Sensor Fusion in Autonomous Navigation using Fast SLAM 3.0 – An improved SLAM Method

by

Zheng Wang

B.Sc. in Mechanical Engineering

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Aerospace Engineering

Carleton University
Ottawa, Ontario

Abstract:

This thesis introduces three important components of autonomous navigation: visual odometry and image fusion; Kalman filtering and its application; simultaneous localization and mapping (SLAM), and presents Fast - SLAM 3.0: an approach to SLAM that combines the advantages and eliminates the disadvantages of Fast SLAM 2.0 and Extended Kalman Filter (EKF) SLAM. The Fast SLAM 3.0 models the particles as the robot pose mean of a Gaussian distribution, which keeps the error covariance matrix (P) of pose estimation propagating as normal EKF SLAM. The usage of those fully functional Extended Kalman Filters in each particle allows uncertainty to be remembered over the whole trajectories, avoiding Fast SLAM 2.0's tendency to become over-confident and keeping the best feature of Fast SLAM that locally avoids linearization of the robot model and provides a high level of robustness to the clutter and ambiguous data association. Extensive simulations show that the Fast SLAM 3.0 significantly outperforms both Fast SLAM 2.0 and EKF SLAM.

Acknowledgements

I would like to thank my supervisor, Professor Jurek Z. Sasiadek for giving me the opportunity to join the Mechanical Engineering program at Carleton University, and persuading me to achieve more and reach my goals in this thesis project. There are not enough words to describe my gratitude to his advice and help during the years of research.

To all my friends who were far and worried about how I was doing far away from home during the completion of my master's degree, and to those who were close encouraging me to keep on working as hard as I could to finish this project.

To my family, who were concerned about me leaving my professional career behind to come back to university and persuade my dream of getting my master's degree. I know that for my family this project must have seemed like a strange idea. However, they were very supportive during all this time.

To Rei, thank you for being there all those long nights when I was working on this project and for supporting me and motivating me to keep on trying once and once again.

Table of Contents

List of Tables

List of Illustrations

List of Appendices

Chapter 1: Introduction

The objective of this thesis is to introduce the commonly used techniques in autonomous navigation as well as present the Fast SLAM 3.0 – a novel SLAM method.

Autonomous navigation is the technique that allow vehicle or robot moving from one place to another based on its received sensor data and control signal from local or remote computation without human aiding. Accurate localization of the agent and mapping of the surrounding environment is the core process in autonomous navigation and the foundation of posterior control signal computation. A variety of odometry and mapping methods with different types of sensors are used in autonomous navigation including wheel odometry, visual odometry, light detection and ranging (LiDAR), radar, gyro and accelerometer, etc.

The visual odometry and mapping system is widely used and studied for its noticeable advantages compared to the other localization and mapping schemes. It could accomplish localization and mapping at the same time, suffers no wheel slippage in wheel odometry, costs less and runs fast than LiDAR and is generally more accurate compared to the inertial measurement units (IMUs).

Besides all the prior mentioned advantages, the performance of visual localization and mapping system could be further improved by image fusion from identical or different types of cameras to accommodate the requirement of different application environments.

Even though the visual system has enormous advantages compared to the other sensors in autonomous navigation, it is still necessary to imply more than one type of sensors in the system. The benefits include higher robustness from the redundant system component and providing more precise estimation via the sensor fusion techniques. Since each sensor has its own noise characters, the sensor fusion techniques like Kalman filtering could

greatly reduce the error in localization and mapping.

Kalman filter is an iterative least square estimation technique that removes one noise from the lumped two noises. It minimizes the trace of the state error covariance matrix and is optimal when the noise distribution is Gaussian. The extended Kalman filter (EKF) simultaneous localization and mapping (SLAM) system and its variants are widely used in autonomous navigation.

There are mainly two categories of the SLAM system. One is online SLAM such as EKF SLAM and Fast SLAM that estimate the map and current pose of an agent. The other is full SLAM like graph-based SLAM that optimizes the whole local or global map and trajectory of an agent. This thesis focusses on the online SLAM and detailed illustrates both the advantages and disadvantages of EKF SLAM and Fast SLAM. The EKF SLAM suffers from the error caused by scene ambiguity and system linearization while Fast SLAM is more robust to this issue. But Fast SLAM has its own problem which is an overconfident issue due to low dimension limitation of particle filter and bad system modelling.

1.1 Thesis Contribution

This thesis presents Fast - SLAM 3.0: an approach to SLAM that combines the advantages and eliminates the disadvantages of Fast SLAM 2.0 and Extended Kalman Filter (EKF) SLAM. The Fast SLAM 3.0 models its particles as the mean of an EKF SLAM which keeps the error covariance matrix (P) propagating instead of setting it to zero after sampling. The usage of those fully functional Extended Kalman Filters in each particle allows uncertainty to be remembered over the whole trajectories, avoiding Fast SLAM 2.0's tendency to become over-confident and keeping the best feature of Fast SLAM that locally

avoids linearization of the robot model and provides a high level of robustness to the clutter and ambiguous data association. Extensive simulations in the randomly generated environment show that the Fast SLAM 3.0 significantly outperforms both Fast SLAM 2.0 and EKF SLAM. The advantages of Fast SLAM 3.0 are most pronounced in those long trajectories of the cluttered environments where either pure approach has high chances to fail.

1.2 Thesis Summary

The remainder of the paper is structured as follows. Chapter two introduces the visual odometry and the main elemental procedures that combines a visual odometry system and their advantages and disadvantages. Chapter three gives a brief review of image fusion and demonstrates some examples of image fusion applications that are useful in autonomous navigation. In Chapter four, there is a brief introduction of the Kalman filter and then follows a lot of application simulation examples which mainly focus on sensor fusion and navigation purposes. Chapter five describes the EKF SLAM, Fast SLAM system and detailed illustrate their advantages and disadvantages. Then demonstrates the novel Fast SLAM 3.0 algorithm and its potential advantages compared to the other on-line SLAM methods. Chapter six is the conclusion.

Chapter 2: Visual Odometry

Visual odometry is the technique that uses images to estimate the 3D or 2D motion of the agent pose, which is a critical and fundamental component of autonomous navigation.

