z_{k|k-1}, P_{k|k-1}, W$)
Data: y^i measurements, U set of all measurements, $n_y = |U|$ total number of measurements, $V_{k|k-1}$ predicted scale matrix, $v_{k|k-1}$ predicted degrees of freedom, $z_{k|k-1}$ state prediction, $P_{k|k-1}$ state covariance prediction and W is the sensor noise covariance.
Result: $V_{k|k}, v_{k|k}, z_{k|k}, P_{k|k}, \eta_k, S_k, \bar{y}$
begin
  $\bar{y} = \frac{1}{n_y} \sum_{y^i \epsilon U} y^i$
  $Y_k = \frac{1}{n_y} \sum_{y^i \epsilon U} (y^i - \bar{y})(y^i - \bar{y})^T$
  $\hat{X} = \frac{V_{k|k-1}}{v_{k|k-1} - 2d - 2}$
  $\bar{Y} = \frac{W + \hat{X}_k}{n_y}$
  $[z_{k|k}, P_{k|k}, \varepsilon_k, S_k, \eta_k] = CKF_{update}(\bar{Y}, \bar{y}, z_{k|k-1}, P_{k|k-1})$
  $\hat{N} = \hat{X}^{\frac{1}{2}} S_k^{-\frac{1}{2}} \varepsilon_k \varepsilon_k^T (S_k^{-\frac{1}{2}})^T (\hat{X}^{\frac{1}{2}})^T$
  $\hat{Y} = \hat{X}^{\frac{1}{2}} \bar{Y}_k^{-\frac{1}{2}} Y_k (\bar{Y}^{-\frac{1}{2}})^T (\hat{X}^{\frac{1}{2}})^T$
  $V_{k|k} = V_{k|k-1} + \hat{N} + \hat{Y}$
  $v_{k|k} = v_{k|k-1} + n_y$
end

Next is to update the states of all Gaussians that have survived from the previous step $k-1$ according to the Random Matrix Prediction in Algorithm 1 as well as their corresponding weights,

$$w_{u(k|k-1)} = p_s \, q_{k|k-1} \, w_{u(k-1|k-1)}. \tag{4.7}$$

After updating the surviving Gaussians, the new birthing Gaussians are formed as described in Algorithm 4 added to the set of Gaussians. The final step in the prediction is to update the full set of wights that now consists of both the predicted wights of the surviving Gaussians as well as the new birthing Gaussians as follows:

$$\tilde{w}_{u(k|k)} = Q_d \, \hat{w}_{u(k|k-1)} \, \lambda_c \, pdf_c, \tag{4.8}$$

where Q_d is the probability of a missed detection in measurements and is derived as $Q_d = 1 - P_d$, P_d being the predefined probability of detecting a target; λ_c is the average number of uniform clutter detections per radar scan where the number of clutter detections per scan is assumed to be Poisson distributed; and pdf_c is the uniform clutter density which is calculated as

$$pdf_c = \frac{1}{V_{clutter}}, \tag{4.9}$$

where $V_{clutter}$ is the volume of the region where the clutter detections are expected to appear in the sensor space.

The algorithm for the prediction steps is summarised in the Algorithm 3.

Algorithm 3: Bernoulli Gaussian Sum Filter Prediction

function $BGSF_{pred}$ $(y_k, V_{u(k-1|k-1)}, v_{u(k-1|k-1)}, z_{u(k-1|k-1)}, P_{u(k-1|k-1)}, N_{u(k-1)}, w_{u(k-1)}, birthflag)$

Data: y_k all measurements at instant k, $V_{k|k-1}$ predicted scale matrix, $v_{k|k-1}$ predicted degrees of freedom, $z_{k|k-1}$ state prediction, $P_{k|k-1}$ state covariance prediction.

Result: $V_{u(k|k)}, v_{u(k|k)}, z_{u(k|k)}, P_{u(k|k)}, q_{k|k}$

begin

$q_{k|k-1} = p_b(1 - q_{k-1|k-1}) + p_s\, q_{k-1|k-1}$

for $i = 1$ **to** $N_{u(k-1)}$ **do**

$[V^{(i)}_{u(k|k-1)}, v^{(i)}_{u(k|k-1)}, z^{(i)}_{u(k|k-1)}, P^{(i)}_{u(k|k-1)}] = RMP(V^{(i)}_{u(k-1|k-1)}, v^{(i)}_{u(k-1|k-1)}, z^{(i)}_{u(k-1|k-1)}, P^{(i)}_{u(k-1|k-1)})$

$w_{u(k|k-1)} = p_s\, q_{k|k-1}\, w_{u(k-1|k-1)}$

end

$[w_{b(k)}, z_{b(k)}, P_{b(k)}, n_b, v_{b(k)}, V_{b(k)}] = Birth(p_b, q_{k-1|k-1}, w_b, birthflag)$

$\tilde{w}_{u(k|k-1)} = [w_{b(k)} \;\; w_{u(k|k-1)}]$, $\tilde{z}_{u(k|k-1)} = [z_{b(k)} \;\; z_{u(k|k-1)}]$, $\tilde{v}_{u(k|k-1)} = [v_{b(k)} \;\; v_{u(k|k-1)}]$, $\tilde{V}_{u(k|k-1)} = [V_{b(k)} \;\; V_{u(k|k-1)}]$, $\tilde{P}_{u(k|k-1)} = [P_{b(k)} \;\; P_{u(k|k-1)}]$

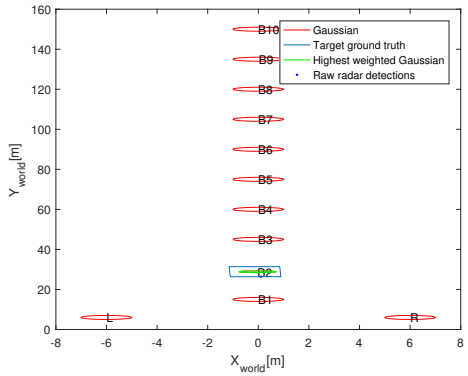
$\tilde{w}_{u(k|k)} = Q_d\, \tilde{w}_{u(k|k-1)}\, \lambda_c\, pdf_c$

end

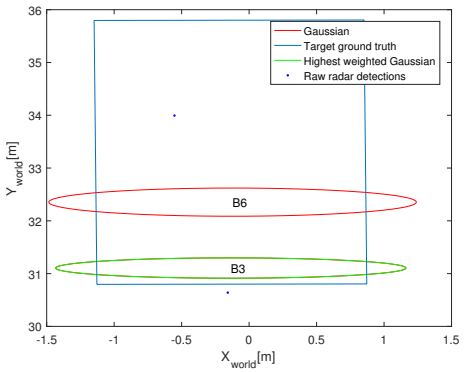
4.2.4.2 Strategic Placement of Gaussians

In a Bernoulli based birthing algorithm, there is an explicit step to create new Gaussians in the FOV to iteratively search for the target. This means that for initialising the new Gaussians, there is a need to look for the target in the right places. As suggested earlier, this may be done by placing them in each resolution bin, but this is highly complex. As a trade-off between convergence time and the complexity of the algorithm, a strategic placement for the new Gaussians was found, shown in Figure 4.3a. There are 12 new birthing Gaussians, of which 2 are placed on the right and left of the radar close to the boundary of its FOV in order to look for vehicles overtaking from the sides and the others are placed directly in front of the radar.

The Figure 4.3a is the prior from the first instance of a scenario when new Gaussians were placed and it illustrates the performance of this method, since the highest weighted Gaussian, which is the one that will survive to the next instant is very close to the actual target. The other Figure 4.3b shows the posterior from the 10^{th} instant during the same scenario, when the birth happens (existence probability has reached a value greater than 0.99). The Gaussian that has got the highest weight has converged to the rear of the vehicle, and this information is sufficient to invoke the reflector model. The reason the Gaussian has converged to the rear of the vehicle is due to the fact that there are more detections from there since the vehicle exists in the FOV already in this case. That means that when the target enters the FOV from the sides, those sides are the ones that the Gaussian converges to. This is illustrated



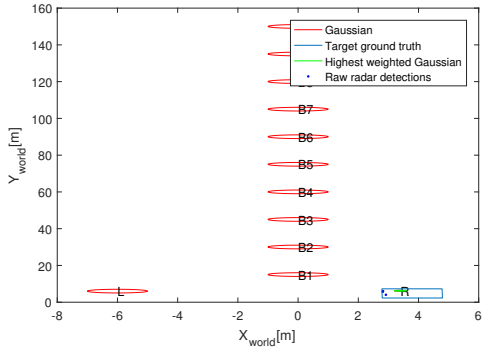
(a) New birthing Gaussians with their IDs and their distribution in the FOV, showing the B_2 Gaussian getting maximum weight.



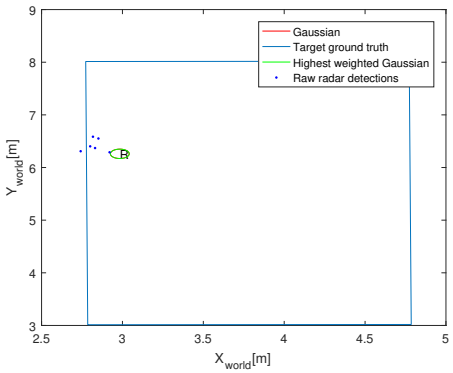
(b) Gaussian converging to the rear of the vehicle while the target is already present in the FOV.

Figure 4.3: An example for Gaussian converging while using the Random Matrix Approach with an elliptical spatial model, when the target is already present in the FOV of radar.

in Figure 4.4a showing the prior during the first instance and Figure 4.4b that shows the posterior at the 8^{th} instant when the switch to the reflector takes place. Using this knowledge that the Gaussians converge to the sides when the target overtakes, this prior information to the reflector model is rotated correspondingly.



(a) New birthing Gaussians with their IDs and their distribution in the FOV, showing the Gaussian with the ID R getting maximum weight.



(b) Gaussian converging to the left side of the vehicle while the target overtakes the host from the right. Here we can see the birthing instance, where a birth is formed on the left front while of the target

Figure 4.4: An example for Gaussian converging while using the Random Matrix Approach with an elliptical spatial model, when the target overtakes the host from the right.

Algorithm 4: Birthing New Gaussians

1 **function** Birth($p_b, q_{k-1|k-1}, w_b, birthflag$)
  Data: p_b probability of detection, $q_{k-1|k-1}$ existence probability from previous instance of the surviving Gaussian, $birthflag$ flag used to define the birthing conditions.
  Result: $w_{b(k)}, z_{b(k)}, P_{b(k)}, n_b, v_{b(k)}, V_{b(k)}$
2 **begin**
3 **if** $birthflag = 0$ **then**
4 $n_b = 0$
5 $w_{b(k)} = z_{b(k)} = P_{b(k)} = v_{b(k)} = V_{b(k)} = \varnothing$
6 **else**
7 $n_b = 12$
8 $w^{(i)}_{b(k-1)} = \frac{1}{n_b}, i = 1, 2, ..., n_b$
9 $w^{(i)}_{b(k)} = p_b(1 - q_{k-1|k-1})w^{(i)}_{b(k-1)}, i = 1, 2, ..., n_b$
10 $[z_{b(k)}, P_{b(k)}] = Place\ Gaussians$
11 $[v_{b(k)}, V_{b(k)}] = initialise\ spatial\ model$
12 **end**
13 **end**

The functions *Place Gaussians* in line 10 and *initialise spatial model* in line 11 in Algorithm 4 are the functions that create the new Gaussians to search for the target. The parameter $z_{b(k)}$ is a vector of states, placed as explained in this Section with some predefined fixed uncertainty $P_{b(k)}$ for all those Gaussians/states. The spatial models parameters as explained in Section 4.2.3 are given those parameters in a vector whose length is equal to the number of newly birthed Gaussians, which means that all spatial models, as extended objects, get the same extent and scaling.

4.2.4.3 Gaussian IDs

During the birthing each new Gaussian gets an ID that is tracked with it. This ID is then used when a birth is triggered to determine which reflector the birth belongs to. This is done in order to make further use of the available prior information. That is, if a birth is formed from a Gaussian that was originally placed to the left edge of the FOV, the birth most likely belongs to the right front wheel. This assumption is done because in this case the right front wheel will be the first large reflector that comes into view. On the other hand, if a birth is formed from a Gaussian that was originally placed directly in front of the sensor it most likely belongs to the centre of the back of the target. During the merging of two Gaussians, the ID is kept from the Gaussian that had the dominating weight. The IDs that these Gaussians are given are : R for the right Gaussian whose left wheel is seen first, L for the left Gaussian whose right wheel is seen first and B_i for all Gaussians whose back is seen with i denoting their position from the radar. These are the identifiers that are shown in Figures 4.3 and 4.4.

4.2.4.4 BGSF update

The first step in the update is to filter the incoming measurements based on the estimated range rate so that only moving radar detections are used for the update. This is done in order to simplify the complexity of the filter and is done under the assumption that the target of interest is moving. The remaining detections are then clustered together using the DBSCAN as explained in Section 4.2.2.

Each Gaussian in the set of Gaussians that was formed in the prediction steps is then updated with each cluster formed in the DBSCAN using the steps shown in the Random Matrix Update in Algorithm 2. This update forms a new set of Gaussians, the corresponding weights need to be updated based on the measurement likelihood $q_k(y)$ which denotes how likely it is that each updated Gaussian belongs to the corresponding measurement. The likelihood $q_k(y)$ and the weights for the new Gaussians are computed as shown in Equations 4.10 and 4.11.

$$q_k^{(i,j)}(y_c^i) = \mathcal{N}(y_c^{(i)}, \eta_k^{(j)}, S_k^{(i,j)}), \tag{4.10}$$

$$w_{mu(k|k)}^{(i,j)} = P_d \tilde{w}_{u(k|k-1)} q_k^{(i,j)}. \tag{4.11}$$

The Gaussians that were formed during the measurement update are then added to the set that was formed during the prediction step making the total number of Gaussians $N_h = (n_c \times n_{gauss}) + n_{gauss}$, where n_c is the number of clusters formed and n_{gauss} is the number of Gaussians in the set that was formed during the prediction.

The existence likelihood that denotes the likelihood of the target existing in the FOV can now be updated as follows

$$q_{k|k} = \frac{q_{k|k-1} \sum_{i=1}^{N_h} \tilde{W}_{u(k|k)}^{(i)}}{\lambda_c \, pdf_c (1 - q_{k|k-1}) + q_{k|k-1} \sum_{i=1}^{N_h} \tilde{W}_{u(k|k)}^{(i)})}. \tag{4.12}$$

The final step is then to normalise the full set of weights be for the full set of Gaussians under go the pruning,merging and capping steps explained in Section 4.2.4.5. The algorithm for the update steps is summarised in Algorithm 5.

4.2.4.5 Prune, Merge and Cap

The pruning is done by having a lower threshold on the normalised weights, so all Gaussians that have lower weights than the threshold ($\hat{w}_{u(k|k)}^{(j)} < w_{threshold}$) get pruned away (removed). The remaining Gaussians are then merged if the Mahalanobis distance (Equation 2.22) between them is lower than a threshold value. All values from ($\tilde{Z}_{u(k|k)}, \tilde{P}_{u(k|k)}, \tilde{v}_{u(k|k)}$ and $\tilde{V}_{u(k|k)}$) belonging to the Gaussians that are determined suitable for merging are matched based on the Mahalanobis distance between them. These Gaussians are then merged by taking a weighted sum of all parameters based on the corresponding weights with the exception of their ID, where dominating weight determines the ID of the new Gaussian. If the number of Gaussians that are left after the pruning and merging exceeds a threshold value, only a limited number of Gaussians with the highest weights are kept for the next iteration/time step. This is done in order to limit the maximum complexity of the BGSF algorithm.

Algorithm 5: Bernoulli Gaussian Sum Filter Update

function $BGSF_{update}$ $(y_k, V_{u(k-1|k-1)}, v_{u(k-1|k-1)}, z_{u(k-1|k-1)}, P_{u(k-1|k-1)}, N_{u(k-1)}, w_{u(k-1)}, birthflag)$

Data: y_k all measurements at instant k, $V_{k|k-1}$ predicted scale matrix, $v_{k|k-1}$ predicted degrees of freedom, $z_{k|k-1}$ state prediction, $P_{k|k-1}$ state covariance prediction.

Result: $V_{u(k|k)}, v_{u(k|k)}, z_{u(k|k)}, P_{u(k|k)}, q_{k|k}, \hat{w}_{u(k|k)}$

begin

$y_{thres(k)} = velocity\_threshold(y_k)$

$[Y_c^{(i)}, n_c] = DBSCAN(y_{thres(k)})$

for $i = 1$ **to** n_c **do**

for $j = 1$ **to** n_{gauss} /* $n_{gauss} = n_b + N_u$ */ **do**

$[V_{mu(k|k)}^{(i,j)}, v_{mu(k|k)}^{(i,j)}, z_{mu(k|k)}^{(i,j)}, P_{mu(k|k)}^{(i,j)}, \eta_k^{(j)}, S_k^{(i,j)}, y_c^{(i)}] = RMU(Y_c(i), \tilde{V}_{u(k|k-1)}(j), \tilde{v}_{u(k|k-1)}(j), \tilde{z}_{u(k|k-1)}(j), \tilde{P}_{u(k|k-1)}(j))$

$q_k^{(i,j)}(y_c^i) = \mathcal{N}(y_c^{(i)}, \eta_k^{(j)}, S_k^{(i,j)})$

$w_{mu(k|k)}^{(i,j)} = P_d \tilde{w}_{u(k|k-1)} q_k^{(i,j)}(z_{k|k}^{(i,j)})$

end

end

$\tilde{W}_{u(k|k)} = [\tilde{w}_{u(k|k)} \;\; w_{mu(k|k)}^{(i,j)}]$, $\tilde{Z}_{u(k|k)} = [\tilde{z}_{u(k|k-1)} \;\; z_{mu(k|k)}^{(i,j)}]$, $\tilde{P}_{u(k|k)} = [\tilde{P}_{u(k|k-1)} \;\; P_{mu(k|k)}^{(i,j)}]$, $\tilde{v}_{u(k|k)} = [\tilde{v}_{u(k|k-1)} \;\; v_{mu(k|k)}^{(i,j)}]$, $\tilde{V}_{u(k|k)} = [\tilde{V}_{u(k|k-1)} \;\; V_{mu(k|k)}^{(i,j)}]$ $, \forall i \leq n_c, \; \forall j \leq n_{gauss}$

$\hat{w}_{u(k|k)}^{(j)} = \frac{\tilde{W}_{u(k|k)}^{(j)}}{\sum_{i=1}^{N_h} \tilde{W}_{u(k|k)}^{(i)}}$ /* $N_h = (n_c \times n_{gauss}) + n_{gauss}$ */

$q_{k|k} = \frac{q_{k|k-1} \sum_{i=1}^{N_h} \tilde{W}_{u(k|k)}^{(i)}}{\lambda_c \, pdf_c (1 - q_{k|k-1}) + q_{k|k-1} \sum_{i=1}^{N_h} \tilde{W}_{u(k|k)}^{(i)}}$

$[V_{u(k|k)}, v_{u(k|k)}, z_{u(k|k)}, P_{u(k|k)}, q_{k|k}, \hat{w}_{u(k|k)}] = PMC(\tilde{V}_{u(k|k)}, \tilde{v}_{u(k|k)}, \tilde{Z}_{u(k|k)}, \tilde{P}_{u(k|k)}, \hat{w}_{u(k|k)})$ /* Prune, Merge and Cap */

end

5

Trajectory Estimation and Smoothing

The final aim is to compute the optimal trajectory that the target vehicle takes, without considering the causality requirements. Therefore, smoothing becomes an integral part of the scope within which this thesis lies. Smoothing reduces the posterior uncertainty by passing the state estimates through a time-inversed filter, which only uses the motion model. Thus, the vehicle trajectory evens out, and there is a possibility to get rid of jerky filter estimates since these can be avoided by re-enforcing the motion model. Smoothed estimates are generally used to have an overview of the entire trajectory that has been achieved. These estimates are less uncertain than their forward filtered counterparts, and therefore can even be used to evaluate performances of a forward-running filter.

Other than just smoothing the trajectory or position estimates, the smoother also accomplishes the smoothing of other states, like velocity, yaw and yaw rate which are generally very erratic due to the noise in the measurements. Several solutions are available, and these are investigated starting from the basic ones to the ones that are more efficient. The Section 5.1 explains the Rauch-Tung-Striebel smoother, and Section 5.2 explains the IMM based smoother. These solutions are applied to the final state estimates emerging from the filters explained prior to this chapter, and the results are shown in Chapter 6.

5.1 Rauch–Tung–Striebel Smoother

The Rauch–Tung–Striebel (RTS) smoother [15] was developed by Rauch, Tung and Striebel and follows from the use of the forward Kalman filtering equations. For this thesis, the CKF was used as the forward filter, and thus, for the backward smoothing problem too, a CKF alongside the RTS shall be used. Using the two moments from the forward filtering case for the posteriors $\hat{z}_{k|k}, P_{k|k}$, the backward smoothing is done. The whole set of the forward filtered posteriors is z_K, P_K with sampling time T and motion model noise Σ. The algorithm for the RTS smoother

is given in Algorithm 6.

Algorithm 6: RTS Smoother

function RTS($\hat{z}_K, P_K, \Sigma, T$)
Data: $\hat{z}_K$ is the complete state vector after the forward filtering, P_K is the set of the posterior covariance at each step during the forward filtering, Σ contains the motion noise covariance for the noise driven states.
Result: $\hat{z}_{k|K}, P_{k|K}$
begin
  for $k = K-1, ..., 1$ **do**
    $\mathcal{Z}_k^{(i)} = \hat{z}_k + \sqrt{n_z}(P_k^{\frac{1}{2}})_i, i = 1, 2, ..., n_z$
    $\mathcal{Z}_k^{(i+n_z)} = \hat{z}_k - \sqrt{n_z}(P_k^{\frac{1}{2}})_i, i = n_z + 1, 2, ..., 2\,n_z$
    $\hat{\mathcal{Z}}_{k+1}^{(i)} = f_{k-1}(\mathcal{Z}_k, T), i = 1, 2, ..., 2n_z$
    $Q_k =$ computed as shown in A.3 and A.4
    $\hat{z}_{k+1} = \frac{1}{2n_z}\sum_{i=1}^{2n_z}\hat{\mathcal{Z}}_{k+1}^{(i)}$
    $P_{k+1} = Q_k + \frac{1}{2n_z}\sum_{i=1}^{2n_z}(\hat{\mathcal{Z}}_{k+1}^{(i)} - \hat{z}_{k+1})(\hat{\mathcal{Z}}_{k+1}^{(i)} - \hat{z}_{k+1})^T$
    $D_{k+1} = \frac{1}{2n_z}\sum_{i=1}^{2n_z}(\mathcal{Z}_k^{(i)} - z_k)(\hat{\mathcal{Z}}_{k+1}^{(i)} - \hat{z}_{k+1})^T$
    $G_k = D_{k+1}P_{k+1}^{-1}$
    $\hat{z}_{k|K} = \hat{z}_k + G_k(\hat{z}_{k+1|K} - \hat{z}_{k+1})$
    $P_{k|K} = P_k - G_k(P_{k+1} - P_{k+1|K})G_k^T$
  end
end

The RTS smoother is a well known smoother. No contributions were made to this during the thesis work, but the steps were adapted to include the CKF logic and were implemented as explained in Algorithm 6. There is another smoother that can be employed, which lies within the scope of this thesis. That is the IMM smoother, which is explained in detail in the next sections.

5.2 Interactive Multiple Model Smoother

Since an IMM filter was used, there is also the possibility to use an IMM based smoother, as suggested in [12]. This is only applicable when the forward filter was based on an IMM filter. This is because the knowledge about the forward estimates of how much weight each model got needs to be known for the smoothing. If there is more weight in one state during the filtering, the smoother would of course use that information to know which has better estimates and form the smoothing weights similar to this way based on those estimates. Thus, the entire posteriors of the 10×1 vector, the μ_k^j mode probabilities and forward transition probability matrix π_{ji} are required for the IMM smoother.

The Section 5.2.1 discusses the lagless assumption that is made so that there is no lag in the smoothed estimates, and then in Section 5.2.2, the steps involved in the IMM smoother are explained step by step.

5.2.1 Lagless Smoother Assumption

The IMM smoother that has been referred to addresses a case when we assume a lag L in the smoother such that $k - L \leq t \leq k$, where k represents the instances where the trajectory is smoothed and t is the instant where the filter estimates the trajectory. For the algorithm presented, the lag is assumed to be zero, and thus only instances when the forward filtering values are available are smoothed. Also, the references are still made in the forward sense, such that $k + 1 > k$ with respect to time, as in the filter. Thus, as the algorithm for the smoother presented progresses, the instances of time reduce from $k + 1$ to k. In order to distinguish the forward filter estimates from the backward smoothed estimates at the same instant, a change in subscript is used, where instances denoted using t represent the filter and k represents the smoother. The two time notations however denote the same time instant, i.e. $k = t$, and only differ with respect to the quantity they represent being from the filter or smoother.

5.2.2 Algorithm

The algorithm for the IMM smoother is described in this Section. The inputs as explained earlier are the state estimates $\hat{z}^j_{k|k}$, posterior estimates $P^j_{k|k}$, mode probabilities μ^j_k and transition probability matrix π_{ji} where $i, j = 1, 2, ..., N_r$. N_r, as before is the total number of state vectors, which is $N_r = 2$. Using these definitions, it is possible to employ the lagless IMM smoother.

1. Using the forward TPM π_{ji} and the forward mode probabilities μ^i_k, the backward TPM $\{b_{ij}\}^{N_r}_{i,j=1}$ is calculated as

$$b_{ij} = \frac{\pi_{ji}\mu^j_k}{\sum^{N_r}_{l=1} \pi_{li}\mu^l_k}. \tag{5.1}$$

2. The backward mixing probability $\{\nu^{ij}_{k+1|K}\}^{N_r}_{i,j=1}$ is calculated using the backward TPM and the backward mode probability from the next instant(in terms of causal consideration of time) or the previous iteration (in terms of the smoother)

$$\nu^{ij}_{k+1|K} = \frac{b_{ij}\nu^i_{k+1|K}}{\sum^{N_r}_{l=1} b_{lj}\nu^l_{k+1|K}}. \tag{5.2}$$

3. This step is similar to the forward filtering case, and the mixed smoothed estimates are calculated as

$$\hat{z}^{0j}_{k+1|K} = \sum_{i=1}^{N_r} \nu^{ij}_{k+1|K}\hat{z}^i_{k+1|K}, \tag{5.3}$$

$$P^{0j}_{k+1|K} = \sum_{i=1}^{N_r} \nu^{ij}_{k+1|K}\{P^i_{k+1|K} + [\hat{z}^i_{k+1|K} - \hat{z}^{0j}_{k+1|K}][\hat{z}^i_{k+1|K} - \hat{z}^{0j}_{k+1|K}]^T\}. \tag{5.4}$$

4. The estimates from the filtered output $\{\hat{z}^j_{k|k}\}^{N_r}_{j=1}$ from Equation 3.4 are used to calculate the mode matched smoothed estimates for both the state vectors.

A parameter called *Smoothing gain* is calculated using the posterior state covariance values from the forward filter as

$$A_k^j = D_{k+1}(P_{k+1|k}^j)^{-1}. \tag{5.5}$$

Using this parameter, the smoothed estimates are calculated as:

$$\hat{z}_{k|K}^j = \hat{z}_{k|k}^j + A_k^j(\hat{z}_{k+1|K}^j - \hat{z}_{k+1|k}^j), \tag{5.6}$$

$$P_{k|K}^j = P_t^j - A_k^j(P_{k+1|K}^{0j} - P_{k+1|k}^j)(A_k^j)^T. \tag{5.7}$$

The matrices D_{k+1}, $P_{k+1|k}^j$ and the state vector $\hat{z}_{k+1|k}^j$ are computed in the same manor as shown in lines 4 to 10 in the RTS Algorithm 6.

5. As in the forward filter case, the mode probabilities $\{\nu_{k|K}^j\}_{j=1}^{N_r}$ are calculated for the smoothing as well using:

$$\nu_{k|K}^j = \frac{\Lambda_k^j \mu_{k|k}^j}{\sum_{i=1}^{N_r} \Lambda_k^i \mu_{k|k}^i}. \tag{5.8}$$

Here, the parameter $\{\Lambda_k^j\}_{j=1}^{N_r}$ involves the usage of the forward TPM π_{ji},

$$\Lambda_k^j = \sum_{i=1}^{N_r} \pi_{ji} \mathcal{N}(\hat{z}_{k+1|K}^i; \hat{z}_{k+1|k}^j, P_{k+1|k}^j). \tag{5.9}$$

6. The final estimates are calculated using the weighted sum of the separate state vector estimates

$$\hat{z}_{k|K} = \sum_{j=1}^{N_r} \nu_{k|K}^j \hat{z}_{k|K}^j, \tag{5.10}$$

$$P_{k|K} = \sum_{j=1}^{N_r} \nu_{k|K}^j \{P_{k|K}^j + [\hat{z}_{k|K}^j - \hat{z}_{k|K}][\hat{z}_{k|K}^j - \hat{z}_{k|K}]^T\}. \tag{5.11}$$

6

Results and Discussion

In this chapter results from the different test scenarios will be presented. The implemented algorithms were tested on data collected from multiple scenarios. For analyses purposes, the tracking estimates from the algorithms will be compared to two different tracking systems. Where available, the output of the algorithms will be validated against data from the RT-range system explained in Section 6.1 and for scenarios where data from the RT-range system is not available results will be compared to the output from the Fusion system explained in Section 6.2. The results from the tracking when there are no guardrails is shown in Section 6.4 and the tracking for a scenario with guardrails is shown in Section 6.5.

Also to be noted is the fact that the validation data (both the GPS and fusion data) is available for the target while the GPS and fusion estimates track the vehicle on the rear centre. Though the target frame is defined at the centre of the vehicle, the tracking is still done at this point (rear centre) in order to visualise the results in terms of errors at the same point on the target, thereby keeping the accuracy of the validation systems intact.

The error comparisons that have been provided are the absolute errors between the estimates and the assumed ground truth. These errors have been provided for the estimates of the lateral and longitudinal positions, along with the heading, velocities and total positioning.

6.1 RT-range system

The RT-range system is a bolt-on system to the RT inertial and GNSS-aided navigation systems. It can accurately measure the relative position, velocity and heading between multiple vehicles. The reported accuracy given in the system specifications is given in Table 6.1. It shall be noted here that on top of this there is some added inaccuracy between the comparisons due to limitation on time matching. This mismatching in timing is due to data limitations and is estimated to be from $0ms$ to $60ms$. Because the RT-range system reports the relative states of the host and the target the additive error that is the result of the time mismatching is highly dependent on the relative velocity and acceleration between the host and target vehicles.

Table 6.1: Sensor system specifications as provided by the manufacturer (OXTS) home page.

Field	Accuracy
Longitudinal Range	0.03 $[m]$
Lateral Range	0.03 $[m]$
Longitudinal Velocity	0.02 $[\frac{m}{s}]$
Lateral Velocity	0.02 $[\frac{m}{s}]$
Heading	0.1 $[deg]$

6.2 Fusion system

The sensor system that is here referred to as the fusion system is a sensor system that is being developed by Delphi. This system is mounted on the host vehicle in all test cases. The fusion system main sensors are the mid-range radar and a forward looking camera. As mentioned in the introduction, the radar is good at measuring range and range rate but lacks the ability to measure azimuth angle accurately, the camera sensor on the other hand is poor at measuring range and range rate but excellent at measuring azimuth angle. By fusing these two sensors the fusion system is able to robustly report the position and velocity of target vehicles in front of the host as well as being less vulnerable to radar clutter. This means that the fusion system is capable of reporting the targets position and velocity accurately even in difficult scenarios where there are multiple objects in the FOV as long as it has both radar and vision "lock" on the target.

6.3 Naming conventions for plots

For validating all the filters considered, comparisons between different filters are made for the two overall scenarios, which are the cases with and without the guardrails. For validating the results for the scenario without the guardrail, explained in Section 6.4, the comparisons are made between the *Reflector model*, *RMA model* and *Reflector model with birthing*. The *Reflector model* is the standalone radar resolution model explained in Section 2.2, the *RMA model* is the spatial model (ellipse) filtered using the Random Matrix Approach (RMA) explained in Section 4.2.3 and the *Reflector model with birthing* is the BGSF based switching model explained in Section 4.2. For the scenario with the presence of guardrails, explained in Section 6.5, the comparisons are made between the standalone radar resolution model or reflector model called *Reflector model Raw* and the one that works with the IMM called *Reflector model with IMM*.

6.4 Scenarios without guardrails

This section deals with validating the chosen extended target tracking method, which is the *Reflector model*. This is done for three scenarios: (1) when the host is following

the vehicle in an almost straight trajectory, (2) when the target overtakes the vehicle and (3) when the target turns and side reflectors are visible to the host. Graphs are provided of the entire filtered trajectory along with the comparisons of errors of the positions, velocities and heading estimates with the available "ground truth" for that setup.

6.4.1 Following

A scenario where the host vehicle follows the target is presented here. The reason this is interesting is because of the fact that the comparison is now made between two models (*Reflector model* and *RMA model*) which assume the target to be an extended target, with the assumption on one of them (the *Reflector model*) that it is composed of reflectors, and the other (*RMA model*) that it is an elliptical object. This means that the *Reflector model* filter, while following, only sees the back reflectors which means that there are very few/almost no measurements from the sides of the vehicle. The very reason for implementing it was to be able to use these detections from the sides to get better heading estimates. Therefore, while only seeing the rear of the target, the *Reflector model* can possibly get detections from the three reflectors on the targets rear, thereby restricting it to work as a simple extended target tracking method (like the *Spatial model*), with just a spatial $2 - dimensional$ distribution of measurements. This type of filter is exactly what is used for comparison. In other words, the *Reflector model* and the *RMA model* converge for the case when the host follows the target and gets detections that are only associated with the rear reflectors. The map of the entire trajectory is shown in Figure 6.1.

As seen from the Figure 6.1, the data used for comparison is GPS data of the target. There should be attention paid to the axes of the trajectory plots, since visually, it might seem like the filter estimates are wrong even though the errors range from anywhere between 0 *to* $1m$. This is due to the axes being illustrated such that the $y - axis$ is over $400m$ while the $x - axis$ is around $3m$.

It must be noted that the case presented here is not absolute head-on following, which means that the relative angle between the sensor heading and the yaw angle of the target is not 0, i.e., the target moves in such a way that it has a lateral displacement from the mean 0 angle view of the sensor in its FOV. However, it is very close to a real world following scenario. This means that there is a possibility of seeing more of some reflectors on the rear than another. To that extent, the *Reflector model* must perform better than the *RMA model* while filtering due to the fact that the discrimination between reflectors still helps the cause of better tracking. This is only possible when the strengths of the reflecting surfaces are defined with a ratio that explains their behaviour better, and that the expected return signal amplitude is a smooth function instead of the square signal that is used here. The reasoning for this is simple: if the angular dependencies of the radar detections on the vehicle were defined better, then the reflectors would get a better description of their return signal amplitude. This would then allow for one reflector to be more probable of giving a detection than the other, resulting in better tracking.

