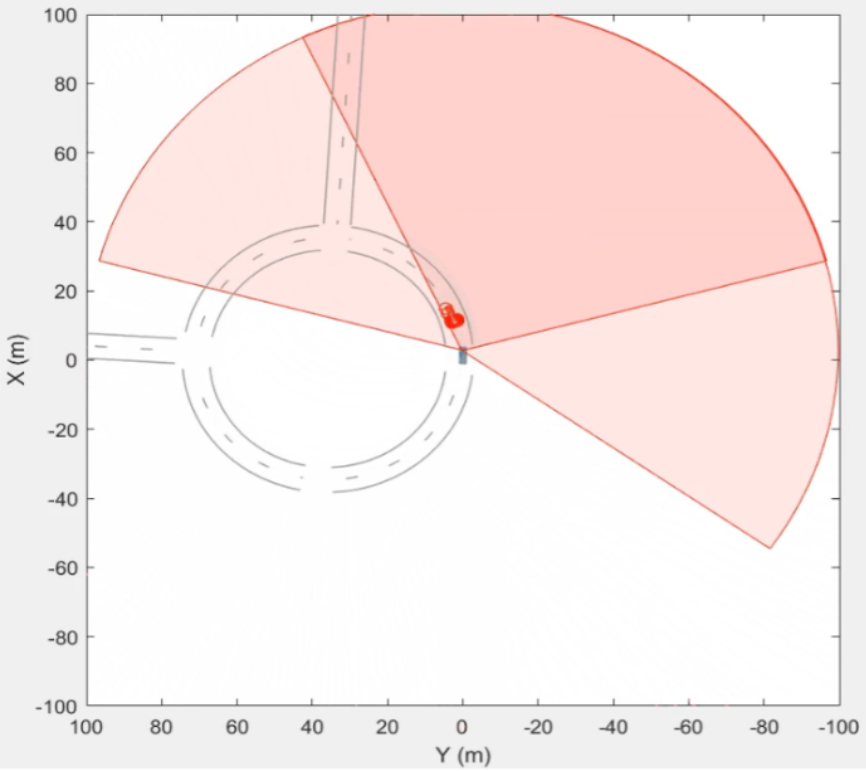

Adaptive Techniques for Tuning of Process Noise in the Kalman filter

Embedded Electronic System Design

Athulya Jose
Shreya Raghunath Banthi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
AND UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Adaptive Techniques for Tuning of Process Noise in the Kalman filter

ATHULYA JOSE
SHREYA RAGHUNATH BANTHI

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Embedded Electronic System Design
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden 2021

Adaptive Techniques for Tuning of Process Noise in the Kalman filter
ATHULYA JOSE
SHREYA RAGHUNATH BANTHI

Company Supervisor: Niclas Carlstrom, APTIV.
Chalmers Supervisor: Lena Peterson, Department of Computer Science and Engineering
Examiner: Per Larsson-Edefors, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Embedded Electronic System Design
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Adaptive Techniques for Tuning
of Process Noise in the Kalman filter

ATHULYA JOSE
SHREYA RAGHUNATH BANTHI
Embedded Electronic System Design
Department of Computer Science and Engineering

Abstract

This thesis work focuses on tuning the process noise of the Kalman filter for surrounding object detection and speed estimation in automotive driver assistance applications. We have created a simple road scenario in Matlab, where an ego-vehicle follows a target-car moving at different speeds and yaws in different road scenarios to collect the required data for the Kalman filter. The tracking algorithms help in predicting the future position of moving objects based on the measurements of RADAR detections. The RADAR sensors attached to the ego-vehicle directly measure the range, azimuth angle and range rate of the target object. We considered the compensated range rate and azimuth angle measurements from the RADAR sensors. Since the RADAR measurements are noisy, the Kalman filter is used to obtain a better result. It is well known that the covariance matrices of the process noise (Q) and the measurement noise (R) have a significant impact on the performance of the Kalman filter in estimating dynamic states. We begin by implementing the conventional ad-hoc approaches to estimating the covariance matrices. However, these approaches for estimating the covariance matrices, may not be suitable for achieving the best filter performance. To address this problem, we propose a adaptive filtering approach to estimate Q based on innovation and residual to improve the accuracy of the dynamic state estimation of the extended Kalman filter (EKF). We also present theoretical investigation methods for optimizing the algorithm when it is to be implemented in an embedded platform. Following that we present the results obtained in our work after comparing the fixed and the adaptive Kalman filter, which clearly points that the adaptive Kalman Q is better in its response time and adaptivity.

RADAR, Kalman filter, Process Noise, Adaptive Tuning, Extended Kalman filter(EKF).

Acknowledgements

First, we want to thank APTIV for giving us the opportunity to do our thesis at a great company. Also, we want to thank the Department of Computer Science and Engineering for letting us be a part of the group. A special thanks to our highly dedicated and helpful supervisors Lena Peterson, Niclas Carlstrom, Hanna Darnfors, Elvira Ramle, Jeanette Warnborg and Deepak Kiran and their support throughout. This thesis wouldn't have been carried out the way that it did without them. We would also like to thank our examiner Per Larsson-Edefors for his appreciated support. Together with these people we also would like to take the opportunity to thank all other persons that have helped us both during this spring, but also throughout our university studies. At last, we would like to thank our respective families for their support and encouragement throughout the entire journey.

Athulya Jose and Shreya Raghunath Banthi, Gothenburg, October 2021.

Contents

1
Introduction

Advanced Driver Assistance Systems (ADAS) are being developed to improve road safety by helping drivers perform complex driving functions. This requires a description of the environment in order to make accurate decisions. Therefore, the vehicle is equipped with multiple sensors such as radar, LiDAR and cameras to collect information from the vehicle environment. The data collected from the sensors can be used to estimate features such as range, speeds, accelerations and future positions of the surrounding objects [1].

However, we only consider radar for target tracking. From now on, we will refer to the vehicle equipped with radar sensors as the ego-vehicle. Since the radar detections are subject to noise due to errors in the radar system or external factors that affect the radar measurements, the Kalman filter is used. The Kalman filter algorithm provides estimates of unknown variables that are more accurate than values obtained from measurements alone. It uses a set of measurements observed over time that may contain statistical noise and other inaccuracies. To estimate the current state, the algorithm does not need a history of observations or estimates, but rather the estimated state from the previous time step and the current measurement. Even if we cannot observe or measure all internal states, the Kalman filter can estimate the internal state from the available measurements.

The increasing use of Doppler radars in automotive applications opens up new possibilities for object tracking. An ADAS helps in detecting and tracking objects in the environment with the help of Kalman filter algorithm used for state estimation problem. Kalman filters can optimally estimate the internal states of a system in the presence of uncertain and indirect measurements. An estimate of the system state is obtained by incorporating the dynamic model of a system and a series of measurements of the state using a two-step prediction and updating process.

At a given time, the true state is measured with a noisy sensor. We now have two estimates of the state of the system that are likely to differ: a predicted state and a measured state. The question now is which of the two estimates is correct.

Since both are subject to uncertainty, the question should really be: How can we combine the two based on their relative uncertainties?

A Kalman filter determines how much confidence or weighting must be applied to both the prediction and the measurement so that the corrected state lies exactly at

the optimal point between the two. This balancing act depends on a mathematical representation of the uncertainty, which we obtain in the form of the covariance.

The Kalman filter algorithm combines a sensor measurement with an erroneous prediction from a process model, both the measurement and the process model are subject to uncertainty. The uncertainty resulting from the process model deviations is called process noise, and the noise associated with the measurement model is called measurement noise. For example, a process model assumes that the car is moving at a constant speed because its acceleration is zero, but in reality it has an acceleration, that is, its speed varies from time to time. This change in the car's acceleration is an uncertainty/error that we introduce into our system with the process noise.

A basic discrete-time model often used for Kalman filtering is,

$$x_{k+1} = f(x_k) + w_k \tag{1.1}$$

$$y_k = h(x_k) + v_k \tag{1.2}$$

In (1.1), $f(x_k)$ is considered the process model and defines the evolution of the state (x) from time k to $k+1$. Similarly in (1.2), $h(x_k)$ represents the measurement model and defines the relationship between the state and the corresponding measured value (y) at the current time step k. In the above model, w_k and v_k are stochastic normally distributed noise variables with known covariance matrices Q and R, respectively[2]. The noise variables are responsible for the inaccuracies associated with the models. Therefore, the performance of the Kalman filter may deteriorate significantly under practical conditions when the covariance matrices of the process and measurement noise are unknown. Furthermore, the traditional non-adaptive Kalman filter algorithm is not suitable for the time-varying noise covariances encountered in maneuver tracking [3].

Furthermore, the tracking accuracy is highly dependent on the performance of the filter. In this scenario, the basic Kalman formulation with fixed known noise covariance matrices is not sufficient to achieve good tracking accuracy in a complex tracking environment. Choosing the covariance of process and measurement noise, also known as tuning the filter to optimize the performance of the filter with respect to a performance index, is a challenging task [4]. Determining the measurement noise v_k, is usually not the big problem, since a deep knowledge of the sensors is often available. On the other hand, careful estimation of the covariance matrices of the process noise is required for better performance of the Kalman filter to improve the overall tracking performance.

1.1 Objective

One of the most important requirements of ADAS, which includes subsystems such as collision avoidance/collision mitigation, adaptive cruise control, stop-and-go-assistant, or blind-spot detection, etc. is to reliably estimate the positions of other

vehicles. This can be achieved using the Kalman filter or one of its derivatives. However, in order for the Kalman filter to quickly adapt to the changes in the tracking environment, we can use some adaptive techniques to optimize the process noise and reduce the computation time of the algorithm. The aim of this work is to formulate adaptive methods to tune the process noise, make a comparison between different methods, and reduce the computation time of the algorithm. This aim includes:

- Formulation of a filter that easily adapts to the changes in the true state and provides good accuracy by reducing the mean error between the estimates and the true states.
- Integration of an algorithm that is optimal in terms of computational complexity into an embedded platform to reduce execution time.

1.2 Problem Formulation

The company Aptiv is developing ADAS in a vehicle that uses advanced technologies to assist the driver. ADAS includes autonomous emergency braking, blind spot warning, lane assist, etc. and is capable of reducing human error and thus traffic accidents. It is capable of making decisions to achieve both safety and comfort functions with the help of object tracking system.

Kalman filtering is an important component of the Aptiv object tracker and for ADAS in general. However, the complexity of the environment in which object tracking is performed for ADAS applications is high. This is because different types of objects with different motion patterns need to be tracked, including stationary objects such as trees, guardrails, etc., as well as non-stationary objects such as vulnerable road users, different types of vehicles, etc. Furthermore, the motion pattern of the objects depends on the state of the object, e.g. the maximum acceleration of a vehicle may depend on its current speed and gain. All these factors may affect the process noise and make it difficult to determine it. Therefore, an advanced method for determining the covariance matrix Q is required to optimize both tracking performance and computational complexity so that the tracker can respond quickly to changes in actual states.