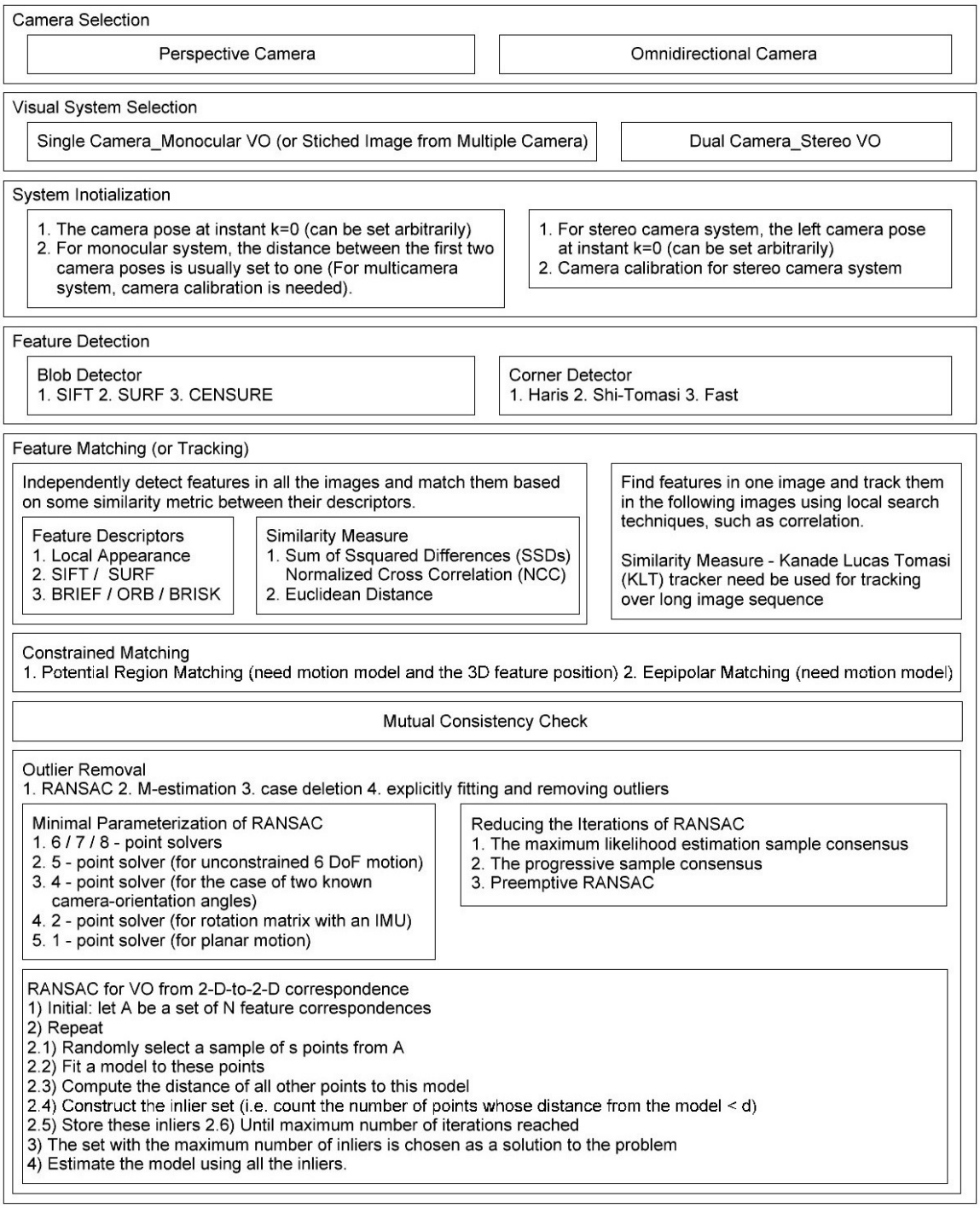


Figure 1: Diagram of feature-based visual odometry (a)

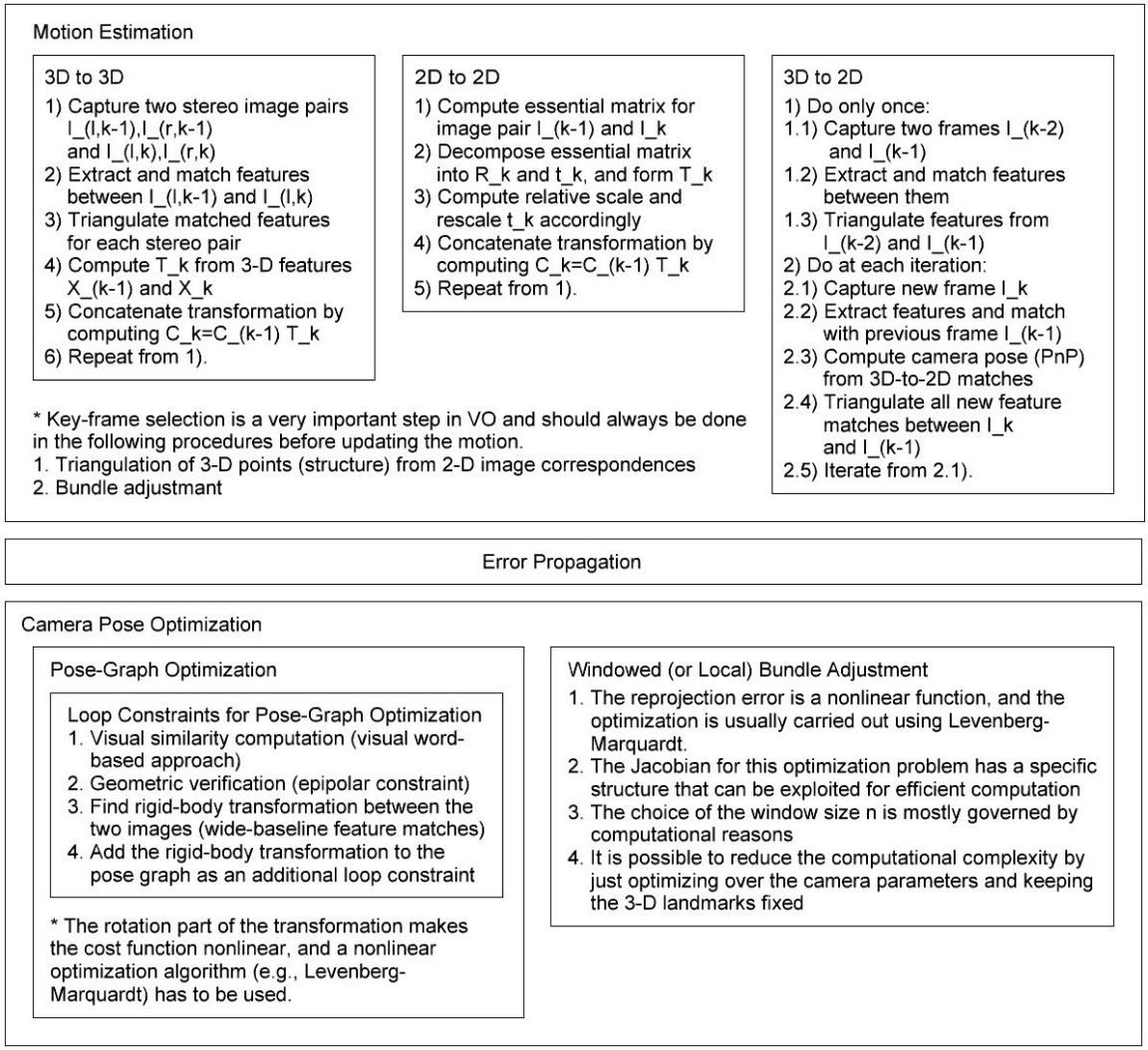


Figure 2: Diagram of feature-based visual odometry (b)

Visual odometry has noticeable advantages compared to the other odometry methods. For example, wheel odometry suffers from the precision problems where wheels tend to slip and slide on the low-friction surface, which leads to the difference between wheel odometry data and true trajectory.

2.1 Camera Selection

The first step of building a visual odometry system is to decide the type of camera that depends on the operating environment. The Omnidirectional camera has a wide field of view (could be more than 180°), which could capture more features for robust visual

odometry but also has decreased per-pixel accuracy and severe image distortion which would introduce more modelling error. The Perspective camera has a smaller field of view which leads to higher per-pixel accuracy and has small image distortion. Hence, its camera modelling error is smaller than the Omnidirectional camera. Illumination situations is another important consideration for camera type selection. The Infrared camera is better than the optical camera in the night operation environment [1]. The urban street has a highly intensive illumination variance which requires the camera has high dynamic ranges.

2.2 Visual System Selection

The visual system in VO could be classified into two categories: monocular (single-camera or stitched images from multiple cameras) VO and stereo VO [2].

In monocular VO, since the baseline of two imaging poses is not known so only bearing information is available. The disadvantage is that motion can only be recovered up to a scale factor [3]. The absolute scale can then be determined from direct measurements (e.g., measuring the size of an element in the scene), motion constraints, or from the integration with other sensors, such as IMU, air-pressure, and range sensors [3]. While in stereo VO, the relative 3-D position of the features is directly measured by triangulation at every robot location and used to compute the relative motion. Trinocular methods belong to the same class of algorithms [3]. Another advantage of the stereo camera scheme is that matches need to be computed only between two views instead of three views as in the monocular scheme. Additionally, since the 3D structure is computed directly from a single stereo pair rather than from adjacent frames as in the monocular case, the stereo scheme exhibits less drift than the monocular one in case of small motions [2]. In that case, it could be used as a complementary sensor for monocular VO to limit the drift of the monocular VO system,

detecting obstacles nearby or working in very small movement speed.

Stereo VO can degenerate to the monocular case when the distance to the scene is much larger than the stereo baseline (i.e., the distance between the two cameras). In this case, stereo vision becomes ineffective and monocular methods must be used [3].