Figures 6.2a and 6.2b show the error comparison of the lateral and longitudinal

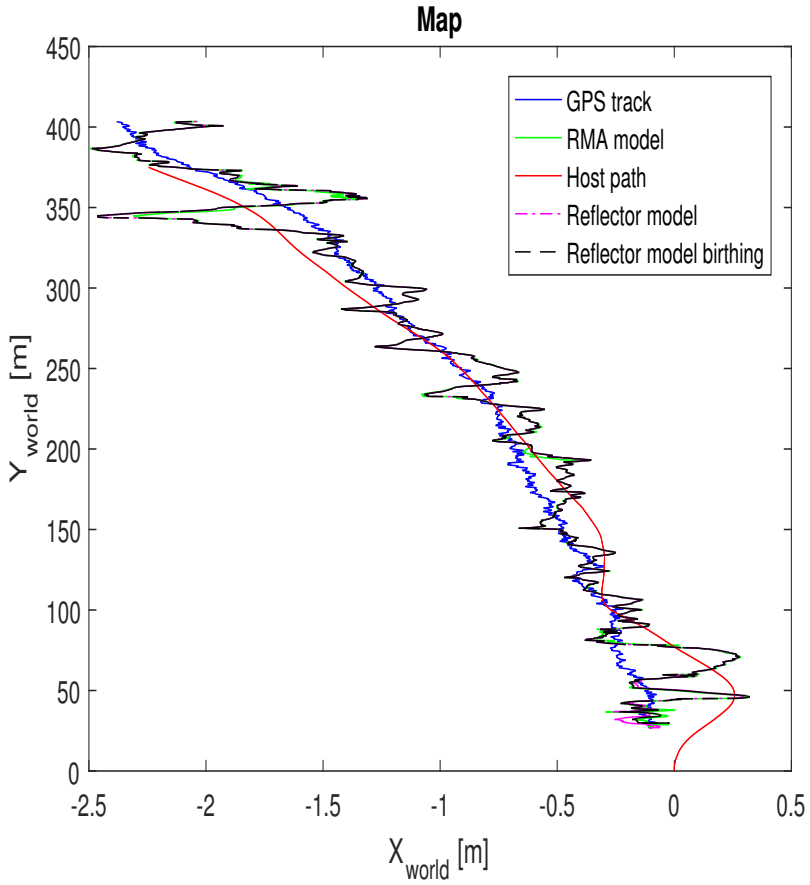
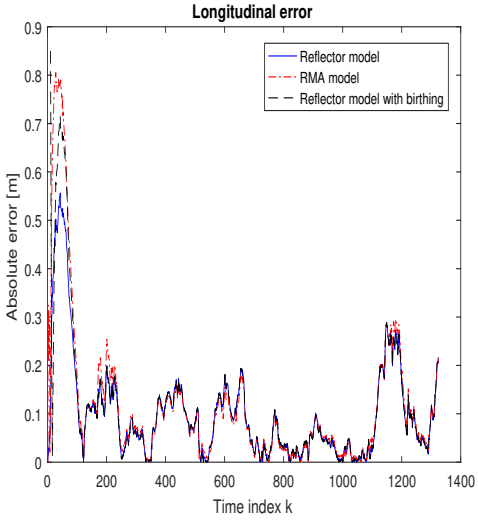


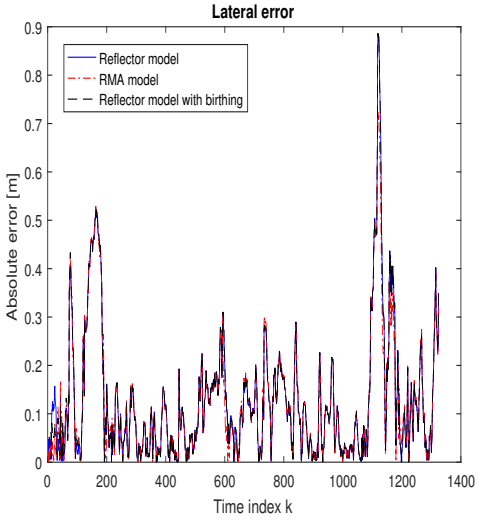
Figure 6.1: Map of the entire filtered trajectory of the host and the target while being tracked with different filters for the following scenario. The host sees only the rear of the target.

positions of the target compared in the world frame when using the *Reflector model* and the *RMA model*. It is quite clear from the Figures that the performance of both the filters in this scenario is more or less similar, as expected. The *RMA CKF* and *Reflector model* are given the same prior. Ignoring the first few indices, the rest of the performance is similar. The reason for this initial irregularity in the *RMA model* is due to the fact that the filter gains have not converged to the optimal value of estimating the object to be localised to the rear of the vehicle, as shown in Figure 4.3b. The *Reflector model*, on the other hand, already has this information fed to it at the prior due the predefined reflector positions. This is more evident in the longitudinal comparison because of the offset that the *RMA model* would have until it converges to the rear, while simultaneously mapping the posteriors to the middle of the vehicle. This offset is defined similarly for the two models, which is the reason why they converge. But this is also a reason for why the initial performance of the *RMA model* is worse.

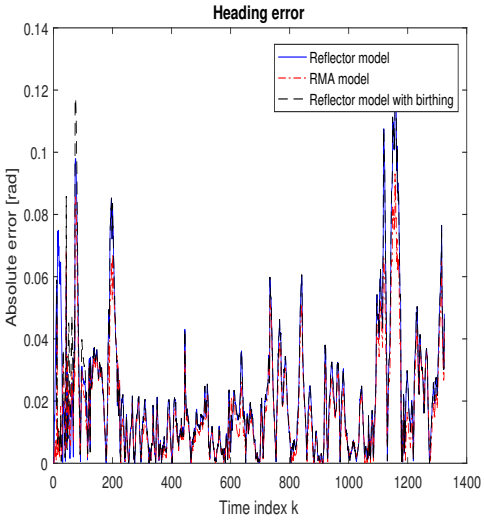
The other interesting aspect to the comparisons is the *Reflector model with birthing*, which starts a few indices later since the start of the tracking due to the switch from the *RMA model* to the *Reflector model*. Thus, the plot only shows the trajectory errors after the switch has taken place. During this scenario, the target



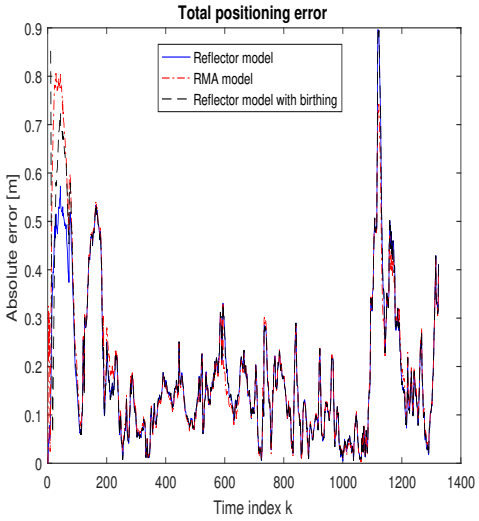
(a) Longitudinal error. Enlarged Figure A.1 can be found in appendix



(b) Lateral error. Enlarged Figure A.2 can be found in appendix



(c) Heading error. Enlarged Figure A.3 can be found in appendix



(d) Total positioning error. Enlarged Figure A.4 can be found in appendix

Figure 6.2: Results from the following scenario. Error comparisons of filtered estimates of the target. All figures present the absolute error in the world frame.

is never lost and is therefore, not killed. The behaviour is also almost exact to the other two filters which are fed the prior. This convergence is proof that the switching has taken place correctly and in a desirable manner due to the fact that eventually, feeding a predetermined prior and determining it online using a birthing function, amounts to the same output. The switching also happens within a few indices, and to be specific, the instant of switching is at the 24^{th} index. This means that the measurements from the radar come from the target from almost the first instance (it is known that the vehicle already exists in the FOV of the radar while following),

since the detections are not wrongly associated and the Bernoulli filter is quick to suggest that the existence probability has reached a value greater than 0.99.

From Figures 6.2c and 6.2d, which show the errors in heading and the overall positioning, a similar behaviour is noticed. The maximum error throughout the entire run of the scenario is in the range of 0.13 $rad \approx 7.45°$. Considering the fact that the GPS data was used for validating the performance, the uncertainty in the GPS data also contributes to some estimates being wrongly judged erroneous since the absolute ground truth is not known. With that in mind, it is fair enough to conclude that the heading estimates are reasonable, also because in the scenario where the host follows the target, the heading estimates are only based on the movement of the target since only the rear is seen.

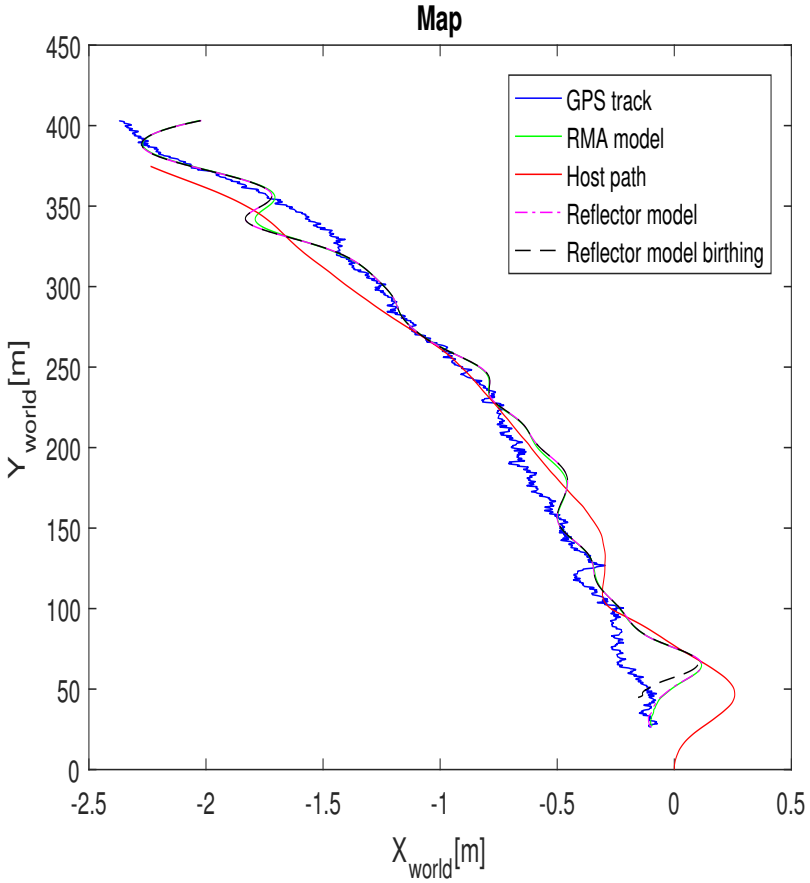


Figure 6.3: Map of the entire smoothed trajectories of the host and filtered trajectories shown in Figure 6.1 of the target while being tracked with different filters for the following scenario. The host sees only rear of the target.

The errors from the trajectory are reduced even further by smoothing them out using an *RTS* smoother. A scenario where a guardrail was not present, does not need the use of an IMM filter working alongside the *Reflector model* since no updates are made for the heading of the target. Thus, an IMM filter would give the same result since both the states would hold the same value with the weights summing up to 1. And since the IMM filter is thus not relevant, an IMM smoother is not

applied to the filtered estimates since it would work the same as the *RTS smoother*. The smoothed trajectories of all the filters are plotted in Figure 6.3. Comparing the Figure 6.4 which shows the errors in longitudinal, lateral, heading and overall positions of the smoothed estimates to the Figure 6.2, it is seen that the errors are smaller due to the reduced uncertainty after smoothing. It should be noted that the posterior estimates from both the *Reflector model* and *RMA model* were smoothed using the RTS. These estimates are, as before, the position estimates of the back of the centre of the target and thus, no knowledge of any extended target was used for smoothing the trajectory.

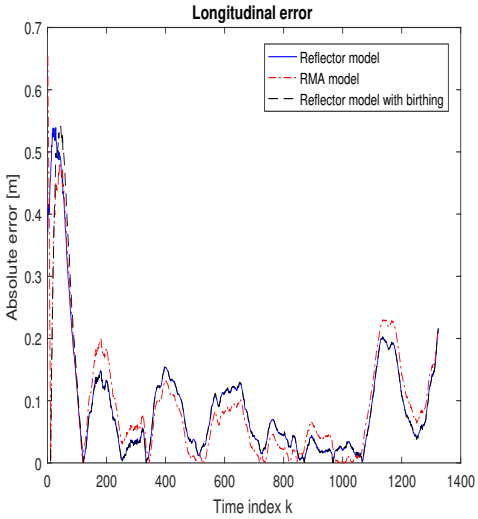
From Figure 6.4a, it is seen that the difference in errors in longitudinal estimates, if any, are in the range of ~5 cm, but the average performance is almost the same. These differences have their source most probably from either of these two options: from the uncertainties in the GPS data or the centre of the vehicle being mapped with some offset which emerges from the fact that there is a mismatch in time samples between the smoothed data and the GPS data. Thus, since the difference in errors is low and the reason for these differences is tough to comment on, the longitudinal error plot is assumed to give equal performance for both the filters. The lateral positioning errors are more or less similar for both filters, and the maximum error is around 35 cm given the inaccuracy of the validation data, which is quite reasonable.

Figure 6.4c shows the smoothed heading estimate errors of both the filters, and from this, it is seen that the maximum error is around 0.045 $rad \approx 2.6°$ and for most of the indices, the error is less than 0.02 $rad \approx 1.15°$. This is a reasonable result, and considering the total positioning error shown in Figure 6.4d, where the maximum error is 0.5 m, it can be concluded that the smoothed estimates satisfactorily improve the estimates of the target states from the forward filtering.

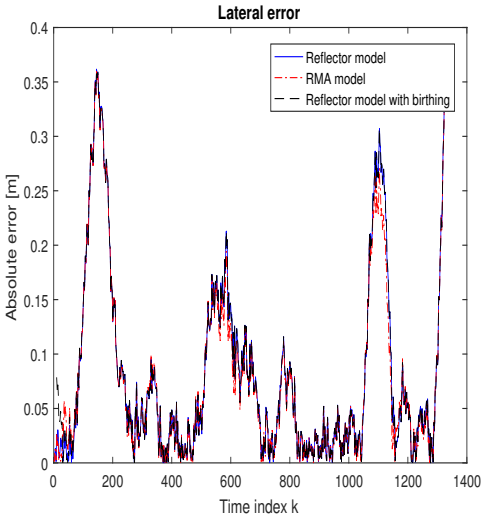
A comparison of the errors in the target velocity estimates can be seen in Figure 6.5. It is estimated that the high noise factor in these velocity plots is a result of the time mismatch discussed in the beginning of this chapter. In the start of the scenario (from $k = 1$ to $k \approx 180$), the smoothing appears to have a negative impact on the velocity estimate since the error increases from about 0.8 m/s *to* 1 m/s. This is estimated to be caused by two main factors, one at the start of the test the posterior covariance in the forward filtering has not converged to its true value. The second factor is that during the smoothing any error in the position estimate will affect the smoothed velocity output and as seen in Figure 6.2d, there is a relatively high error in the total positioning estimates. For the rest of this scenario (from time $k \approx 180$ to the end of the test) the high noise factor in the validation data makes any comparison difficult.

The similarity in performances between the two extended target tracking methods and models was further confirmed by running several different tests, and observing the results. The results of the forward filtered estimates for the following scenario for the tests are summarised in Table 6.2. It is seen that even though there is a marginally better performance in the *Reflector model*, it is fair to conclude that the numbers suggested by these two models are very close.

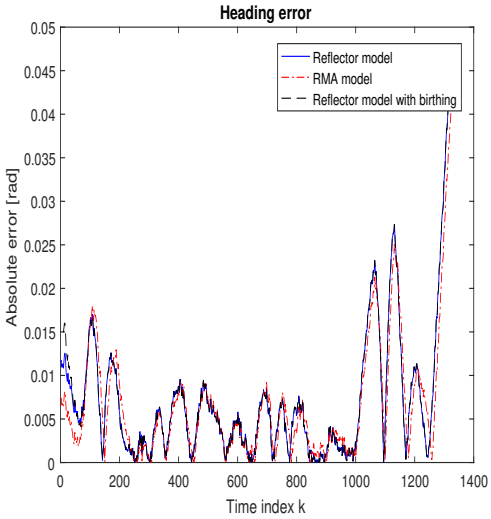
The similarities were even more obvious once the smoothing was performed. The results from the smoothing are presented in Table 6.3.



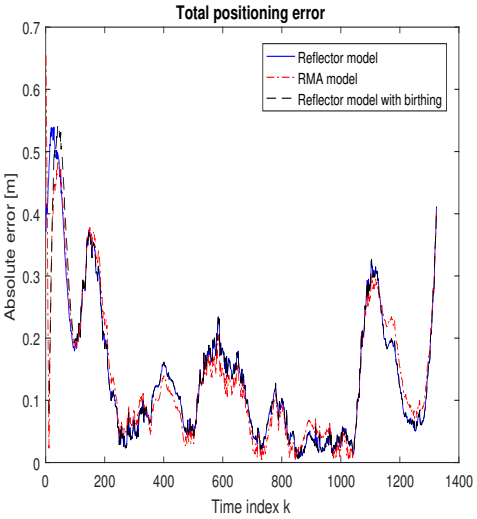
(a) Longitudinal error, after smoothing. Enlarged Figure A.5 can be found in appendix



(b) Lateral error, after smoothing. Enlarged Figure A.6 can be found in appendix

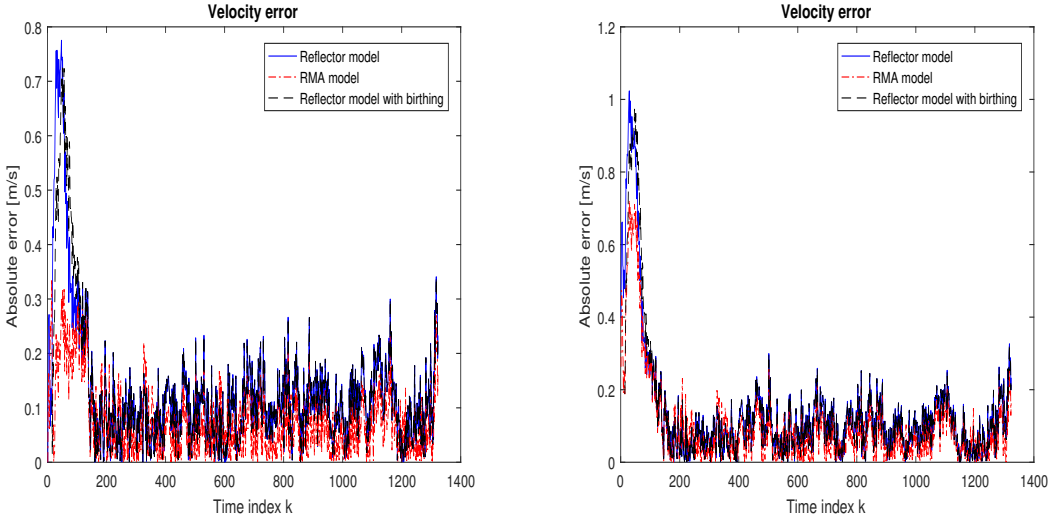


(c) Heading error, after smoothing. Enlarged Figure A.7 can be found in appendix



(d) Total positioning error, after smoothing. Enlarged Figure A.8 can be found in appendix

Figure 6.4: Results from the following scenario. Error comparisons after smoothing of filtered values shown in Figure 6.2. All figures present the absolute error in the world frame.



(a) Velocity error. Enlarged Figure A.9 can be found in appendix

(b) Velocity error, after smoothing. Enlarged Figure A.10 can be found in appendix

Figure 6.5: Error comparisons of filtered and smoothed estimates of velocity for the following scenario. The velocity comparison is of the absolute velocity in the world frame.

Model	Longitudinal	Lateral	Total	Heading	Total within $1m$
Reflector model	0.165[m]	0.186[m]	0.279[m]	0.0168[rad]	98.4%
RMA model	0.205[m]	0.184[m]	0.311[m]	0.0142[rad]	96.9%

Table 6.2: (Following) The results presented in this table are from 26 independent tests spanning 29000 time samples after the forward filtering. The range from host to target is 20 m to 110 m, the date is the average offset from the ground truth.

Model	Longitudinal	Lateral	Total	Heading	Total within $1m$
Reflector model	0.15[m]	0.142[m]	0.231[m]	0.005[rad]	99%
RMA model	0.152[m]	0.142[m]	0.232[m]	0.005[rad]	98.9%

Table 6.3: (Following) The results presented in this table are from 26 independent tests spanning 29000 time samples after smoothing. The range from host to target is 20 m to 110 m, the date is the average offset from the ground truth.

6.4.2 Turning

The target is modelled as being composed of 11 reflectors in the *Radar Resolution model*, and until this point, from the following scenario, only the three rear ones have been visible, and hence, the next investigation needs to be made in order to look at the other reflectors and look at how well the algorithm performs to include them. The perfect setting for that would be to steer the target to a sharp right or left turn while in the FOV of the sensor, and try to map those detections that emerge to the sides of the vehicle and observe the behaviour of the proposed solutions. the results presented here are from a scenario when the target took a sharp right turn, and the trajectories are plotted for the filters and the GPS data in Figure 6.6.

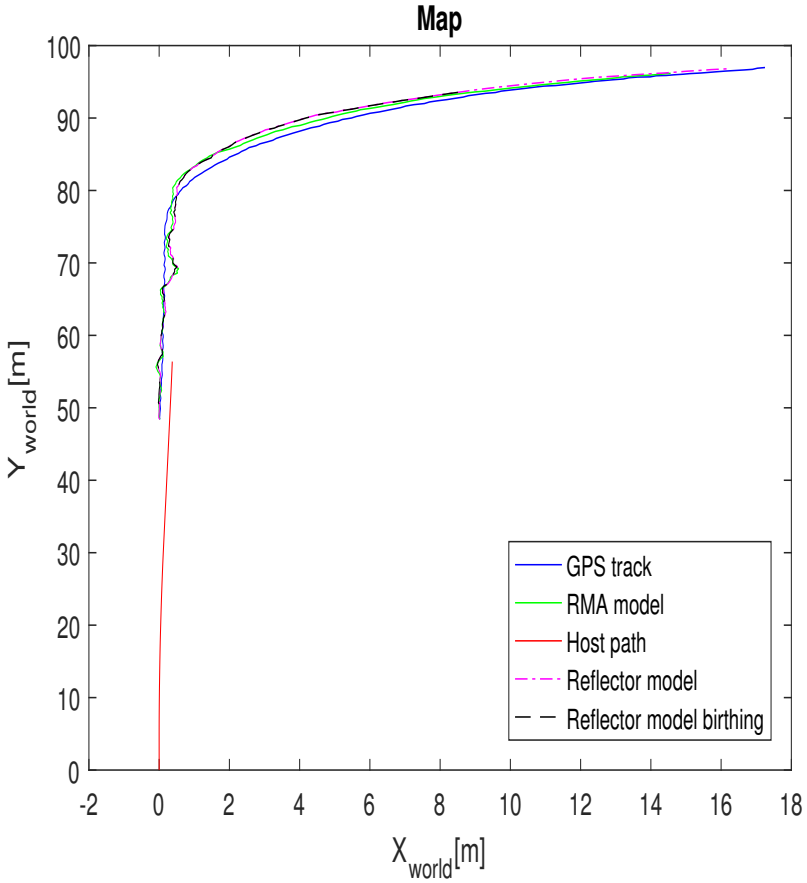
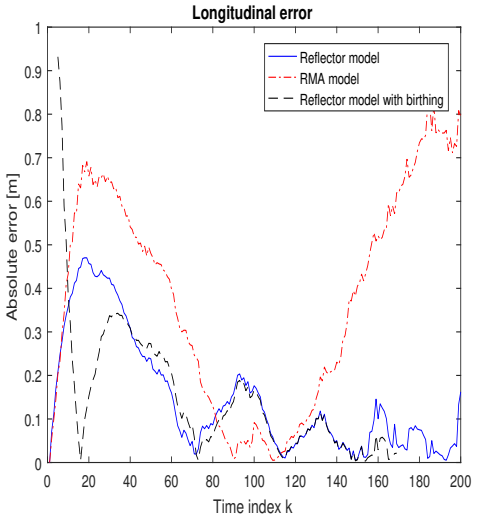


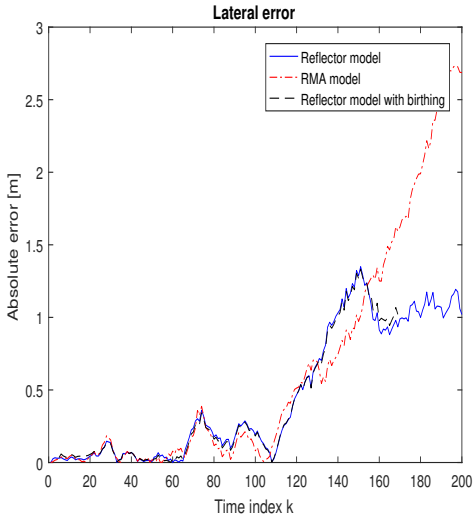
Figure 6.6: Map of the entire trajectory of host and the target while being tracked with different filters for the turning scenario. The target takes a sharp right turn in the FOV of the host.

The host tracks the target initially at a range of about 50 m, after which both the host and target travel at about the same speed for about 30 m at which point the target takes the sharp right turn. The target then exposes its side reflectors to the radar thereby allowing for the *Reflector model* to be able to map these reflectors. The error analyses for this scenario are presented in Figure 6.7. What is also interesting in this scenario is the *Reflector model with birthing* created using the birthing function, gets killed through the process. This allows investigations into the drawbacks of

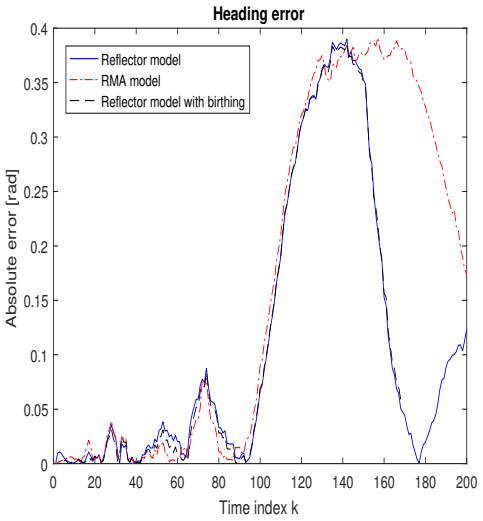
using a birthing function based a *RMA model* since the target still exists in the FOV but gets killed.



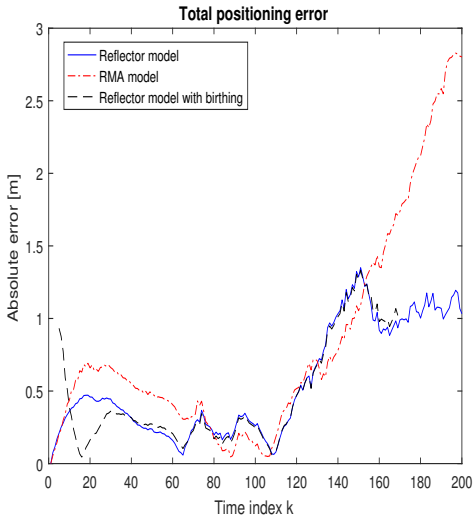
(a) Longitudinal error. Enlarged Figure A.11 can be found in appendix



(b) Lateral error. Enlarged Figure A.12 can be found in appendix



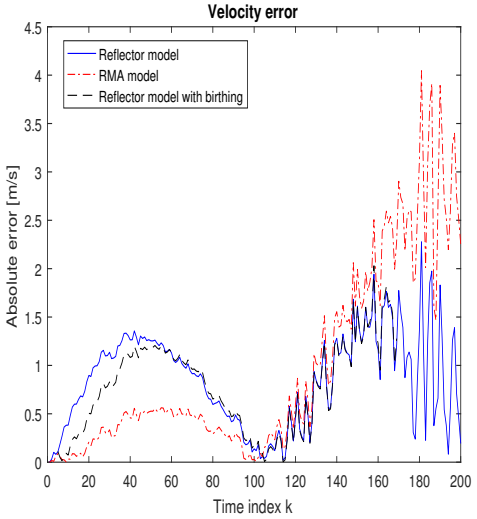
(c) Heading error. Enlarged Figure A.13 can be found in appendix



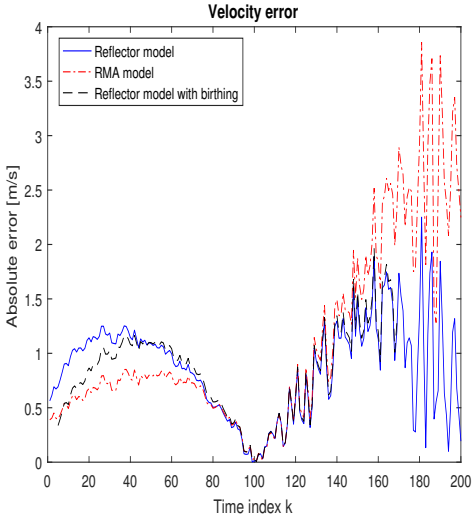
(d) Total positioning error. Enlarged Figure A.14 can be found in appendix

Figure 6.7: Results from the turning scenario. Error comparisons of filtered estimates of the target. All figures present the absolute error in the world frame.

The initial performance of the *RMA model* and *Reflector model* in the longitudinal direction is similar to that observed in the following case, as it should be since the target is being followed. The *Reflector model* has a better performance in this case, with an error of about 0.4 m while the *RMA model* has an error of about 0.7 m. The lateral errors are very small and comparable between the two models for the first few instances. The longitudinal errors, as explained earlier, basically



(a) Velocity error after the forward filtering. Enlarged Figure A.15 can be found in appendix



(b) Velocity error after smoothing. Enlarged Figure A.16 can be found in appendix

Figure 6.8: Error comparisons of filtered and smoothed estimates of velocity of the target for the turning scenario. The velocity comparison is of the absolute velocity in the world frame.

stem from the fact that the optimal values have not been reached yet, and the filter has not converged. The errors pop up in both the models, and is quickly compensated for by the *Reflector model* but takes a few more indices for the *RMA model* to converge. This fixing causes the velocity estimates to go higher, as shown in Figure 6.8. The velocity comparison is of the absolute velocity in the world frame. It is estimated that the high noise factor in these velocity plots is a result of the time mismatch discussed in the beginning of this chapter. This Figure shows that the error is much more in the *Reflector model* due to the *"fixing"* of the longitudinal estimates, whereas the velocity estimates for the *RMA model* is much better. With the *Reflector model with birthing*, since the absolute error is plotted (meaning all errors are in the positive scale), this is due to that the algorithm estimates the positions to be behind the actual position and then moves it forward, crossing the zero-mark and going to the other side, and becoming almost equal in performance to the *Reflector model* given the prior, as expected. This explains the behaviour of this curve going all the way down to 0 before converging to the *Reflector model*.

The initial performances of the filters are sensible in the heading angle, due to the simple case of handling a following scenario as shown in the previous example. The heading estimates, however, become highly erroneous as time progresses, pushing the error up to 0.4 $rad \approx 23°$. The vehicle starts turning at about $k = 100$, which is when the errors start to spike up in the heading estimates. The *Reflector model* also increases in error but is very quick to adapt to the motion of the target as soon as the correction begins due to the immediate mapping of side reflectors. This gets the error down to a low value at about $k = 180$, giving an interval of 80

instances when it had a high error, but in real-time this is about 2.4 s considering the sampling time of the radar. The *RMA model* on the other hand, takes much longer to start correcting for the wrong heading estimates and this is because it does not utilise the prior knowledge about the size and dimensions of the target.

The other interesting thing to be noted about these error plots is the impact that the wrong heading estimates have on the lateral and longitudinal estimates of the target, as well as the velocity estimates in the filters. The *RMA model* has the errors propagated to these estimates too whereas the *Reflector model* does not get influenced by the heading error estimates in the longitudinal direction, but only in the lateral direction. This also, is because of the fact that the *Reflector model* knows that the vehicle has turned and therefore, the wrong estimates can occur in the lateral sense due to the motion suggesting that and not in the longitudinal sense. Due to these reasons, the *RMA model* has very poor performance when it comes to the turning scenario, thereby validating the reason for having chosen the *Reflector model* as an extended target tracking method.

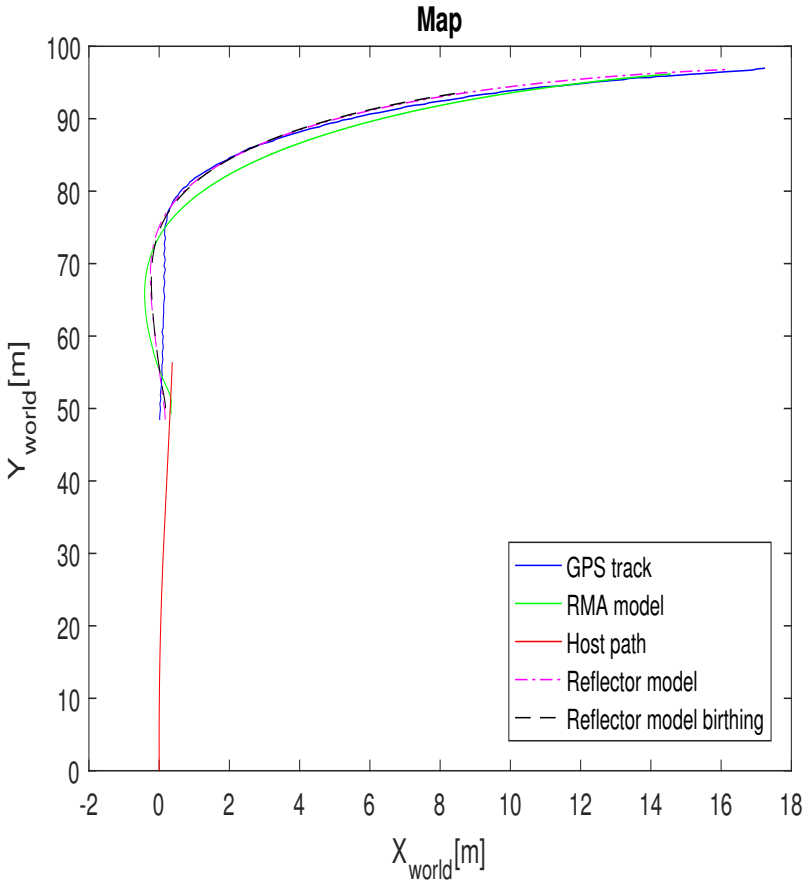
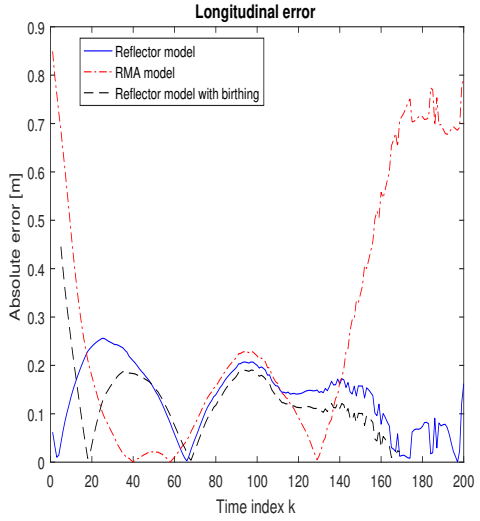
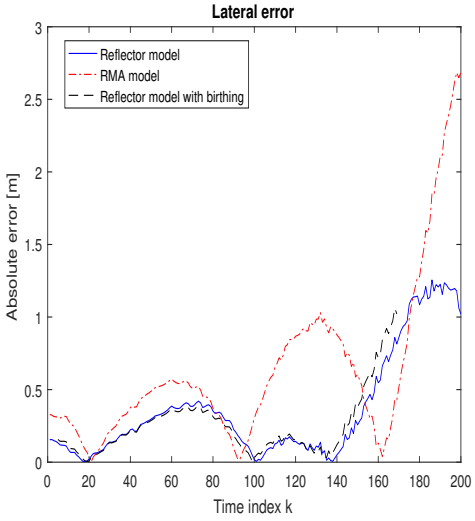


Figure 6.9: Map of the entire smoothed trajectories of the host and filtered trajectories of the target shown in Figure 6.6 for the turning scenario. The target takes a sharp right turn in the FOV of the host.

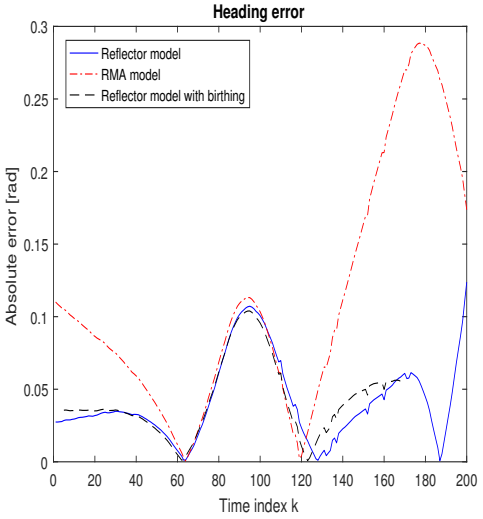
A close look at the *Reflector model with birthing* reveals that the target gets killed during the algorithm run, and is not created again. This is a disadvantage



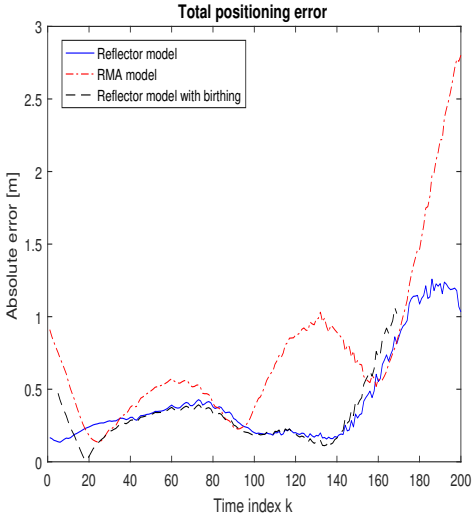
(a) Longitudinal error, after smoothing. Enlarged Figure A.17 can be found in appendix



(b) Lateral error, after smoothing. Enlarged Figure A.18 can be found in appendix



(c) Heading error, after smoothing. Enlarged Figure A.19 can be found in appendix



(d) Total positioning error, after smoothing. Enlarged Figure A.20 can be found in appendix

Figure 6.10: Results from the turning scenario. Error comparisons of the target estimates, after smoothing of filtered values shown in Figure 6.7. All figures present the absolute error in the world frame.

noted from using the *RMA model* to control the birth and death, since the existence is based on the performance of the random matrix approach method on the spatial model (*RMA model*). As discussed, the *RMA model* for this scenario performs bad, and this is evident from the error comparisons in Figure 6.7. The high errors cause the existence probability to go below 0.5 thereby killing the target, while it

is performing equally well as the *Reflector model*. This can be avoided by further work, which will be addressed in Section 7.