1.3 Limitations

In order to limit the scope of the project, we decided to impose several constraints on the tracking problem. The tracking device can only operate in a 2D Cartesian coordinate system, since radars only compute the environment in one plane.
The algorithm will track a single target vehicle, so we assume that all detections are associated with this one target vehicle. These detections may be affected by noise, but our scope will not consider these challenges, i.e. the problem of data association.

1.4 Thesis Outline

Chapter 2 provides an overview of the theoretical foundations relevant to this thesis and introduces to radar measurements and the Kalman filter.

Chapter 3 explains the design approach taken to achieve the objectives stated in Section 1.1. It is divided into: Yaw rate estimation, the Extended Kalman filter with fixed process noise, and Extended Kalman filter with adaptive tuning process noise.

Chapter 4 presents the results of our work, including a comparison between the Kalman filter with fixed process noise and the adaptive process noise.
The final chapter includes discussion and future work related to this thesis.

2
Theory

This chapter describes the theory underpinning this thesis. In the first section, the theory behind the range and velocity measurement of the RADAR sensor mounted on the vehicle is explained. In the following sections, the vehicle motion model and the Kalman filter are presented.

2.1 Sensor Theory

RADAR technology interests and developments were observed and tested in the late 1800s as a major theory of RADAR capability. In today's society, RADAR sensors are common for weather, service, and automotive applications. Sensors can be used to track, monitor, and evaluate the movement of an object. RADAR itself is an electromagnetic sensor that operates on the transmitted signal in a specific waveform and senses the presence of an echo. From this signal, modern RADAR sensors can derive a distance measurement and a speed measurement [5].

2.2 Range Measurements

The time taken for the echo to travel between the target and the sensor depends on the range. The range can be measured by the relationship between time, position, and velocity at a known electromagnetic propagation velocity c. It is defined as the ratio of (2.1), where R is the one-way radial measurement (not same as the measurement covariance matrix referred otherwise), the transmission time T_i is the time a signal is transmitted, and the reflection time T_j is the time corresponding to the return of the signal to the sensor.

$$R = \frac{c(T_j - T_i)}{2} \tag{2.1}$$

2.3 Angle Measurements

To measure angles to objects, RADARs have multiple antennas arranged in rows and columns on the same plane. If an object reflecting a signal back to the antennas is not perpendicular to the RADAR plane, the different antennas will not receive the returning wave at exactly the same time. This results in a phase shift between the received signals from the different antennas. This phase shift can be used to calculate the angle at which the object is located relative to the RADAR. The phase

shifts of the antennas in the vertical columns are used to calculate the elevation angle of the object relative to the RADAR plane, and the phase shifts of the antennas in the horizontal rows are used to calculate the angle of the object in the horizontal plane.

2.4 Velocity Measurements

Velocity estimation from RADAR data is based on signal phase shift. The RADAR signal's phase shift over $2R$ (Range) duration is,

$$\phi = \frac{2\pi}{\lambda} \tag{2.2}$$

where λ is the wavelength. If the range R changes linearly with time, this results in a change in the phase shift of the return echo. This change depends on the relative velocity between the sensor and the object reflecting the signal, and causes a change in the frequency of the echo signal. The Doppler effect is well known, and the associated frequency changes are commonly referred to as Doppler frequency.
In order to find the relative radial velocity $V_d(t)$, the first order derivative is considered on each side as,

$$\frac{d\phi(t)}{dt} = \frac{4\pi}{\lambda}\frac{dR(t)}{dt} \tag{2.3}$$

The radial relative velocity, v_d is then identified as,

$$v_d(t) = \frac{dR(t)}{dt} = \frac{\lambda}{4\pi}\frac{d\phi(t)}{dt} \tag{2.4}$$

with the substitution,

$$\frac{d\phi(t)}{dt} = w_d(t) = 2\pi f_d(t), \tag{2.5}$$

where, $f_d(t)$ is the Doppler frequency. Finally, we find $v_d(t)$ as,

$$v_d(t) = \frac{\lambda f_d(t)}{2} \tag{2.6}$$

2.5 Velocity Induced by the Host

The vehicle is called a host when the RADAR is mounted on a vehicle. Thus, the RADAR moves along with the vehicle. Since the generated wave is moving faster than the vehicle, the range measurements are not affected by the movement of the vehicle. However, the motion of the host has a strong influence on the measurement of the radial velocity of objects, since phase shifts can occur even with a small change in distance. Consequently, for stationary objects the radial velocity is detected by the motion of the host vehicle. If another vehicle has the same velocity as the host vehicle, no radial velocity is measured. The motion of the host RADAR must also be

compensated to determine if the target is stationary or moving since, radial velocity should be measured with respect to the ground and not the host vehicle itself [6].

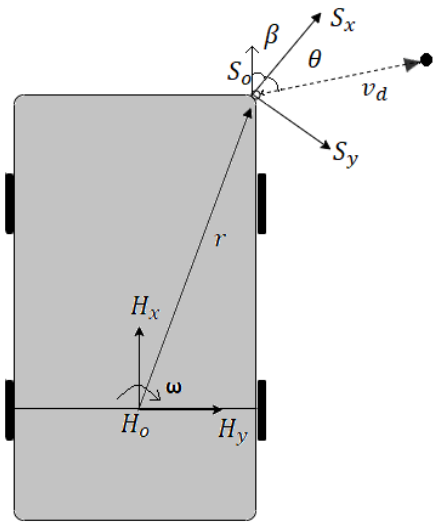

Figure 2.1: Figure shows the position of the RADAR co-ordinate system relative to the host co-ordinate system. This image was adopted from [6].

However, the velocity vectors of targets cannot be extracted directly because only the radial velocity aspect can be computed from a Doppler RADAR. Therefore, it is important to reconstruct the sensor velocity v_S and heading direction α for at least two obtained reflection points. However, it is helpful to include all observed reflection points and use regression for motion estimation to increase its accuracy. The main argument is that the estimated radial velocities form a cosine over the azimuth angle, hereafter referred to as the velocity profile[7].

The origin of the host coordinate system H_o is based on the rear axle of the vehicle as shown in Figure 2.1. The x-axis, H_x is aligned with the vehicle's forward direction. The y-axis, H_y is aligned to the right. Also, there is a sensor coordinate system S_o, which defines the position and orientation of the RADAR installed on the host. The orientation to the x-axis of the RADAR sensor coordinate system S_o to the x-axis of the host coordinate system H_0, is given by β, r is the radial distance from the H_o to the S_o.

As the host turns, the sensor moves along an arch trajectory with radius r. Hence the origin of the S_o includes velocity vector from the turning rate of the host, which is the tangential velocity and it is given by the cross product of ω, the rotational velocity of the host, and r, the position vector of the origin of the sensor coordinate system in the host coordinate system as

$$v_\perp = \omega \times r = \begin{bmatrix} v_{\perp x} \\ v_{\perp y} \\ v_{\perp z} \end{bmatrix} = \omega \begin{bmatrix} -r_y \\ r_x \\ 0 \end{bmatrix} \tag{2.7}$$

To obtain the total velocity of the sensor , velocity due to the forward motion of the host must be added to the tangential velocity as,

$$v_s = \begin{bmatrix} v_{sx} \\ v_{sy} \end{bmatrix} = \begin{bmatrix} v_{\perp x} \\ v_{\perp y} \end{bmatrix} + \begin{bmatrix} v_{Hx} \\ 0 \end{bmatrix} \tag{2.8}$$

In order to relate the induced velocity, expressed in a H_o, to the radial velocities computed in the S_o, the RADAR orientation relative to the host $\Delta\beta$ must be taken into account. Therefore the direction of the host induced RADAR sensor velocity is

$$\alpha_s = \mathrm{a}\tan 2\left(\frac{v_{S_y}}{v_{S_x}}\right) - \Delta\beta \tag{2.9}$$

Therefore the measured host-induced radial velocity, v_r, at an arbitrary azimuth angle, θ, can be expressed as

$$v_r(\theta) = -\Big|v_s\Big| \cos(\theta - \alpha_S) \tag{2.10}$$

2.6 Kalman filter

The linear quadratic Gaussian problem involves estimating the "state" of a linear dynamical system perturbed by white Gaussian noise [8]. The Kalman filter is used to estimate the state of a discrete-time controlled system governed by the linear stochastic difference equation. They are used to improve the estimation of a quantity that cannot be evaluated directly but can be calculated indirectly. In the presence of noise, they are usually used to integrate measurements from different sensors to find the best approximation to the states [9].

Kalman uses the dynamic model of the system (e.g., physical laws of motion), known control inputs, and multiple sequential measurements (e.g., from sensors) to obtain a more accurate estimate of the varying quantities of the system (state). The system dynamics model helps to make an initial estimate of the state of the system. This predicted state is corrected once the measurements from the real sensors become available. To improve the estimate of the state, the filter uses the knowledge of the noise statistics throughout the process.

The process model or dynamic model used to describe the evolution of the system state from time $k-1$ to k is given as follows,

$$x_k = Ax_{k-1} + w_{k-1} \tag{2.11}$$

In (2.11), x_k represents the state vector at time k, A is the state transition matrix applied to the previous state x_{k-1}, w_{k-1} is the process noise vector containing the imperfections associated with the model. The process noise vector is assumed to be Gaussian in origin and to have a covariance matrix given by Q.

In addition to the process model, when a set of measurements from the sensors is available, a measurement model is used to model the sensors and represent the system state in the measurement space. Therefore, the relationship between the measurement and the system state at the current time step k can be described as follows,

$$z_k = Hx_k + v_k \tag{2.12}$$

In (2.12), x_k is the state vector, z_k is the observation vector, H is the measurement matrix that maps the true state space to the measurement space, and v_k is the measurement noise vector that accounts for the uncertainties of the measurement model. The measurement noise vector is also assumed to be zero-mean Gaussian with covariance matrix R.

In order for the Kalman filter to produce an estimate of the system state x_k at time k, we need to provide an initial estimate of the system state x_0, available measurements, and the necessary information about the system, which includes the state transition matrix A, the measurement matrix H, the covariance matrix of the process noise Q, and the covariance matrix of the measurement noise R as described above.

The Kalman filter algorithm operates in two steps, called the prediction step and the measurement update step, as described below,

1. **Prediction Step:** The prediction step contains a model that exploits knowledge of the system dynamics and relays the previous estimates of the state and covariance matrix. This prediction is a timely projection for the next move to obtain the a priori estimates, $\hat{x}_k^-$. A prior estimate of the state $\hat{x}_k^-$ and the state error covariance P_k^- is obtained as follows,

$$\hat{x}_k^- = A\hat{x}_{k-1}^+ \tag{2.13}$$

$$P_k^- = AP_{k-1}^+A^T + Q \tag{2.14}$$

   The prior/predicted state estimate is evolved from the updated previous state estimate $\hat{x}_{k-1}^+$. The prior state error covariance will be larger as it is calculated by adding the process noise covariance. Therefore, after the prediction step the filter will be more uncertain of the estimated state.