2.3 System Initialization

The camera poses at instant $t = 0$ should have an initial value that could be set arbitrarily. For the monocular system, the distance between the first two camera poses is usually set to one and for the multicamera system, camera calibration is needed [3]. For the stereo camera system, the pose of the left camera at instant $t = 0$ should have an initial value that could be set arbitrarily and camera calibration for stereo cameras is needed [4].

2.4 Feature Type Selection

There are three popular pose estimation methods: feature-based method, appearance-based method and the hybrid method which combines two of them [5]. Feature-based methods are based on salient and repeatable features that are tracked over the frames. Appearance-based methods use the intensity information of parts or all the pixels in the image or subregions of it. Hybrid methods use a combination of the previous two [4].

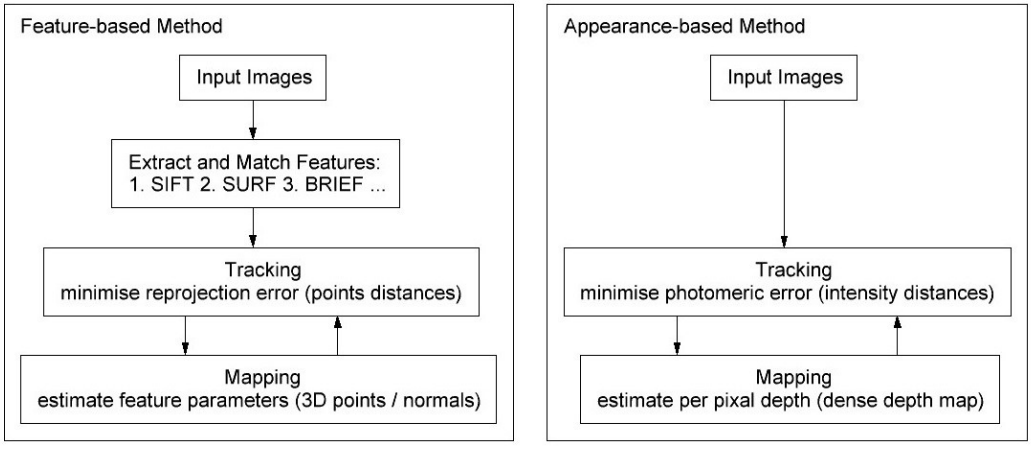


Figure 3: Feature-based and appearance-based method

Most of the appearance (density or global)-based methods are easier of implementation compared to feature-based methods and works better at high speed especially high-speed indoor operation where the image would be blurred seriously. But they are computationally more expensive and has less accuracy in long distance. Appearance-based methods can use and reconstruct the whole image which is good for parallelism but also slower in long trajectory. It is difficult to remove outliers retroactively in these methods and they are not robust to inconsistencies in the model or system (rolling shutter), so they are inflexible compared to feature-based methods. The decisions (linearization point) of global methods based on more complete information, which needs good initialization [6].

Feature-based methods require the ability to robustly match or track features across frames but work better at low speed, large scale environment where there are enough steady cleared features for positioning. Besides, they are faster and more accurate than most global methods in long-distance estimation. Therefore, most VO implementations are feature-based. Feature-based methods only use and reconstruct corners. Hence, they are faster compared to global methods in the long distance. They are also more flexible since the outliers could be removed retroactively and they are robust to inconsistencies in the model or system (rolling shutter). The decisions (Key Point detection) of feature-based methods based on less complete information, which does not need good initialization [6].

It is noticeable that Christian Forster etc. proposed a semi-direct VO [7] that combines the success-factors of feature-based methods (tracking many features, parallel tracking and mapping, keyframe selection) with the accuracy and speed characters of global methods in short distance operation. Their experiment of high frame-rate VO for MAVs showed this method has increased robustness to faster flight maneuvers.

Corner detectors and Blobs detectors are two popular feature detection methods for feature-based VO. Corner detectors are fast to compute but are less distinctive, whereas blob detectors are more distinctive but slower to detect [5]. And corners cannot be redetected as often as blobs after large changes in scale and viewpoint [5]. Hence, the choice of the appropriate feature detector should be carefully considered, depending on the computational constraints, real-time requirements, environment type, and motion baseline (i.e., how nearby images are taken) [4].

Table 1: Comparison of common feature detection methods

	Harris	Shi-Tomasi	Fast	SIFT	SURF	CENSURE
Corner Detector	●	●	●	○	○	○
Blob Detector	○	○	○	●	●	●
Rotational Invariant	●	●	●	●	●	●
Scale Invariant	○	○	●	●	●	●
Affine Invariant	○	○	○	●	●	●
Repeatability	+++	+++	++	+++	+++	+++
Localization Accuracy	+++	+++	++	++	++	++
Robustness	++	++	++	+++	++	+++
Efficiency	++	++	++++	+	++	+++

SSD / NCC / SIFT / SURF / BRIEF / ORB / BRISK / Census are commonly used feature descriptors. Contrary to SSD, NCC compensates well for slight brightness changes. In fact, SSD and NCC are not invariant to any of these changes, therefore, their use is limited to images taken at nearby positions [5]. SIFT, SURF, and CENSURE are not true affine invariant detectors but were empirically found to be invariant up to certain changes of the viewpoint [5]. The SIFT descriptor proved to be stable against changes in illumination, rotation, scale, and even up to 60° changes in viewpoint [5]. It could be computed for corner or blob features. However, its performance will decrease in corners. BRIEF, ORB and BRISK are much faster to compute than SIFT and SURF [5].

Features should cover the image as evenly as possible. The image could be partitioned

into a grid, and feature detectors applied to each cell by tuning the detection thresholds until a minimum number of features are found in each sub-image [8]. As a rule of thumb, 1,000 features are a good number for a 640 x 480-pixel image [5].

An alternative to point features for VO is to use lines or edges, but lines are more difficult to match because lines are more likely to be occluded than points [5]. Furthermore, the origin and end of a line segment of edges may not exist.

2.5 Feature Matching

Feature matching consists of detecting features independently in all the images and then matching them based on some similarity metrics. Feature tracking consists of finding features in one image and then tracking them in the next images using a local search technique, such as correlation [5]. Feature tracking is more suitable when the images are taken from nearby viewpoints, whereas feature matching is more suitable when a large motion or viewpoint change is expected. in the last few decades, the focus has shifted to large-scale environments, and so the images are taken as far apart as possible from each to limit the motion-drift-related issues [5].

2.6 Outlier Removal

Matched points are usually contaminated by outliers which are wrong data associations. Possible causes of outliers include image noise, occlusions, blur, and changes in viewpoint and illumination for which the mathematical model of the feature detector or descriptor does not account for [5]. For the camera motion to be estimated accurately, it is important that outliers be removed.

Random sample consensus (RANSAC) is a standard method for motion estimation in the presence of outliers. If the camera motion is unconstrained, the minimum number of

points to estimate the motion is five and the 5-point RANSAC (or the 6 / 7 / 8-point one) should be used. Compare to 6 / 7 / 8-point solvers, the 5-point solver has the advantage that it works also for planar scenes [5]. Preemptive RANSAC [9] has been the most popular one because the number of iterations can be fixed a priori, which has several advantages when an on-line operation is necessary [5]. If the high-speed requirement is the most concerned issue, using a minimal point set is better than using a nonminimal set. However, even the 5-point RANSAC may not be the best idea if the image correspondences are very noisy. In this case, using more points than a minimal set is proved to give better performance in terms of accuracy and number of inliers [10] [11].

In many cases of SIFT Matching, it might be beneficial to skip the ratio test and let RANSAC take care of the outliers [5].

2.7 Motion Estimation

3D to 3D / 3D to 2D / 2D to 2D Point Registration are three commonly used motion estimation methods [3]. Comport et al. relied on the quadrifocal tensor, which allows motion to be computed from 2D to 2D image matches without having to triangulate 3D points in any of the stereo pairs. The benefit of using directly raw 2D points instead of triangulated 3D points lays in a more accurate motion computation. The 2D to 2D case requires a minimum of 5-point correspondences. However, only 3 correspondences are necessary in the 3D to 2D motion case. This lower number of points leads to much faster motion estimation.