The Figure 6.9 shows the trajectories of the smoothed values of both the filters along with the *Reflector model with birthing*. The errors shown in Figure 6.10 are smoothed out considerably well, and it is then very obvious that the *Reflector model* outperforms the *RMA model*. Due to lack of validation data, tests for validating similar performances for all turning scenarios could not be performed.

6.4.3 Overtaking

To consider the scenario in which the target overtakes the host, the host vehicle was made to move at a low velocity for a few seconds before the target overtook it from another lane on its right side. Then the host was moving in a way so as to follow the target for some time before the target started to move towards the left and into another lane. This scenario presents good grounds of comparisons for two important cases: (a) the comparison between the *RMA model* and the *Reflector model*, and (b) the birthing algorithm implemented and its performance. Until this point, the results that were observed were only for the case when the target already existed in the FOV of the radar. But in this scenario, the whole target comes fully into view only after some indices after starting the filter. The birthing Gaussians, as explained in Section 4, are placed such that there is one available on the right side of the FOV of the radar. In this scenario the Gaussian that was placed on the right triggers the birth resulting in that the tracking gets initialised with the states of that Gaussian as a prior as explained in Section 4.2.4.3. The entire trajectories of the host and the target are shown in the Figure 6.11. The target estimates are obtained from the three available filtering algorithms: *Reflector model*, *RMA model* and *Reflector model with birthing*.

This scenario presents a minor issue with respect to the filters which do not use the birthing function, not being able to run unless they have a prior. This is not the case with the *Reflector model with birthing* since it can create the first instance of the object in order to do the tracking. If it was run, the starting point could be anywhere in the log, especially even before the vehicle has started existing in the FOV because the very reason of implementing the birthing function is to iteratively search for a valid object. But in the case of the other filters, a prior needs to be defined, and that prior should be within the scope of the radar measurements. If this does not hold, then the reflector mapping and the sensor model would suggest that no reflector can be seen, thereby causing the uncertainty to be high, pushing the state covariance to be singular. Thus, for this scenario, the plots and discussions start at the point where the target has just started existing in the FOV of the radar. The *Reflector model with birthing* still searches for the target, and thus the switch happens at some point after the scenario begins.

The reasons for the initial errors in the position estimates for the *RMA model* are that the measurements make the ellipse to quickly form the extended object in the region of the first incoming measurements, which is from the front of the vehicle. That means that, since we are tracking the car at its rear end (and that is the prior to the *RMA model*), and the first measurements are from the front, the

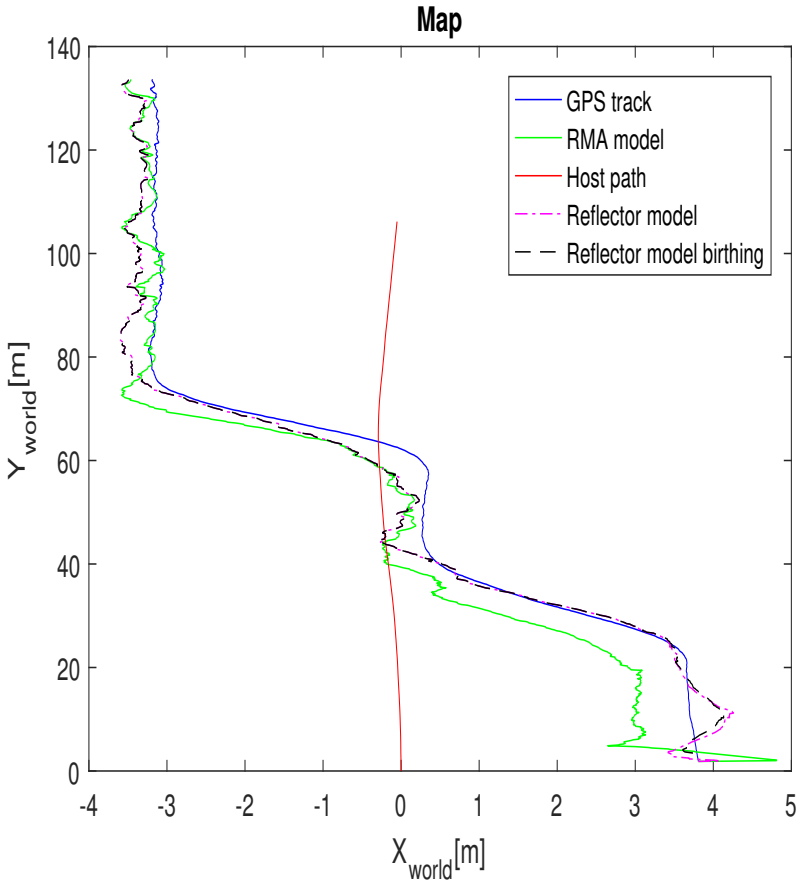
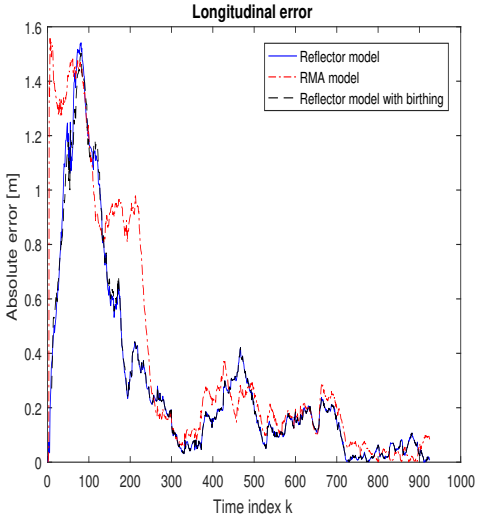


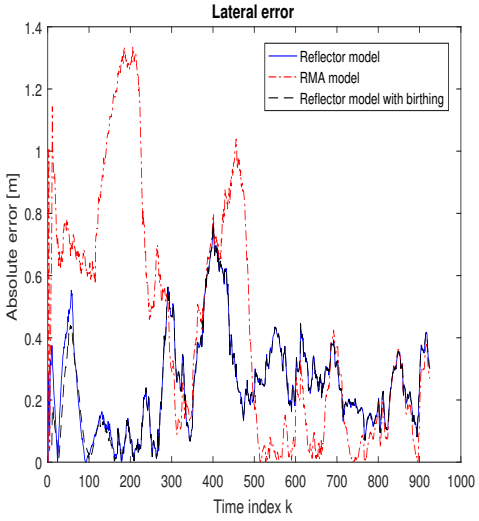
Figure 6.11: Map of the entire trajectory of host and the target while being tracked with different filters for the overtaking scenario. The target overtakes the host from the right.

errors suggest a longitudinal error somewhere between 0 and the length of the car, which is around 4.6 m, and lateral error of about half the width of the car (1.9 m) which is about 0.85 m. The longitudinal estimates improve until the detections come from all over the side of the vehicle that is seen, which then reduces the longitudinal error, and then eventually the entire vehicle comes in front of the target at instant of about $k = 180$ *to* 190. The subsequent error at that range of instances in the lateral estimates is due to the detections that are coming in are from all over the target, even from the rear due to its motion to the front of the host, making the prediction that the vehicle moves faster than it actually is. This is shown in the velocity errors displayed in Figure 6.13a. It is estimated that the high noise factor in these velocity plots is a result of the time mismatch discussed in the beginning of this chapter. The estimates go on to get better, and after that behave similar to a scenario where the *RMA model* tracks the rear.

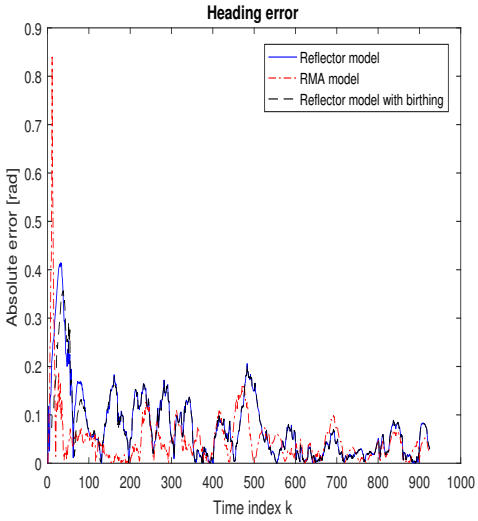
On the other hand, for the first few instances in the *Reflector model*, the prior knowledge is known that since the detections come from the side of the vehicle, these are mapped to the front left wheel, and other side reflectors, hence resolving them to the back of the vehicle gives us the right estimates. The peak in the longitudinal and lateral estimates originate from the wrong heading estimates at about $k = 50$.



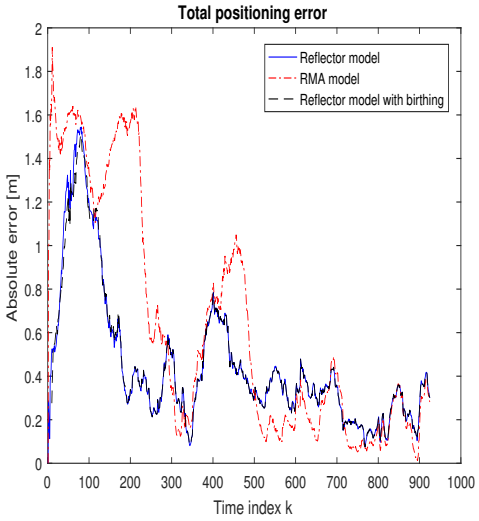
(a) Longitudinal error. Enlarged Figure A.21 can be found in appendix



(b) Lateral error. Enlarged Figure A.22 can be found in appendix



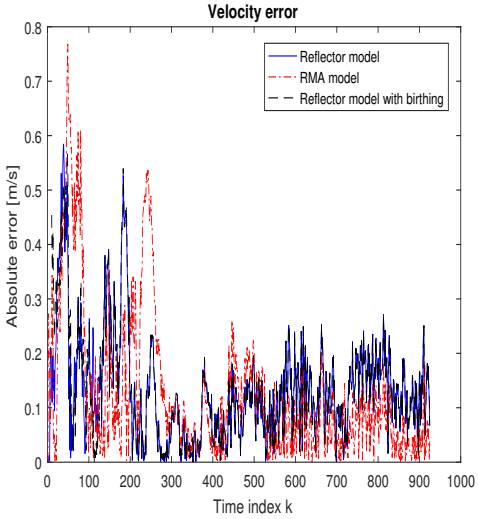
(c) Heading error. Enlarged Figure A.23 can be found in appendix



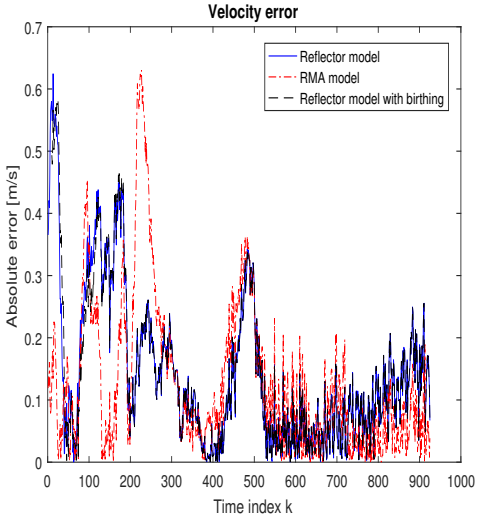
(d) Total positioning error. Enlarged Figure A.24 can be found in appendix

Figure 6.12: Results from the overtaking scenario. Error comparisons of filtered estimates of the target. All figures present the absolute error in the world frame.

This is due to the measurements coming at a steep angle to the radar, making the azimuth more erroneous than it normally is. As for the instant when the vehicle comes into view of the host, the jump in wrong lateral estimates is not noticed in the *Reflector model* due to the correct mapping of the reflectors, and this is similar in the longitudinal estimates too. The Figure 6.12 shows that most of the time in the initial steps (around 0 *to* 200 indices), the *Reflector model*, as an extended target tracking method, outperforms the simple *RMA model* in this scenario, except for that one peak where they perform equally bad. The impact of using a *RMA model*



(a) Velocity error after the forward filtering. Enlarged Figure A.25 can be found in appendix



(b) Velocity error after smoothing. Enlarged Figure A.26 can be found in appendix

Figure 6.13: Error comparisons of filtered and smoothed estimates of velocity for the overtaking scenario. The velocity comparison is of the absolute velocity the world frame.

is especially noticed in the lateral estimates as shown in Figure 6.12b.

The second high peak occurring in the lateral estimates at about $k = 450$ which happens in both the models, is due to the target moving towards the left and away from the line of sight to expose the other side with the other set of reflectors as compared to the initial side when the target came into view. Though this is seen in the longitudinal error as well, it is not as significant as in the lateral estimates, due to the fact that the movement is in the lateral direction. The *RMA model* takes a significant amount of time as compared to the *Reflector model* to adapt to these changes. These changes are also visible in the heading estimates and the velocity estimates while the lane change happens.

Once this is completed, the *RMA model* has better performance in the lateral estimates, whereas the behaviour is almost equal in the longitudinal direction. The reason why the *RMA model* has better performance in the lateral estimates after the lane change is because of the detections that come from all over the rear being resolved using an elliptically shaped distribution that quickly adapts to the lane change, whereas there is always a possibility for the *Reflector model* to wrongly judge the wrong reflectors to be seen more or less, or associate the wrong detections to the wrong reflector, causing a shift, which is what is seen here. This shift is in the range of 0.2 m *to* 0.4 m for about 200 indices from $k = 500$ *to* 700, after which the errors are comparable for both the models.

The *Reflector model with the birthing*, once after being birthed, behaves exactly like the *Reflector model* and this change happens rather soon due to the high number of incoming detections. That means that the switch happened at the right

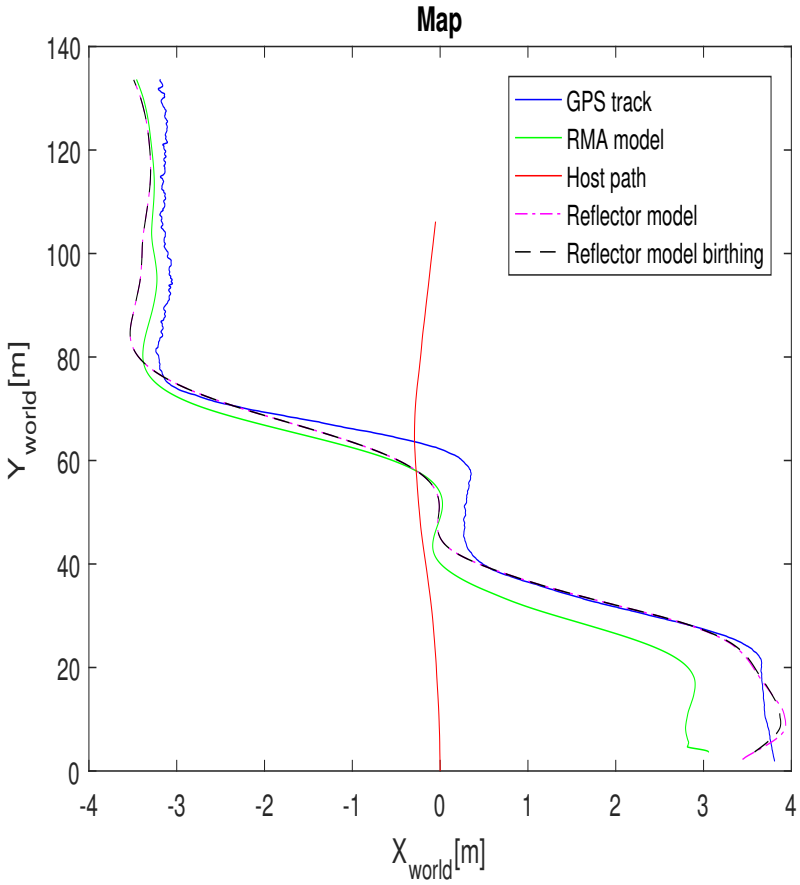
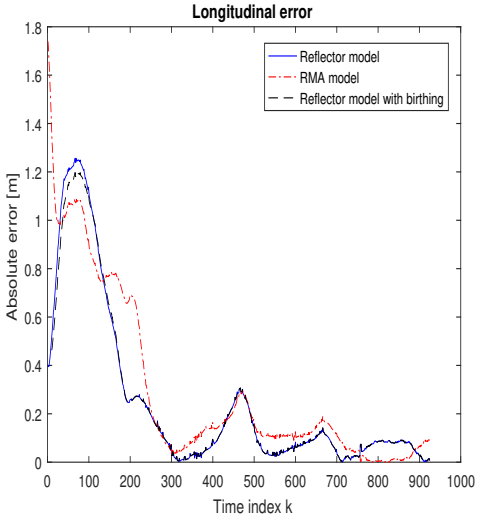


Figure 6.14: Map of the entire smoothed trajectories of the filtered trajectories shown in Figure 6.11 of host and the target while being tracked with different filters for the overtaking scenario. The target overtakes the host from the right.

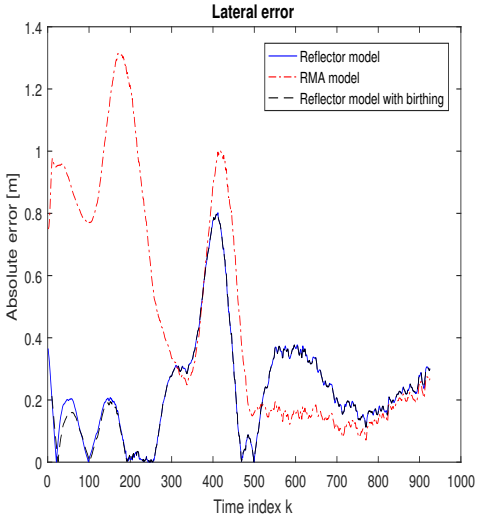
instant and mapping the right reflectors. This is possible due to the definition of the Gaussian IDs presented in Section 4.2.4.3, which allows the Gaussian R being birthed on the right to be immediately mapped in such a way during the switch that the Gaussian gets approximated quickly to the left front wheel instead of as an overall state switch to the middle of the back of the vehicle or anywhere else.

The smoothed trajectories are shown in Figure 6.14, and the corresponding errors in Figure 6.15. The *Reflector model*, for most of the scenario has an evidently better performance. This is true especially with the longitudinal and lateral estimates, but not so much with the heading estimates. The heading estimates are generally better for the *RMA model*, and this is again, because of the wrong resolution of the range rate measurements based on the azimuth measurements. The error goes high up to 0.12 $rad \approx 6.9°$, at the instants when the lane change happens. This gets reflected both in the longitudinal and lateral estimates of both the extended target models.

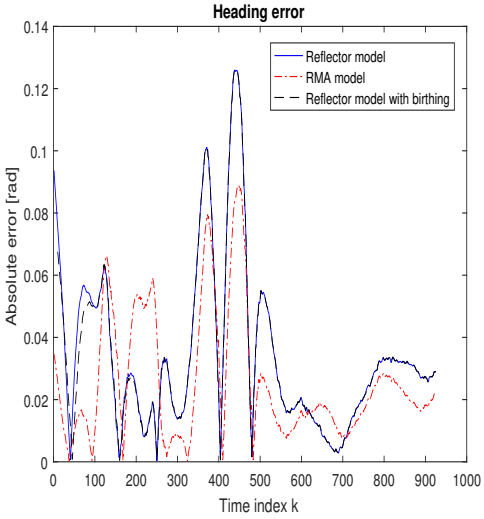
Several tests were run for the overtaking scenario and the results of these tests are presented in Table 6.4. As seen from the table, the *Reflector model* outperforms the simple *RMA model*. The columns represent the errors in that particular state of the target vehicle being tracked. The errors presented in the upcoming tables are



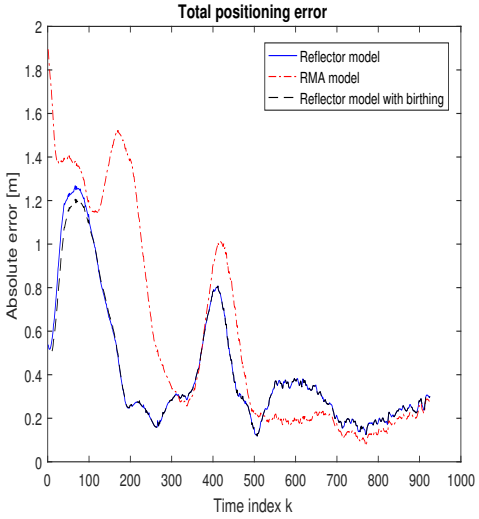
(a) Longitudinal error, after smoothing. Enlarged Figure A.27 can be found in appendix



(b) Lateral error, after smoothing. Enlarged Figure A.28 can be found in appendix



(c) Heading error while, after smoothing. Enlarged Figure A.29 can be found in appendix



(d) Total positioning error, after smoothing. Enlarged Figure A.30 can be found in appendix

Figure 6.15: Results from the overtaking scenario. Error comparisons after smoothing of filtered values of the target shown in Figure 6.12. All figures present the absolute error in the world frame.

all the RMS errors in the host frame. Note that since these are in the host frame, the errors that result from the positioning of the host have been ignored.

Smoothing makes the estimates better for both the models, and this is seen from the smoothing of the test cases compiled in Table 6.4, which is then presented in Table 6.5. The *Reflector model* has a much better performance in this case as

Model	Longitudinal[m]	Lateral[m]	Total[m]	Heading[rad]	Total within $1m$
Reflector model	0.218	0.437	0.538	0.120	88.3%
RMA model	1.288	0.675	1.560	0.100	55.6%

Table 6.4: (Overtaking) The results presented in this table are from 6 independent tests spanning 3417 time samples after the forward filtering. The range from host to target is 2 m to 50 m, the date is the average offset from the ground truth.

well. The overall performance increase of the trajectory estimation after smoothing is around 17% to 19%.

Model	Longitudinal[m]	Lateral[m]	Total[m]	Heading[rad]	Total within $1m$
Reflector model	0.148	0.365	0.435	0.035	95%
RMA model	0.917	0.577	1.177	0.038	65.2%

Table 6.5: (Overtaking) The results presented in this table are from 6 independent tests spanning 3417 time samples after smoothing. The range from host to target is 2 m to 50 m, the date is the average offset from the ground truth.

As a means of comparison for the *Reflector model with birthing* between the following and overtaking scenarios (for which the validation data is available), Table 6.6 presents the numerical values of the different tests that were run for validating their performances for generalisation of results. The first column *Time to birth* represents the average number of indices it takes for the birthing to take place. The second column *Time to converge* shows the average number of indices it takes for the *Reflector model with Birthing* to converge to the *Reflector model*, which has been fed the prior information. These values multiplied by the sampling time (1/30 s) would give the time in *seconds*. The last column presents the number of data logs that were considered for computing the average results.

Scenario	Time to birth [k]	Time to converge [k]	Number of logs
Following	24.6	48.8	26
Overtaking	15.8	NA	6

Table 6.6: The performance results of birthing algorithm for the following and overtaking scenarios.

6.5 Scenario with guardrails

The Interactive Multiple Model filter described in Section 3.2 is implemented for using guardrail estimates. In a scenario where a guardrail is used, the desired result would be the better heading and positioning estimates when the guardrail estimates are valid. This validating logic is provided by the IMM as a weighing algorithm to check if the guardrail estimates should be used or not. In order to visualise this result, a test scenario where the target vehicle moved between two guardrails

(right and left guardrails were present) is considered. The host vehicle followed the target for some distance while the target continued to move in the direction of the contour suggested by the guardrails, until an instance where the target took its own route and went straight while the guardrails went further left and branched out. This scenario is perfect to compare the performances of the reflector model working standalone against it working with the IMM. The map of the trajectories tracked by the various filters is shown in Figure 6.16.

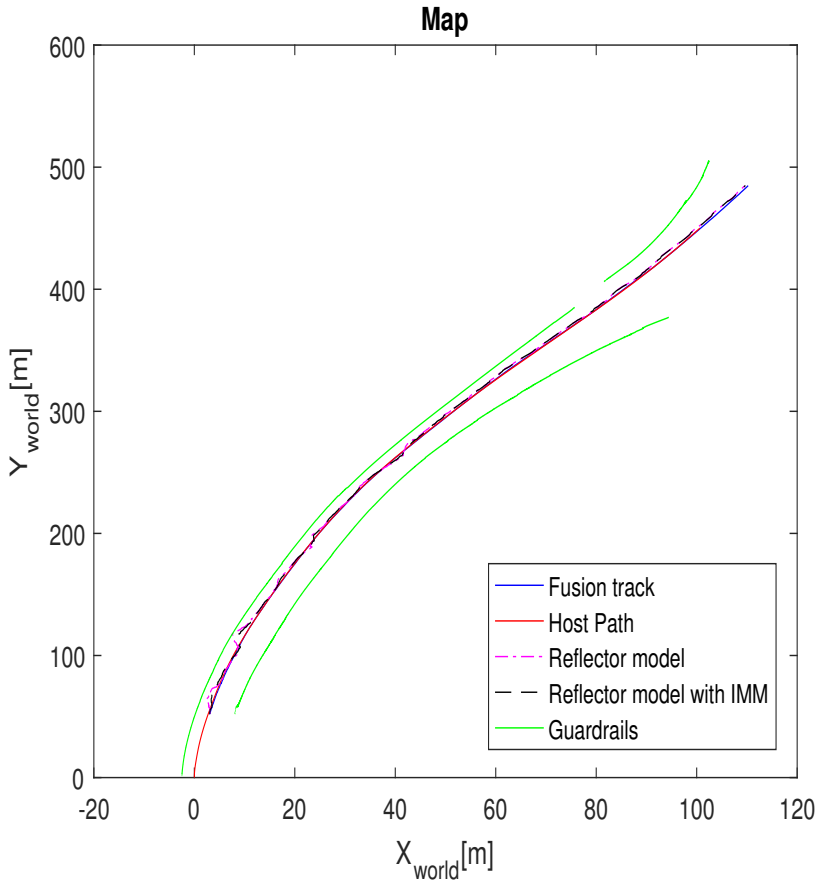
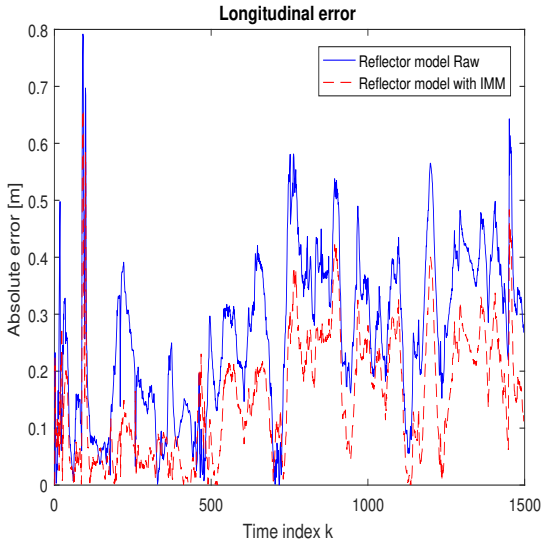


Figure 6.16: Map of the entire trajectory of host and the target for the guardrails scenario. The information from guardrails is utilised for better estimation of target state.

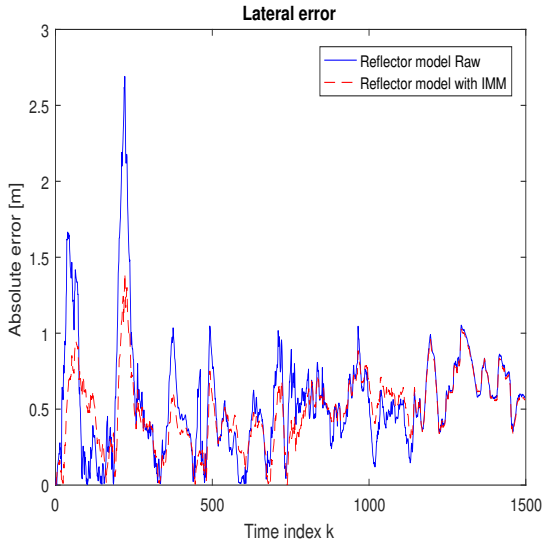
The validation data used for this case is the fusion estimates from the filter developed by Delphi, which gives the fusion estimates between radar and camera. The accuracy of these fusion estimates is not as good as the GPS data of the RT-range system and since GPS data is not available for this scenario, these fusion estimates will have to be used for comparison with a close investigation on the validity of its estimates at certain instances.

Figures 6.17a and 6.17b show the errors in longitudinal and lateral positions between the two filters. The *Reflector model with IMM* has an overall increased performance or similar performance for most of the instances. It is important to remember that the measurement update with the guardrail heading is done only for

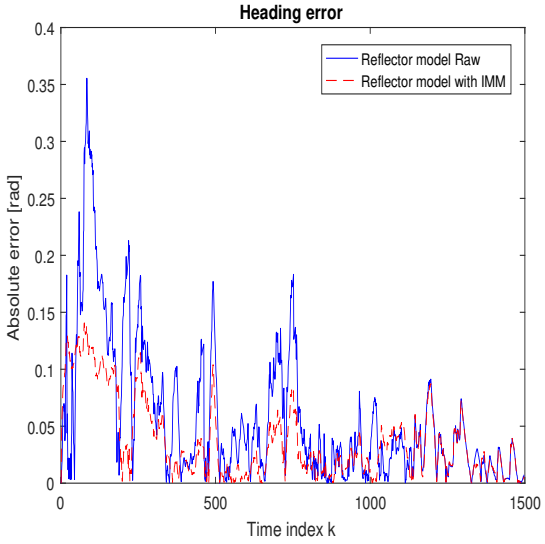
one state space and the weight that each state vector gets for the final estimation of the posterior is given in Figure 6.18 for this case. These weights are chosen, based on the TPM, which is π_{ji}, a predetermined tuning parameter, chosen to be $\pi_{ji} = [0.65\ 0.35; 0.15\ 0.85]$, which was found to be the optimal value. The value chosen for the variance of the barrier estimates was $W_{barr} = 0.3$.



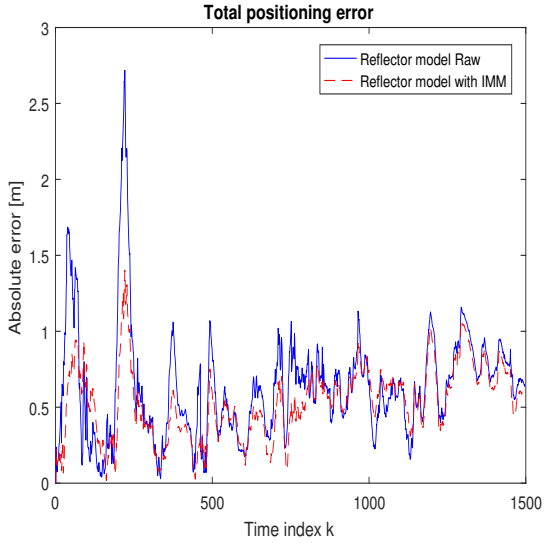
(a) Longitudinal error. Enlarged Figure A.31 can be found in appendix



(b) Lateral error. Enlarged Figure A.32 can be found in appendix



(c) Heading error. Enlarged Figure A.33 can be found in appendix



(d) Total positioning error. Enlarged Figure A.34 can be found in appendix

Figure 6.17: Results from the guardrails scenario. Error comparisons of filtered estimates of the target. All figures present the absolute error in the world frame.

The measurement update as shown in Section 3.3.1 is done only for better heading estimates, and this is reflected in the performance of the *Reflector model with IMM* in both the longitudinal and lateral directions. The better heading es-

timates are valid until about $k = 1100$, when the target starts moving away from the guardrails. This is exactly what is expected of the IMM, in the sense that the guardrails updates stop being given significant weights once the target has reached the point where they are no longer valid (target stops moving parallel to the guardrails). These values are, naturally, highly dependent on the values chosen for the TPM π_{ji} and the variance of the barrier W_{barr}. Once the target has moved away from the guardrails, the estimates of the reflector model with and without the guardrail update converge and behave in a similar way, which is visible from the lateral and heading plots shown in Figure 6.17b and 6.17c. The behaviour in the longitudinal direction is still different for the two models after $k = 1100$, when the *Reflector model with IMM* still performs better than the *Reflector model Raw*, which is the standalone *Reflector model* as shown in the previous results.

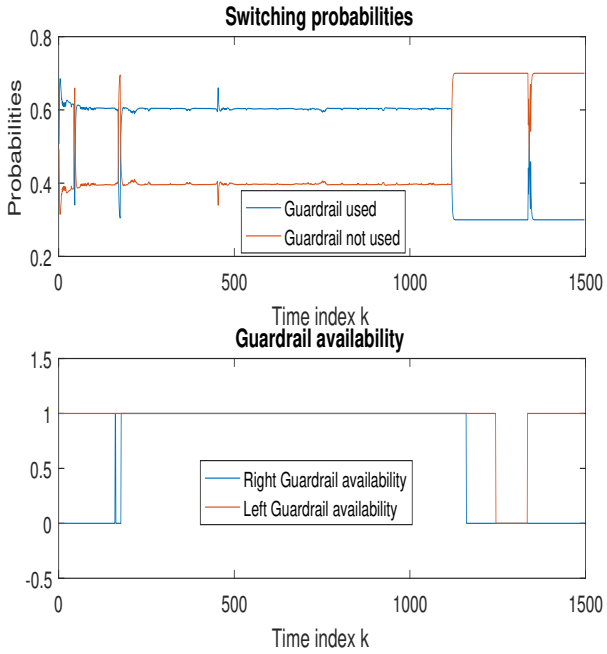


Figure 6.18: Switching mode probabilities and Guardrail estimates availability.

Figure 6.18 shows both the switching probabilities and guardrails (both right and left) availability. The switching mode probabilities are shown for both the states, the one which uses the guardrail update and one that does not. It is fairly clear that any change in the availability of the guardrail estimates shows an immediate response in the mode probabilities. The right guardrail estimate is not available for a few instances in the beginning until about $k = 200$, when the mode probability for the *Reflector model with IMM* gets a jump in the weights. This jump is a result of that at that time instance the right guardrail becomes available and the filter needs to validate this new information. It is expected that until about $k = 1100$, the guardrail estimates are available and valid, which means that they are used for the heading update which is why they get more weight till that point. But this happens much before the guardrail estimates become unavailable, i.e., for a few instances just before $k = 1100$, the *Reflector model Raw* starts getting more weight. This is an

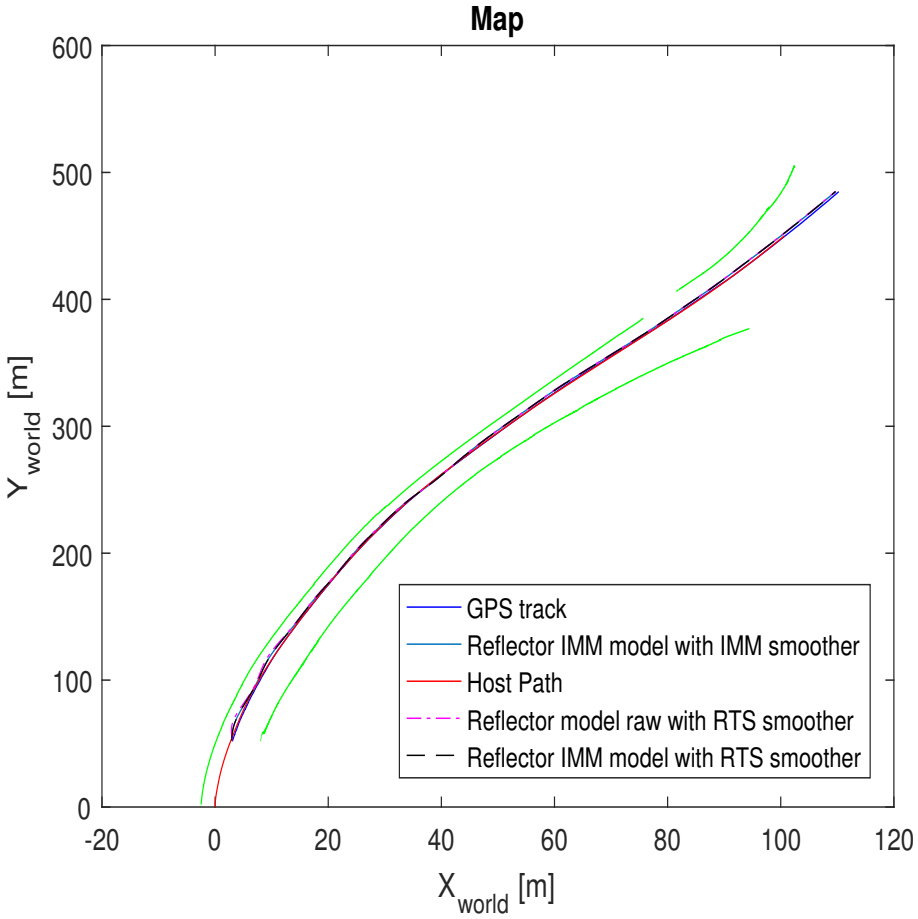
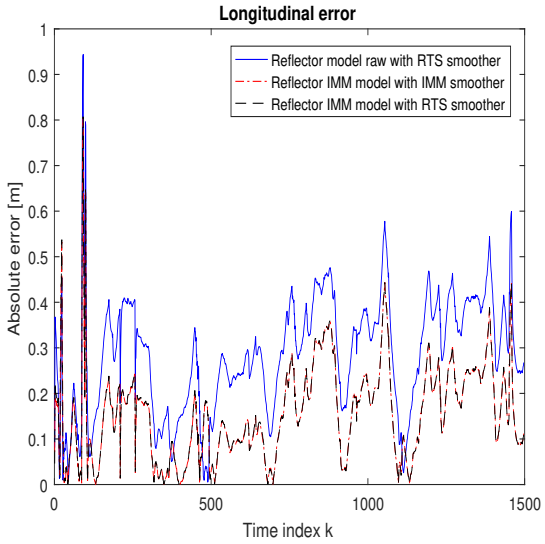


Figure 6.19: Map of the entire smoothed trajectories of the host and filtered trajectories shown in Figure 6.16 of the target for the guardrails scenario. The information from guardrails is utilised for better estimation of target state.