2. **Measurement Update Step**: In the measurement update step, the difference between the true measurement z_k and estimated measurement Hx_k^-, also known as measurement residual or innovation d_k, is calculated first.

$$d_k = z_k - Hx_k^- \tag{2.15}$$

$$K_k = P_k^- H^T(HP_k^- H^T + R)^{-1} \tag{2.16}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k d_k \tag{2.17}$$

$$P_k^+ = (I - K_k H) P_k^- \tag{2.18}$$

In the above set of equations, $z_k \in \mathbb{R}$ is the measured value, K_k is the Kalman gain and $I \in \mathbb{R}^{nxn}$ is the identity matrix.

To obtain the a posteriori computation, $\hat{x}_k^+$, the measurement update step is responsible for measurement feedback, i.e., incorporating the last measurement that defines the system apriori [10].

2.7 Extended Kalman filter

An estimator that linearizes the system dynamics around the current state is known as the extended Kalman filter (EKF) [11]. This yields a linear model that can be used to estimate the current state from the available measurements and previous state. Its mean value provides the best estimate for white noise. The expected values of both v_k and w_k are 0 as they are zero-mean. Therefore, the linearization around 2.19 and 2.20 can be used to obtain a linearized model which is done using Jacobians as given in 2.21 and 2.22,

$$x_k = f(x_{k-1}) + w_{k-1} \tag{2.19}$$

$$z_k = h(x_k) + v_k \tag{2.20}$$

$$A_k = \frac{\partial f(x_{k-1})}{\partial x}|_{\hat{x}_k = \hat{x}_{k-1}^+} \tag{2.21}$$

$$H_k = \frac{\partial h(x_k)}{\partial x}|_{\hat{x}_k = \hat{x}_k^-} \tag{2.22}$$

A description of the iterative procedure follows,

1 Prediction Step:

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+) \tag{2.23}$$

$$P_k^- = A_{k-1} P_{k-1}^+ A_{k-1}^T + Q. \tag{2.24}$$

2 Measurement Update Step:

$$d_k = z_k - H_k(\hat{x}_k^-) \tag{2.25}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1} \tag{2.26}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k d_k \tag{2.27}$$

$$P_k^+ = (I - K_k H_k) P_k^-. \tag{2.28}$$

2.8 Model for the Vehicle Motion

Different models are employed to predict the movement of the vehicles. A constant velocity model is one of the simplest models for vehicle velocity estimation. Certainly, this model does not represent all the dynamics of the processes involved. But this assumption provides strong predictions with a finite period of time since the velocity and yaw rate (angular velocity) are continuous functions that are constantly changing in time due to the underlying mechanics of the vehicle. This defined as the Constant Turn Rate Velocity (CTRV) model is shown below.

$$v_{t+1} = V_t + r_t, \tag{2.29}$$

$$\omega_{t+1} = \omega_t + n_t, \tag{2.30}$$

Here r_t and n_t are the process noise indicating the uncertainty of the model.

The dynamics that are not considered in the model are modeled as process noise. These models that do not consider the dynamics, should increase the process noise than usual to account for the missing dynamics.

2.9 Adaptive Estimation of Process Noise Q

It is well known that the covariance matrices of the process noise Q and the measurement noise R strongly affect the efficiency of the Kalman filter in estimating dynamic states. A critical problem to be solved when using the Kalman filter is the proper determination of the covariance matrices of Q and R. An incorrect choice of Q and R can significantly degrade the performance of the Kalman filter and even cause the filter to deviate.

Generally, an ad hoc method is used in which Q and R are assumed to be constant during estimation and are manually adjusted by trial-and-error approaches. However, since noise levels may vary for different applications, it may be very difficult to adjust Q and R correctly using such an ad hoc approach. Moreover, the standard method for calculating Q and R requires an accurate knowledge of the process noise and computational error, usually based on extensive empirical research [2]. In practice, the values are usually fixed and applied to the entire application. The performance of embedded systems suffers from this inflexibility.

To address this problem, an estimation approach that adaptively adjusts Q and R at each step is used to reduce the impact of Q and R definition errors and increase accuracy [12]. In [13], adaptive estimation approaches are classified into four categories: Bayesian, correlation, covariance adjustment, and maximum likelihood approaches. Covariance adjustment is one of the well-known adaptive estimation approaches that adjusts the covariance matrix of the innovation or residual based on their theoretical values.

2.9.1 Measurement Residuals

In the Kalman filtering algorithm, the difference between the actual measured value and the predicted value is termed as innovation or pre-fit residual. Similarly, the difference between the actual measured value and its estimated value using the information available in time step k is known as post-fit residuals or residual. The covariance of the process noise Q can be adaptively estimated based on the innovation sequence. Theoretically, the innovation and residual sequences extracted from the filter are correlative.

In general, the innovation and residual analysis provides a more objective view of the actual filter performance. If the dynamic system of the Kalman filter were perfectly modeled, both the innovation and residual series should be zero-mean white noise processes. Unlike the state covariance matrix generated by the Kalman filter itself, the statistics of the innovation and residual series are independent and reliable indicators of filtering quality.

2.9.2 Sliding Window

Streaming motion statistics can be computed using the sliding window approach, where a window (or windows) containing the most recent read data is always kept and only this data is considered relevant for learning. A critical issue with this approach is the choice of window size. The first and simplest strategy is to set (or ask the user to specify) a window size W and maintain it during the execution of the algorithm [14]. Another variation of the window method uses variable-length windows and considers the evolution of the model error to decide whether a change has occurred. In this case, the a priori assumption about the rate of change is usually hidden in the way the decision is made. A window whose length is dynamically adjusted to reflect changes in the data. If changes occur, as indicated by a statistical test, the window is reduced in size to preserve only the data that still appear to be valid. If the data appears to be stationary, the window is enlarged to work with more data and reduce the variance.

2.9.3 Forgetting Factor

The statistics of streaming signals can also be calculated using the exponential weighing method that uses a forgetting factor. The algorithm recursively applies a set of calculated weights to the data sample. Thus the weighing /forgetting factor decreases exponentially as the data become older [15]. This will impose more influence to the recent data than the old data on the statistical calculations. Selection of the forgetting factor determine how much importance will be given to the recent data and old data. If the recent data need to have more influence on the statistics the forgetting factor should be closer to zero. This will also help to detect small changes in data sets with fast variations.

2.10 Code Optimization for an Embedded Platform

This section discusses the problems encountered in translating and optimizing the computer programs to be compiled on an embedded platform. As code optimization was one of the scope of our thesis, which we could not fulfill due to time constrictions, we planned to provide the theoretical insights that will help achieve an optimal code. So, this will not be discussed in our Approach/Result sections.

To begin with, embedded systems are characterized by a number of constraints that are not present in the world of standard computers. Most of these constraints are related to cost. The reason is that embedded systems are characterized by a set of constraints such as system timings, code size, RAM, and power consumption that are not common in standard PCs. Embedded platforms often use idiosyncratic processors designed to maximize their performance for a limited class of applications. The applications are often very power sensitive. The main difference is the limited amount of memory designed to maximize performance for a limited class of applications, and optimizing each of these functions requires its own methods and techniques.

2.10.1 Optimization Goal

When developing code for an emedded platform it is also important to set an optimization goal that is good enough for the current application, as it can be a waste of resources and time to simply try to go as low as possible. Also, reaching this goal means that the optimization is complete. If your program is working, you may already know or have a pretty good idea of which subroutines and modules are most important to the overall efficiency of the code. Interrupt routines, high-priority tasks, computations with adaptive deadlines, and functions that are either computationally intensive or called frequently are all likely candidates [16]. A so-called profiler, included in some software development packages, can be used to narrow the focus to the routines that the program spends most (or too much) time on.

2.10.2 Software Tools Required

Optimizing a system can be extremely difficult without the right measurement tools. For example, energy optimization can be difficult if we do not have accurate means to measure the energy consumed by the system or by the microcontroller. If we do not separate these two different energy measurements and try to minimize the energy of the microcontroller, then we wont see much reducing in energy consumption. While there are several tools that can help us achieve the same, we should carefully choose non-intrusive optimization tools that will not change the application. When we use the help of compilers for code optimization, they can be designed to help us achieve the desired optimization. Considering the choice of compiler, compiler settings etc.

2.10.3 Optimization Tradeoffs

First, we need to identify the area that needs improvement, which is either the speed or the size of the code. To improve speed, critical sections need to be identified so that time is not wasted on executing non-critical sections.

Typically, 80 percent of a program's execution time is spent running 20 percent of the code [17]. So optimizing this hot spot will improve performance, even if we have to do it at the expense of slowing down the other area of the code. In addition, unless absolutely necessary, optimization should not be done at the end of the development cycle, as this adds to the cost of an initial design change and wastes further time and resources in debugging.

Code optimization means writing or rewriting code so that a program consumes as little memory or disk space as possible, minimizes its CPU time or network bandwidth, or makes optimal use of extra cores. In practice, we sometimes resort to another definition: writing less code.

Execution speed is usually only important within certain time-critical and/or frequently executed sections of code. There are many things you can do manually to improve the efficiency of those sections. But code size is hard to affect manually, and the compiler is in a much better position to make that change for all your software modules.

Since memory is an important aspect, if we need to optimize in terms of memory, we need to consider how the data is handled by considering the use of RAM, the use of data types, avoiding unnecessary type conversions, considering the benefits of unsigned types, etc.

2.10.4 Optimization Techniques

Some important tricks to keep in mind are,

- Signed and unsigned data are also considered different data types, since it takes processing cycles to convert from one type to the other. Therefore, it is preferable to use unsigned types for division and reminders, array indexing, and loop counters, while signed types are used for converting integers to floating point numbers.
- The local variable is preferred to the global variable in a function. In general, global variables are stored in memory, while local variables are stored in the register. Since register access is faster than memory access, implementing local variables results in faster operation speed. In addition, the portability of the code also argues for the use of local variables. However, if there are more local variables than available registers, the local variables are temporarily stored in the stack.
- Also, a large number of parameters can be costly due to the number of pushes and pops on each function call. Therefore, it is more efficient to pass structure

references as parameters to reduce this overhead.

- The return value of a function is stored in a register. If this return data is of no use, time and space are wasted storing this information. The programmer should define the function as void to minimize the additional processing overhead.
- In C++, the keyword inline can be added to any function declaration. This keyword makes a request to the compiler to replace all calls to the indicated function with copies of the code that is inside. This eliminates the run-time overhead associated with the actual function call and is most effective when the inline function is called frequently but contains only a few lines of code. Inline functions provide a perfect example of how execution speed and code size are sometimes inversely linked. The repetitive addition of the inline code will increase the size of your program in direct proportion to the number of times the function is called. Obviously, the larger the function, the more significant the size increase will be. The resulting program runs faster, but now requires more ROM in which to be stored.
- Re-usability measures whether existing assets - such as code - can be reused. Assets are easier to reuse if they have properties such as modularity or loose coupling. Re-usability can be measured by the number of dependencies. Running a static analyzer can help you identify these dependencies.