As pointed out by Nistér et al., motion estimation from 2D to 2D and 3D to 2D correspondences is more accurate than from 3D to 3D correspondences, because it minimizes the image reprojection error instead of the 3D to 3D feature position error. In

monocular 3D to 2D point registration, the main challenge is to maintain a consistent and accurate set of triangulated 3D features and to create 3D to 2D feature matches for at least three adjacent frames [3]. In the monocular scheme, the 2D to 2D method is preferable compared to the 3D to 2D case since it avoids point triangulation. However, in practice, the 3D to 2D method is used more often than the 2D-to-2D method. The reason lies in its faster data association [3].

2.8 Pose Optimization

Pose optimization in VO includes global optimization (V-SLAM) and local optimization (Bundle Adjustment). A V-SLAM method is potentially much more precise, because it enforces many more constraints on the path, but not necessarily more robust (e.g., wrong data association in loop closing could severely affect the map consistency) [3]. Besides, it is more complex and computationally expensive. In the end, the choice between VO and V-SLAM depends on the tradeoff between performance and consistency, and simplicity in implementation [3]. Although the global consistency of the camera path is sometimes desirable, VO trades off consistency for real-time performance, without the need to keep track of all the previous history of the camera [3].

2.9 VO along / VO with Other Sensors

Olson et al. [12], [13] showed that the use of camera ego-motion estimate alone results in accumulation errors with growth faster than a linear function in the distance travelled, leading to increased orientation errors. Conversely, when an absolute orientation sensor is used, the error growth can be reduced to a linear function of the distance travelled. Olson et al. [12] also showed that with the assistance of an absolute orientation sensor, the linearization error of vehicle pose could be greatly reduced.

Chapter 3: Image Fusion

Image fusion is the technique that fuses two or more images into one that the output image contains more information than either of the single image to help the following manual or computer process. In this case, the benefits of image fusion include an extended range of operation, extended spatial and temporal coverage, reduced uncertainty, higher reliability, robust system performance and compact representation of information [14].

Image fusion of multiple cameras with different dynamic ranges could provide the image with more features than a single camera which leads to a more robust visual odometry. Figure 4 shows the image fusion example of 4 images taken at 1/30, 1/4, 2.5 and 15 seconds. The outcome image demonstrates better visibility than any of the original images.

Figure 4: High dynamic image fusion

Image fusion could make the visual odometry more accurate because the single-camera visual odometry has high accuracy on horizontal and vertical direction but large uncertainty

in depth while a wider view (larger than 180°) would greatly reduce the error on depth. Figure 5 shows a mosaic image stitched from three sub-images.

Figure 5: Multiple image fusion

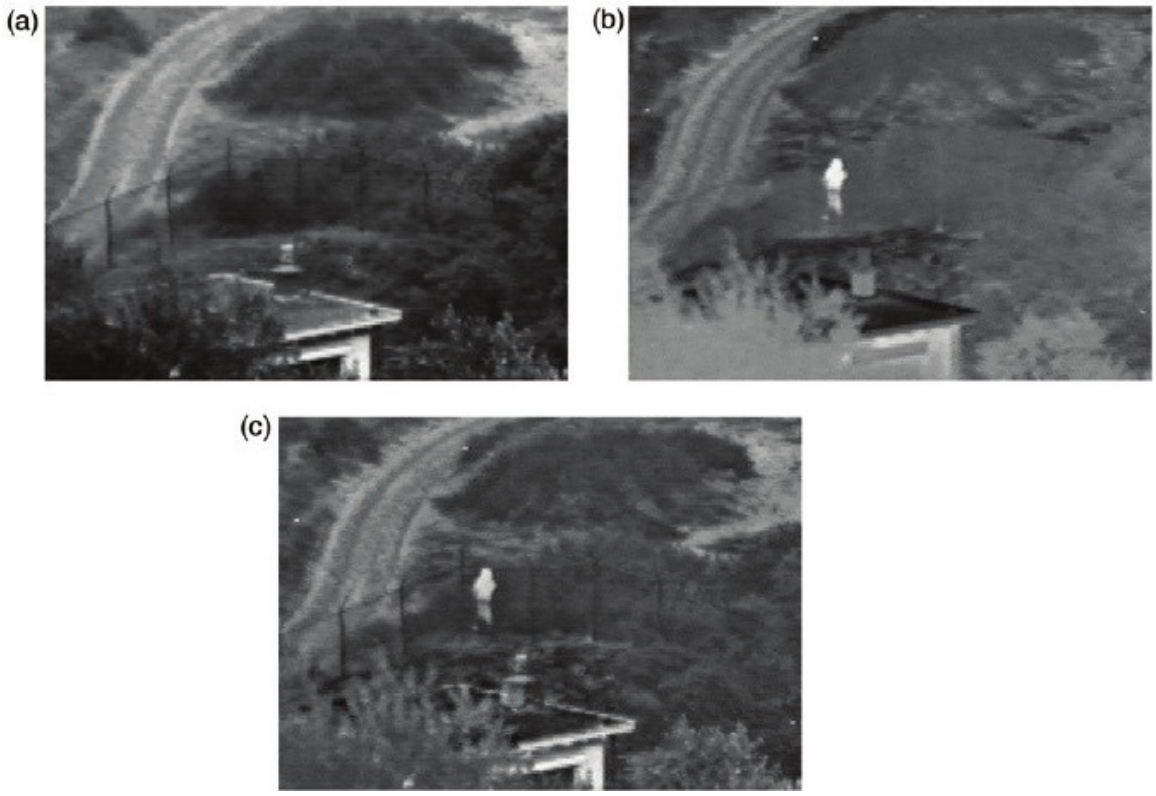


Figure 6: Optical-infrared image fusion

The third example illustrates a head-tracked vision system for night vision applications.

Its multiple imaging sensors employed can enhance the user's overall situational awareness. Figure 6 (a) shows the image-intensified CCD (IICCD) sensor image of the scene, and Figure 6 (b) shows the corresponding thermal imaging forward-looking-infrared (FLIR) sensor image of the same scene. These images contain complementary features as illustrated by the fused image shown in Figure 6 (c) [15].