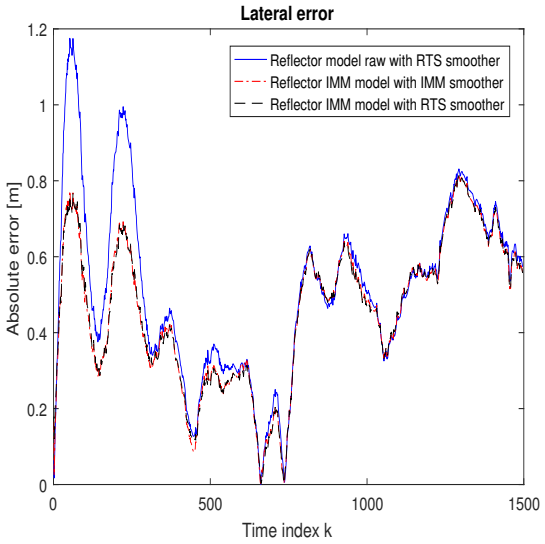
interesting region in the log, where the performance of the IMM is such that even if the guardrail estimates are available, the update is not done because they are no longer valid for the target, and are thus, ignored. Eventually, the guardrail stops being available, and this causes no effect since the states with the guardrail updates are already being given low weights.

There is no effect until about $k = 1350$, where the guardrail estimates are suddenly available again. When this happens, there is a "search" by the IMM to see if these updates are valid again. This is checked for a few instances and the *Reflector model with IMM* is updated with the barrier information, but this also, quickly dies out as a possibility because they are not valid anymore. The target has moved away from the guardrails and thus, this is exactly what is required from the algorithm implemented there for, it can be concluded that the overall performance is much better and as expected from the IMM.

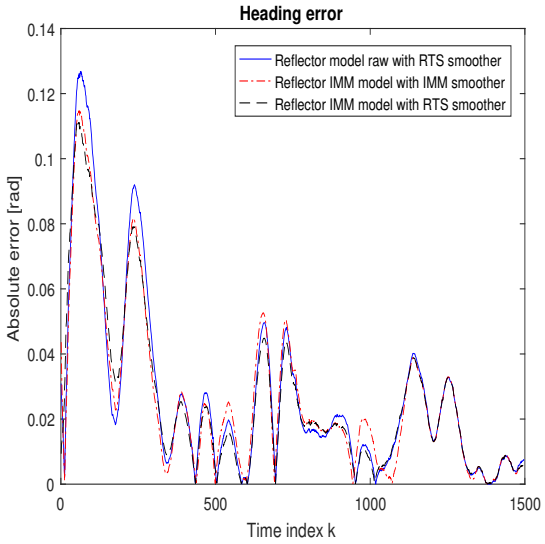
Figure 6.19 shows the smoothed trajectories of the various smoothers applicable for this scenario. Figure 6.20 shows the smoothed error plots for the above filters. There is a possibility to smooth the IMM filtered estimates using the IMM smoother as explained in Section 5.2. The Figure 6.20 shows the errors from the filtered error values from the previous Figure 6.17, smoothed using various techniques. Since the



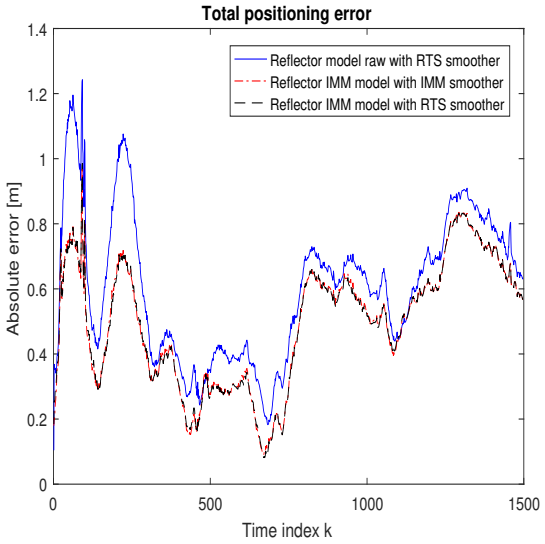
(a) Longitudinal error, after smoothing. Enlarged Figure A.35 can be found in appendix



(b) Lateral error, after smoothing. Enlarged Figure A.36 can be found in appendix



(c) Heading error, after smoothing. Enlarged Figure A.37 can be found in appendix

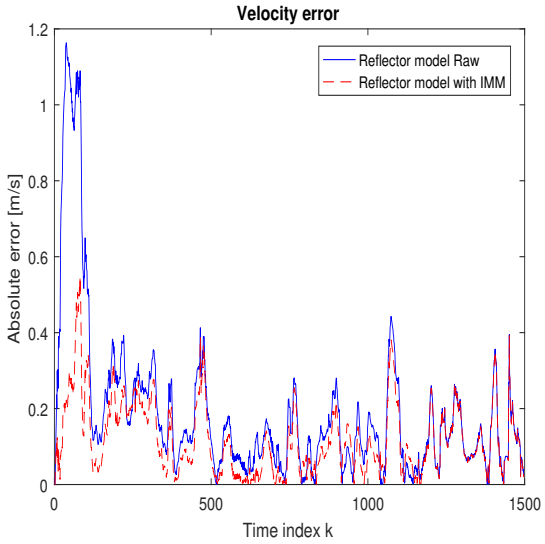


(d) Total positioning error, after smoothing. Enlarged Figure A.38 can be found in appendix

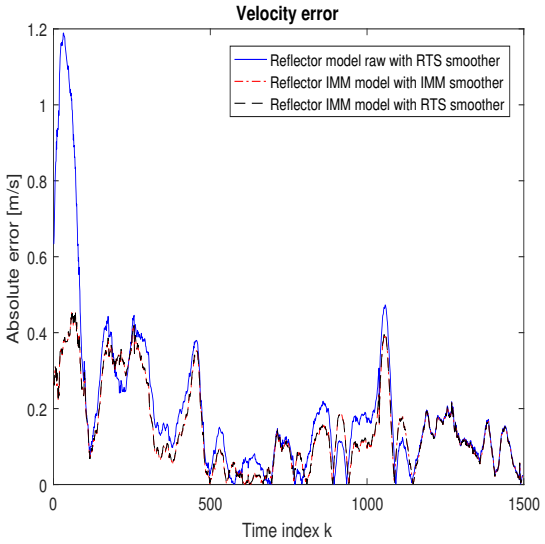
Figure 6.20: Results from the guardrails scenario. Error comparisons of smoothed estimates of filtered values of the target shown in Figure 6.17. All figures present the absolute error in the world frame.

Reflector model Raw does not have IMM estimates, it is not possible to smooth it using the IMM smoother, and thus, all the possible combinations, which includes reflector smoothed using RTS, reflector with IMM smoothed using RTS and reflector with IMM smoothed using IMM are shown here.

From the positioning errors shown in Figure 6.20d, it can be seen that the



(a) Velocity error during the forward filtering. Enlarged Figure A.39 can be found in appendix



(b) Velocity error after the smoothing. Enlarged Figure A.40 can be found in appendix

Figure 6.21: Comparison of the velocity errors for the guardrails scenario. The velocity comparison is the absolute velocity the world frame.

overall performance of the IMM smoother is approximately the same as when the same data is smoothed with an RTS smoother. The same can be seen In Figure 6.21 where the gain in using the IMM filter in the forward filtering is clear in the start of the scenario. But when comparing the two different smoothing methods on the IMM forward filtered data the performance is almost identical. But these values, as stated before, cannot be assumed to be compared to absolute ground truth, so the actual evaluation cannot be done since they are too close and the absolute accuracy of the fusion estimates is unknown. The conclusion that can be drawn from these analyses is that the potential performance gain in using the IMM smoother rather than the RTS is minimal, even though the lack of accurate ground truth data for this scenario sets constraints on the evaluation.

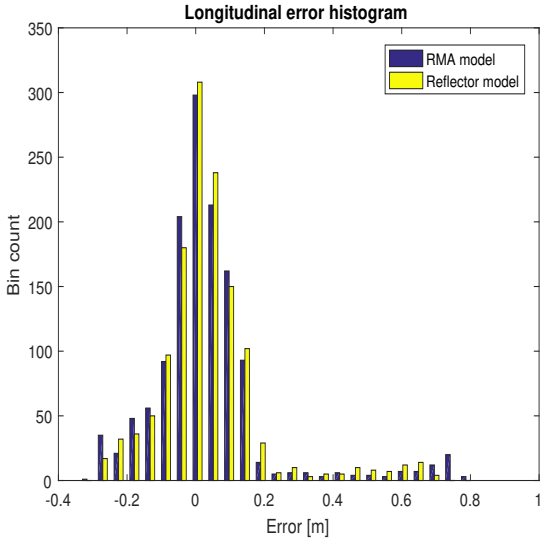
6.6 Error analyses

As the final analysis to interpret the errors, the empirical distributions in the form of histograms of the errors in the various scenarios have been plotted in this section. The reasons to do these analyses are as follows:

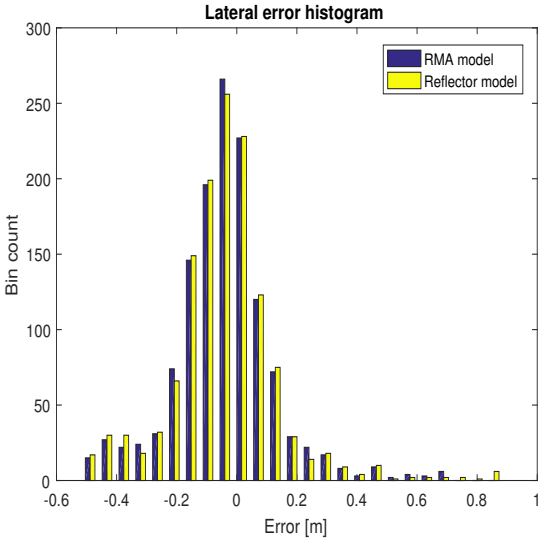
- First, such empirical distributions can highlight differences between different filters, to show which gives smaller errors.
- Furthermore, if the empirical distributions of the errors are Gaussian, it is a sign that there is no more information left in the data. If the empirical distributions are significantly non-Gaussian, it could be a sign that the is some bias, or some other problem, that has not been captured enough by the models and the filtering.

6.6.1 Following

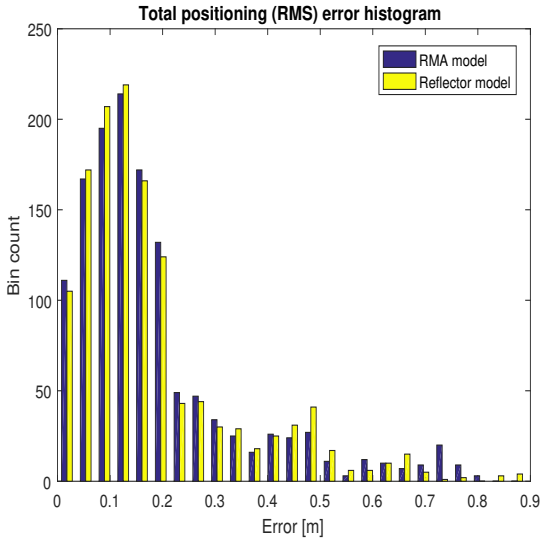
In Figures 6.22 the errors of the two filter types are compared in histograms. As discussed in section 6.4.1 the performance of the two different filters is almost the same. The histograms in Figures 6.22a, 6.22b and 6.22d look fairly Gaussian distributed which indicates that most of the information available is being used by the two tracking algorithms.



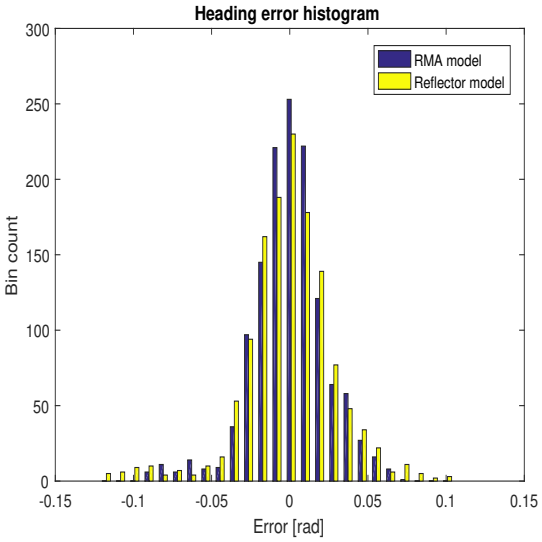
(a) Longitudinal error.



(b) Lateral error.



(c) Total positioning error.



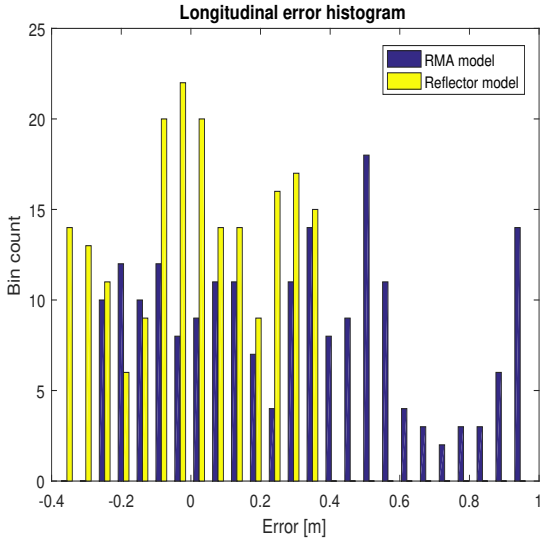
(d) Heading error.

Figure 6.22: Histograms for the errors from the following scenario.

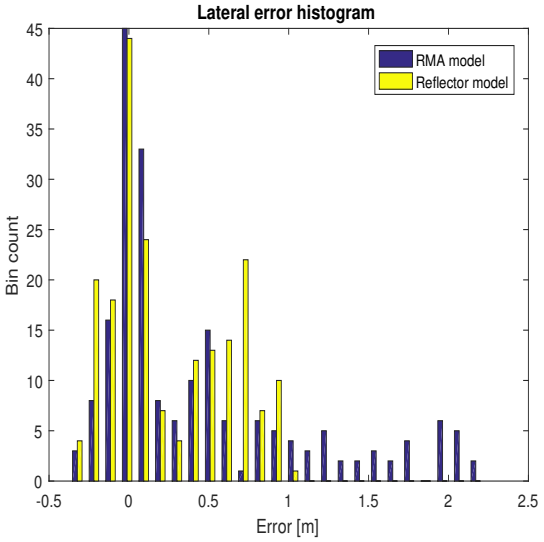
6.6.2 Turning

The Figures in 6.23 show the errors from the turning scenario plotted in histograms. From these figures it can be seen that the error of the *Reflector model* has a tighter

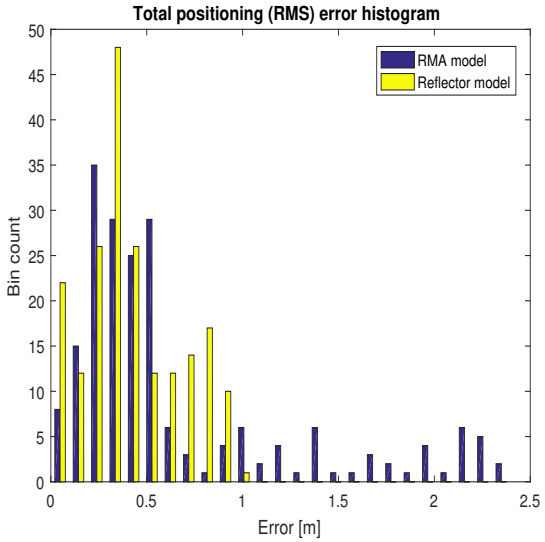
distribution in the positions as well as being closer to zero mean. Figure 6.23c illustrates the distribution of the total position error, which shows that the reflector model not only is closer to a zero mean it also has a closer resemblance to a Gaussian distribution indicating that it is utilising more of the information available. Not all of these show a Gaussian distribution of the errors especially in Figure 6.23a where the the distribution is closer to being uniform than Gaussian for both models. This is expected due to the non-linearity of the problem as well as the fact that the number of data points for this scenario is only 200 samples which is possibly not sufficient to capture the true error distribution.



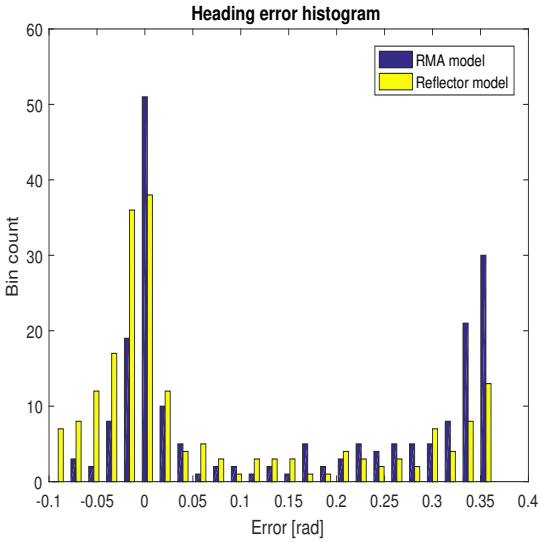
(a) Longitudinal error.



(b) Lateral error.



(c) Total positioning error.

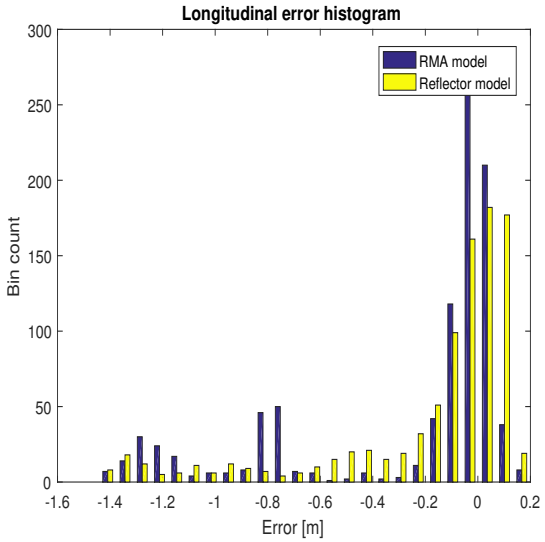


(d) Heading error.

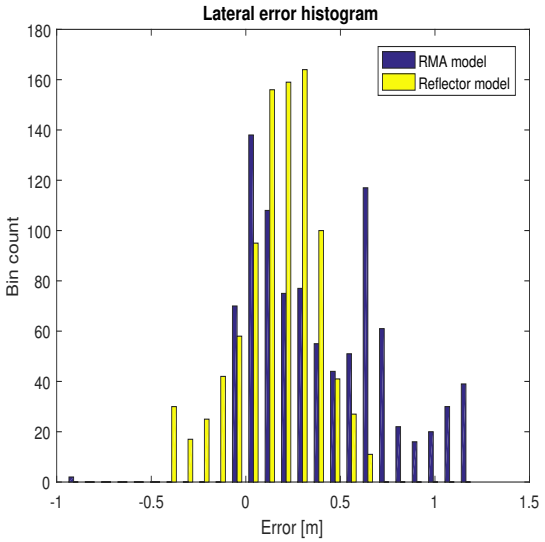
Figure 6.23: Histograms for the errors from the turning scenario.

6.6.3 Overtaking

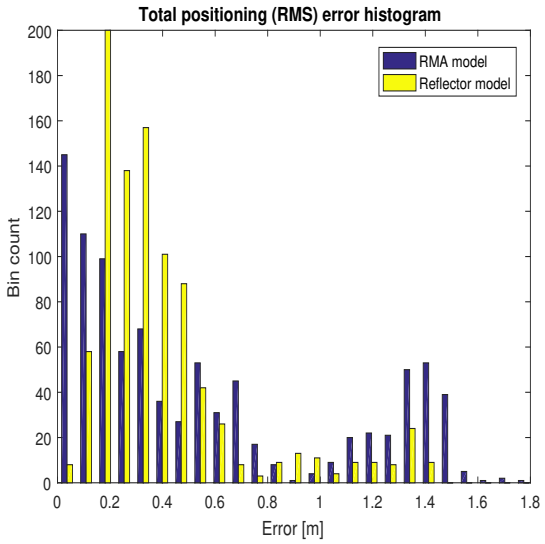
The Figures in 6.24 show the errors from the overtaking scenario plotted in histograms. In general we can see that the *Reflector model* error distribution has a closer resemblance to a Gaussian distribution, indicating that most of the information available is being utilised in the tracker.



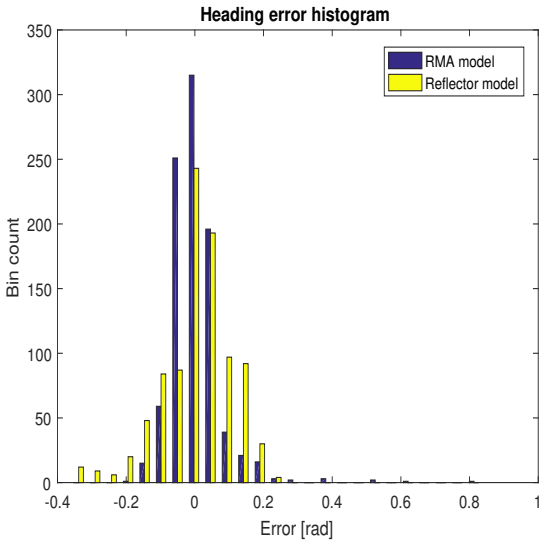
(a) Longitudinal error.



(b) Lateral error.



(c) Total positioning error.



(d) Heading error.

Figure 6.24: Histograms for the errors of the overtaking scenario.

7

Conclusion and Future Work

The thesis was initially aimed at employing an extended target tracking method, and also use guardrail information. To optimise the trajectory, smoothing solutions were to be used. Additionally, to be able to create a prior for the target, birthing and killing techniques were to be developed. With the final implementation where all the algorithms work in parallel, these objectives were mostly satisfied. Some cutbacks were made in order to make the algorithm more computationally efficient though there was not much focus on this in the beginning. The algorithm presented shows promising results for all scenarios considered, irrespective of the availability of guardrails and can start efficiently tracking a target when its prior is not known by searching the entire area of interest. As soon as this prior is created, the radar resolution model tracks the target, in which the dimensions and shape of the vehicle have been taken into account. This gives better tracking than using a point filter or a simple extended model (spatial model) especially when more than one side of the target is seen by the host. In the scenario where the host follows the target as presented in Section 6.4.1, further analyses on the angular dependencies of the expected RCS values of the reflectors might be able to improve the tracking accuracy even further.

The multiple hypotheses formulation can be included in the radar resolution model to observe how much better the solutions are, and then formulate a trade-off point where the number of hypotheses could only be made for the moving detections against all the visible reflectors and completely ignoring the clutter. Also, the dimensions of the vehicle are assumed to be roughly known, but this is for medium-sized cars. So, an online estimation of vehicle dimensions may be developed in order to place the reflectors correctly.

The IMM filter helps to make use of guardrail updates, and this algorithm has clear benefits when used, in correcting both the heading and position estimates. This implementation on an online system is a task that needs a lot of attention since the causality requirements for an online system need to be kept in mind. This, along with of course other algorithms, needs to be tested for robustness before a real-time implementation is made.

For birthing, initially an assumption was made that the target was a point target. This was later replaced by a spatial target which is an extended target with rough dimensional assumptions so that the switch between the birthing algorithm and the radar resolution model would not be as radical. With the spatial model, it was noted that the tracking and switching were much better, but it also had some drawbacks in the form of killing the target even though it still exists in the FOV of the radar. This is due to the switching logic being based on the spatial model.

All the scenarios considered and algorithms developed work for the case when there is only one target that is being tracked, but in reality there are numerous vehicles on the road. In order to track all of them (or at least the ones that are in the immediate surrounding of the host), an MTT (Multi-Target Tracking) algorithm needs to be developed, which can track multiple targets using the radar resolution model.

It was noted that using the IMM smoother did not have any significant impact on getting better estimates than the RTS smoother. Smoothing tasks are usually preferred for analysing the trajectory that a particular vehicle has taken, and this could be expanded by also mapping the road, and using that information to smooth the trajectory. With this, the posterior would be much closer to the ground truth. Naturally, these ideas can be stretched to other sensors and fusing them finally together to complete the task of perceiving the environment around the host vehicle. The algorithms also need to be optimised for being able to run on an online system performing real-time tracking and filtering.

Bibliography

[1] Y. Bar-Shalom, K. C. Chang, and H. A. P. Blom, "Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, no. 2, pp. 296–300, 1989.

[2] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", AAAI Press, 1996, pp. 226–231.

[3] E. P. Blasch and M. Hensel, "Fusion of distributions for radar clutter modeling", DTIC Document, Tech. Rep., 2004.

[4] Y. Bar-Shalom, S. Challa, and H. A. P. Blom, "IMM estimator versus optimal estimator for hybrid systems", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 3, pp. 986–991, 2005.

[5] M. Feldmann and D. Franken, "Tracking of extended objects and group targets using random matrices - a new approach", *Proceedings of the International Conference on Information Fusion*, 2008.

[6] W. Koch, "Bayesian approach to extended object and cluster tracking using random matrices", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 3, pp. 1042–1059, 2008.

[7] U. Orguner and M. Demirekler, "Maximum likelihood estimation of transition probabilities of jump markov linear systems", *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 5093–5108, 2008.

[8] M. Feldmann and D. Franken, "Advances on tracking of extended objects and group targets using random matrices", *Proceedings of the International Conference on Information Fusion*, pp. 1029–1036, 2009.

[9] W. Koch, M. Feldmann, and D. Franken, "Bayesian approach to extended object and cluster tracking using random matrices", *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1409–1420, 2011.

[10] L. Hammarstrand, F. Sandblom, and L. Svensson, "Extended object tracking using a radar resolution model", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2371–2386, 2012.

[11] M. Mallick, V. Krishnamurthy, and B.-N. Vo, *Integrated tracking, classification, and sensor management: Theory and applications*. River Street, Hoboken: Wiley-IEEE Press, 2012.

[12] N. Nadarajah, R. Tharmarasa, M. McDonald, and T. Kirubarajan, "IMM forward filtering and backward smoothing for maneuvering target tracking", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2673–2678, 2012.

[13] B. T. Vo, C. M. See, N. Ma, and W. T. Ng, "Multi-sensor joint detection and tracking with the bernoulli filter", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 2, pp. 1385–1402, 2012.

[14] B. Ristic, B.-T. Vo, B.-N. Vo, and A. Farina, "A tutorial on bernoulli filters: Theory, implementation and applications", *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3406–3430, 2013.

[15] S. Särkkä, *Bayesian filtering and smoothing*. Shaftesbury Road, Cambridge: Cambridge University Press, 2013.

[16] M. Roth, G. Hendeby, and F. Gustafsson, "EKF/UKF maneuvering target tracking using coordinated turn models with polar/cartesian velocity", in *Information Fusion (FUSION), 2014 17th International Conference on*, IEEE, 2014, pp. 1–8.

[17] K. Granström, A. Natale, P. Braca, G. Ludeno, and F. Serafino, "Gamma gaussian inverse wishart probability hypothesis density for extended target tracking using x-band marine radar data", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 22, pp. 6617–6631, 2015.

[18] S. Hamid Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid, "Joint probabilistic data association revisited", in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3047–3055.

[19] A. Pettersson and V. Svensson, "Road geometry estimation for longitudinal vehicle control", Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2015.

[20] K. Granstrom and M. Baum, "Extended object tracking: Introduction, overview and applications", *arXiv preprint arXiv:1604.00970*, 2016.

[21] DBSCAN Clustering in MATLAB. Accessed: 2017-04-25, [Online]. Available: `http://yarpiz.com/255/ypml110-dbscan-clustering`.

[22] B. T. Vo. BA Tuong Vo Matlab Codes. Accessed: 2017-04-25, [Online]. Available: `http://ba-tuong.vo-au.com/codes.html`.

A

Appendix 1

A.1 CT model

We use a CT motion model, where the state vector is

$$z_k = \begin{bmatrix} x_k & y_k & v_k & \theta_k & \omega_k \end{bmatrix}^T, \tag{A.1}$$

where x_k is the x-coordinate, y_k is the y-coordinate, v_k is the unresolved velocity, θ_k is the heading angle and ω_k is the angular velocity at any instant k.

The discretisation of the motion model is a nonlinear function $z_k = f(z_{k-1}, T)$, which is suggested in [16], given by

$$z_k = f(z_{k-1}, T) = \begin{bmatrix} x_{k-1} + \frac{2v_{k-1}}{\omega_{k-1}} \cos(\theta_{k-1} + \omega_{k-1}\frac{T}{2}) \sin(\omega_{k-1}\frac{T}{2}) \\ y_{k-1} + \frac{2v_{k-1}}{\omega_{k-1}} \sin(\theta_{k-1} + \omega_{k-1}\frac{T}{2}) \sin(\omega_{k-1}\frac{T}{2}) \\ v_{k-1} \\ \theta_{k-1} + \omega_{k-1}T \\ \omega_{k-1} \end{bmatrix}^T. \tag{A.2}$$

A.2 Motion Model Noise

The model noise Q_{k-1} is formed as suggested in [16].

$$G = \begin{bmatrix} \frac{T^2 cos(\theta_{k-1|k-1})}{2} & 0 \\ \frac{T^2 sin(\theta_{k-1|k-1})}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix} \tag{A.3}$$

$$Q_{k-1} = G\Sigma G^T \tag{A.4}$$

A.3 CKF prediction

The state prediction at instant k is done using the prior at that step, which is $\hat{z}_{k-1|k-1}$. Since we use a Cubature Kalman filter (CKF), we have $2n_z$ sigma points,

all which need to be propagated through this step. $n_z = 5$, where n_z is the dimensionality of z_k, since the CT model is used. The CKF prediction logic as provided in [15] is shown in the algorithm below.

Algorithm 7: Cubature Kalman Filter Prediction

function $CKF_{pred}(z_{k-1|k-1}, P_{k-1|k-1}, \Sigma, T)$
Data: $z_{k-1|k-1}$ posterior state estimate from previous instant, $P_{k|k-1}$ posterior state covariance estimate from previous instant, Σ contains the motion noise covariance for the nose driven states and T is the sample time.
Result: $z_{k|k-1}, P_{k|k-1}$
begin
  $\mathcal{Z}_{k-1}^{(i)} = \hat{z}_{k-1|k-1} + \sqrt{n_z}(P_{k-1|k-1}^{\frac{1}{2}})_i, i = 1, 2, ..., n_z$
  $\mathcal{Z}_{k-1}^{(i+n_z)} = \hat{z}_{k-1|k-1} - \sqrt{n_z}(P_{k-1|k-1}^{\frac{1}{2}})_i, i = n_z + 1, 2, ..., 2\,n_z$
  $\hat{\mathcal{Z}}_{k|k-1}^{(i)} = f_{k-1}(\mathcal{Z}_{k-1}, T), i = 1, 2, ..., 2n_z$
  Q_{k-1} = computed as shown in A.3 and A.4
  $\hat{z}_{k|k-1} = \frac{1}{2n_z}\sum_{i=1}^{2n_z}\hat{\mathcal{Z}}_{k|k-1}^{(i)}$
  $P_{k|k-1} = Q_{k-1} + \frac{1}{2n_z}\sum_{i=1}^{2n_z}(\hat{\mathcal{Z}}_{k|k-1}^{(i)} - \hat{z}_{k|k-1})(\hat{\mathcal{Z}}_{k|k-1}^{(i)} - \hat{z}_{k|k-1})^T$
end

A.4 Measurement model

Given the positions of the target (x_k, y_k) from the CT state vector z_k of the target at any instant k, the mapping from the state space to the measurement space is computed as

$$h(z) = \begin{bmatrix} \sqrt{(x_k - s_{x(k)})^2 + (y_k - s_{y(k)})^2} \\ v_k \\ \arctan\left(\frac{y_k - s_{y(k)}}{x_k - s_{x(k)}}\right) \end{bmatrix}, \tag{A.5}$$

where $(s_{x(k)}, s_{y(k)})$ are the positions of the sensor or the host in the x and y axes at instant k.

A.5 CKF update

Algorithm 8: Cubature Kalman Filter Update

function $CKF_{update}(R, y, z_{k|k-1}, P_{k|k-1}))$

Data: $z_{k|k-1}$ predicted state estimate, $P_{k|k-1}$ predicted state covariance estimate, R measurement model noise covariance matrix, y measurement.