These are some of the most common suggestions that may be considered for the particular application. But, never make the mistake of assuming that the optimized program will behave exactly like the unoptimized one. You must completely retest your software at each new optimization level to be sure that the behavior has not changed. It is important to document and keep all test results. Comparing and evaluating the results before and after optimization helps maintain the integrity of the software.

3
Approach

Tracking algorithms help in predicting the future location of moving objects based on measurements from sensor systems. The RADAR sensors attached to the vehicle measure the distance, azimuth angle and range rate of the target object. Since the RADAR also moves with the vehicle, the distance rate measured by the RADAR is relative to the host vehicle. Therefore, the motion of the sensor should be compensated to estimate the velocity of the target object relative to the ground. Then, the yaw rate of the target is estimated using the compensated range rate and the azimuth angle measurement of the RADAR sensors. Since the RADAR measurements are noisy, the Kalman filter is used to obtain a better estimate of the target's position and orientation. The main objective of this work is to estimate the covariance of the process noise of the Kalman filter in real time to improve the performance of the Kalman filter.

3.1 Yaw Rate Estimation

During each measurement cycle, the RADAR sensor receives a series of reflections from the target object. The yaw velocity of the target can then be calculated from the measurements of at least two Doppler RADARs [18]. This can be achieved by exploiting the relationship between radial velocity and azimuth angle measurements. The radial velocity forms a sinusoidal shape over the azimuth angle and is referred to as the velocity profile. To estimate the yaw velocity of the target, it is assumed that:

- At least two RADAR sensors receive reflections from the target object, as shown in Figure 3.1.
- Each sensor $j \in \{1, 2\}$ receives a number of reflections $i \in \{1, ...N_j\}$ from the target object, where N_j is the number of reflections received by sensor j.
- Position of the RADARs (x_j^S, y_j^S) in a common coordinate system is known.

For a rigid body, the velocity vector is defined by the instantaneous center of rotation (ICR), which is the stationary point about which the body is currently rotating. Using the ICR position (x_{ICR}, y_{ICR}) and the yaw rate ω, the 2D state of motion of the target object can be specified as

$$\begin{bmatrix} v_{j,i}^x \\ v_{j,i}^y \end{bmatrix} = \omega \begin{bmatrix} y_{j,i}^P - y_{ICR} \\ x_{ICR} - x_{j,i}^P \end{bmatrix} = \omega \begin{bmatrix} y_j^s + r_{j,i} \sin(\theta_{j,i}) - y_{ICR} \\ x_{ICR} - r_{j,i} \cos(\theta j, i) - x_j^s \end{bmatrix} \tag{3.1}$$

where $r_{j,i}$ and $\theta_{j,i}$ represent the range and azimuth angle measured by the RADAR

sensor. Similarly, $(x^P_{j,i}, y^P_{j,i})$ is the position of the reflection point in the target object. The velocity vector of the target object can be mapped to the radial velocity measurement of the RADAR as [18].

$$v^D_{j,i} = \begin{bmatrix} \cos(\theta_{j,i}) & \sin(\theta_{j,i}) \end{bmatrix} \begin{bmatrix} v^x_{j,i} \\ v^y_{j,i} \end{bmatrix} \tag{3.2}$$

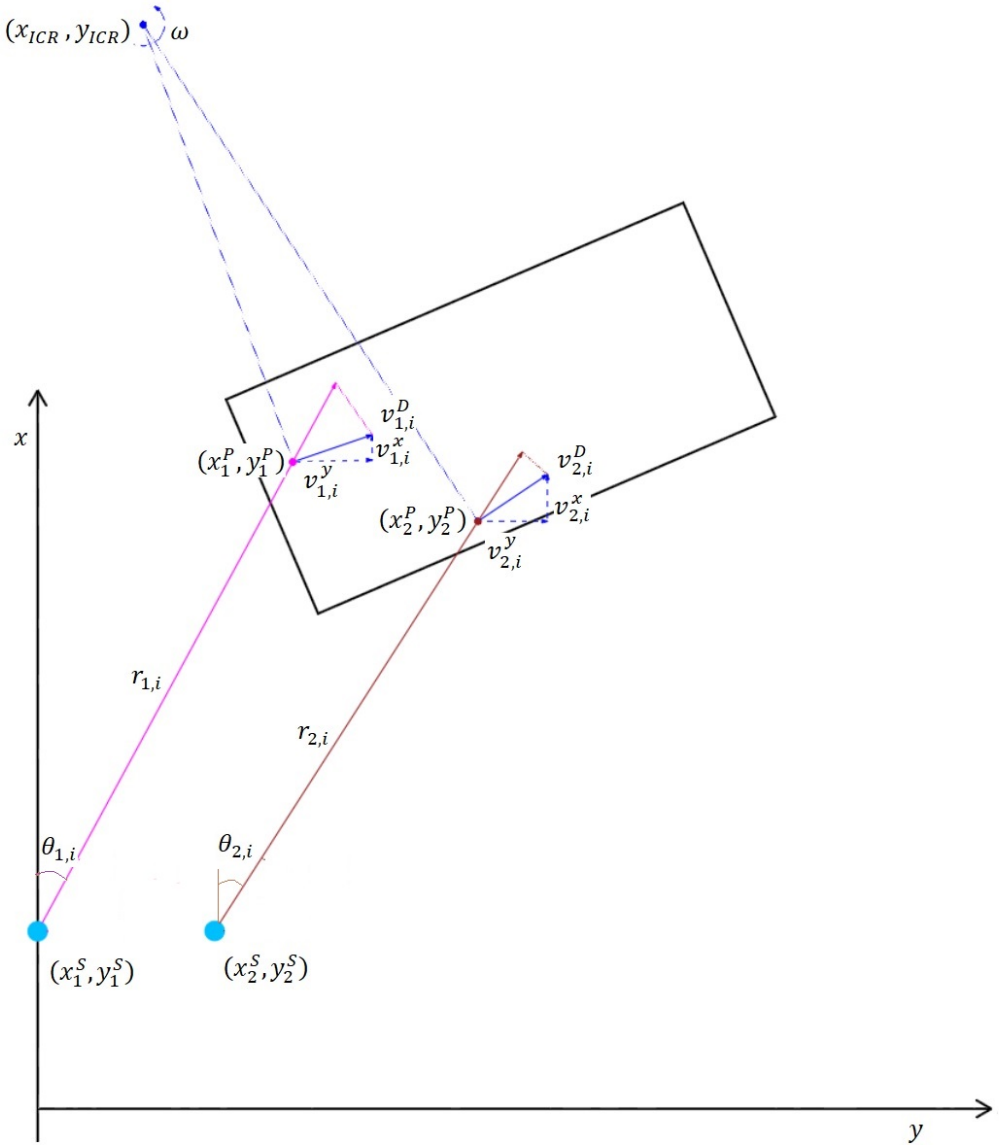

Figure 3.1: Estimating the 2D motion of an object using two Doppler RADAR sensors with known position in a common coordinate system. This image was adopted from [18]

Inserting 3.1 in 3.2 gives

$$v^D_{j,i} = \underbrace{\omega(y^s_j - y_{ICR})}_{C_j} \cos(\theta_{j,i}) + \underbrace{\omega(x_{ICR} - x^s_j)}_{S_j} \sin(\theta_{j,i}) \tag{3.3}$$

Equation 3.3 is known as the velocity profile and it depends on the azimuth angle $\theta_{j,i}$ and does not dependent on the position of the reflection point on the target. This equation can be re-written as

$$v_{j,i}^{D} = C_j \cos(\theta_{j,i}) + S_j \sin(\theta_{j,i}) \tag{3.4}$$

Using regression analysis, we can calculate C_j and S_j separately for each sensor which results in two systems of linear equations. Another linear regression based on C_j and S_j will yield the motion parameters [18]:

$$\min_{\omega^{-1},\, x_{ICR},\, y_{ICR}} \left\| \begin{bmatrix} -y_1^S \\ x_1^S \\ -y_2^S \\ x_2^S \end{bmatrix} - \begin{bmatrix} -C_1 & 0 & -1 \\ -S_1 & 1 & 0 \\ -C_2 & 0 & -1 \\ -S_2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega^{-1} \\ x_{ICR} \\ y_{ICR} \end{bmatrix} \right\|_2^2 \tag{3.5}$$

3.2 Dynamic Model

The state vector and dynamic model used for casting the object tracking problem into the Kalman filter framework are described here. The state vector chosen for the Kalman filter includes the longitudinal and lateral positions px and py, respectively, the linear velocity v, the yaw ψ, and the yaw rate ω, as described in (3.6):

$$x = \begin{bmatrix} px \\ py \\ v \\ \psi \\ \omega \end{bmatrix} \tag{3.6}$$

The dynamic model chosen for predicting the states in the prediction step is a coordinated curve model that assumes constant velocity and yaw rate. The model is represented in (3.7), which describes the relationship between the elements in the state vector.

$$f(x) = \begin{bmatrix} x(1) + Tx(3)\cos(x(4)) \\ x(2) + Tx(3)\sin(x(4)) \\ x(3) \\ x(4) + Tx(5) \\ x(5) \end{bmatrix} \tag{3.7}$$

In (3.7), x(1) corresponds to the longitudinal position, x(2) is the lateral position, x(3) represents the velocity, x(4) represents the yaw and x(5) is the yaw rate in the state vector and T represents the time between RADAR frames.

3.3 Measurement Model

The RADAR sensor provides measurements in polar coordinates, which are converted to Cartesian coordinates because our state vector is described in Cartesian

coordinates. The relationship between the converted RADAR sensor measurement and the state vector is described by the measurement model given in (3.8)

$$h(x) = \begin{bmatrix} x(1) \\ x(2) \\ x(5) \end{bmatrix} \tag{3.8}$$

where x(1) is the longitudinal position, x(2) is the lateral position and x(5) is estimated yaw rate using the measurements from the RADAR sensor. The velocity measurement is not included in the measurement vector as it is more noisy.

3.4 Extended Kalman filter with Fixed Process Noise

The block diagram of the extended Kalman filter using fixed process noise is shown in Figure 3.2.