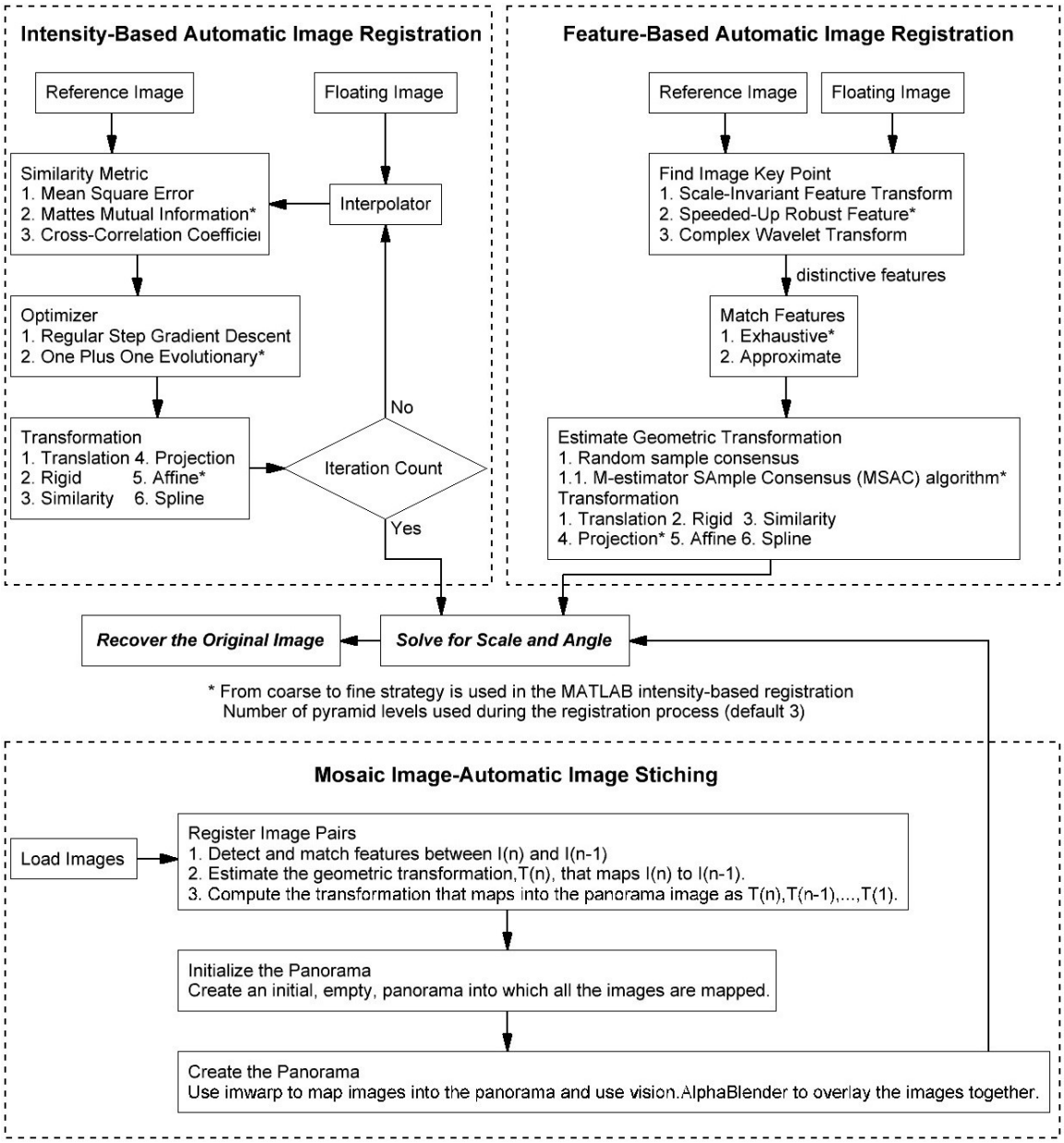


Figure 7: Diagram of image fusion

Image registration is often used as a preliminary step in image fusion. There are two widely used image registration scheme: feature-based registration and intensity-based registration. Intensity-based registration achieves the transformation by optimizing the similarity metric of per-pixel intensity value based on an initial guess. While feature-based registration calculates the transformation depends on the matched features. Intensity-based registration could be improved by starting with more simple transformation types like 'rigid', and then use the resulting transformation as an initial condition for more complicated transformation types like 'affine' [14]. Hierarchical Registration may be used for images to undergo local deformations.

3.1 Pixel Fusion Algorithms and Their Applications

After the input images are spatially and temporally aligned, semantically equivalent and radiometrically calibrated, fusion techniques that rely on simple pixel operations on the input image values are then applied to the images. The techniques include the basic arithmetic operations, logic operations and probabilistic operations as well as slightly more complicated mathematical operations [14]. The image values include pixel gray-levels, feature map values and decision map labels. Although more sophisticated techniques are available, simple pixel operations are widely used in many image fusion applications.

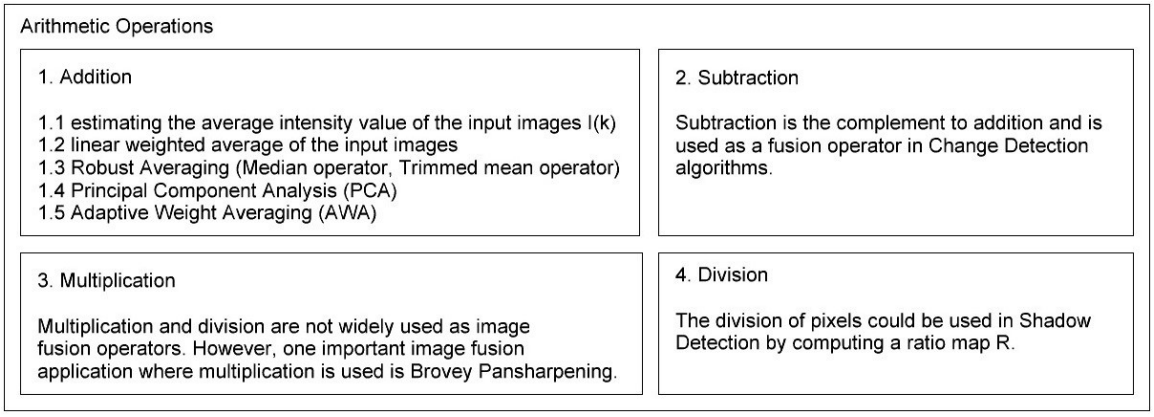


Figure 8: Diagram of pixel-level fusion (a)

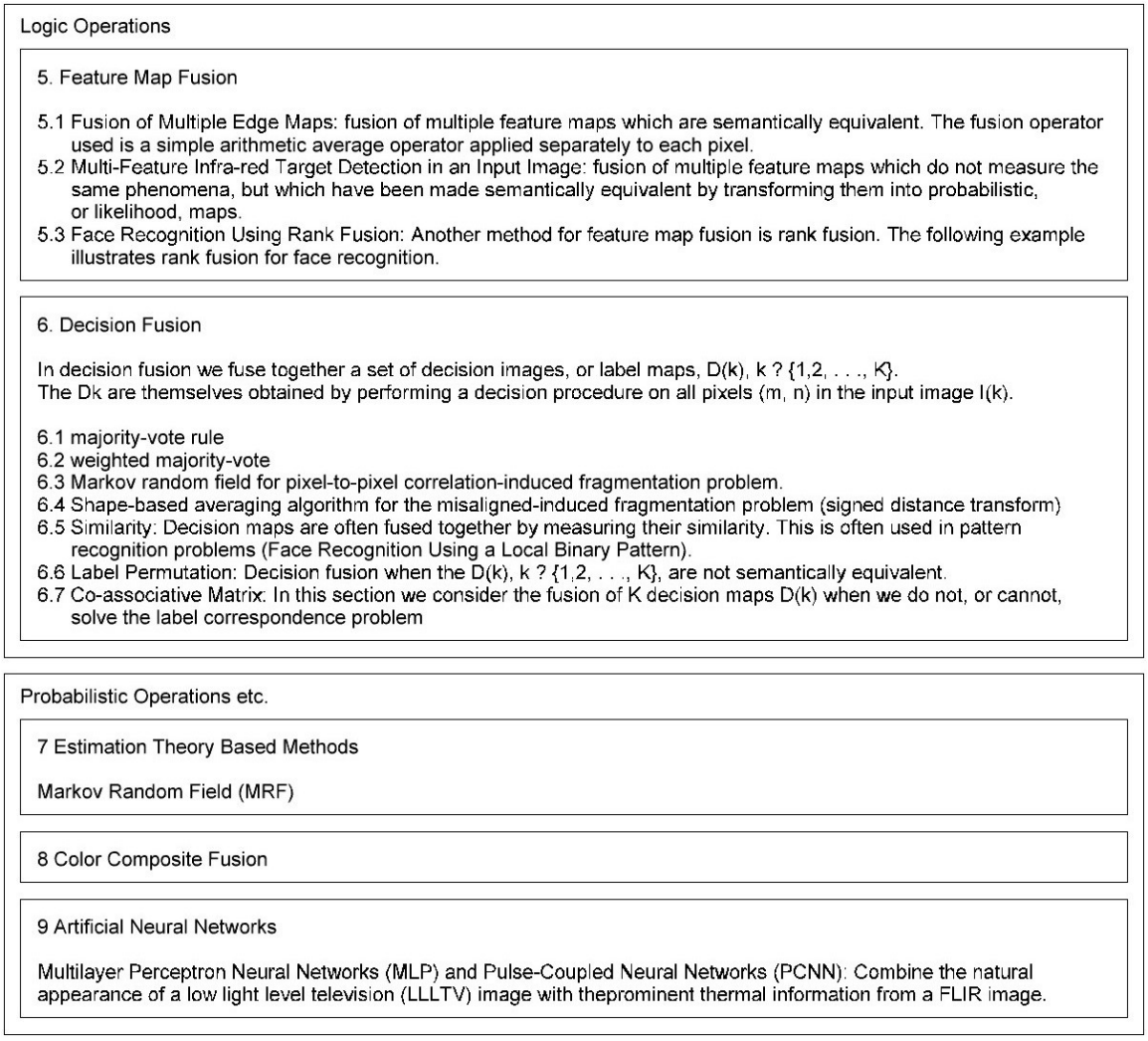


Figure 9: Diagram of pixel-level fusion (b)

3.2 Multiscale Image Transform

Discrete Wavelet Transform (DWT) provides a framework for the multi-resolution analysis of an input image by decomposing an input image into a sequence of wavelet planes and a residual image [14]. It is widely used in feature extraction (SIFT), image registration, image fusion such as fusion of an optical image and an infrared image, pan-sharpening in which we fuse a high spatial resolution panchromatic image with a low spatial resolution multi-spectral image and scale-invariant feature transform [14].