Result: ε_k innovation, S_k innovation covariance, $z_{k|k}$ posterior state estimate, $P_{k|k}$ posterior state covariance estimate, η_k predicted measurement

begin

$\mathcal{Z}_k^{(i)} = \hat{z}_{k|k-1} + \sqrt{n_z}\sqrt{P_{k|k-1}}, i = 1, 2, ..., n_z$

$\mathcal{Z}_k^{(i+n_z)} = \hat{z}_{k|k-1} - \sqrt{n_z}\sqrt{P_{k|k-1}}, i = n_z + 1, 2, ..., 2\,n_z$

$\hat{\mathcal{Y}}_k^{(i)} = h(\mathcal{Z}_k^{(i)})$

$\eta_k = \sum_{i=1}^{2n_z} \hat{\mathcal{Y}}_k^{(i)}$

$S_k = R + \frac{1}{2n_z}\sum_{i=1}^{2n_z}(\hat{\mathcal{Y}}_k^{(i)} - \eta)(\hat{\mathcal{Y}}_k^{(i)} - \eta_k)^T$

$C_k = \frac{1}{2n_z}\sum_{i=1}^{2n_z}(\mathcal{Z}_k^{(i)} - x_{k|k-1})(\hat{\mathcal{Y}}_k^{(i)} - \eta_k)^T$

$\varepsilon_k = y_k - \eta_k$

$K_k = C_k S_k^{-1}$

$\hat{z}_{k|k}^i = \hat{z}_{k|k-1}^i + K_k(\varepsilon_k)$

$P_{k|k}^i = P_{k|k-1}^i - K_k S_k K_k^T$

end

A.6 Enlarged result figures

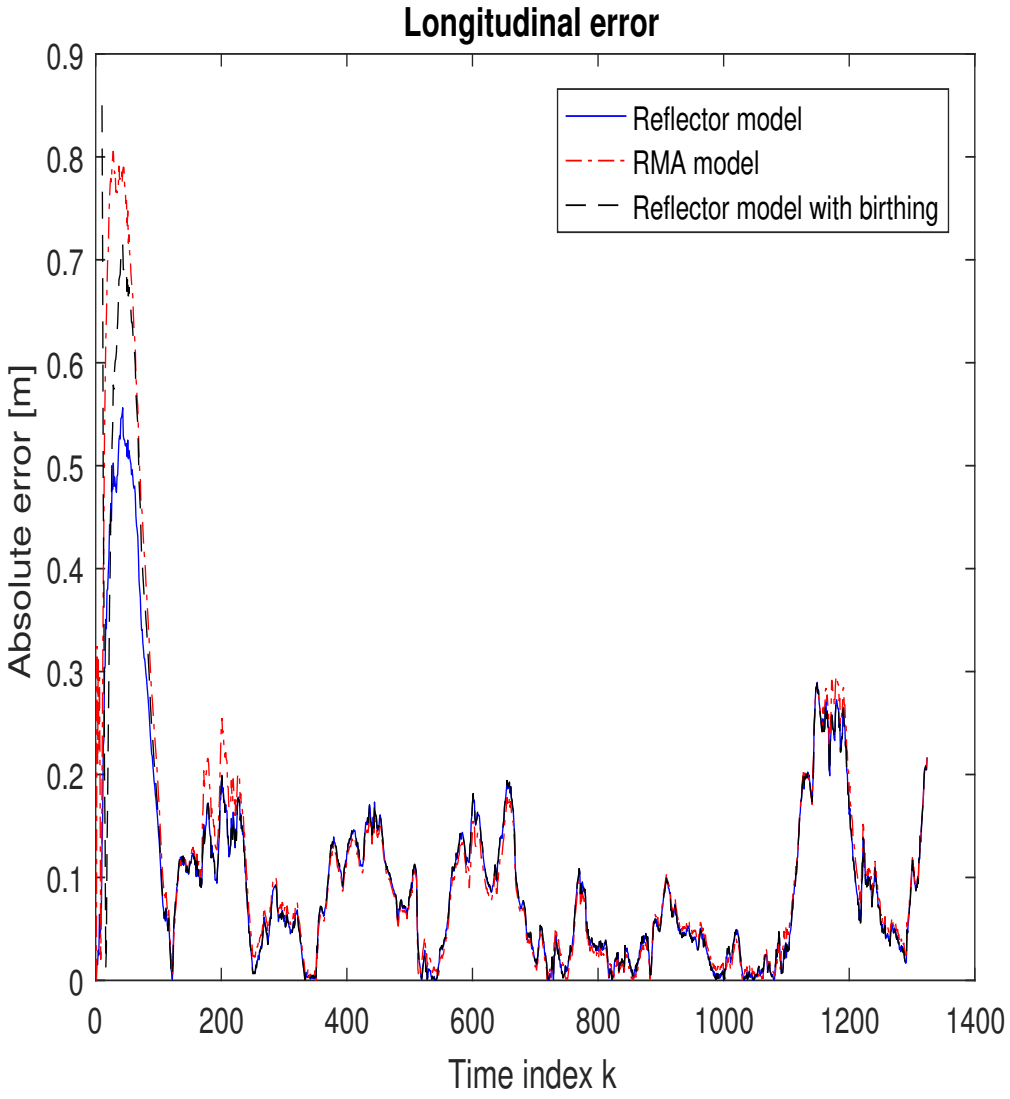


Figure A.1: Longitudinal error following scenario. Enlargement of Figure 6.2a

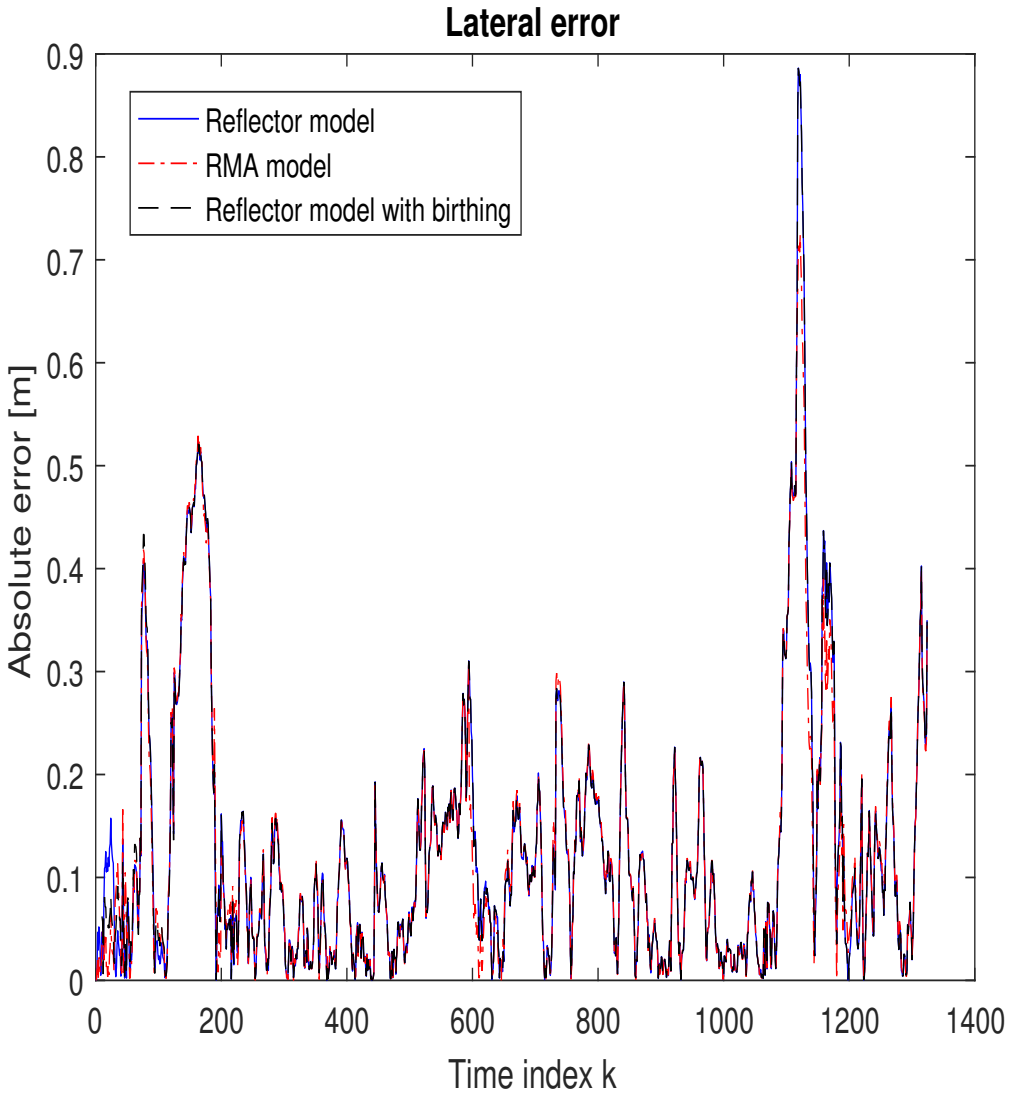


Figure A.2: Lateral error following, scenario. Enlargement of Figure 6.2b

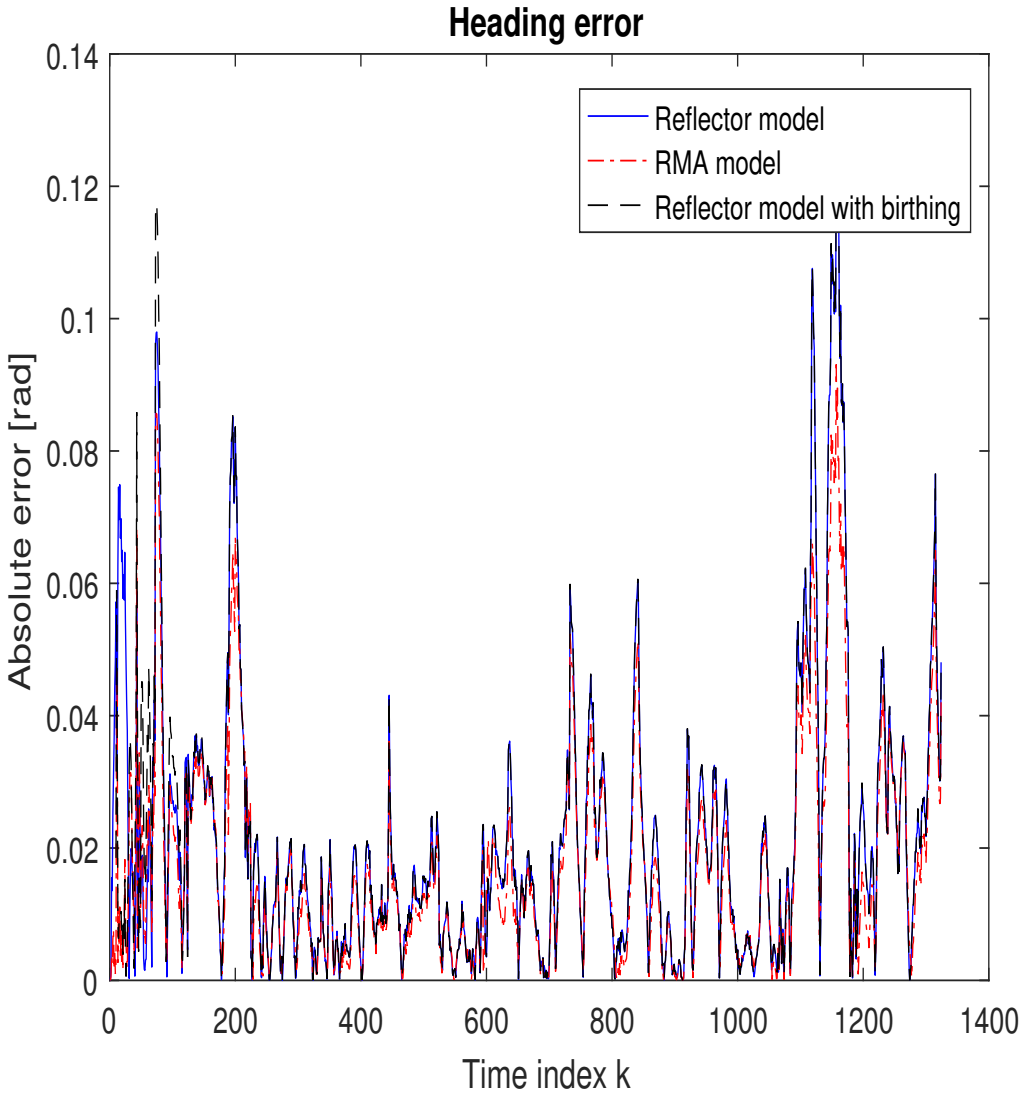


Figure A.3: Heading error following, scenario. Enlargement of Figure 6.2c

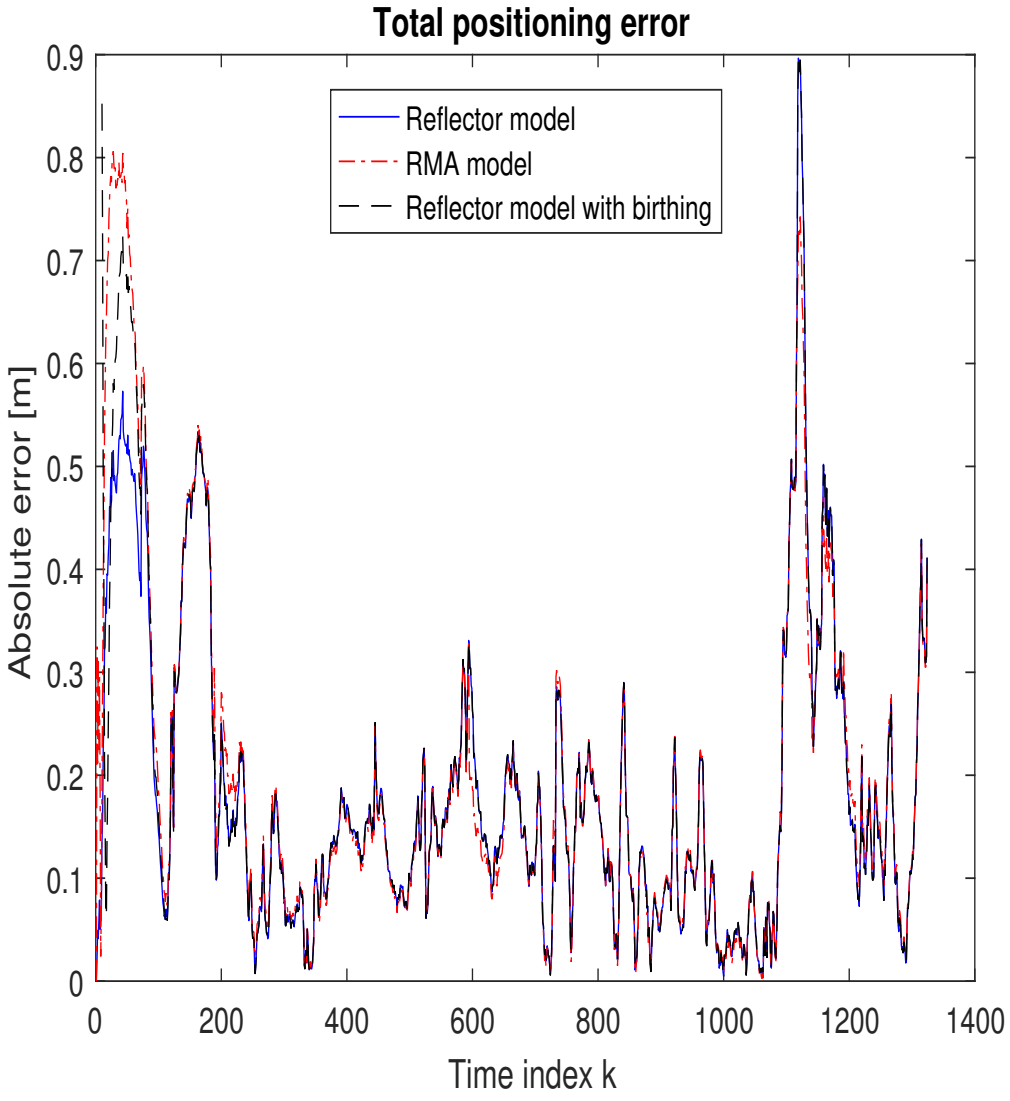


Figure A.4: Total positioning error, following scenario. Enlargement of Figure 6.2d

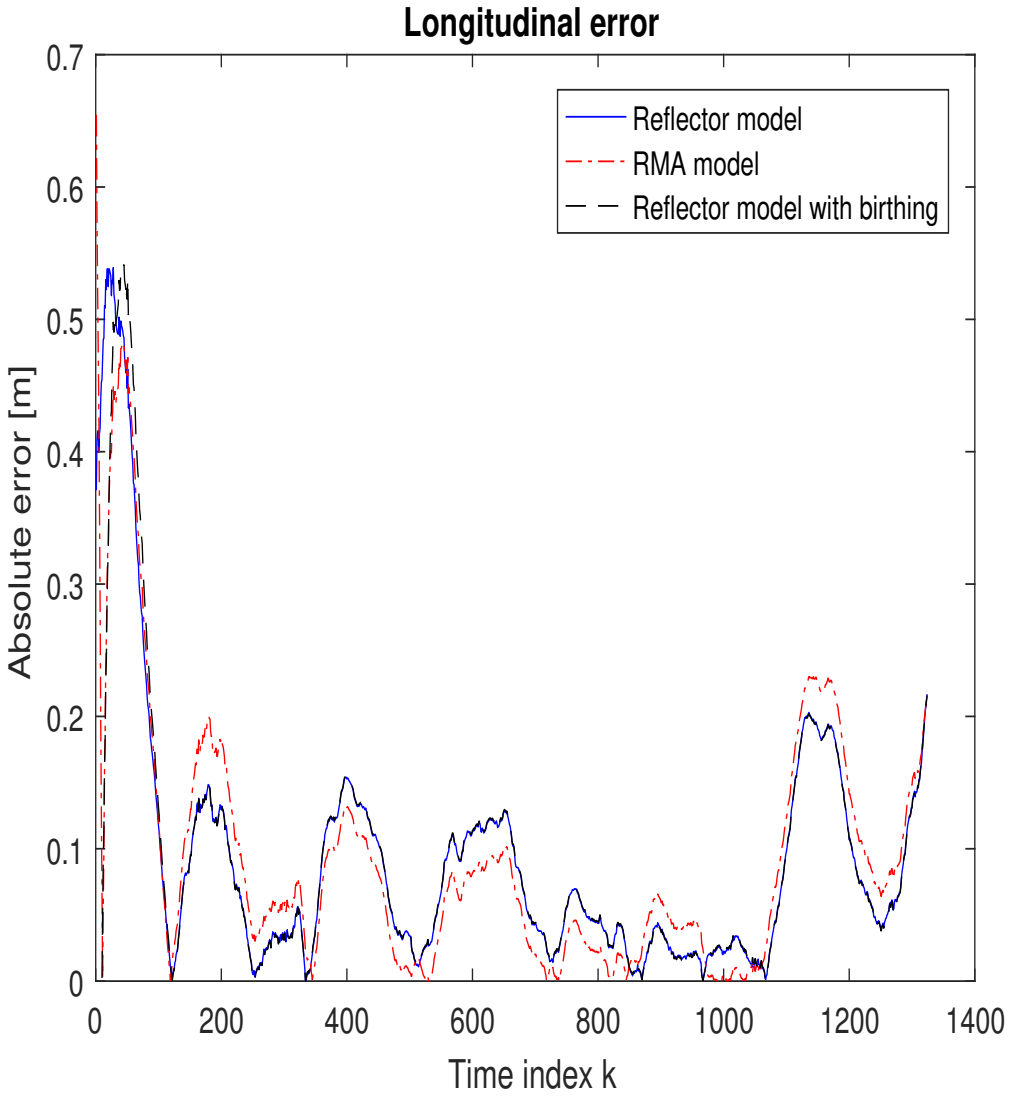


Figure A.5: Longitudinal error, after smoothing, following scenario. Enlargement of Figure 6.4a

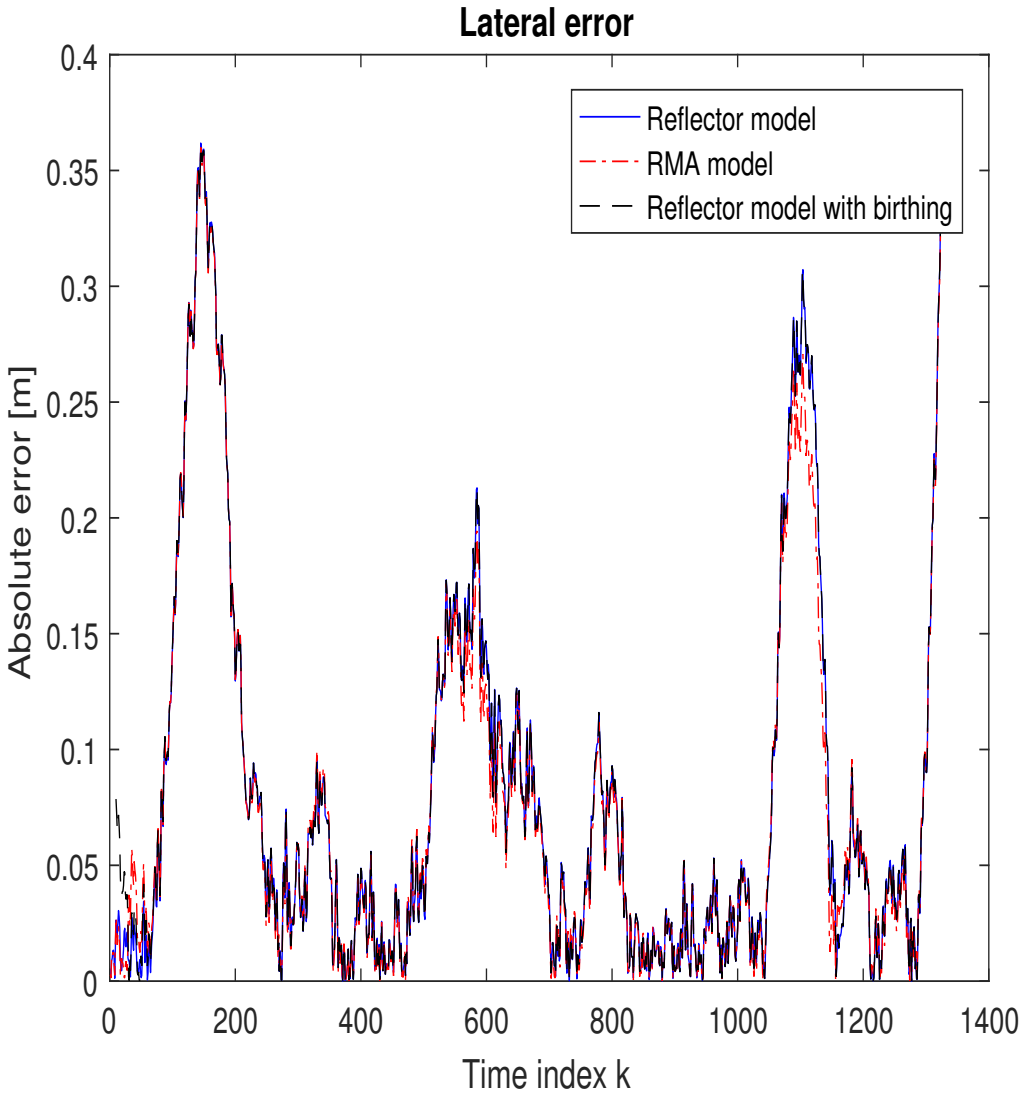


Figure A.6: Lateral error, after smoothing, following scenario. Enlargement of Figure 6.4b

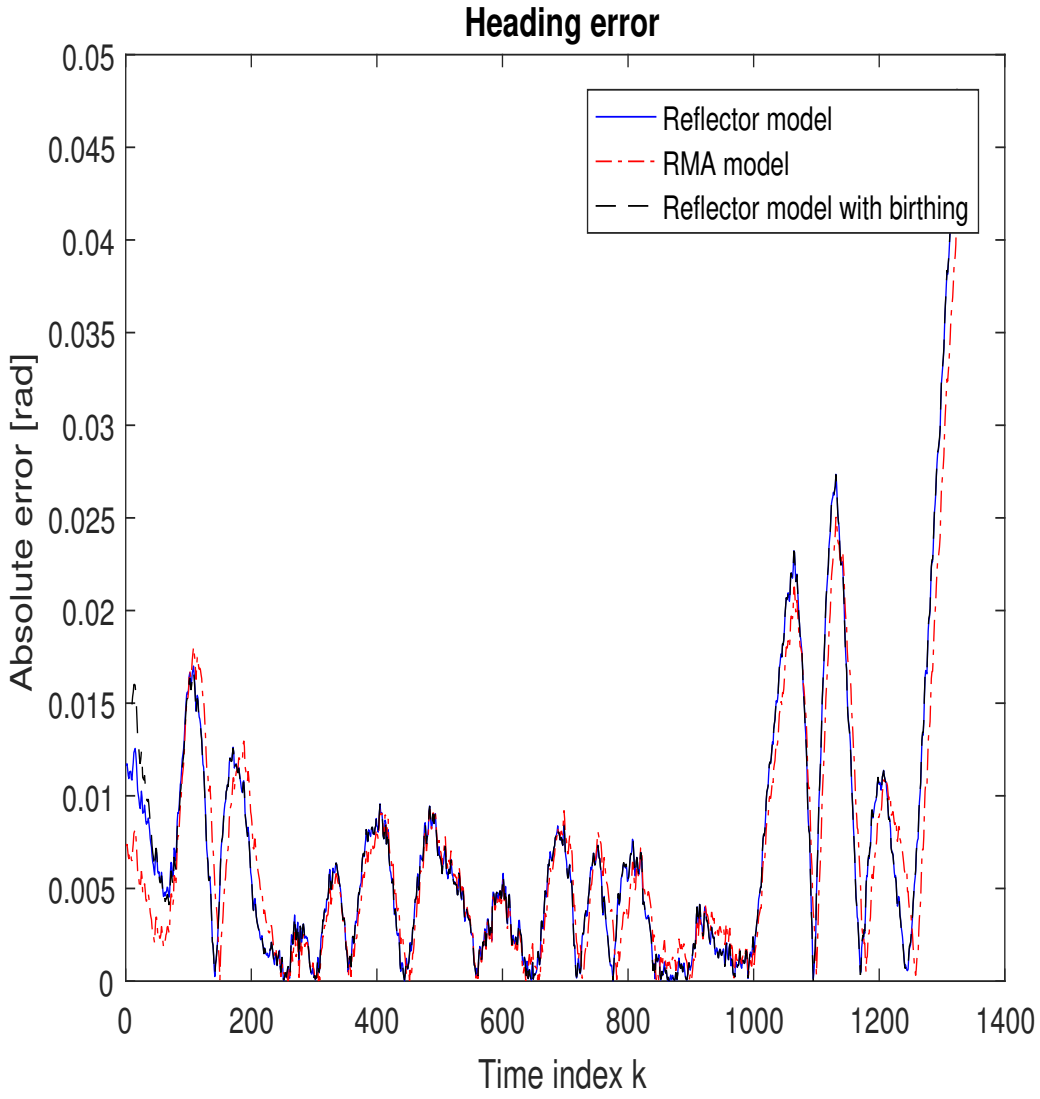


Figure A.7: Heading error, after smoothing, following scenario. Enlargement of Figure 6.4c

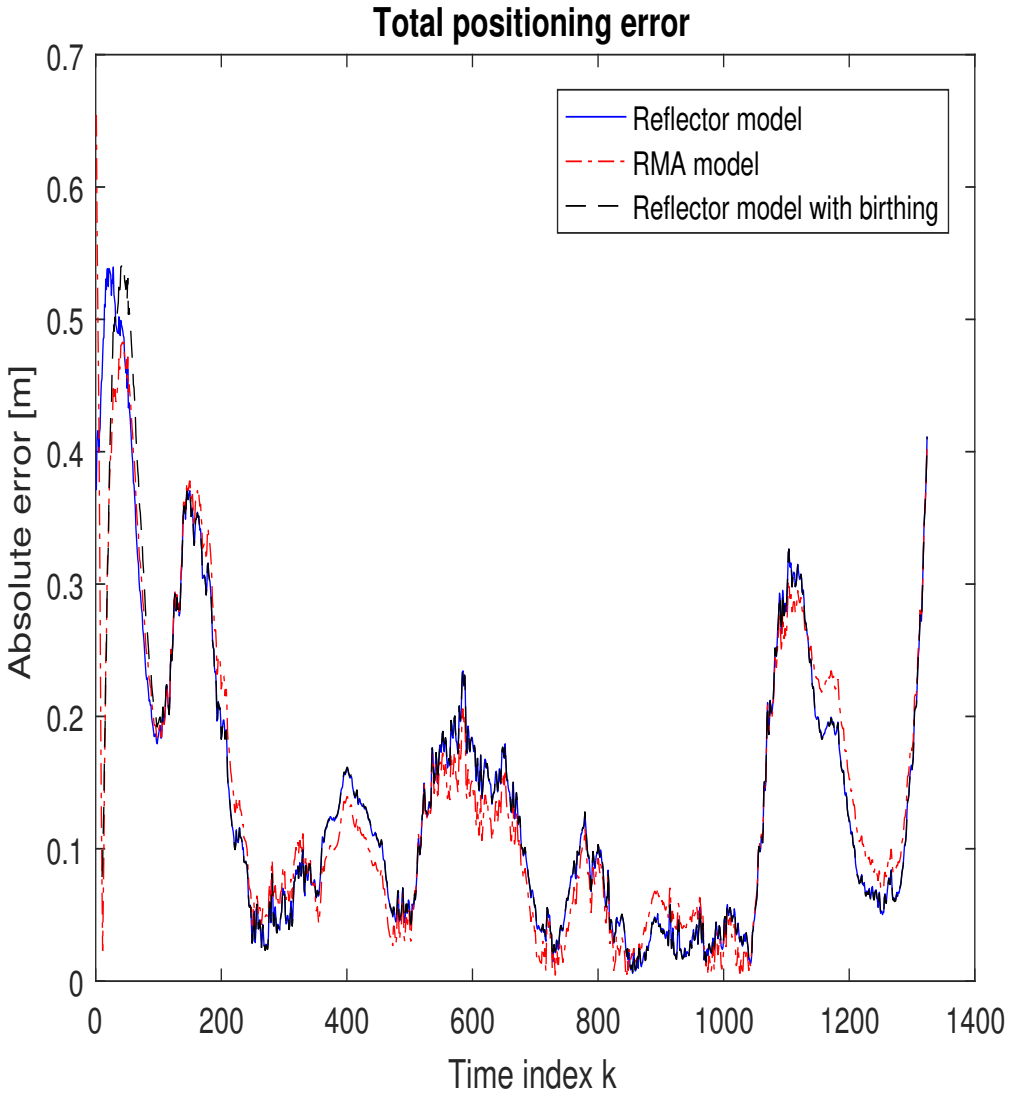


Figure A.8: Total positioning error, after smoothing, following scenario. Enlargement of Figure 6.4d

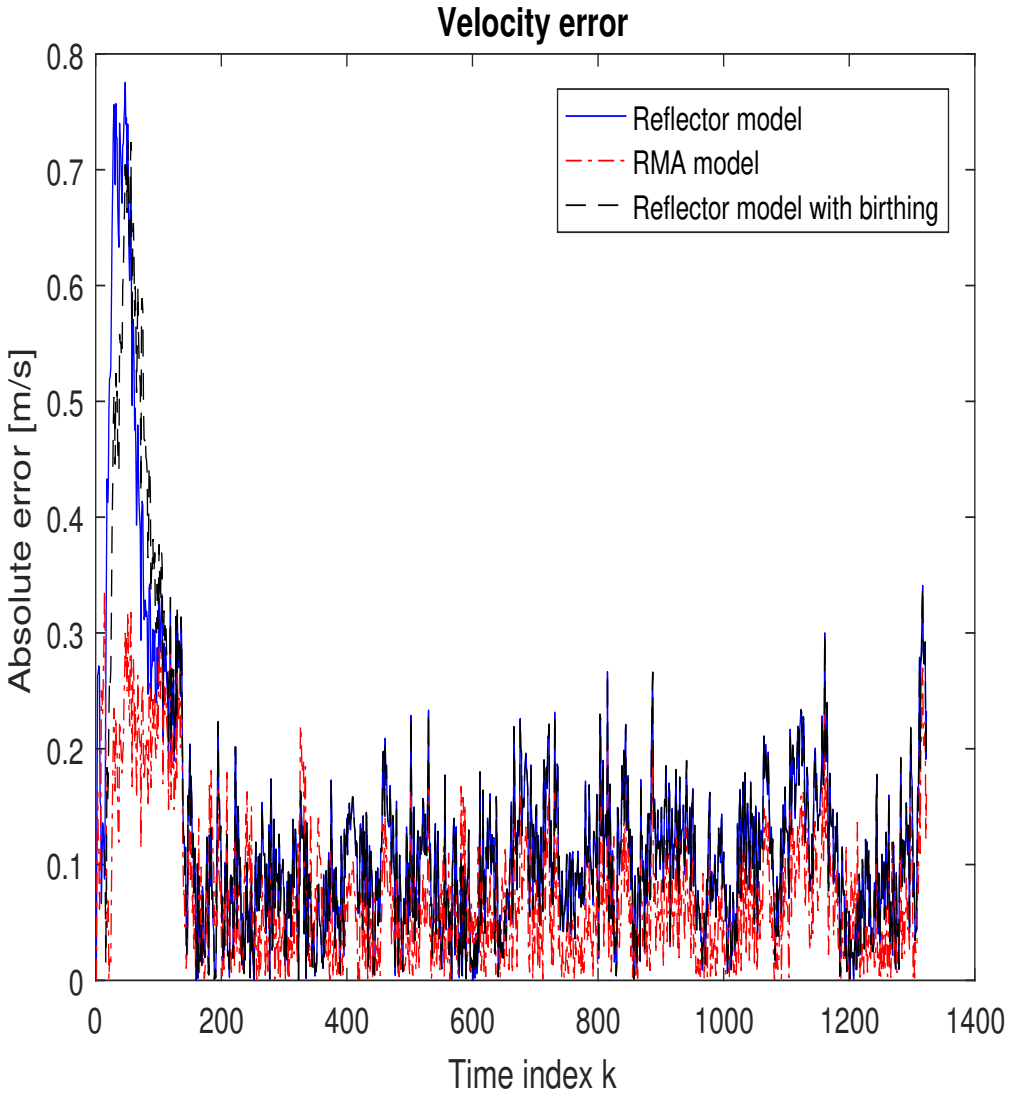


Figure A.9: Velocity error, following scenario. Enlargement of Figure 6.5a

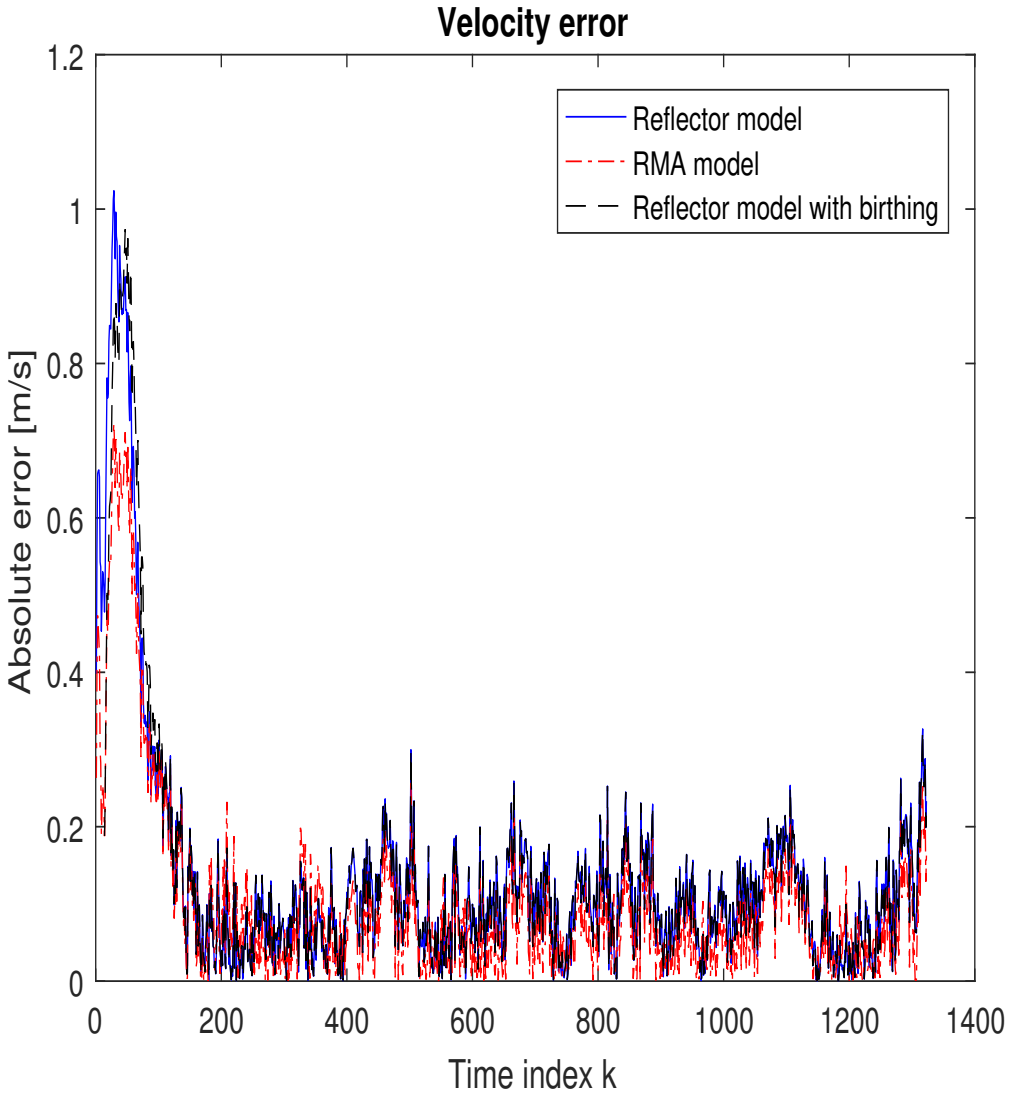


Figure A.10: Velocity error, after smoothing, following scenario. Enlargement of Figure 6.5b