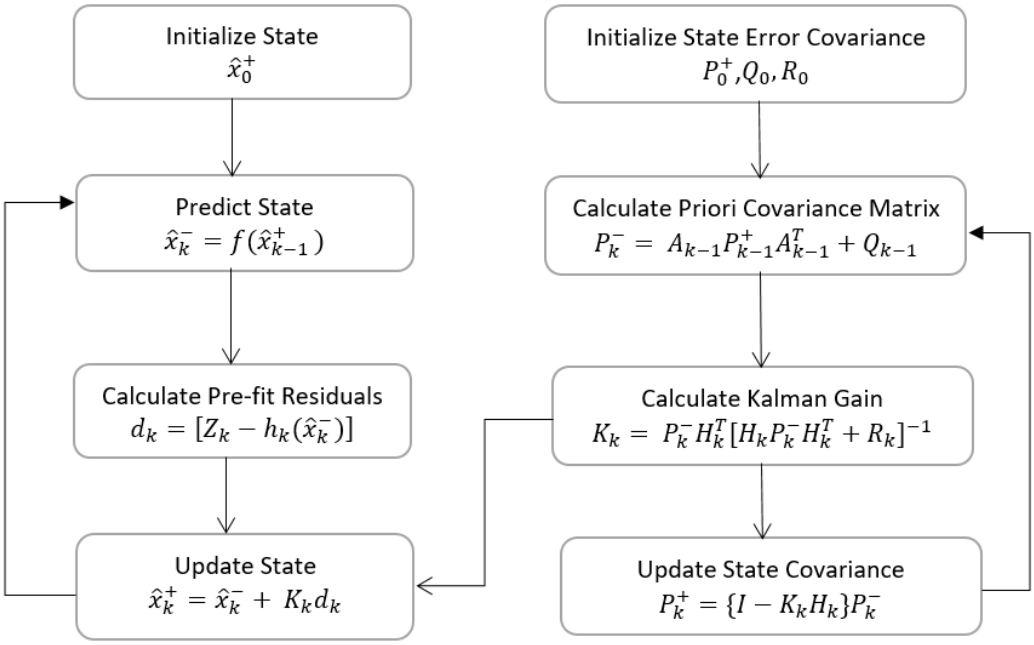

Figure 3.2: Block diagram of extended Kalman filter using fixed process noise

As seen in the block diagram (3.2), at first the Kalman filter needs to be provided with an initial guess of the state estimate $\hat{x}_0^+$, state error covariance P_0^+, process noise covariance Q and measurement noise covariance R.
The state vector is initialized with zero as,

$$\hat{x}_0^+ = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.9}$$

The initial state error covariance P_0^+ is set to a large value of 10^4. We selected the process noise and measurement noise covariance matrices by trial and error as,

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{3.10}$$

$$R = \begin{bmatrix} 0.0001 & 0 & 0 \\ 0 & 0.0001 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{3.11}$$

In the block diagram (3.2) the initialization step is followed by the prediction step in which, state at time $k-1$ is used along with the motion model to predict the state at time k. And a prior calculation of the state error covariance is also made in the prediction step as seen in the block diagram.

The prediction step is followed by the measurement update step, where the predicted state is used together with a measurement model specified in 3.8 to estimate the current measurement. The difference between the estimated measurement and the actual measurement, as well as the Kalman gain, are used to update the predicted state. Similarly, the prior covariance calculated in the prediction step is also updated in the update step. The updated state and the state error covariance are passed to the prediction step to predict the next state (see block diagram). During all time steps, the process noise covariance used in the prediction step is the same as the Q selected in the initialization step.

3.5 Extended Kalman filter with Adaptive Tuning Process Noise

The idea which was proposed in [19] will be described in this section. The block diagram of the proposed algorithm is shown in Figure 3.3. As can be seen in the block diagram, the initial value of Q, denoted as Q_0, needs to be specified by the user in the first step. In the subsequent steps, Q is updated. Compared to the block diagram of the Kalman filter with fixed process noise in Figure 3.2, the new algorithm has an additional block for estimating the covariance of the process noise Q. Therefore, the value of Q used in the prediction step is updated at each time step. This adaptive estimation of Q_k can be achieved using the innovation-based method.

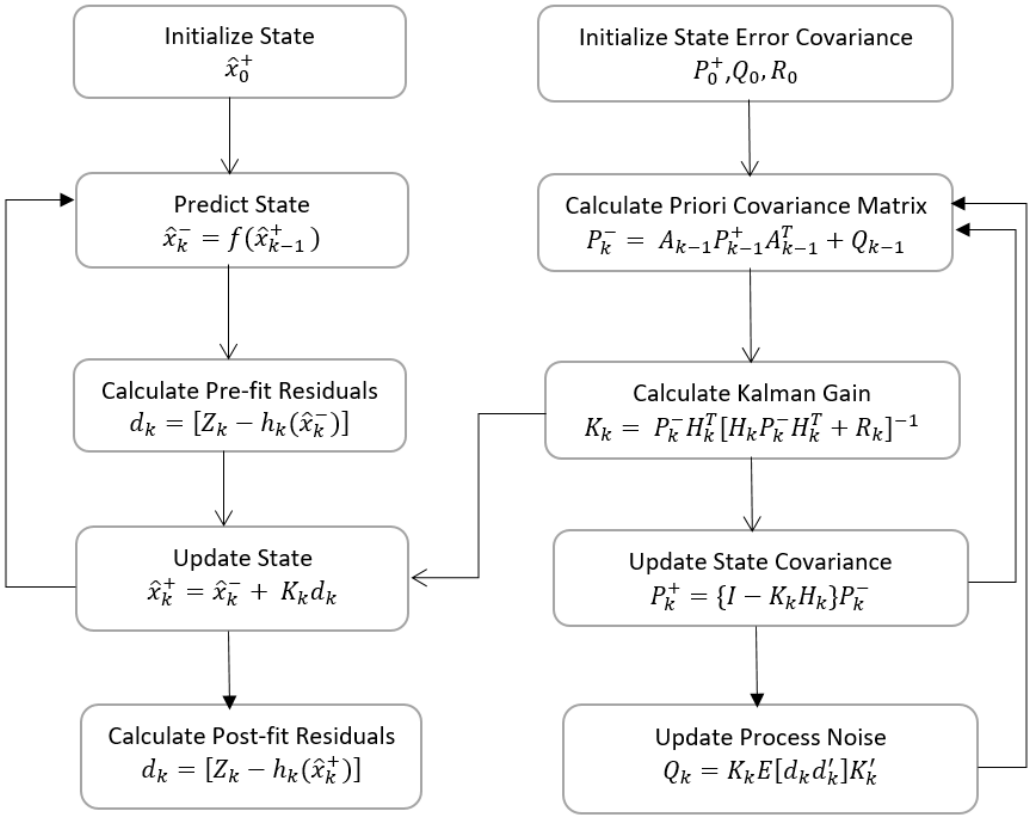

Figure 3.3: Block diagram of Kalman filter with adaptive tuning process noise

3.5.1 Innovation Based Method

Process noise estimation from the Kalman filter equations explained in theory chapter is described here [19].
From 2.19 process noise w_{k-1} can be written as:

$$w_{k-1} = x_k - f(x_{k-1}) \tag{3.12}$$

$$\hat{w}_{k-1} = \hat{x}_k^+ - f(\hat{x}_{k-1}^+) \tag{3.13}$$

using 2.23 we can rewrite the process noise as:

$$\hat{w}_{k-1} = \hat{x}_k^+ - \hat{x}_k^- \tag{3.14}$$

applying 2.27 in 3.14 will result in:

$$\hat{w}_{k-1} = K_k d_k \tag{3.15}$$

Then the process noise covariance matrix Q_k can be calculated as:

$$\mathrm{E}[\hat{w}_{k-1}\hat{w}_{k-1}^T] = K_k \mathrm{E}[d_k d_k^T] K_k^T \tag{3.16}$$

$$Q_k = K_k \mathrm{E}[d_k d_k^T] K_k^T \tag{3.17}$$

3.5.2 Sliding Window Method and Forgetting Factor Method

The expected value operation $\mathrm{E}[d_k d_k^T]$ can be performed using either the sliding window method or the forgetting factor method [20]. In the sliding window method, a window of size N is used. As new data become available, the window is shifted to accommodate the current sample and $N-1$ previous samples. The expected value is calculated by taking the average of the innovation samples over the window, as shown below,

$$\mathrm{E}[d_k d_k^T] = \frac{1}{N} \sum_{i=k-N+1}^{k} d_k d_k^T \tag{3.18}$$

Inserting 3.18 in 3.17 gives,

$$Q_k = K_k \left(\frac{1}{N} \sum_{i=k-N+1}^{k} d_k d_k^T \right) K_k^T \tag{3.19}$$

The length of the data required to compute the expectation operation is determined by the length of the window. For rapidly changing data sets, a small window size is recommended, while for data sets that change slowly, a large window size is recommended [20]. Since the method uses $N-1$ previous samples along with the current samples, the sliding window method implementation must remember the previous samples, which in turn results in more memory being required to store the $N-1$ innovations.

The forgetting factor method applies a certain weight to the older data and the current data to calculate the expectation of the data set. The forgetting factor α determines how much weight to give to the current value and the earlier values. In general, less weight is given to the older data, and the weight decreases as the data gets older. Consequently, the most recent data has a greater impact on the calculations than the older data. Using the forgetting factor method, we can calculate Q_k as:

$$Q_k = \alpha Q_{k-1} + (1-\alpha)(K_k d_k d_k^T K_k^T) \tag{3.20}$$

This method is comparatively better suited for embedded applications as it does not need to remember a large amount of previous residuals.

3.5.3 Improving the estimation of Q

In the Kalman filter algorithm, the innovation d_k is the difference between the actual measurements and the estimated measurements. It is also called the residuals or pre-fit residuals and can be calculated as:

$$d_k = z_k - \hat{z}_k \tag{3.21}$$

In 3.21, $\hat{z}_k$ is the estimated measurement and is calculated using the predicted state, which can be written as:

$$\hat{z}_k = h(\hat{x}_k^-) \tag{3.22}$$

After the update step, we can calculate the residuals using the updated or corrected state as:

$$\hat{z}_k = h(\hat{x}_k^+) \tag{3.23}$$

The residuals calculated using the corrected state are called post-fit residuals. The adaptive Q estimates can be improved by using the corrected state in the calculation of the residuals, resulting in a smoother state estimate than using the predicted state [21].

4

Results

To evaluate the performance of the adaptive tuning extended Kalman filter algorithm, the data was collected using MATLAB simulation. The Automated driving toolbox provided by MATLAB [22] is used to create an artificial driving scenario in which the target vehicle is driving through a roundabout in front of the host vehicle equipped with two RADAR sensors. The collected noisy RADAR detections were further processed using extended Kalman filter to obtain the states of the target vehicle which includes position, velocity, yaw and yaw rate. The resulting states of the target vehicle are presented for both fixed process noise and adaptive process noise cases. In the fixed process noise case, we used a constant process noise, which we selected by trial and error method, throughout the estimation process. For the adaptive tuning case, the process noise is calculated in adaptive during each epoch using the innovation based method instead of using a fixed process noise. The innovation-based method was further improved by replacing predicted state by the corrected state in the calculation.