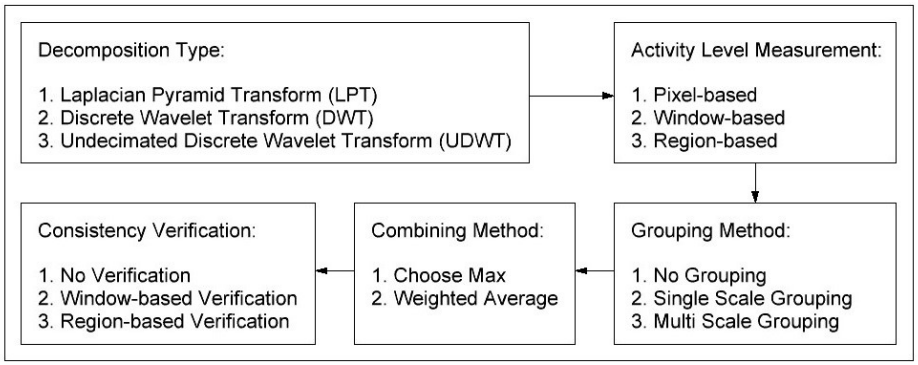


Figure 10: Diagram of multiscale image fusion

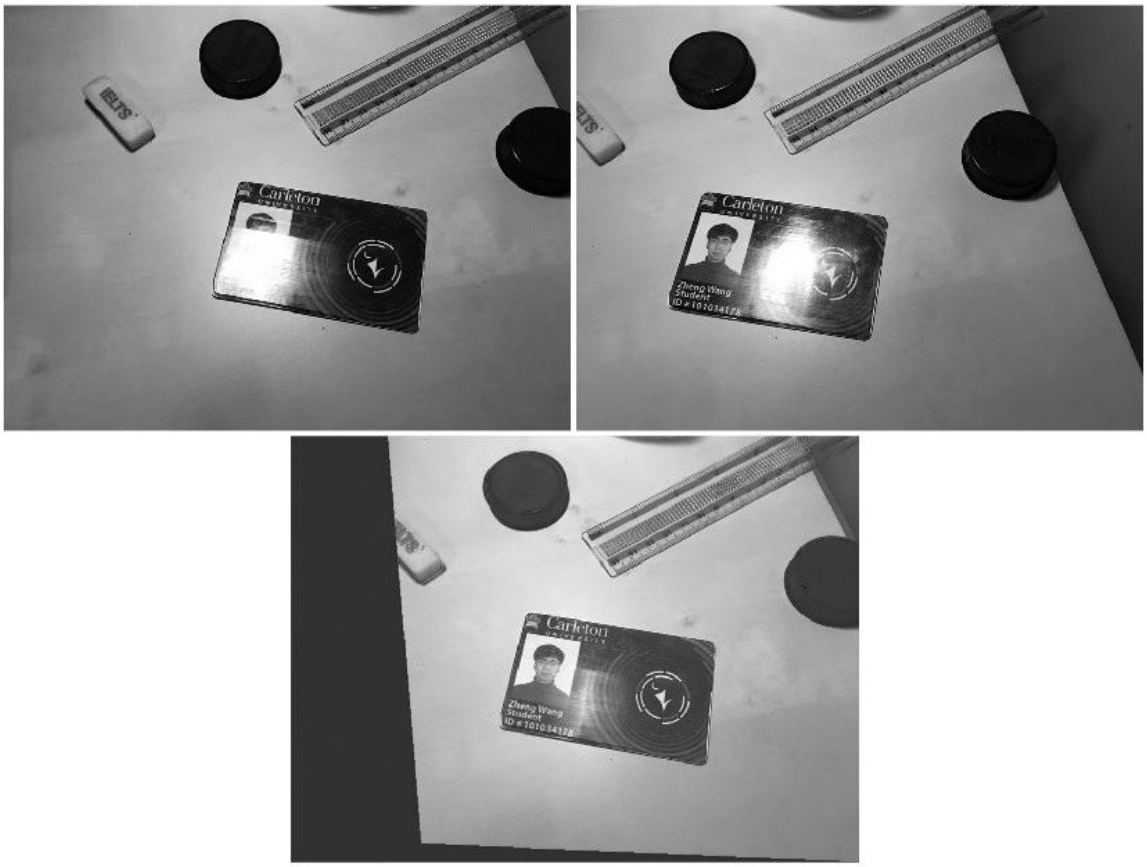


Figure 11: Discrete wavelet transform image fusion

Figure 11 shows an example of removing glare from photos with image fusion and discrete wavelet transform. Two images are taken from a slightly different angle and have a glare effect on a different location. The result shows this method works well in this example, most of the glare effect is removed.

Chapter 4: Kalman Filtering

Kalman filter is a complementary recursive least square filter proposed by R. E. Kalman based on the Bayes filter and Wiener filter that applied to the state space. It combines different sensors' data based on their noise characteristic and minimizes the trace of the state estimate error to generate a better estimation of the trajectory and surrounding,

Bayes filter is the foundation of the Kalman filter which provides a general framework for recursive state estimation based on the given motion and sensor model.

$$\boldsymbol{prediction}: \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1} \quad (1)$$

$$\boldsymbol{correction}: bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t) \quad (2)$$

where x_t is the state of robot and maps at time t and u_t is the control signal from time $t-1$ to t. $\overline{bel}(x_t)$ denotes the prediction of the states, $bel(x_t)$ denotes the corrected state estimate. η is the parameter that normalizes the probability.

Wiener filter is the technique that estimates the desired or target's random process by linear time-invariant (LTI) filtering of an observed noisy process, assuming the known stationary signal, noise spectra and additive noise. The Wiener filter minimizes the mean square error between the estimated random process and the desired process.

$$\bar{S}(s) = G(s)[S(s) + N(s)] \quad (3)$$

where $\bar{S}(s)$ is the estimated random signal, $S(s)$ is the true random process, $N(s)$ is the random noise and $G(s)$ is the filter transfer function. The essence of accurate localization is removing the noise from estimate or measurement. Wiener filter works as Kalman gain in a complementary filtering process that extracts prediction noise from the combination of prediction and measurement noise, then eliminates the prediction noise from $\overline{bel}(x_t)$.