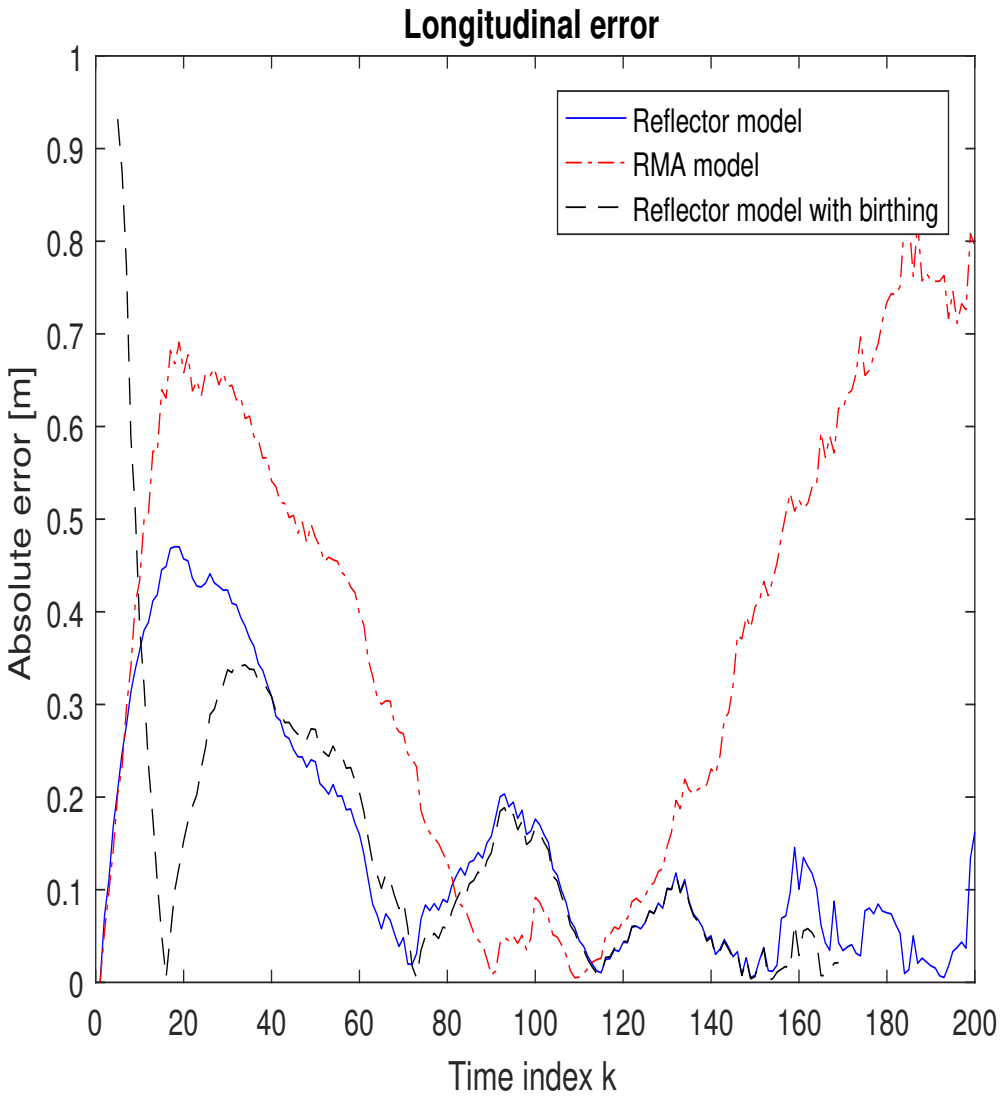


Figure A.11: Longitudinal error, turning scenario. Enlargement of Figure 6.7a

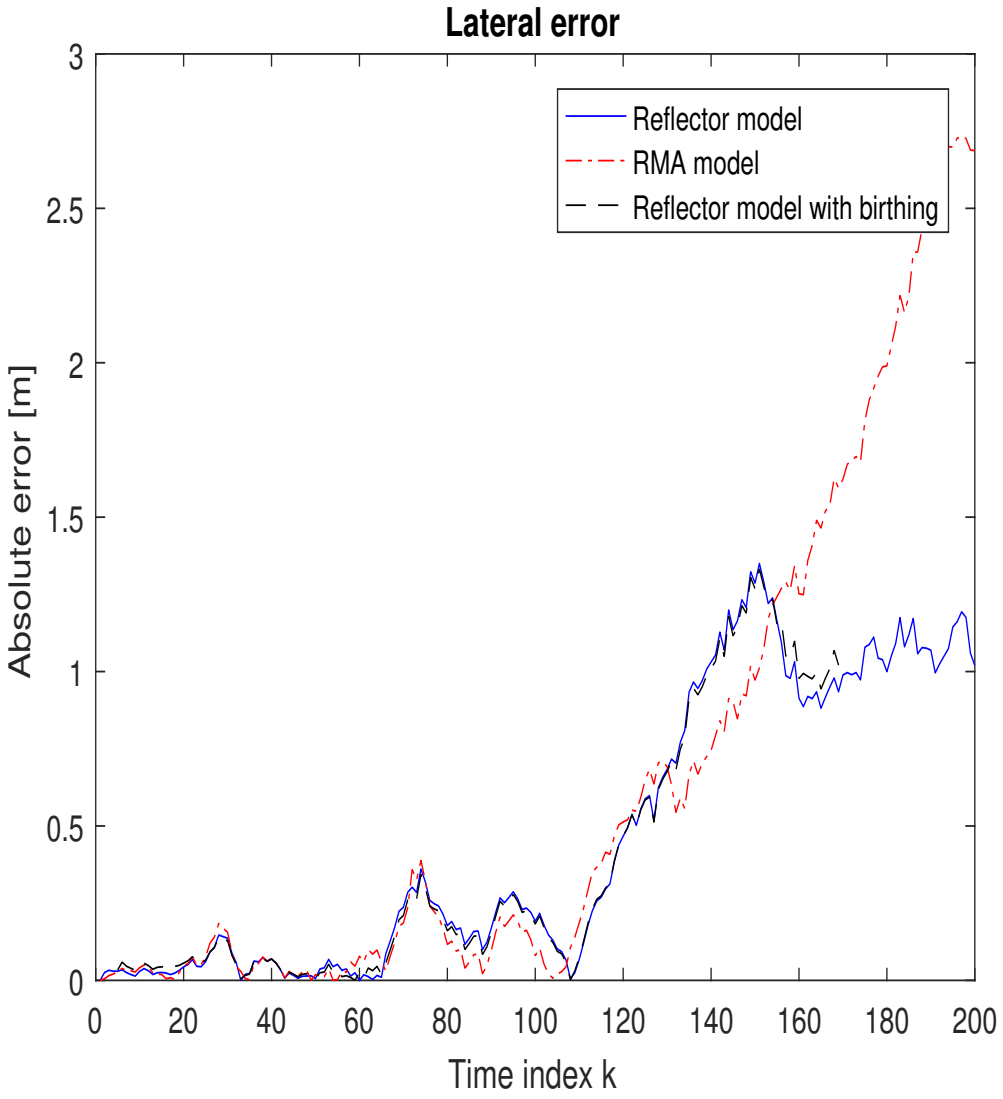


Figure A.12: Lateral error, turning scenario. Enlargement of Figure 6.7b

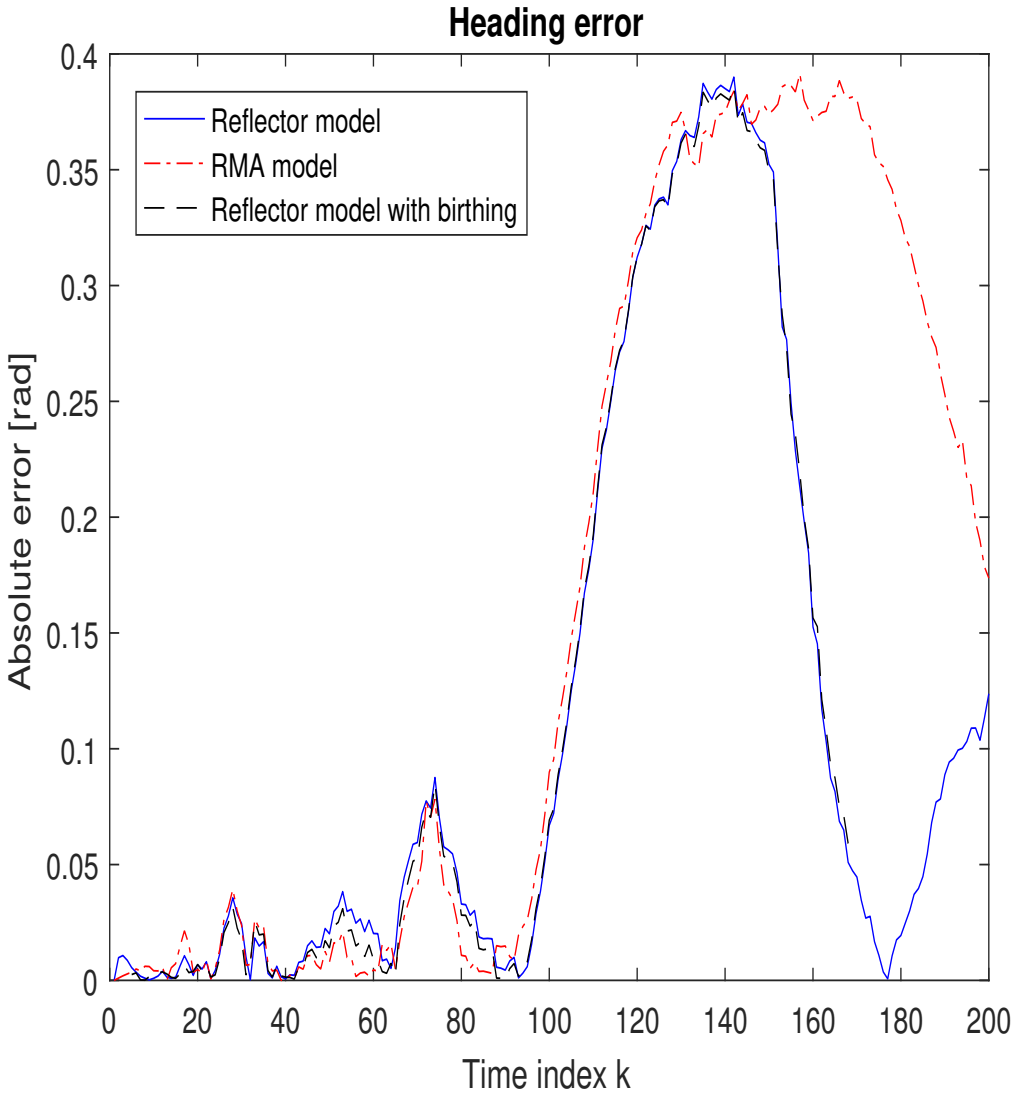


Figure A.13: Heading error, turning scenario. Enlargement of Figure 6.7c

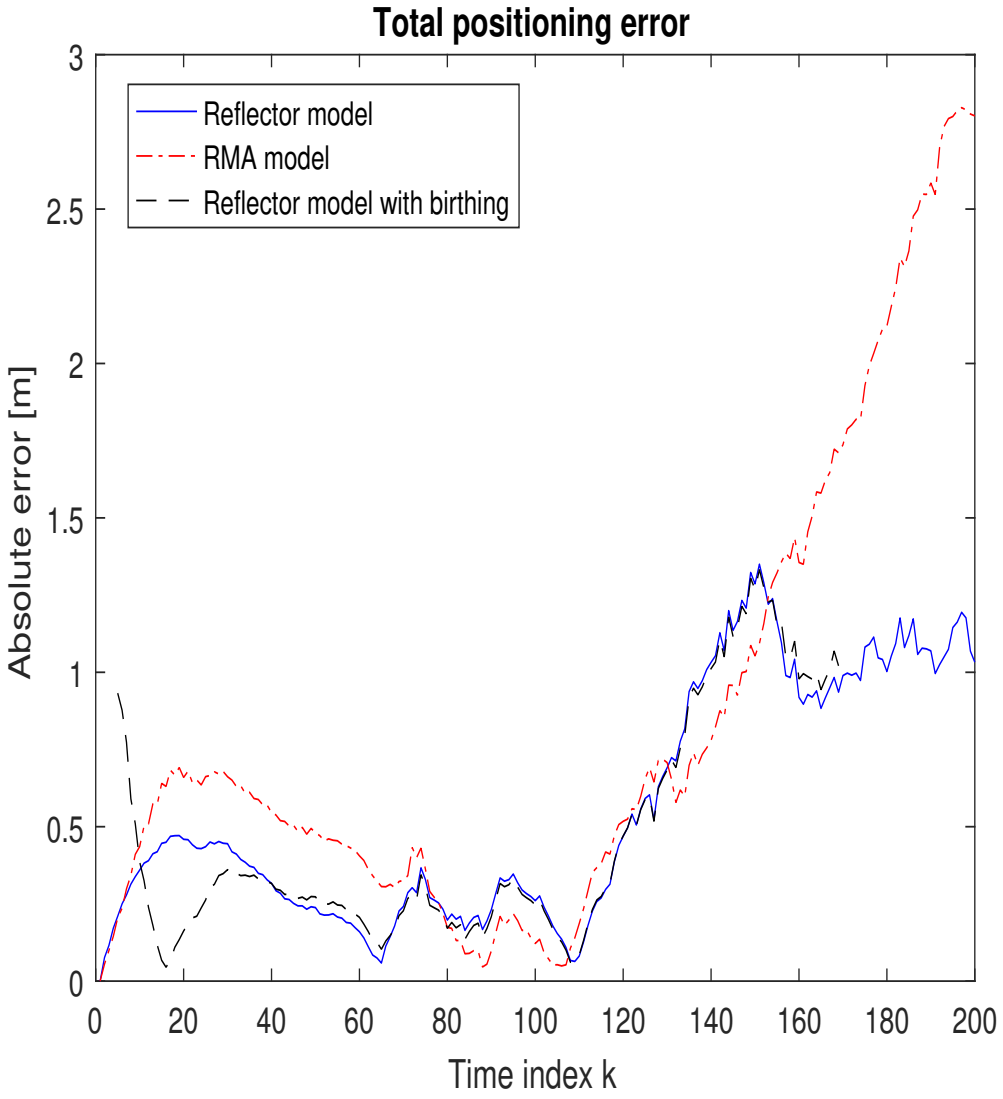


Figure A.14: Total positioning error, turning scenario. Enlargement of Figure 6.7d

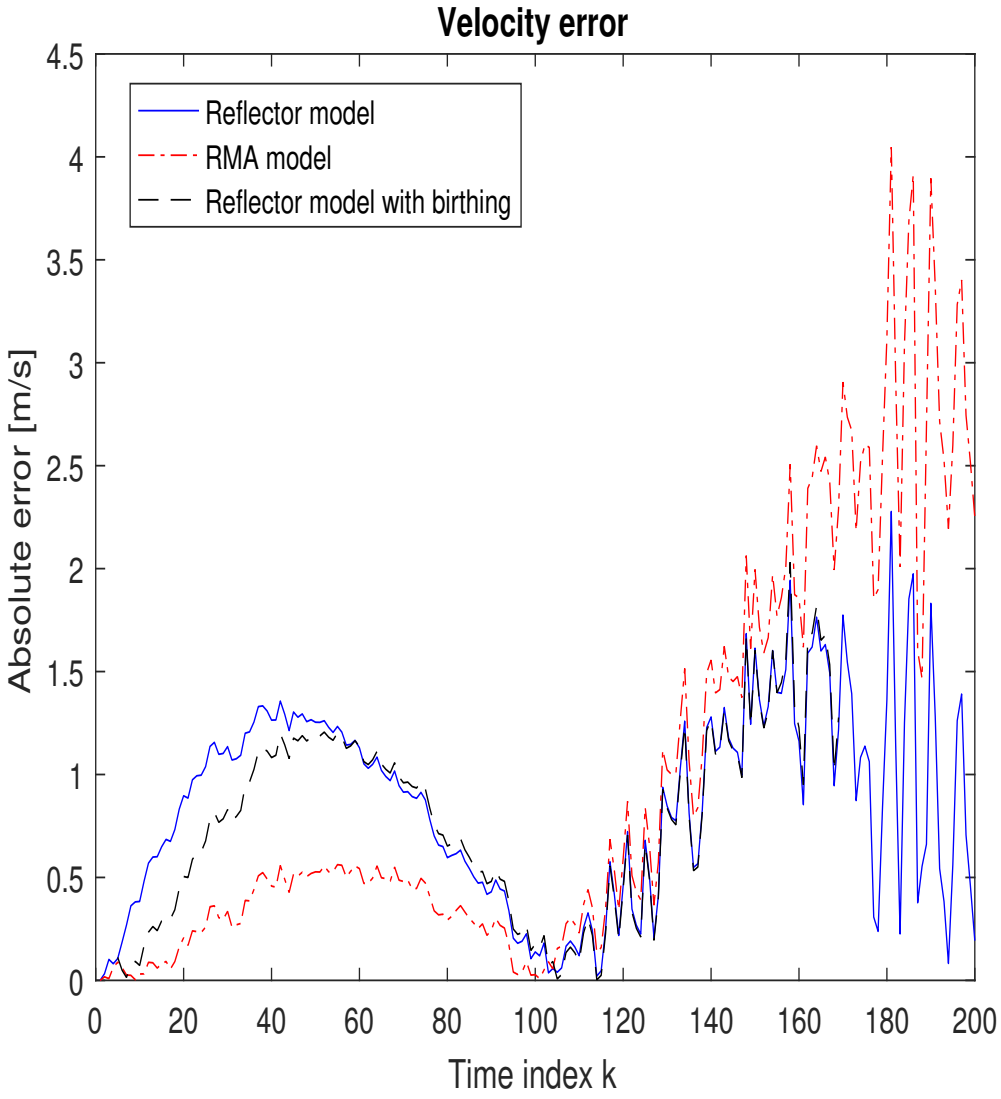


Figure A.15: Velocity error, turning scenario. Enlargement of Figure 6.8a

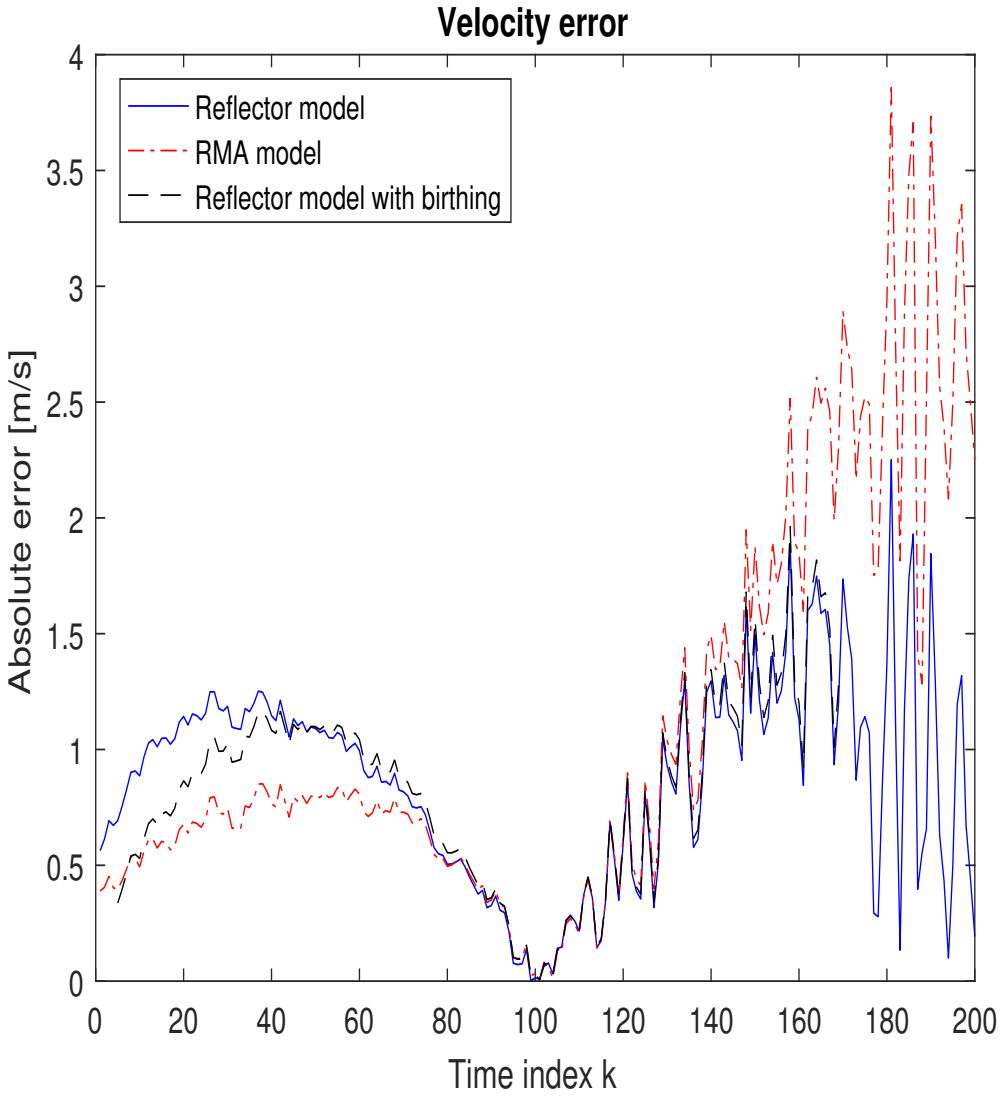


Figure A.16: Velocity error, after smoothing, turning scenario. Enlargement of Figure 6.8b

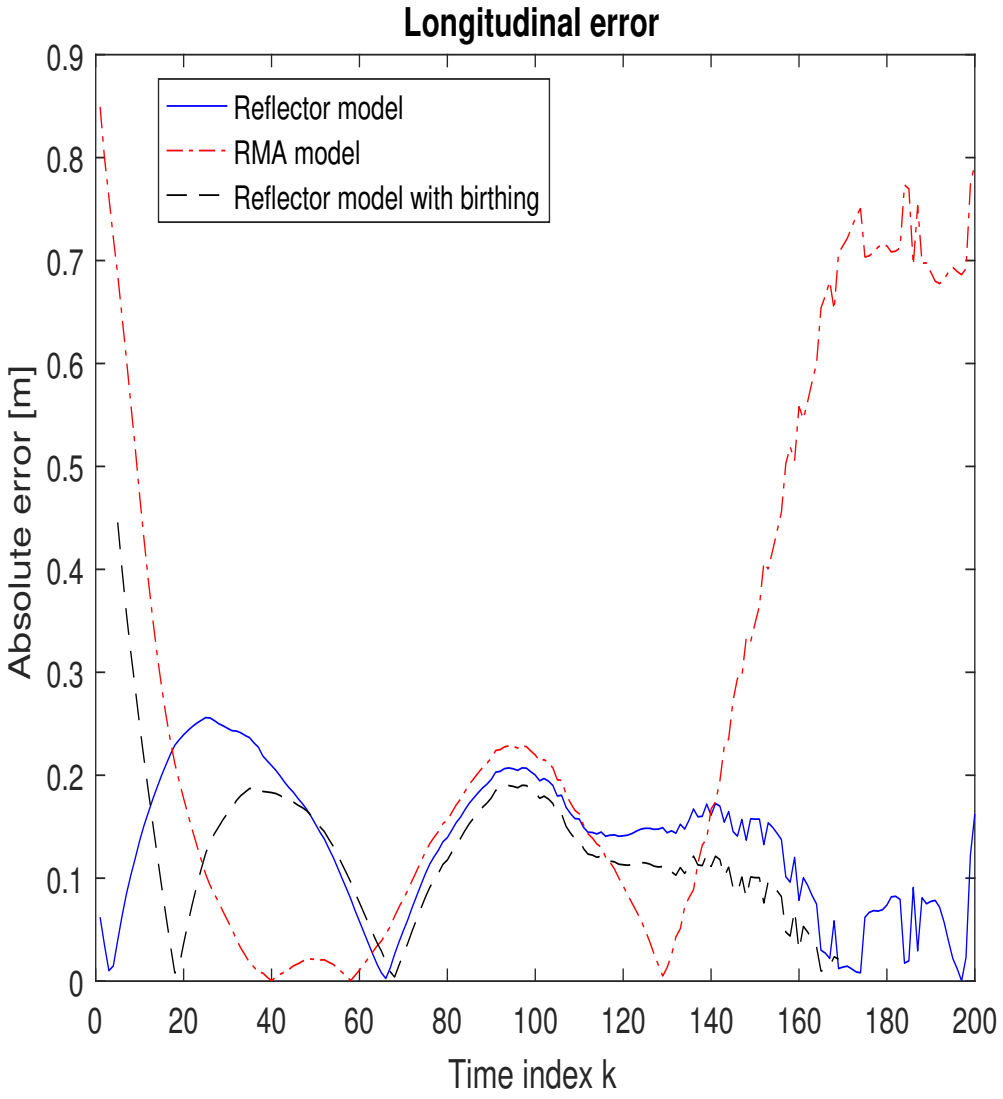


Figure A.17: Velocity error, after smoothing, turning scenario. Enlargement of Figure 6.10a

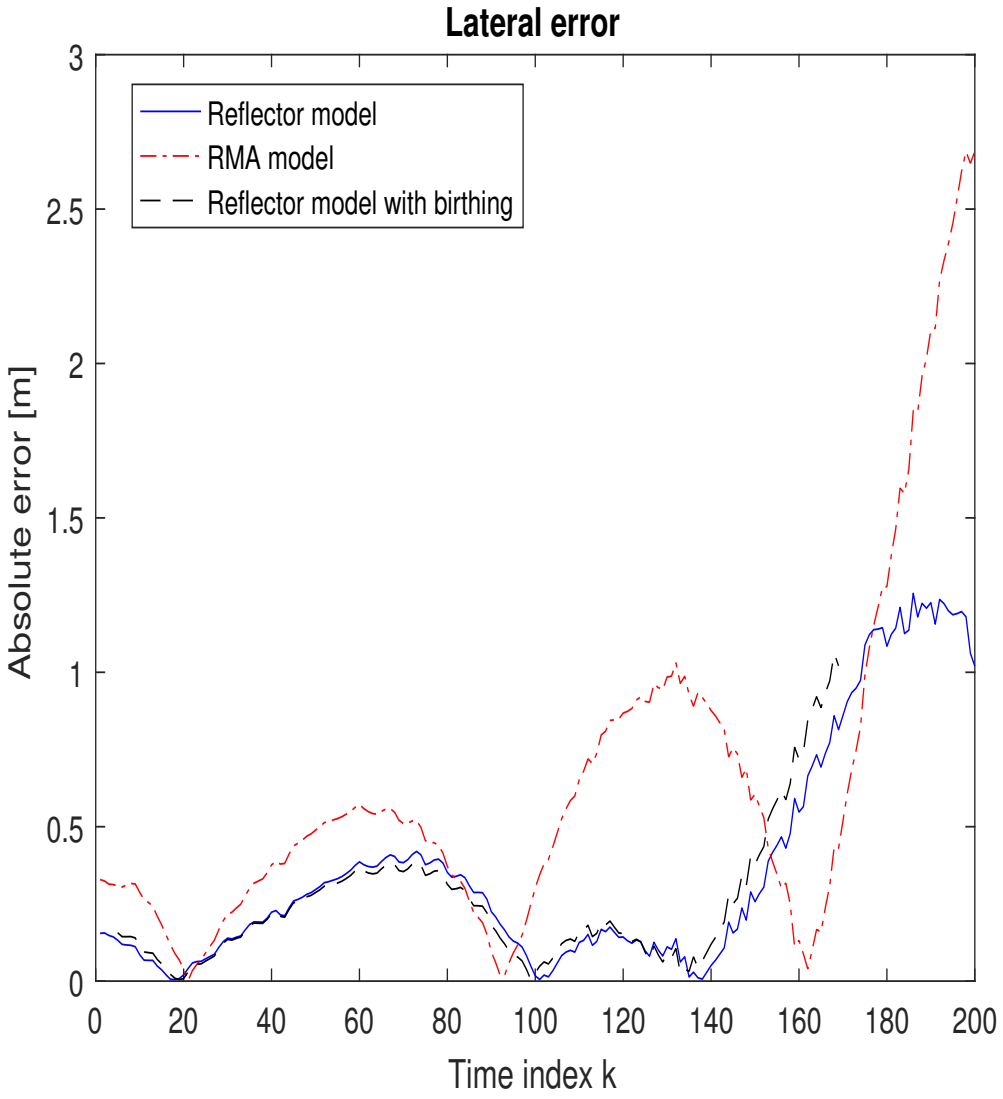


Figure A.18: Velocity error, after smoothing, turning scenario. Enlargement of Figure 6.10b

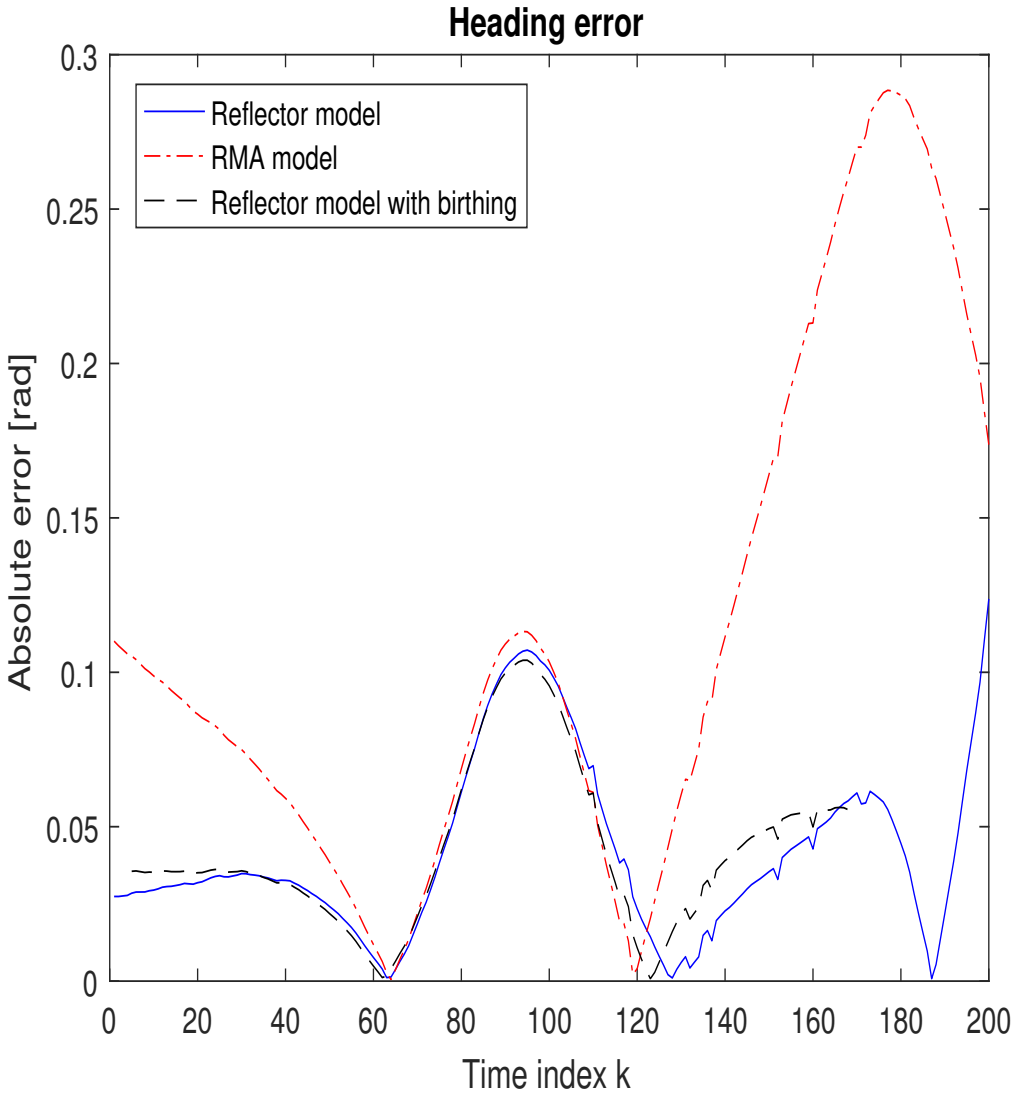


Figure A.19: Heading error, after smoothing, turning scenario. Enlargement of Figure 6.10c

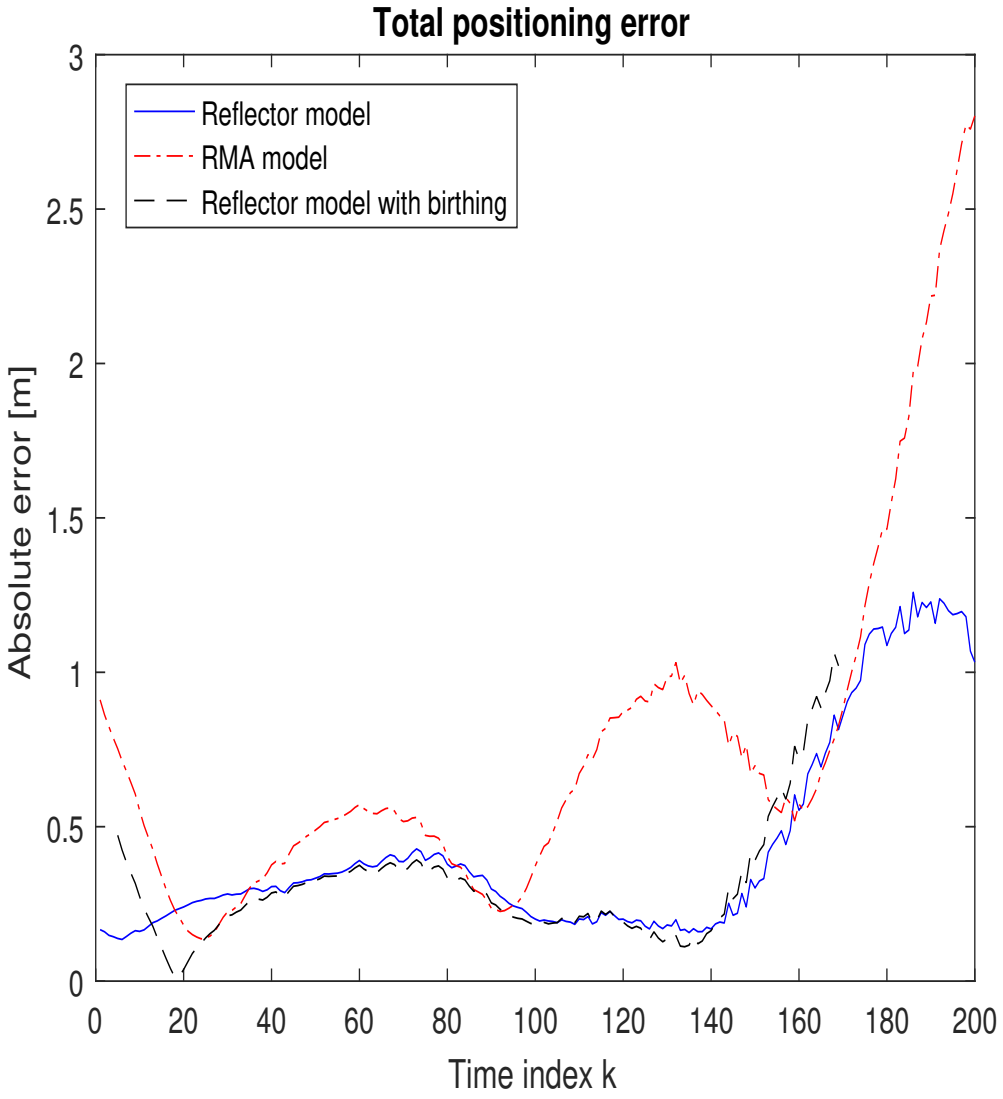


Figure A.20: Total positioning error, after smoothing, turning scenario. Enlargement of Figure 6.10d

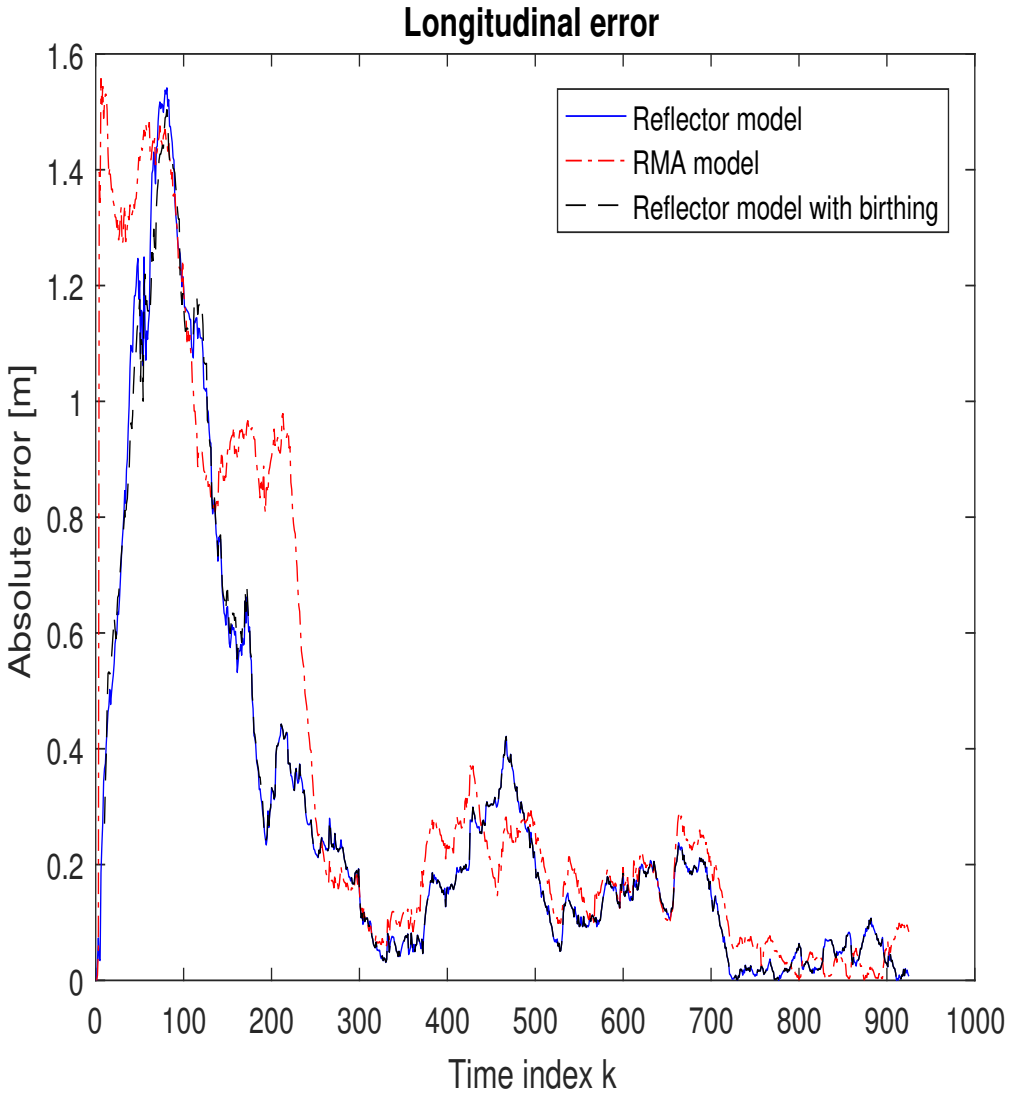


Figure A.21: Longitudinal error, overtaking scenario. Enlargement of Figure 6.12a