4.1 Fixed Process Noise

In the first case we used fixed process noise and measurement noise in the Kalman filter to estimate the states of the target vehicle. The resulting Kalman filtered states of the target vehicle along with the true states are plotted in Figure 4.1. In the figure the top plot shows the difference between the true position of the target and the Kalman filtered position of the target. As shown, the Kalman filtered position of the target seems to align with the true position. True and Kalman filtered velocity of the target is compared in the second plot. It is seen that the Kalman filtered velocity is lagging behind the true velocity of the target. In the third plot a comparison of true yaw and Kalman filtered yaw of the target vehicle is presented. Kalman filtered yaw has a small deviation from the true yaw of the target. Comparison between true yaw rate and Kalman filtered yaw rate can be seen in the last plot. The comparison says that the Kalman filtered yaw rate is not exactly aligned that of the true yaw rate.

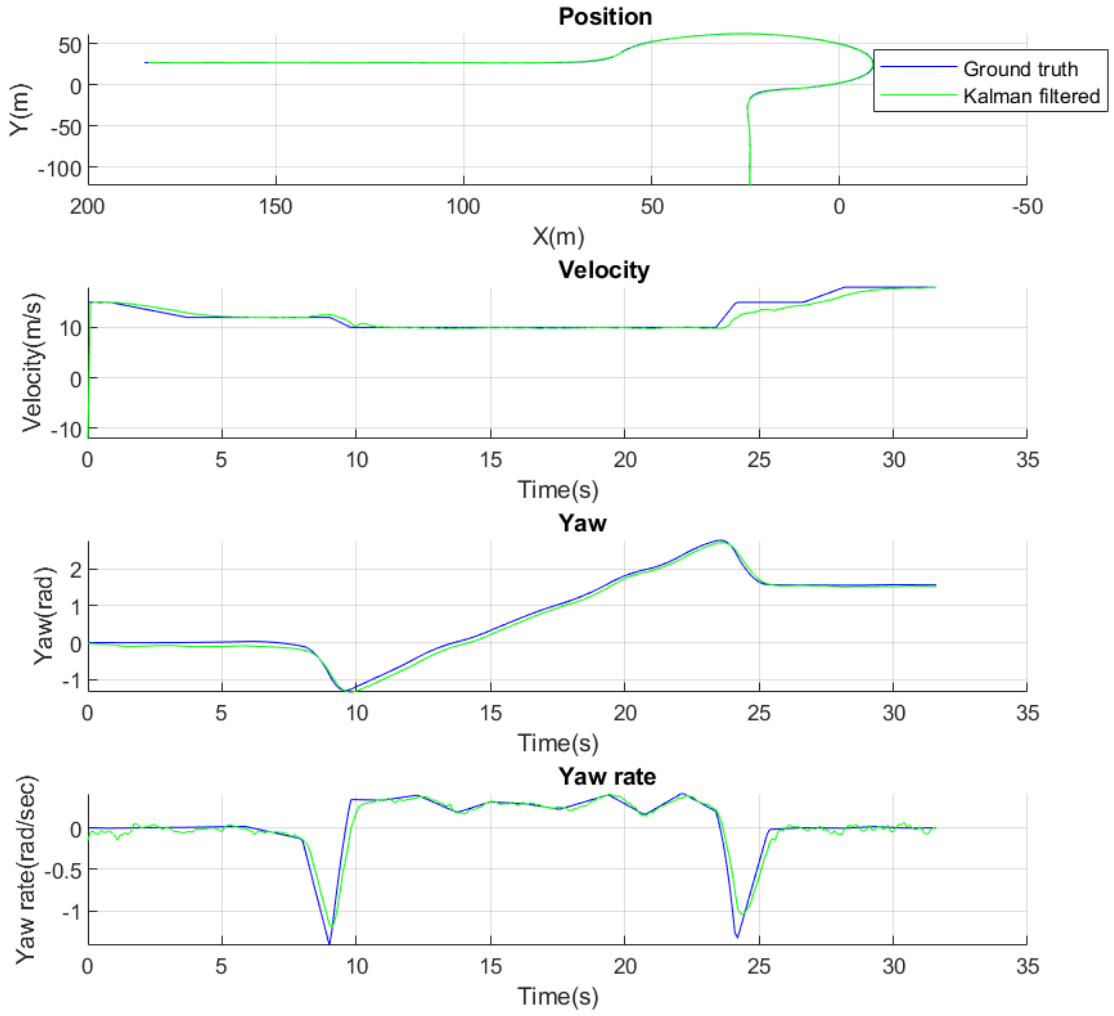

Figure 4.1: Kalman filtered states of the target when using fixed process noise

4.2 Adaptive Tuning Process Noise

In the second case, we use the Kalman filter described in Section 3.5 in which the Kalman filter algorithm is made to tune the process noise in adaptive. The initial process noise is set the same as that of the fixed process noise used in the first case and from the second time step onwards the Kalman filter estimated the process noise in adaptive. The resulting Kalman filtered states and true states are compared in the subsequent sections.

4.2.1 Adaptive Tuning Process Noise Using Pre-fit Residuals

The adaptive process noise for the Kalman filter is calculated using the innovation based method and the resulted Kalman filtered position, velocity, yaw and yaw rate of the target are plotted along with true states in Figure 4.2.

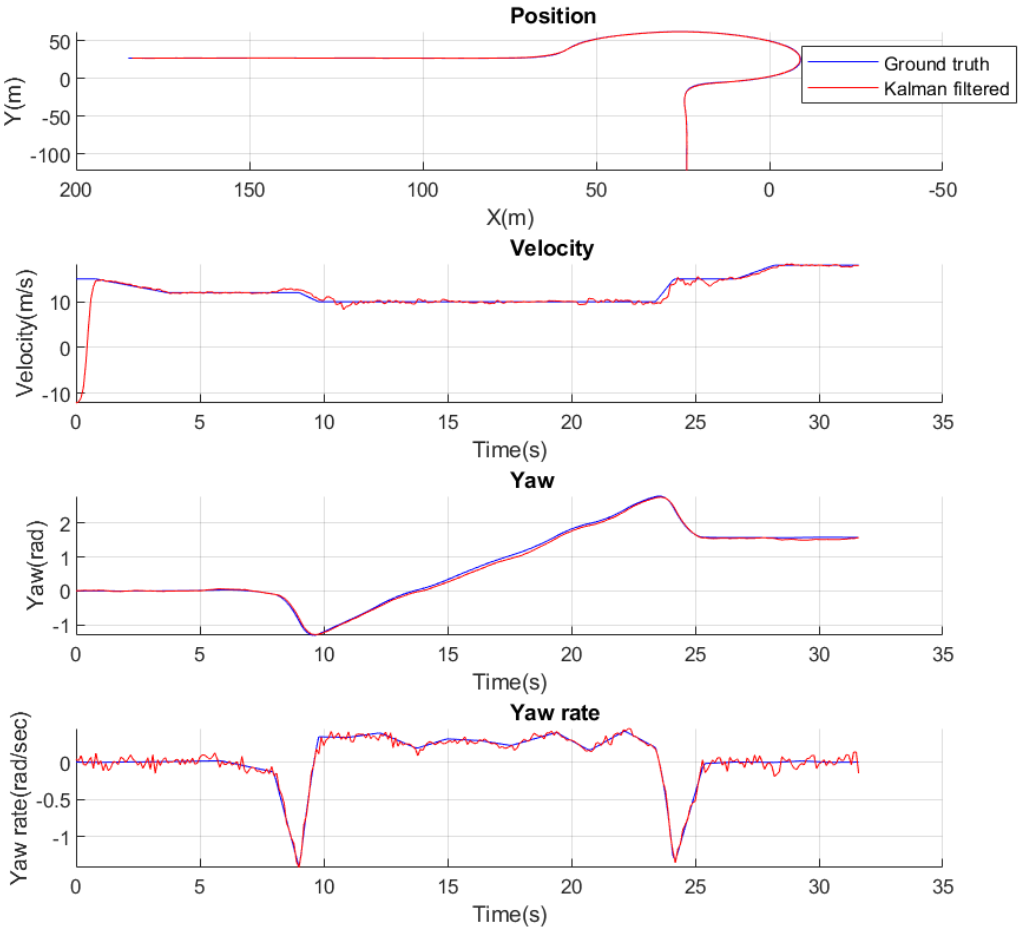

Figure 4.2: Kalman filtered state of the target when using adaptive process noise

From the results, the Kalman filtered position and yaw are aligning closely to the true states of the target. The Kalman filtered velocity is not as fast responsive in the beginning. However, after around 1sec, Kalman filtered velocity follows the true velocity but with few oscillations. Similarly, the Kalman filtered yaw rate of the target is also following the true yaw rate, but the curve seems to be much more oscillating.

4.2.2 Adaptive Tuning Process Noise Using Post-fit Residuals

In order to improve the estimation of process noise hence to reduce the oscillations in the Kalman filtered states when using adaptive process noise, the predicted state is replaced with corrected state to use the post-fit residuals in the estimation of process noise. The results are shown below.

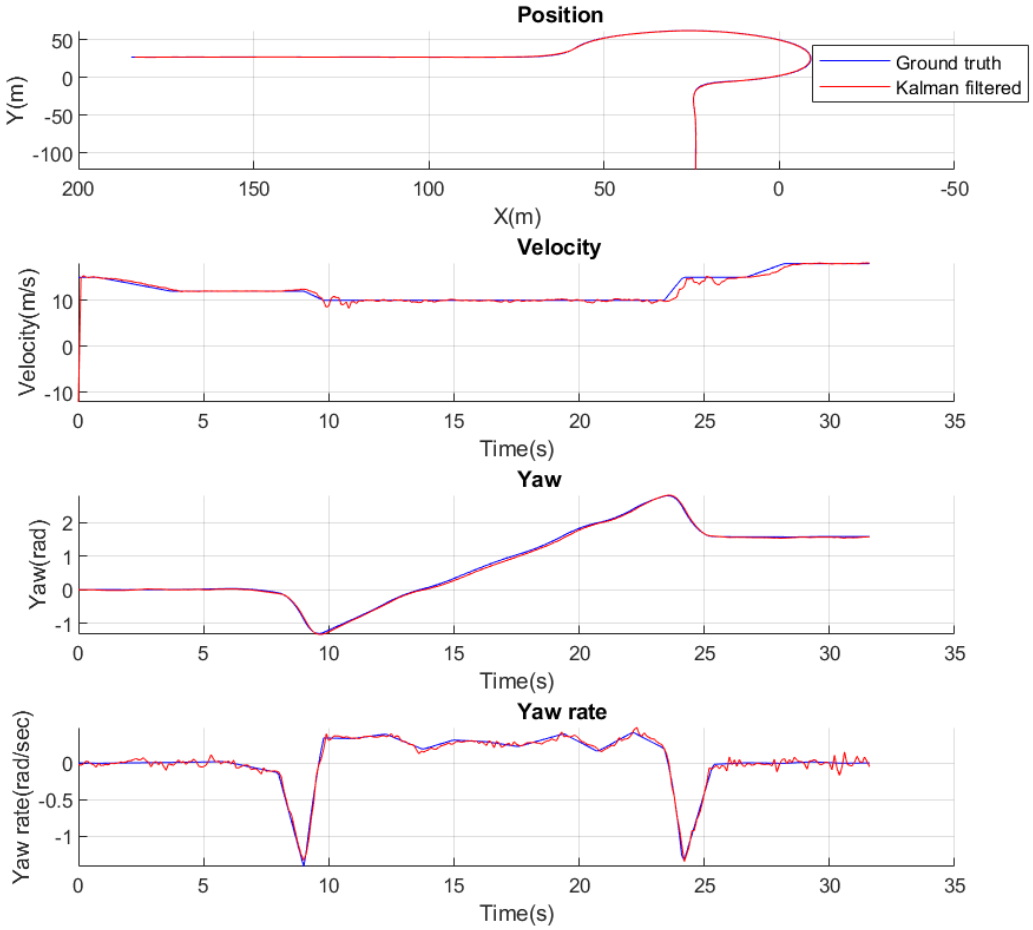

Figure 4.3: Kalman filtered states of target

When compared with estimation of process noise using pre-fit residuals or innovation, the Kalman filtered velocity and yaw rate are improved when used the corrected states in the estimation of process noise. The Kalman filtered velocity has a faster responsive and the curves became smoother than the curve obtained when using innovation.