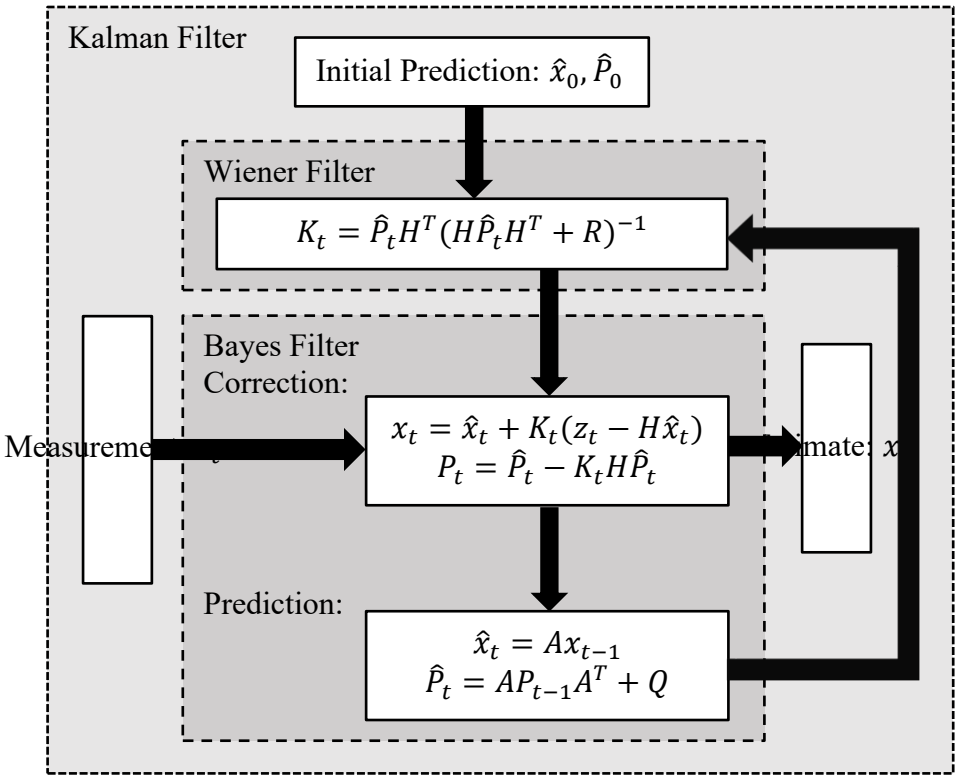


Figure 12: Kalman filter algorithm

External Input	z_t: measurement at time t
Final Output	x_t: state estimate at time t
System Model	A: system state transfer function from t to t+1
	H: transfer function from state to measurement
	Q: error covariance of state transition noise
	R: error covariance of measurement noise
Internal Computation Parameter	$\hat{x}_t$: prediction of the system state of time t
	$\hat{P}_t$: prediction of error covariance of system state estimate of time t
	$\hat{P}_t$: error covariance of system state estimate of time t
	K_t: Kalman gain of time t

In the first step, Kalman gain K is being computed based on the initial prediction new $\hat{x}$ and $\hat{P}$. Then in the correction step, the state vector x and the error covariance of state estimate P are being corrected with the Kalman gain and the measurement data. In the prediction step, the new $\hat{x}$ and $\hat{P}$ are being calculated based on the corrected x and P. Then a new filtering circle for time t+1 is started.

4.1 Recursive Filter

4.1.1 Average Filter

The average filer uses a constant gain to divide all the data to remove noise, it is useful in processing constant data white noise. The following example shows a noise suppression process of the measurement data of a new battery's voltage. The voltage measurement data was gathered for a certain period and then the average value of the data was calculated [16].

Table 2: Truth model and measurement parameters.

Truth Model (Voltage)	14.4	V
Mean Value of the Noise	0	V
Standard Deviation of the Noise	4	V
Measurement Interval	0.2	sec
Simulation Time	10	sec

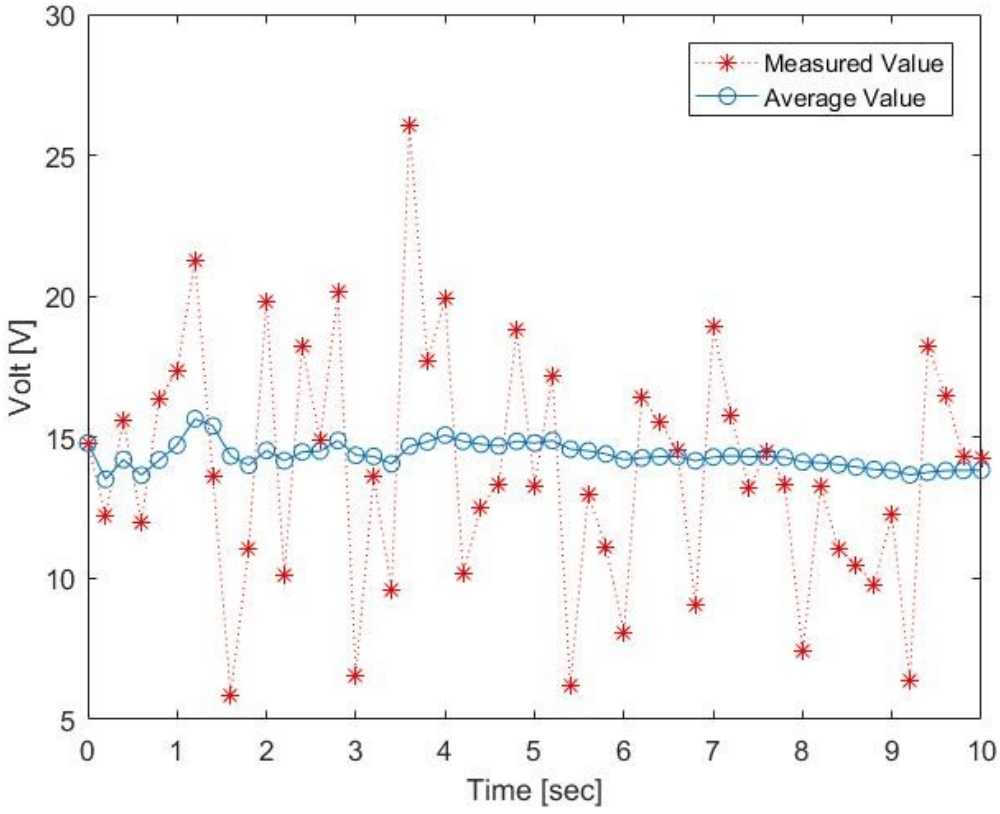


Figure 13: Average filtering for noisy voltage measurement.

The measurement value in Figure 13 shows violent variation but the output value from

the average filter shows a stable pattern. As data accumulates, measurement noise subsides, and the output value approaches the true value of the voltage.

4.1.2 Moving Average Filter

The Moving Average filer uses a factor to divide several adjacent instead of the whole data set to suppress the noise. The following example shows a sonar sensor on a helicopter that generates the measurement data of vertical distance to the ground surface for a certain period. The moving average filter is used to remove the noise due to the vibration of the helicopter, condition of the ground surface etc [16].

Table 3: Measurement parameters and filter setting.

Measurement (Altitude)	discrete sonar measurement data	m
Measurement Interval	0.02	sec
Simulation Time	10	sec
Buffer Size	10	/

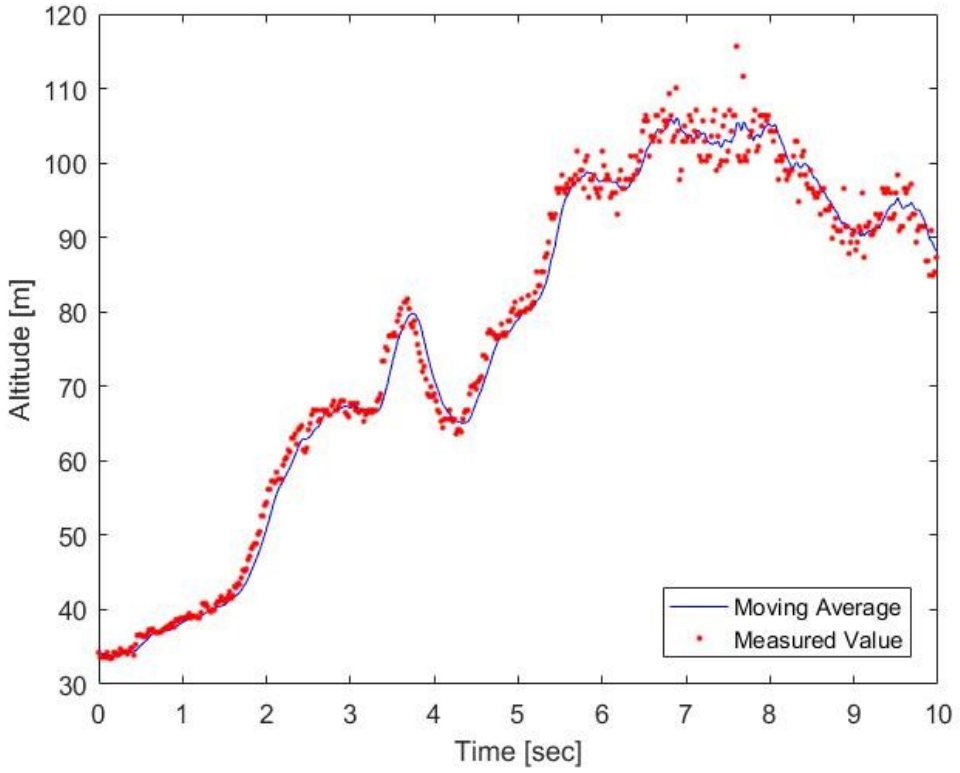


Figure 14: Moving average filtering for noisy sonar measurement.