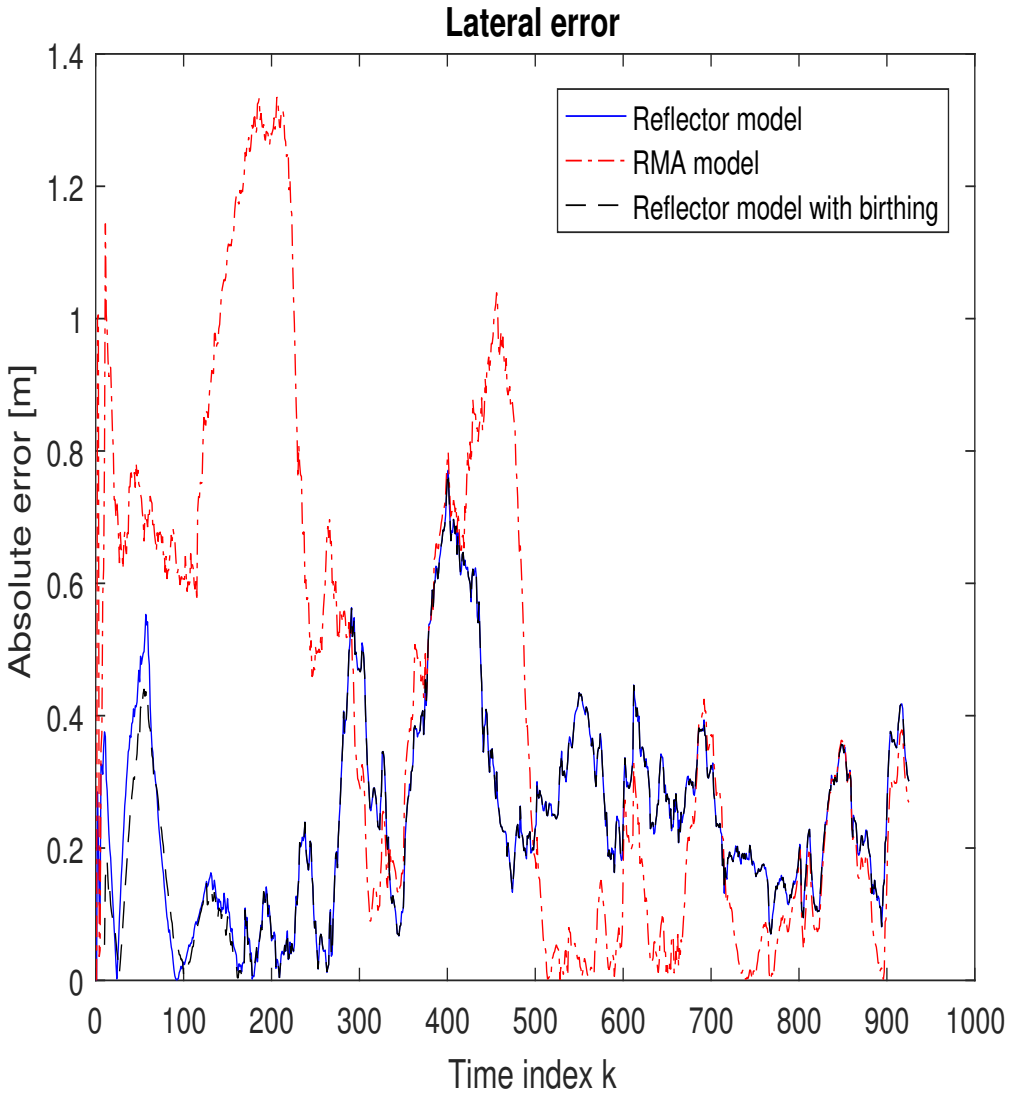


Figure A.22: Lateral error, overtaking scenario. Enlargement of Figure 6.12b

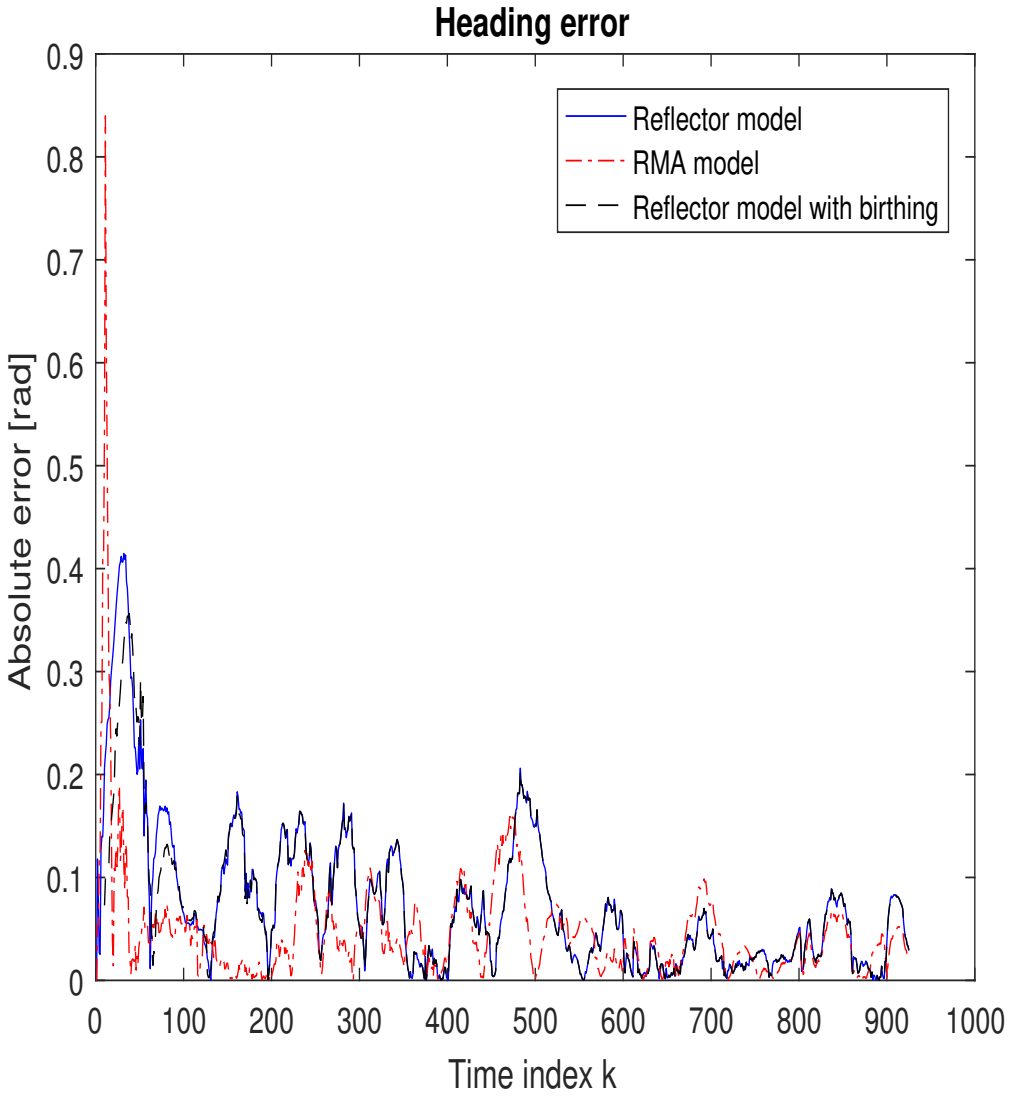


Figure A.23: Heading error, overtaking scenario. Enlargement of Figure 6.12c

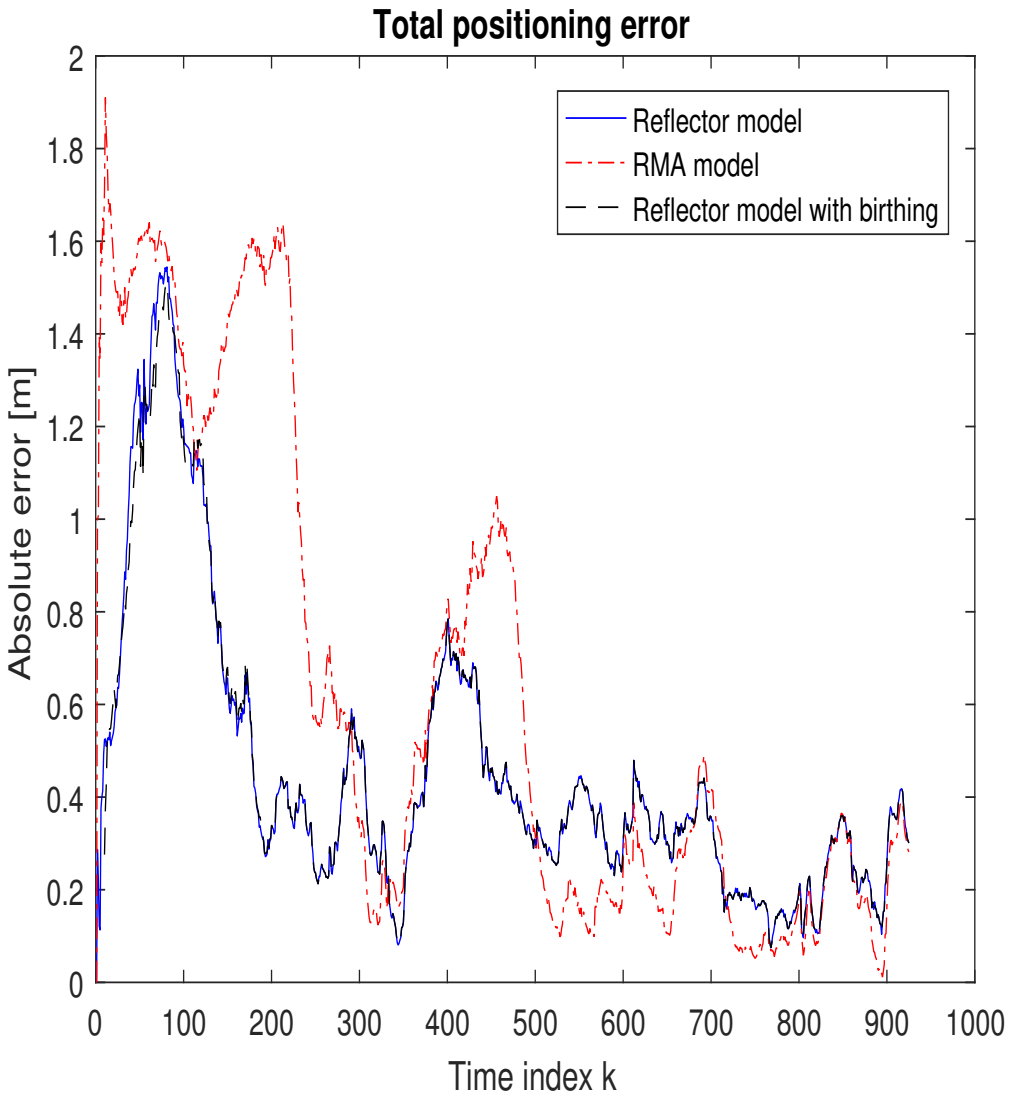


Figure A.24: Velocity error, Overtaking scenario. Enlargement of Figure 6.12d

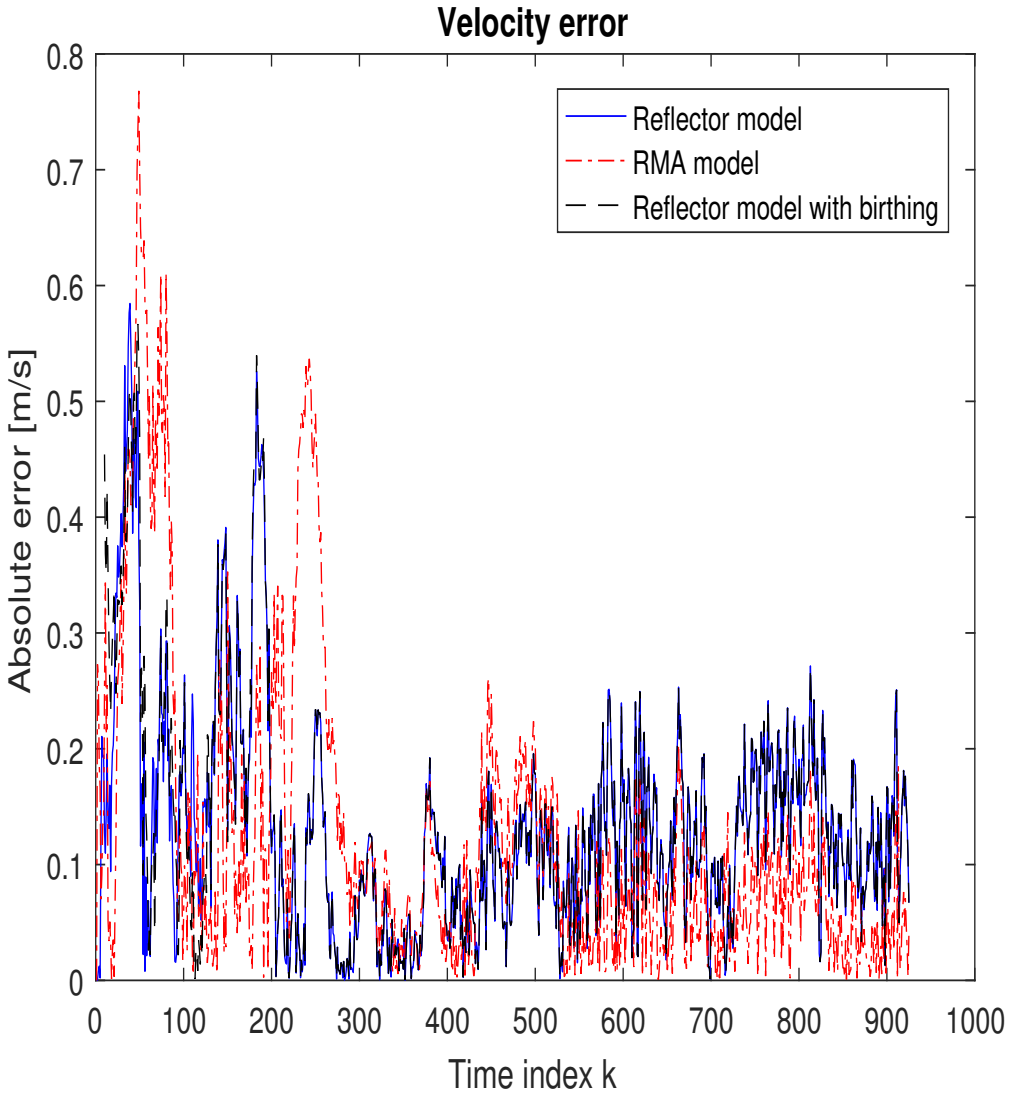


Figure A.25: Velocity error, overtaking scenario. Enlargement of Figure 6.13a

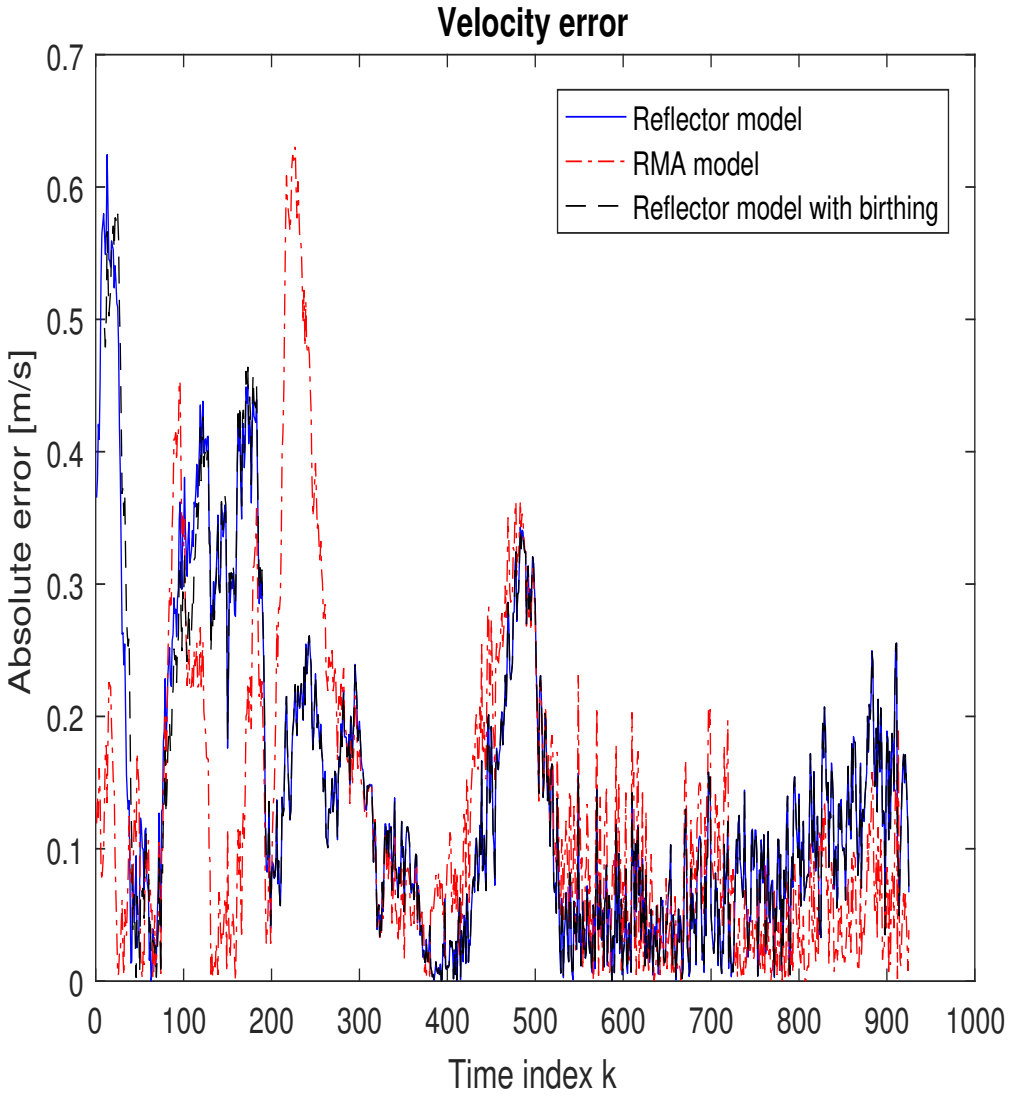


Figure A.26: Velocity error after smoothing, overtaking scenario. Enlargement of Figure 6.13b

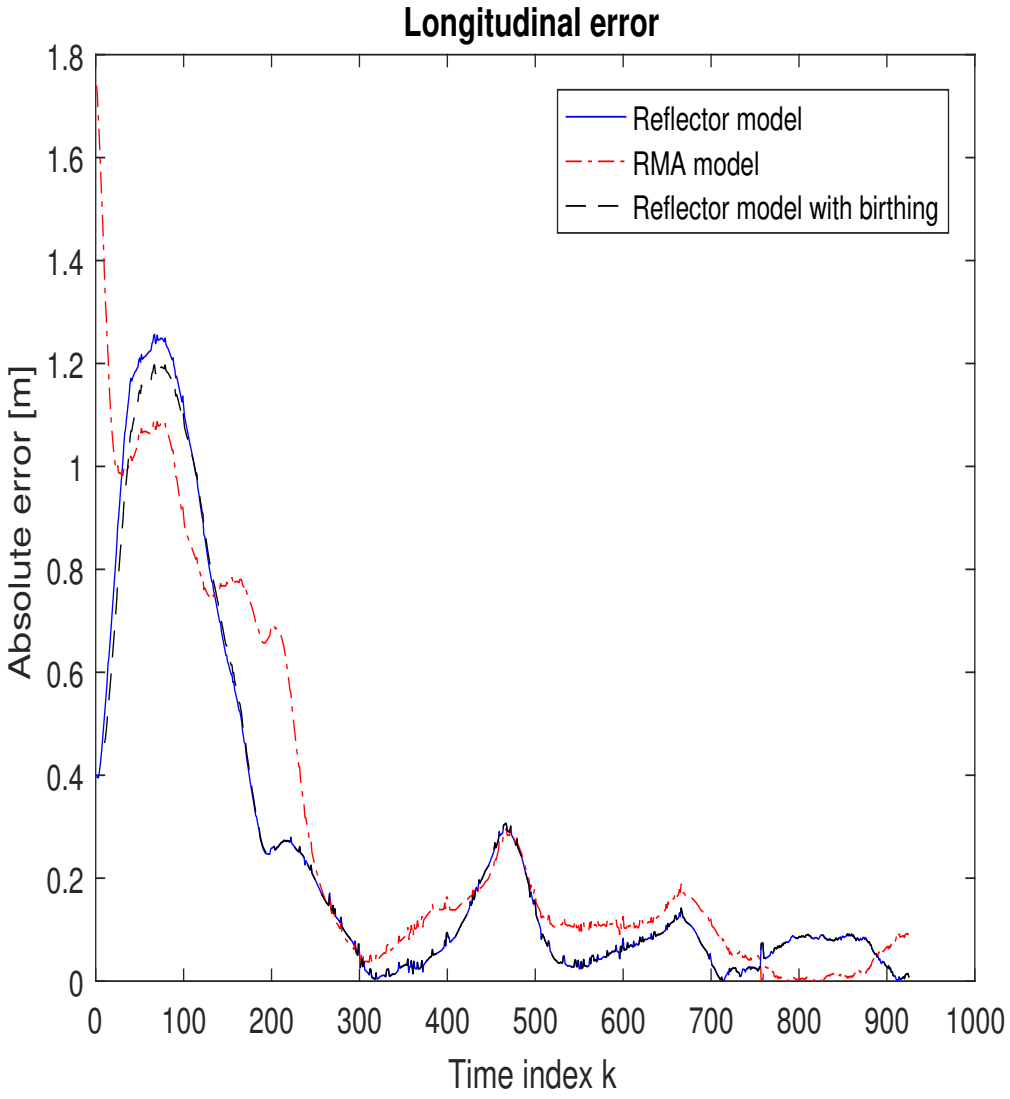


Figure A.27: Longitudinal error, after smoothing, overtaking scenario. Enlargement of Figure 6.15a

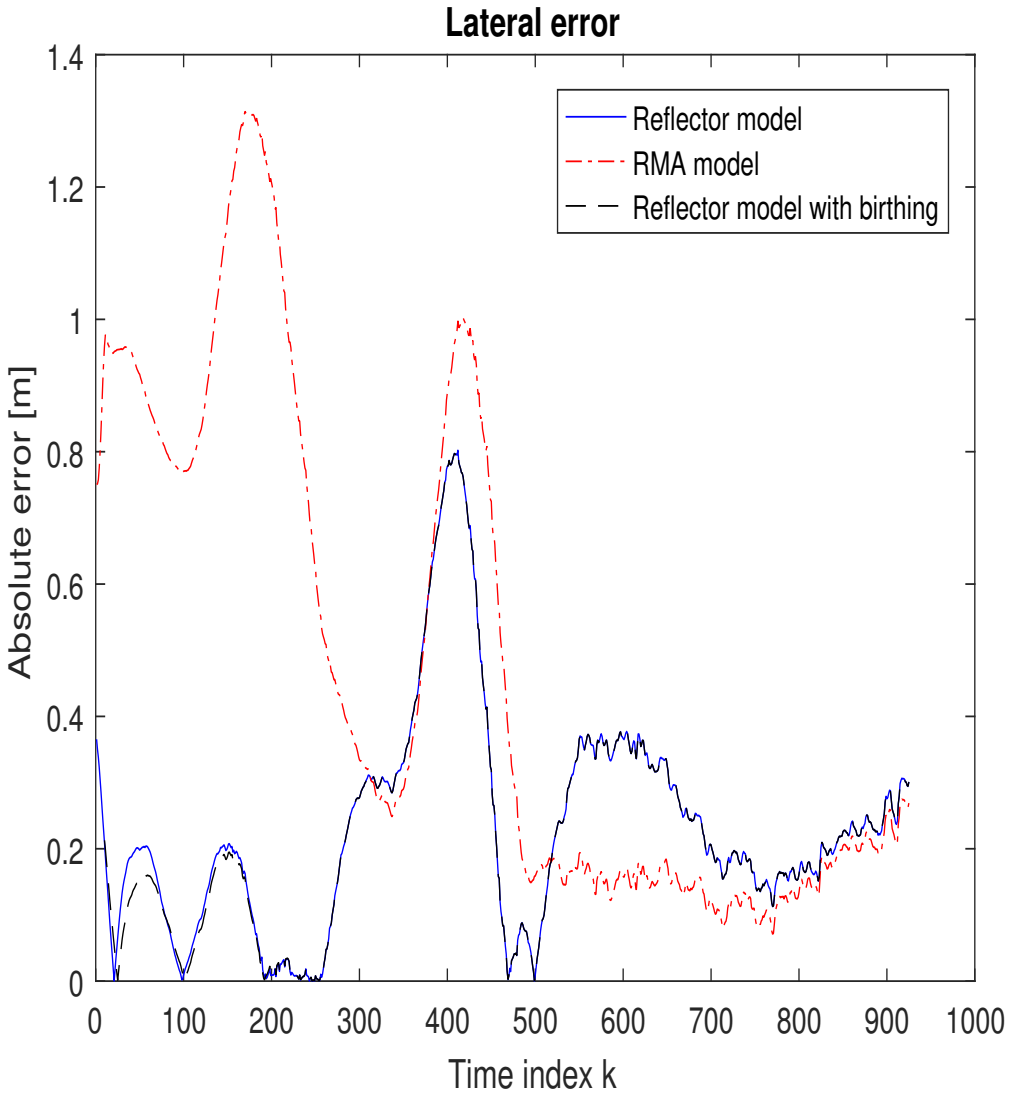


Figure A.28: Lateral error, after smoothing, overtaking scenario. Enlargement of Figure 6.15b

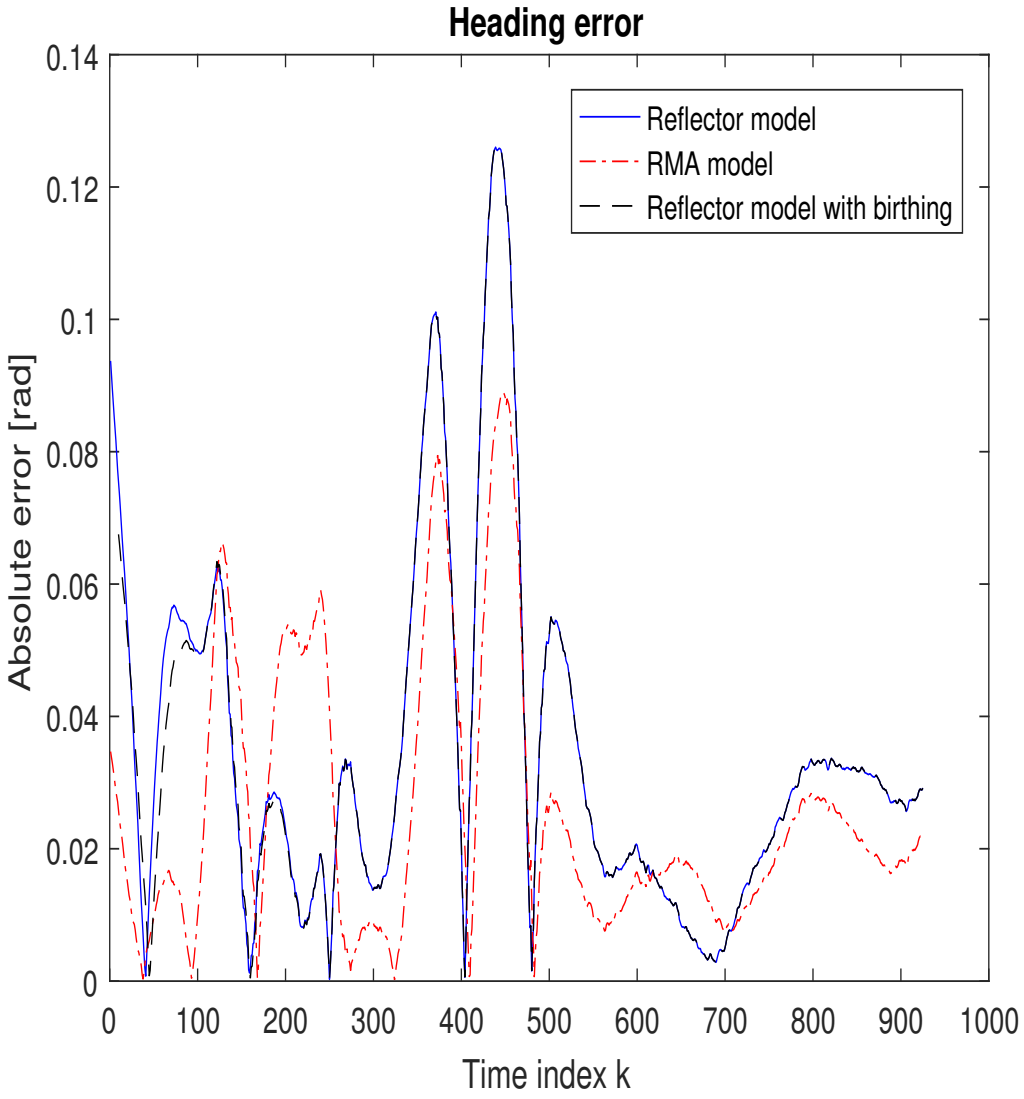


Figure A.29: Heading error while, after smoothing, overtaking scenario. Enlargement of Figure 6.15c

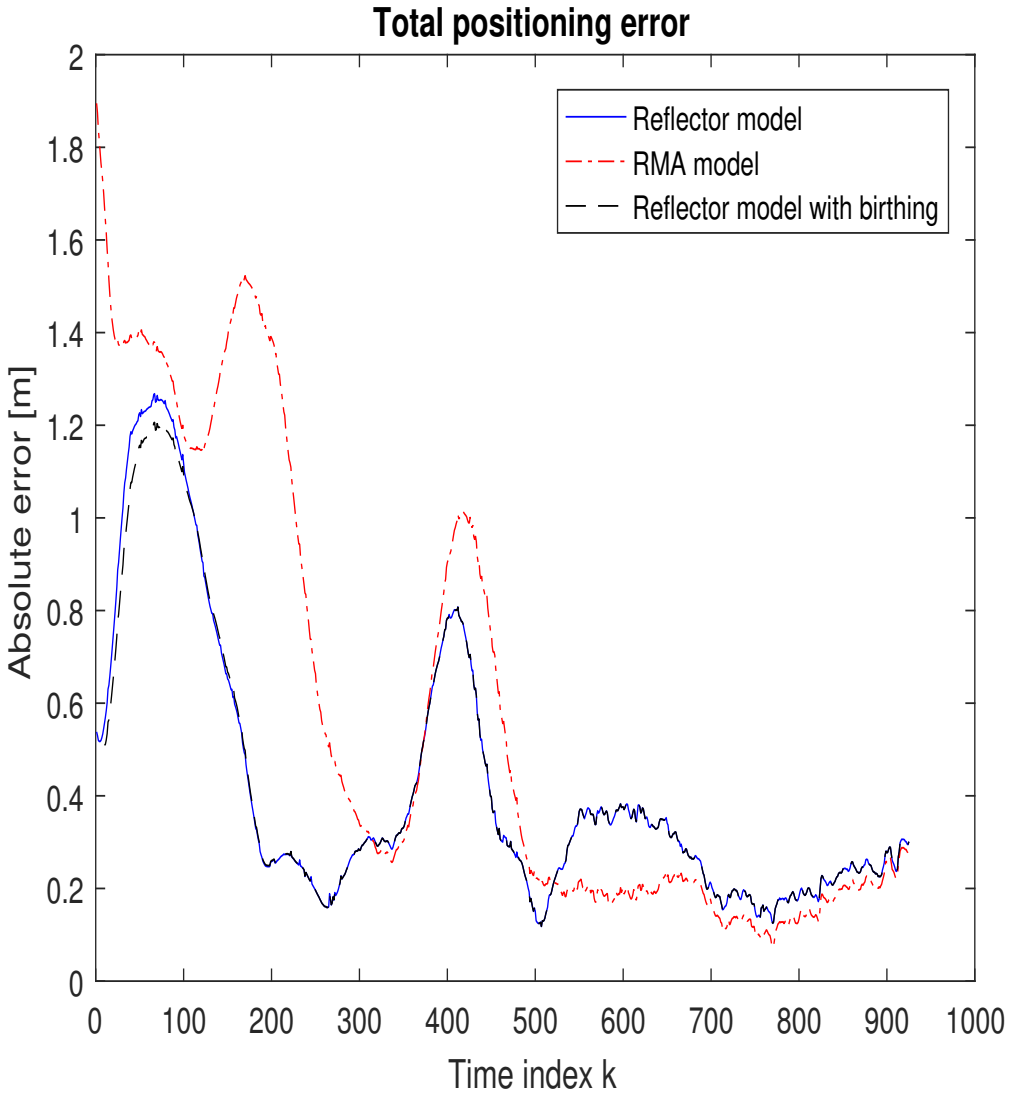


Figure A.30: Total positioning error, after smoothing, overtaking scenario. Enlargement of Figure 6.15d