4.3 Fixed Process Noise vs Adaptive Tuning Process Noise

A comparison of the Kalman filtered state of the target when using fixed process noise and adaptive process noise along with true states is depicted in Figure 4.4.

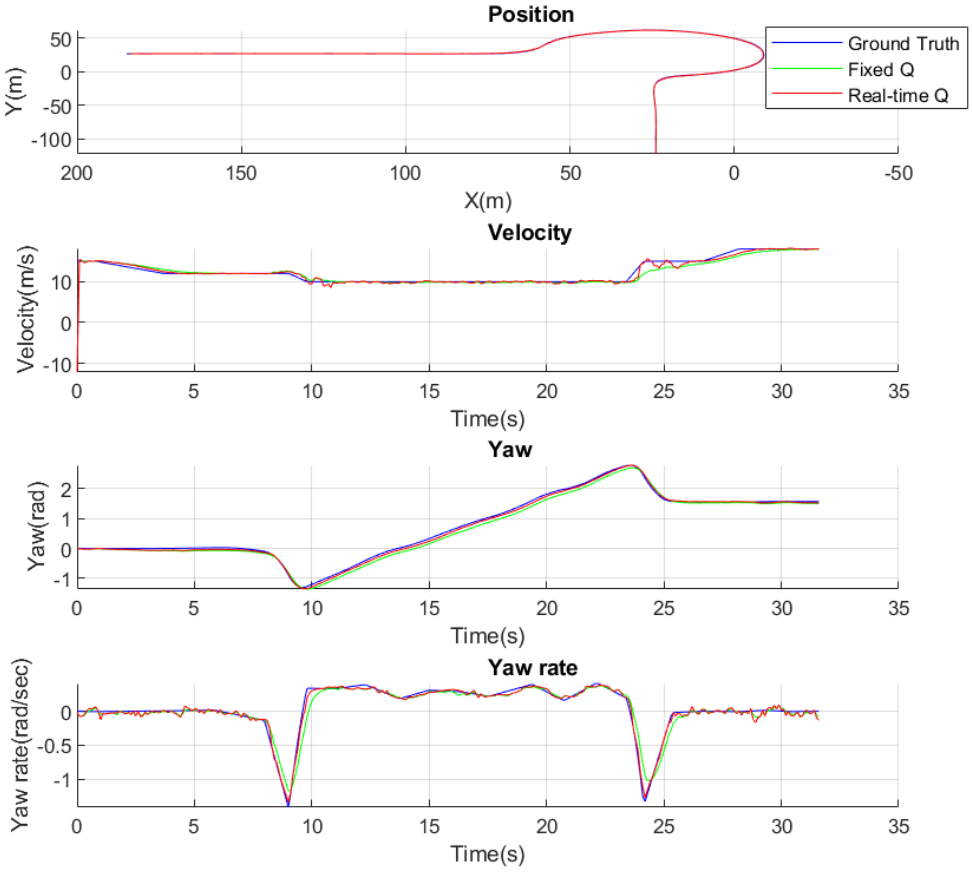

Figure 4.4: Comparison of using Fixed process noise and Adaptive process noise in extended Kalman filter

Compared to the Kalman filtered state using the fixed process noise, a slight improvement is achieved with the Kalman filtered state using the fixed process noise. The corresponding difference is presented in Figure 4.5. The difference is plotted for the velocity, yaw and yaw rate as the improvements are noticeable only in these states. In the beginning of the difference plot, the error or difference is near to zero as the vehicle states are not changing at this point. The difference plot starts to diverge from zero as the vehicle state changes. This is because the filter with adaptive Q adapts slightly faster than the filter with fixed Q.

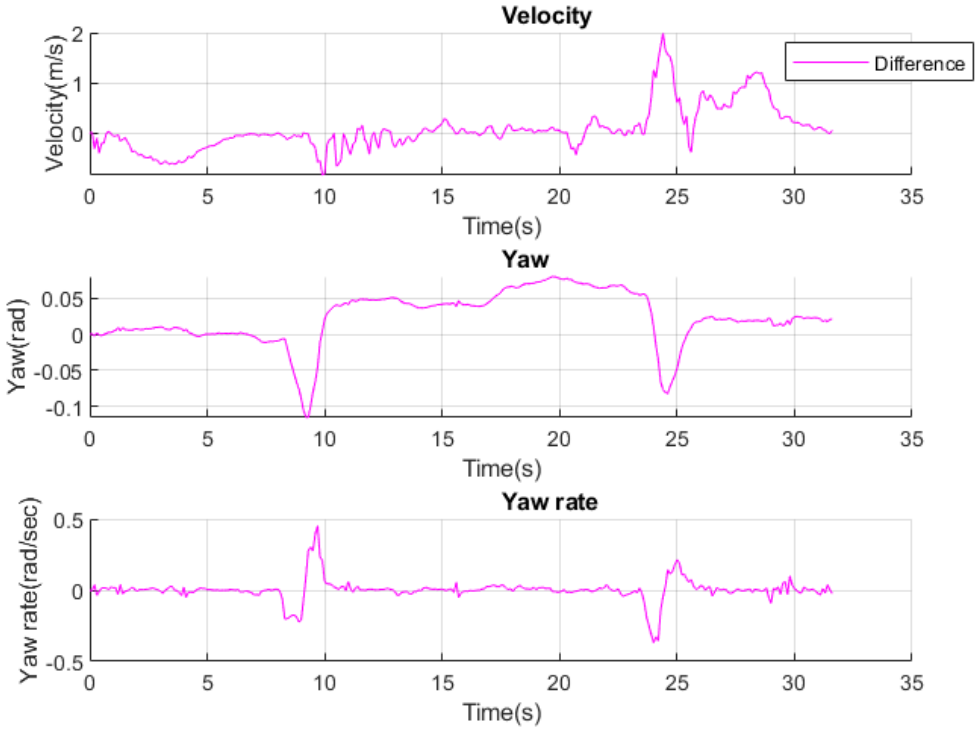

Figure 4.5: Difference between the Kalman filtered states when using fixed process noise and adaptive process noise

4.4 Mean Squared Error

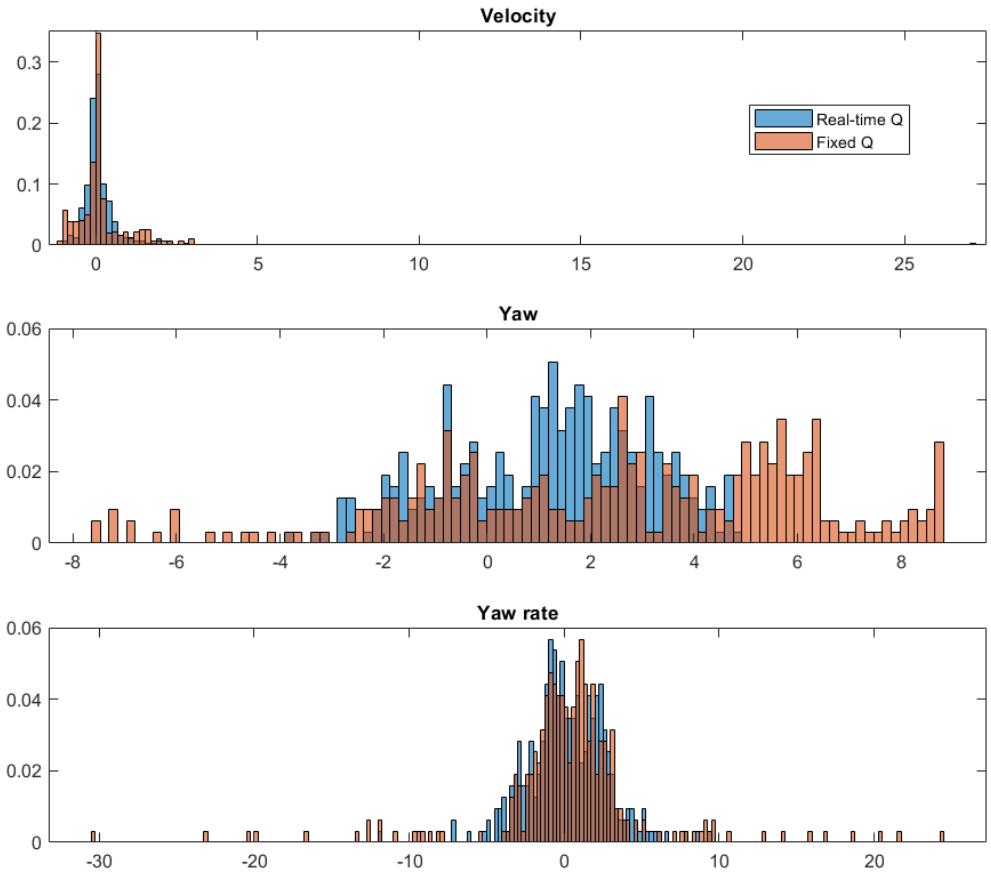

Figure 4.6: Mean squared error

Histogram of error in the Kalman filtered states when used adaptive process noise is shown in Figure 4.7. From the histogram we can see that, for the case of Kalman filter with adaptive Q, most of the error bins are distributed around zero compared to the Kalman filter with fixed Q.

Table 4.1: Comparison of mean squared error

	Kalman filter with fixed Q	Kalman filter with adaptive Q
Velocity	2.89	2.54
Yaw	0.0133	0.0028
Yaw rate	0.0089	0.002

Table 4.1 lists the mean squared error between the true state and Kalman filtered state when using fixed process noise and adaptive process noise. From the table data we can see a slight reduction in the mean squared error when using adaptive Q compared to the fixed Q.