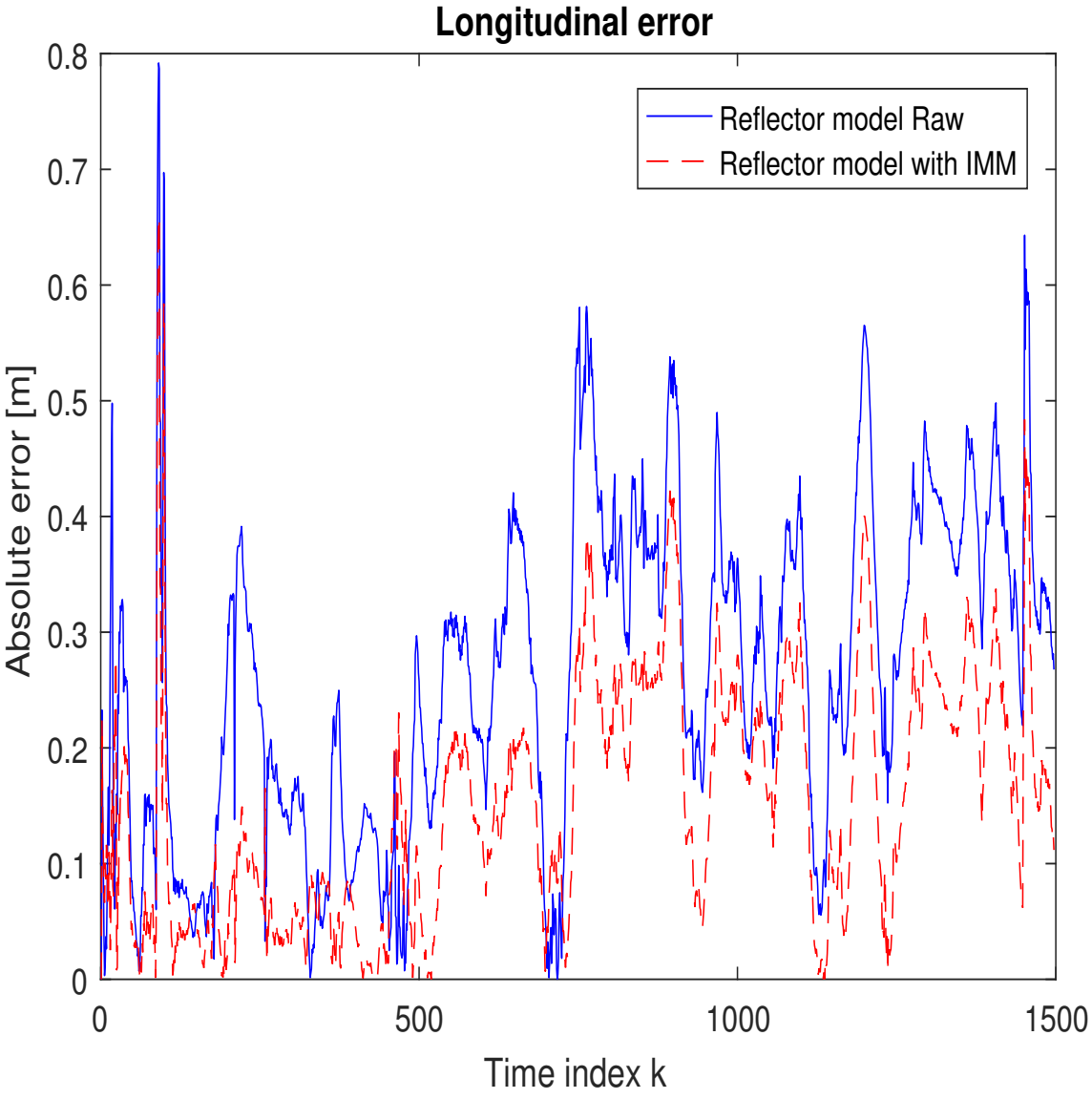


Figure A.31: Longitudinal error, guardrail scenario. Enlargement of Figure 6.17a

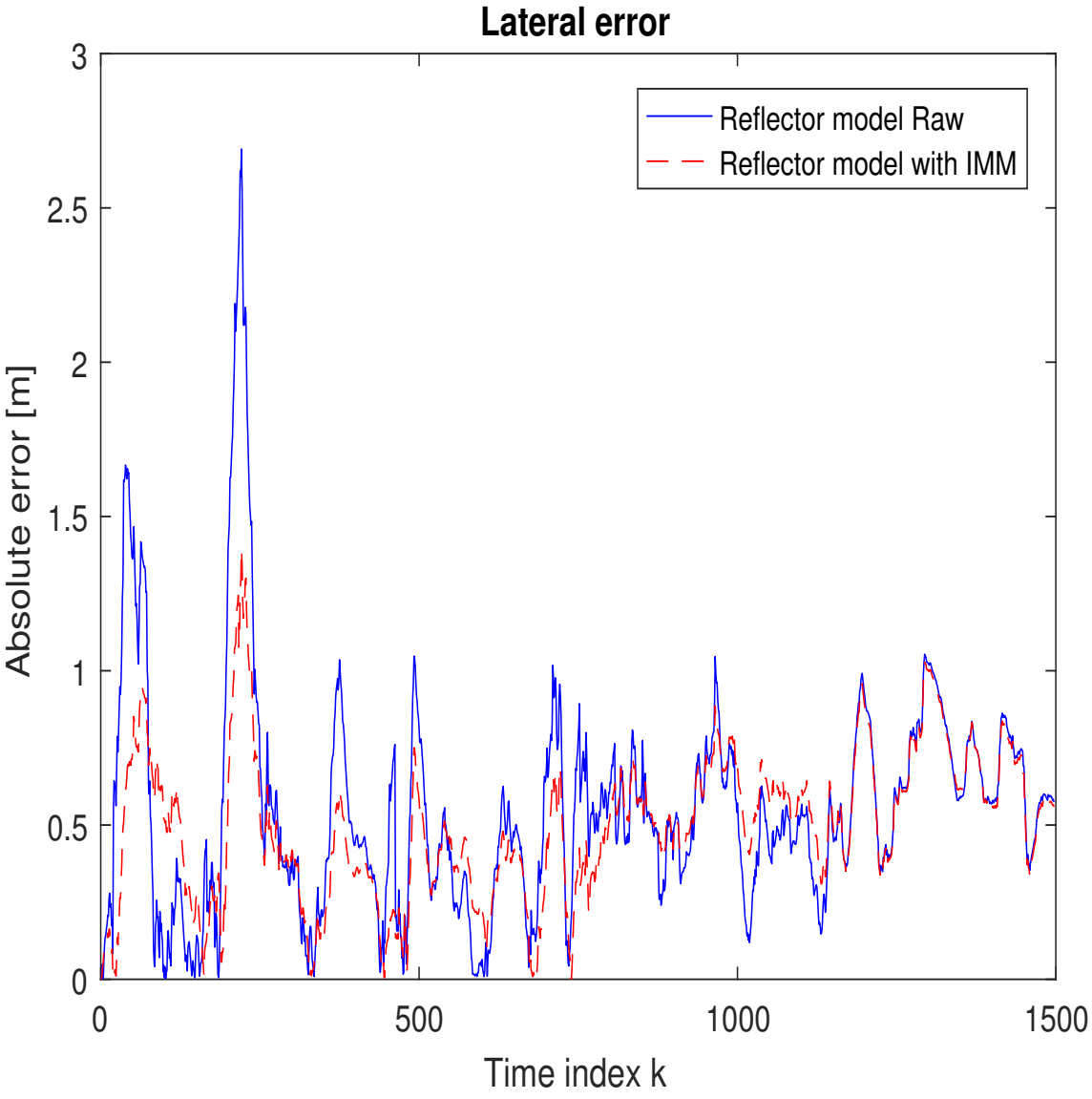


Figure A.32: Lateral error, guardrail scenario. Enlargement of Figure 6.17b

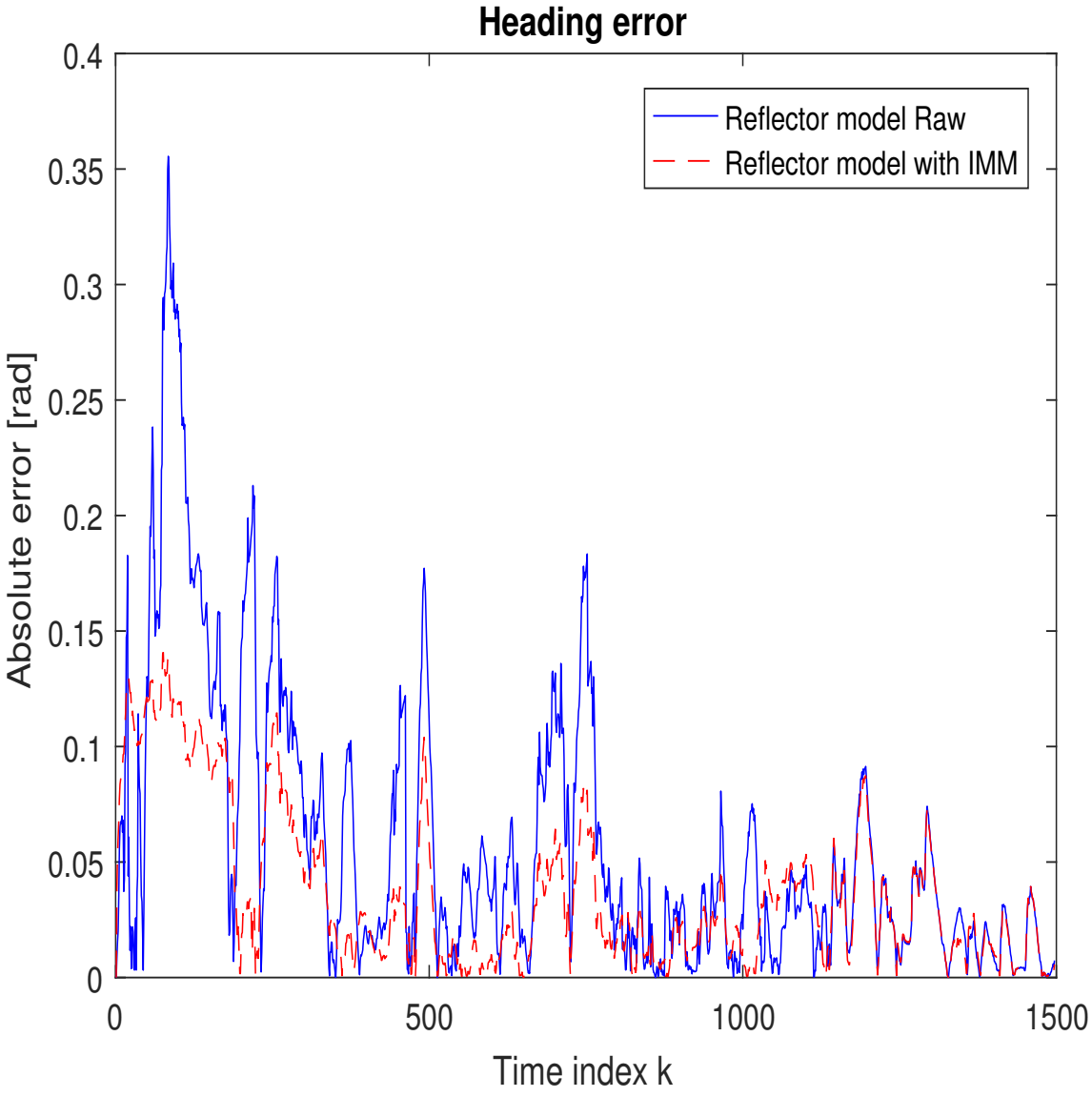


Figure A.33: Heading error, guardrail scenario. Enlargement of Figure 6.17c

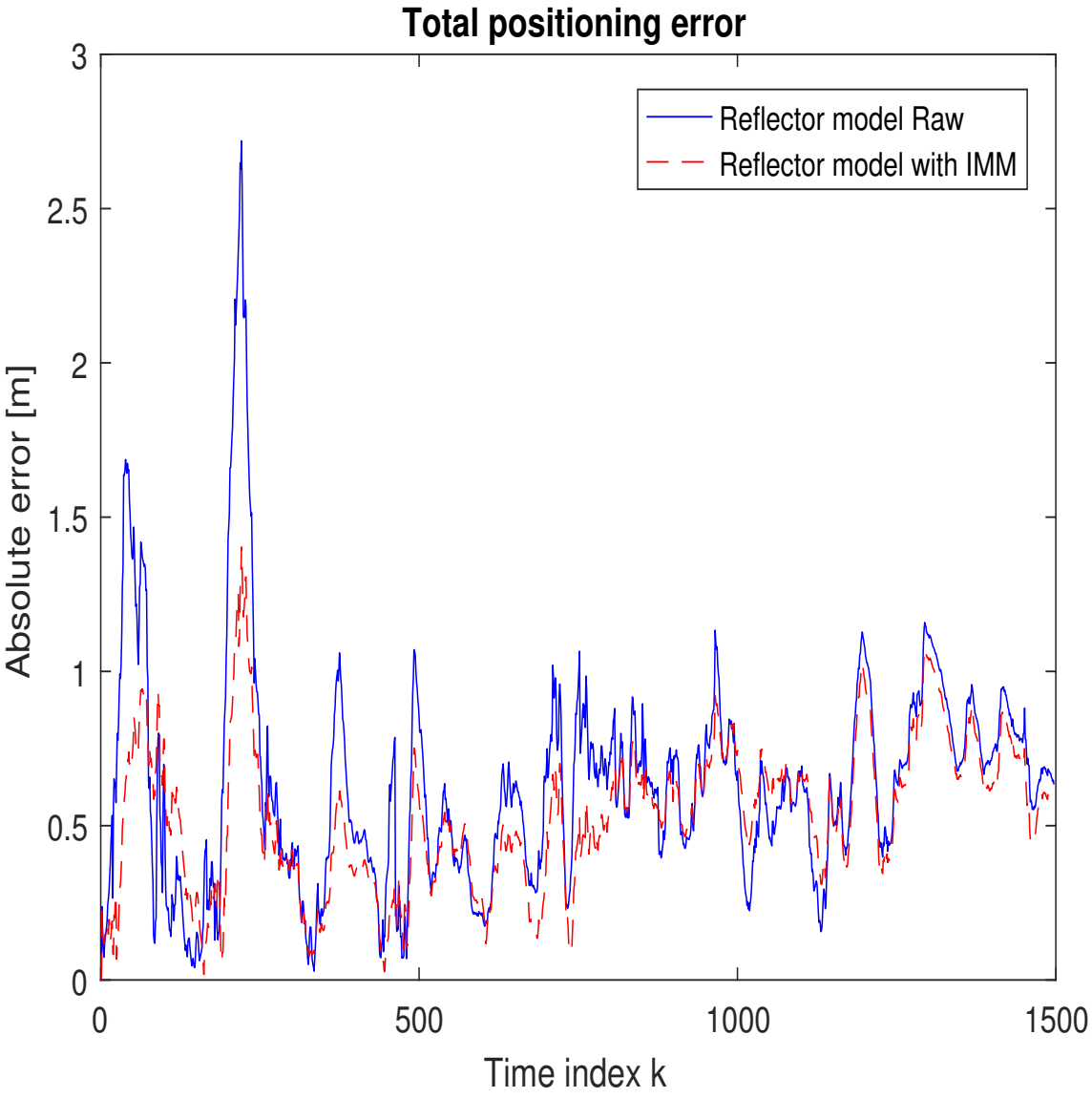


Figure A.34: Total positioning error, guardrail scenario. Enlargement of Figure 6.17d

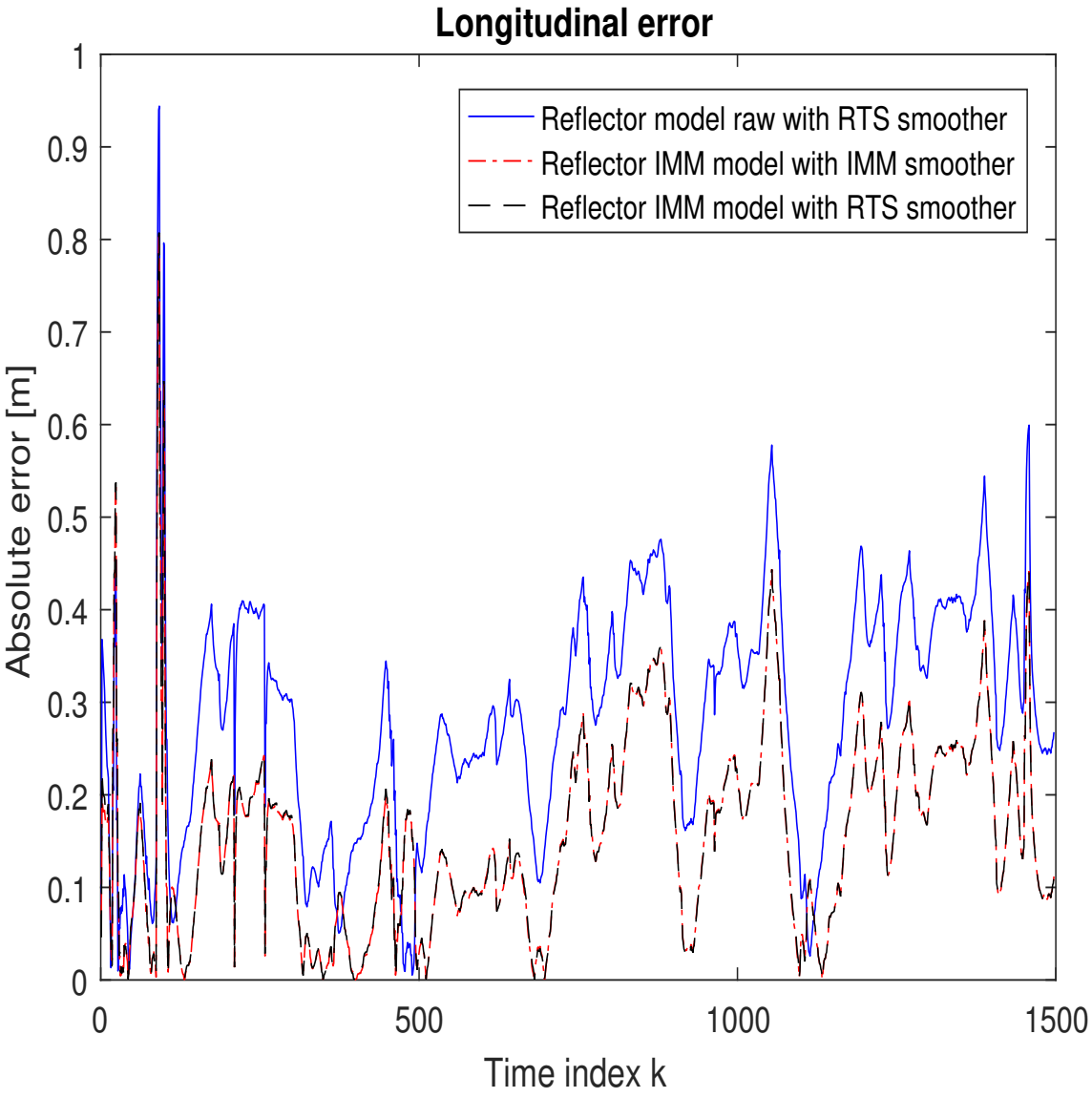


Figure A.35: Longitudinal error, after smoothing, guardrail scenario. Enlargement of Figure 6.20a

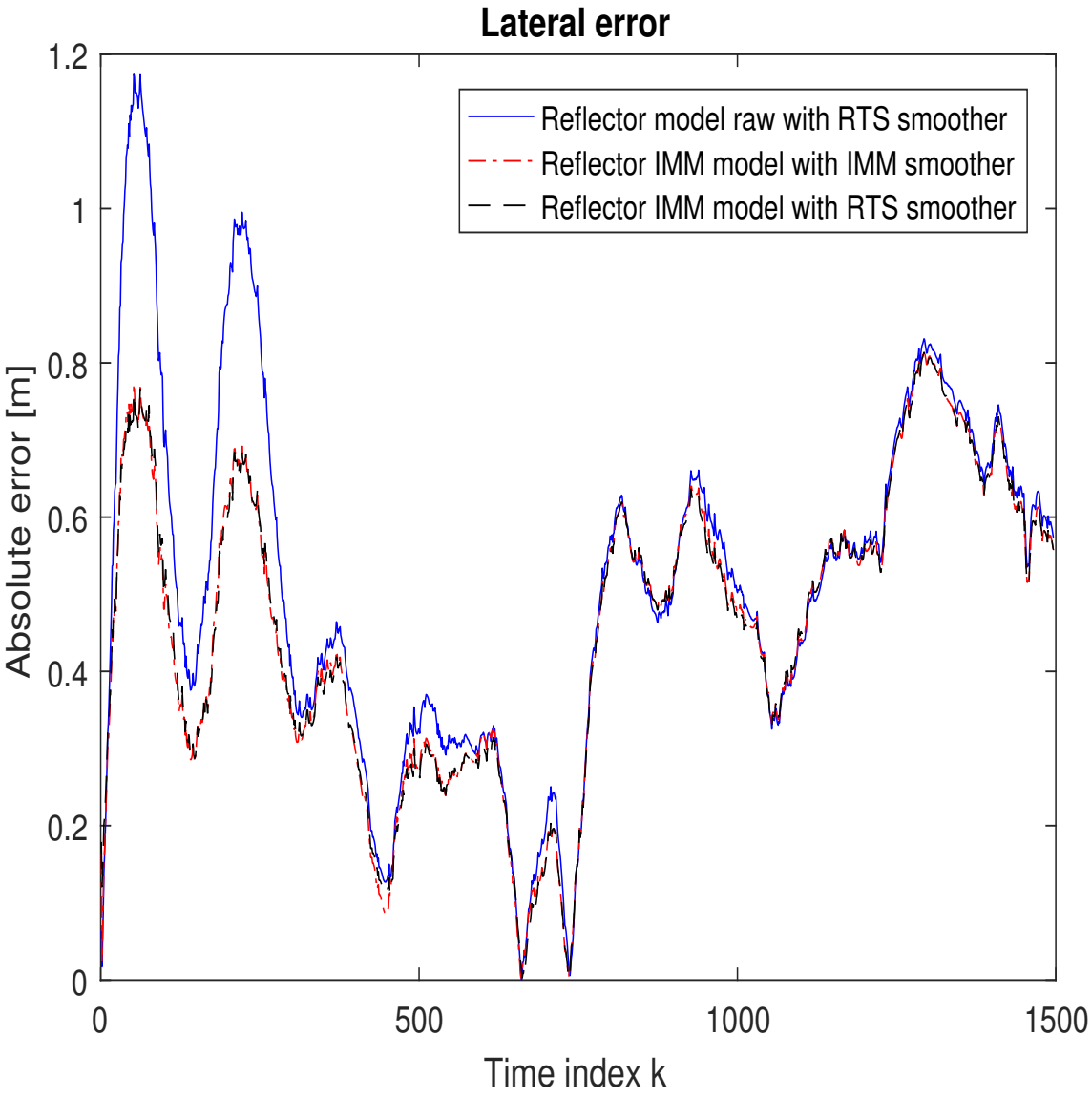


Figure A.36: Lateral error, after smoothing, guardrail scenario. Enlargement of Figure 6.20b

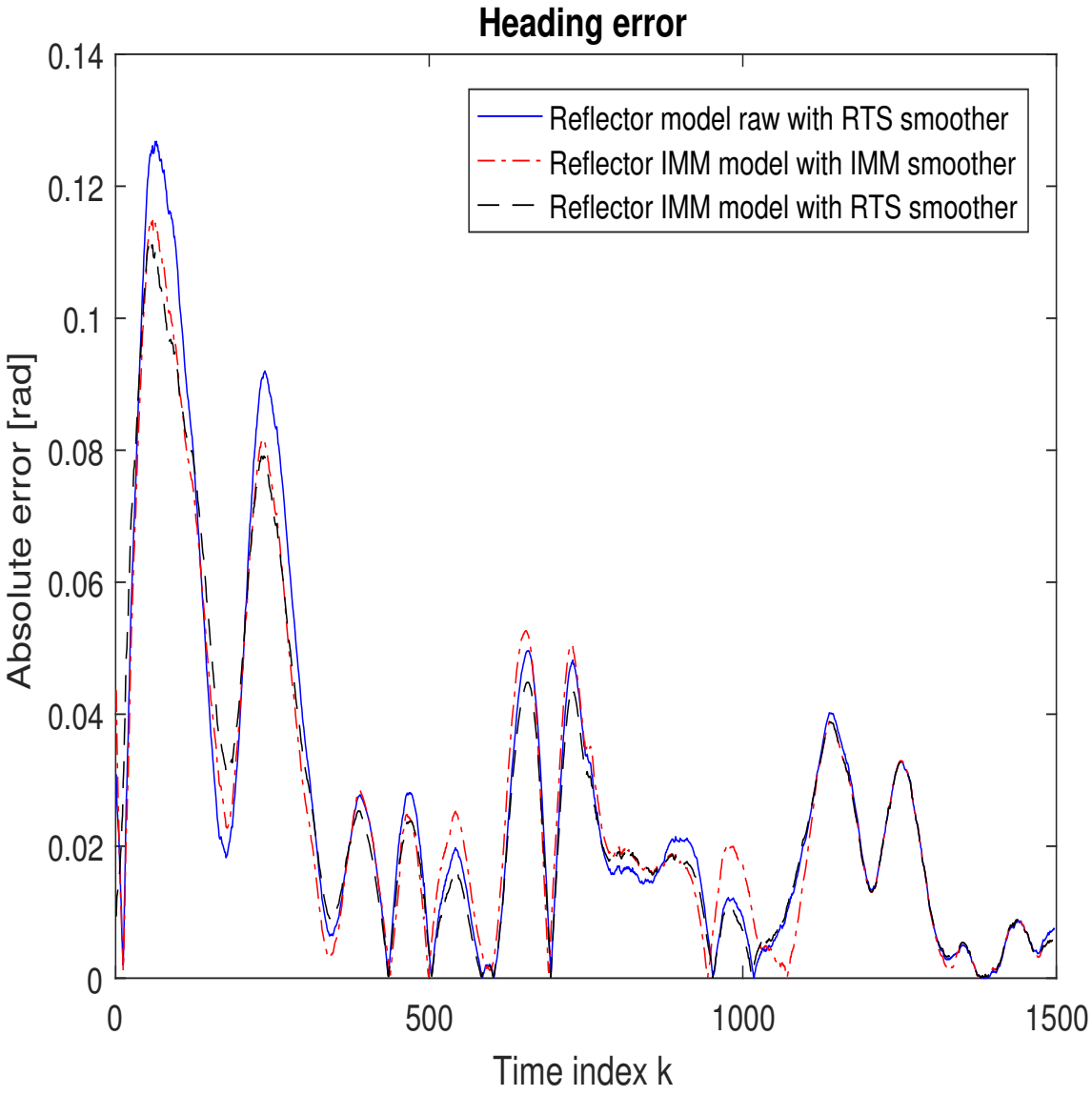


Figure A.37: Heading error, after smoothing, guardrail scenario. Enlargement of Figure 6.20c

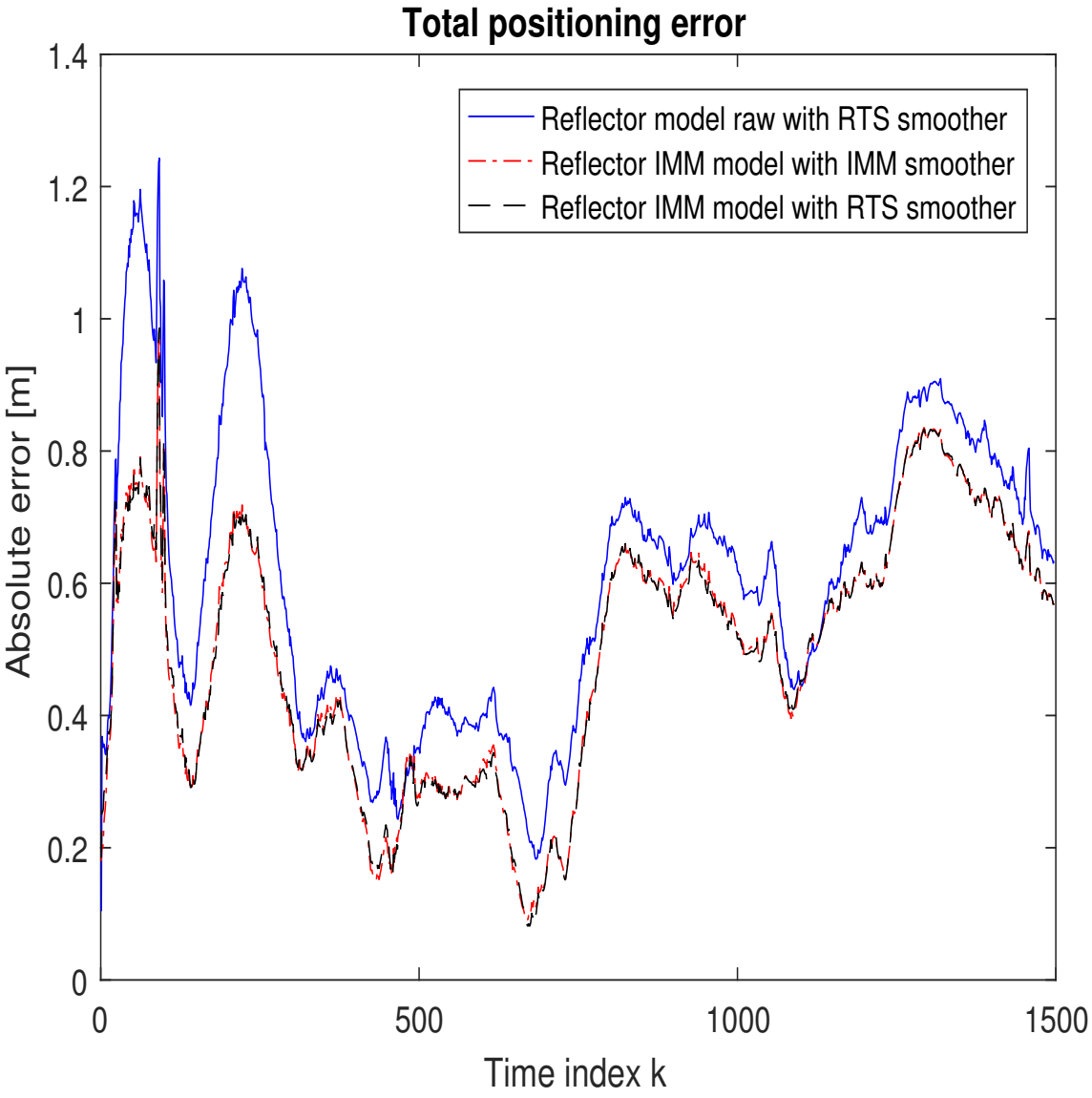


Figure A.38: Total positioning error, after smoothing, guardrail scenario. Enlargement of Figure 6.20d

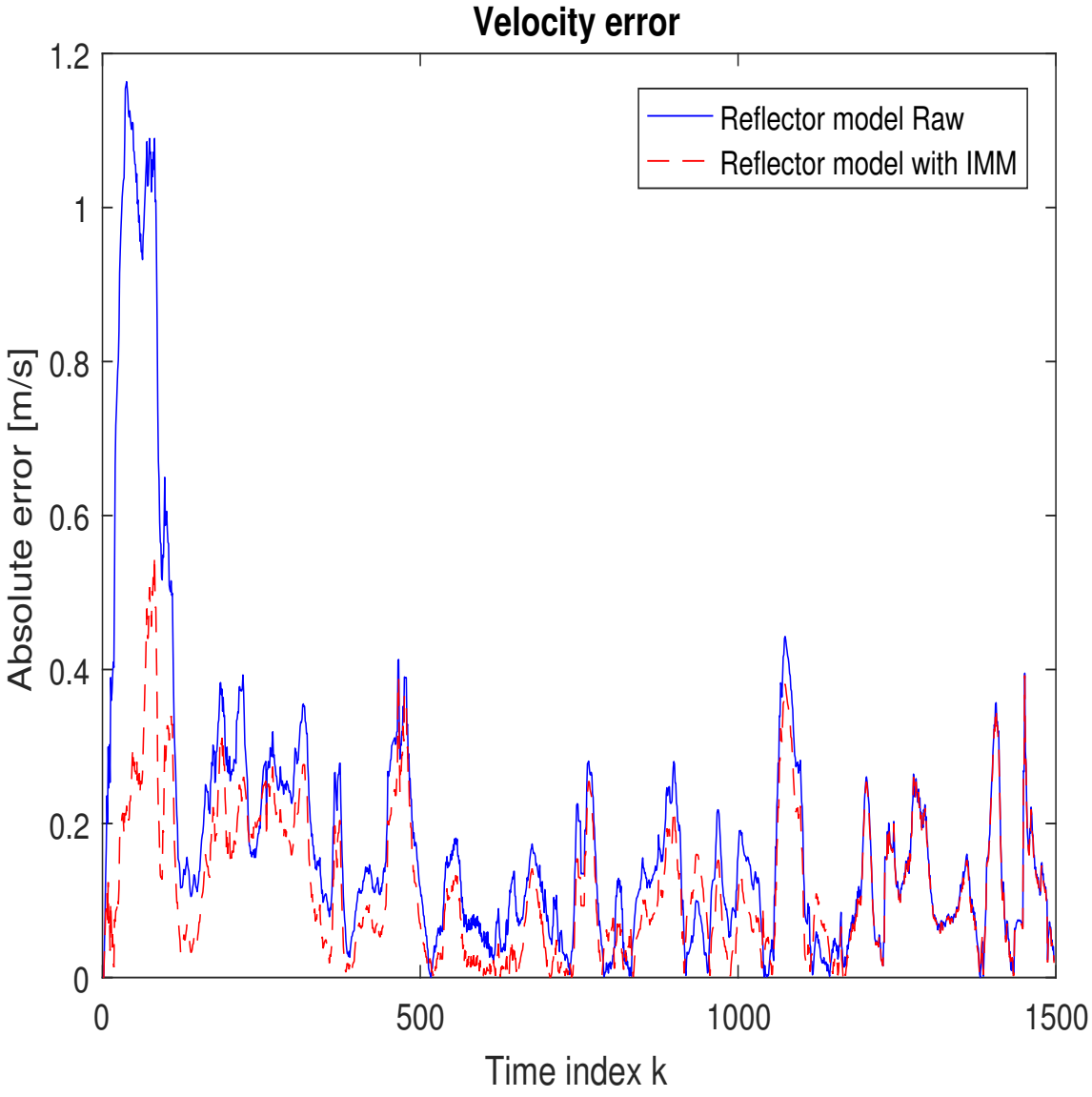


Figure A.39: Velocity error, guardrail scenario. Enlargement of Figure 6.21a

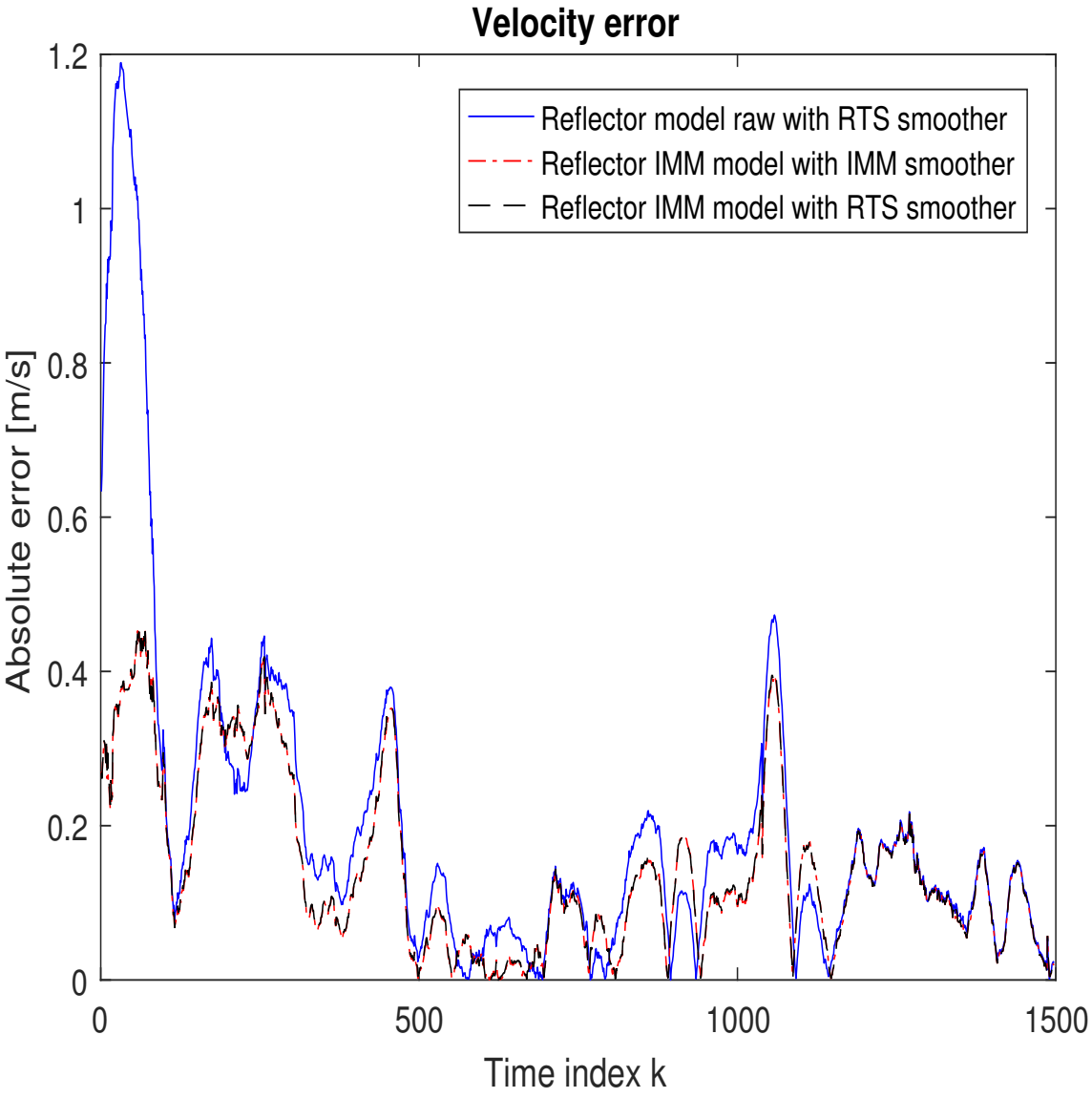


Figure A.40: Velocity error, after smoothing, guardrail scenario. Enlargement of Figure 6.21b