4.5 Process Noise

The diagonal elements of the estimated adaptive Q that include longitudinal and lateral position, velocity, yaw and yaw rate are plotted in the Figure 3.12. From the figure we can see that the process noise is adapting at each time step with the changes in the vehicle state.

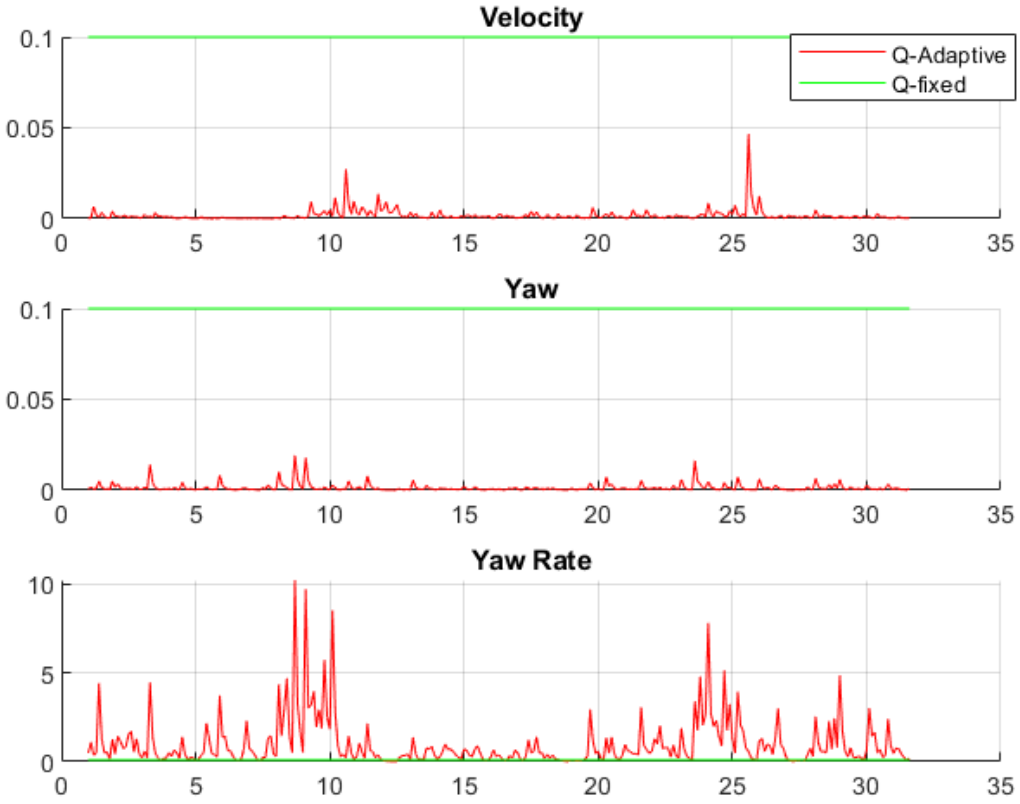

Figure 4.7: Evolution of diagonal elements of process noise covariance matrix over time

5
Discussion

5.1 Fixed Process Noise

The optimality of Kalman filters depends heavily on the assumptions that the model of the linear dynamical system is accurately determined a priori and that the process and measurement noise are zero-centered and jointly standardized Gaussian noise with known covariance matrices. For the Kalman filter to work optimally, we need all the information comprising the system dynamics and the measurement models together with the process and measurement noise. Therefore, when using the Kalman filter, it is crucial to set the covariance matrices of the process and measurement noise correctly. Here, an ad hoc approach is used where these noises are assumed to be constant during estimation and manually adjusted through a lengthy trial-and-error approach.

5.2 Adaptive Tuning Process Noise

Adaptive tuning process noise, deals with nonlinear target tracking problems associated with time-varying covariances of process noise. In this approach, the covariance of process noise and measurement noise are adjusted in real time. This results in better filtering performance compared to fixed process noise. However, compared to fixed process noise, this is a complicated approach. In addition to prediction accuracy, the approximation also has an impact on the covariance matrix. For sensors that measure positions and potentially velocities, the measurement update will not decrease the uncertainty in yaw rate. This leads to unreasonably large yaw rate uncertainties, which affects the tracking performance.

5.3 Adaptive Tuning Process Noise Using Pre-fit and Post-fit Residuals

As discussed, the main idea is to adaptively estimate the uncertain statistical properties of the noise around the nominal values. The Kalman filter can be implemented with the modified covariance matrices. Thus, an adaptive filter formulation addresses the problem of imperfect a priori information and provides a significant performance improvement over the fixed filter through the filter learning process based on the pre-fit residual or innovation sequence. In this case, the perfect knowledge of the a priori information is only of secondary importance since the new measurement

and process covariance matrices are adapted according to the learning history of the filter and the frequent adaptation of the statistical filter information through the residual sequence of the filter goes hand in hand with the idea of a dynamic system in a dynamic environment.

Even though, the estimation of process noise using the pre-fit residuals gives a slight improvement in the filter performance there are a few oscillations in the resulting states and in some cases the filter is found to respond slowly to the changes. These oscillations reduced, when the post-fit residuals were used in the estimation of process noise covariance. The small oscillations in the resulted states are acceptable in the industry and depends on the use cases and the ratio between the Q and R. Also, the filter become fast responsive. The corrected state used in the post-fit residuals will have an effect in smoothing the filtered state.

5.4 Future Work

For simplicity, this work makes a number of assumptions, so it can not really be applied to a real-world scenario yet. Thus, there is still a lot of room to improve the assumptions made and to adjust the process noise of the Kalman filter in a complex environment. Also, there is a need to improve the methods that can handle this huge amount of radar acquisitions and neglect the unwanted erroneous data to tune them in real time.

There is a scope to work with the adaptive tuning of the measurement noise (R) along with the process noise. One of our goals was to implement an adaptive Kalman filtering algorithm in an embedded platform. But we couldn't achieve it due to the lack of time. So it will be a good idea to implement the algorithm in an embedded platform as a future work.

Another disadvantage of the adaptive Kalman filter is a more complex algorithm that leads to an additional estimation block in the Kalman filter algorithm. This disadvantage is acceptable in cases where the highest accuracy is required. However, work can be done to reduce the complexity of the filter.

6
Conclusion

In this thesis work, we generated synthetic radar detections by simulating an artificial driving scenario in MATLAB. The scenario included a target vehicle and a host vehicle that was equipped with radar sensors. Then we implemented an extended Kalman filter to estimate the position, velocity, yaw and yaw rate of the target vehicle from the generated noisy radar detections. The Kalman filter was first implemented with fixed process noise. Later we modified the Kalman filter algorithm to use adaptive process noise.

The goal of this work was to come up with a better adaptive tuning techniques to overcome the drawbacks of fixed process noise, and to integrate it in an embedded platform, but could not implement it in our work.

When we compared the conventional cases with the adaptive cases, the adaptive cases are slightly better than the conventional cases. It is obvious that the adaptive method based on innovation or pre-fit residuals is better in terms of convergence speed. We could improve the response time of the algorithm, which will make it suitable to implement in a real-time system.

The post-fit residuals based method further improved the results obtained using the pre-fit residuals. Overall, all adaptive methods show stable estimation properties, and their stable values are slightly better than those of the conventional methods. Even a slightly better filter contributes significantly to improving the overall performance of the filter.

Bibliography

[1] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*. IEEE, 2015, pp. 74–77.

[2] H. Wang, Z. Deng, B. Feng, H. Ma, and Y. Xia, "An adaptive Kalman filter estimating process noise covariance," *Neurocomputing*, vol. 223, pp. 12–17, 2017.

[3] B. Ge, H. Zhang, L. Jiang, Z. Li, and M. M. Butt, "Adaptive unscented Kalman filter for target tracking with unknown time-varying noise covariance," *Sensors*, vol. 19, no. 6, p. 1371, 2019.

[4] B. M. Åkesson, J. B. Jørgensen, N. K. Poulsen, and S. B. Jørgensen, "A generalized autocovariance least-squares method for Kalman filter tuning," *Journal of Process control*, vol. 18, no. 7-8, pp. 769–779, 2008.

[5] Y. Dodge, *The concise encyclopedia of statistics*. Springer Science & Business Media, 2008.

[6] N. Carlström and A. Öqvist, "Target identification using low level radar measurements," Master's thesis, Chalmers University of Technology, Gothenburg, 2018.

[7] D. Kellner, M. Barjenbruch, J. Klappstein, J. Dickmann, and K. Dietmayer, "Instantaneous ego-motion estimation using Doppler radar," 10 2013, pp. 869–874.

[8] M. S. Grewal, *Kalman Filtering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 705–708.

[9] A. G. Gelen and A. Atasoy, "A new method for Kalman filter tuning," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 2018, pp. 1–6.

[10] G. Welch and G. Bishop, "An introduction to the Kalman filter," USA, Tech. Rep., 1995.

[11] M. I. Ribeiro, "Kalman and extended Kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, p. 46, 2004.

[12] B. Hongwei, J. Zhihua, and T. Weifeng, "IAE-adaptive Kalman filter for INS/GPS integrated navigation system," *Journal of Systems Engineering and Electronics*, vol. 17, no. 3, pp. 502–508, 2006.

[13] R. Mehra, "Approaches to adaptive filtering," *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 693–698, 1972.

[14] L. A. Fokin and A. G. Shchipitsyn, "Innovation-based adaptive Kalman filter derivation," in *2009 International Siberian Conference on Control and Communications*. IEEE, 2009, pp. 318–323.

[15] Y. Liu, X. Fan, C. Lv, J. Wu, L. Li, and D. Ding, "An innovative information fusion method with adaptive Kalman filter for integrated INS/GPS navigation of autonomous vehicles," *Mechanical Systems and Signal Processing*, vol. 100, pp. 605–616, 2018.

[16] K. V. Palem, R. M. Rabbah, V. J. Mooney III, P. Korkmaz, and K. Puttaswamy, "Design space optimization of embedded memory systems via data remapping," in *Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems*, 2002, pp. 28–37.

[17] H. J. Ferreau, S. Almér, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. L. Jerez, G. Stathopoulos, and C. Jones, "Embedded optimization methods for industrial automatic control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 194–13 209, 2017.

[18] D. Kellner, M. Barjenbruch, J. Klappstein, J. Dickmann, and K. Dietmayer, "Instantaneous full-motion estimation of arbitrary objects using dual Doppler radar," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 324–329.

[19] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation," in *2017 IEEE power & energy society general meeting*. IEEE, 2017, pp. 1–5.

[20] "Sliding window method and exponential weighting method," https://se.mathworks.com/help/dsp/ug/sliding-window-method-and-exponential-weighting-method.html, accessed: 2021-10-03.

[21] C. T. Fraser, "Adaptive extended kalman filtering strategies for autonomous relative navigation of formation flying spacecraft," Ph.D. dissertation, Carleton University, 2019.

[22] "Automated driving toolbox." [Online]. Available: https://www.mathworks.com/products/automated-driving.html