It could be seen in Figure 14 that there are slight delays in the output from the moving average filter because of the averaging process. Hence, one should carefully select the size of the buffer in the moving average filter to balance the delay effect and noise suppression.

4.1.3 Low-Pass Filter

A low-pass filter (LPF) is a filter that passes signals with a frequency lower than a selected cut off frequency and refuses signals with frequencies higher than the cut off frequency. The exact frequency response of the filter depends on the filter design. The filter is also called a high-cut filter or treble-cut filter. A low-pass filter is the complement of a high-pass filter. The following example shows a sonar sensor on a helicopter that generates the measurement data of vertical distance to the ground surface for a certain period. The first-order low-pass filter is used to remove the noise due to the vibration of the helicopter, condition of the ground surface etc. Three first-order low-pass filters with different gain parameters are used for this application to show the effect of different gain on the filter output [16].

Table 4: Measurement parameters and filter gains.

Measurement (Altitude)	discrete sonar measurement data	m
Measurement Interval	0.02	sec
Simulation Time	10	sec
Filter Gain	0.4 / 0.7 / 0.9	/

Figure 15 shows the results of three filters. The filter with gain equal to 0.4 assigns more weight on recent measurement. It is more sensitive to the variation of measured data but has a poor noise suppression effect. The filter with gain equal to 0.9 assigns more weight on old measurement. It is not quite sensitive to the variation of measured data but has a better noise suppression effect. The filter with gain equal to 0.7 achieves a good balance between the delay effect and noise suppression.

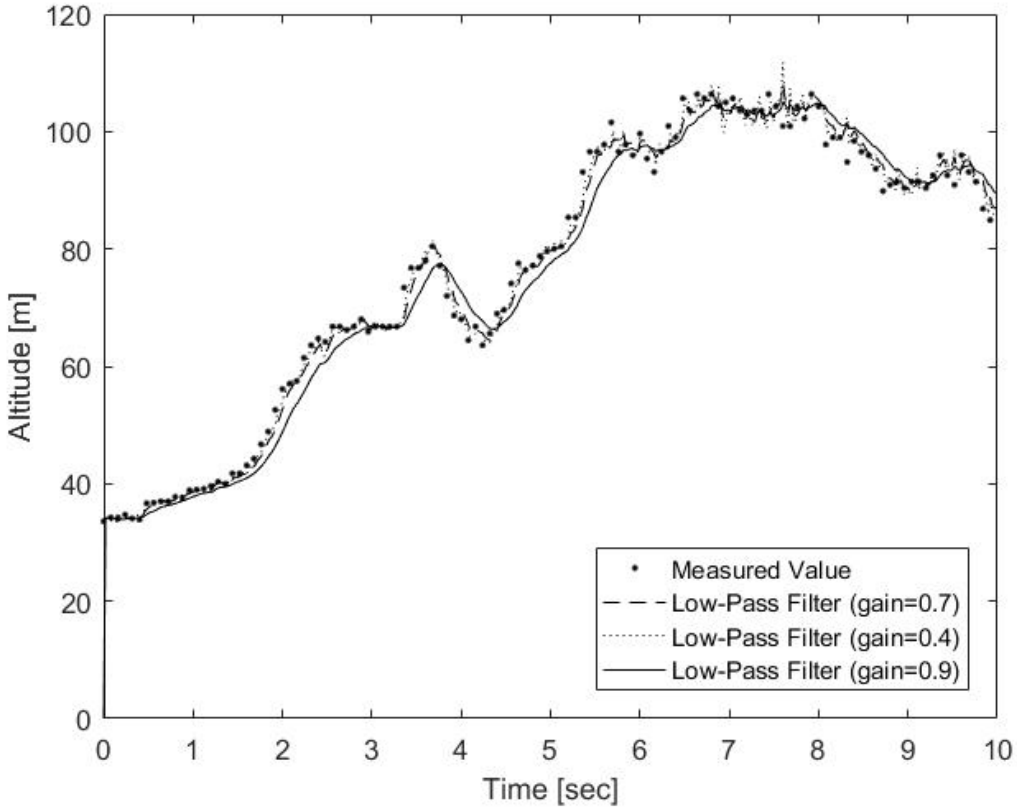


Figure 15: Low-pass filtering for noisy attitude measurement.

4.1.4 High-Pass Filter

A high-pass filter is the complement of a low-pass filter. It passes signals with a frequency higher than a selected cut off frequency and attenuates signals with frequencies lower than the cut off frequency. The following example has a similar background setting as example 4.1.3. except that in this case, one wants to use the high-pass filter to extract the noise due to vibration of the helicopter, condition of the ground surface etc [16].

High-pass filter model:

$$G(s) = \frac{s}{s+a} \text{ where } a = 42.918.$$

Discrete-recursive filter expression in time-domain:

$$x_k = \frac{\tau}{\tau+\Delta t} x_{k-1} + \frac{\tau}{\tau+\Delta t}(z_k - z_{k-1}) \text{ where } \tau = a^{-1}, z \text{ is sonar measurement.}$$

Table 5: Measurement parameters.

Measurement (Altitude)	Discrete sonar measurement data	m
Measurement Interval	0.01	sec
Simulation Time	5	sec

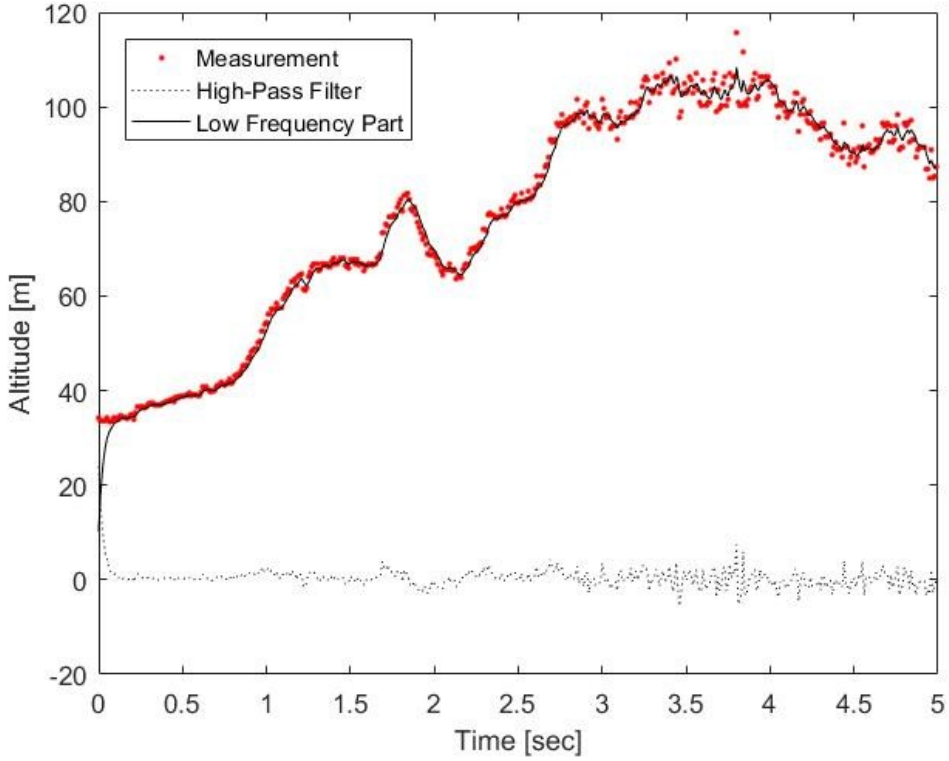


Figure 16: High-pass filtering for noisy altitude measurement.

Figure 16 shows the extracted high pass noise and filter result. The dotted line in the plot shows the output of the high-pass filter. The solid line is the data that the high-pass filter result subtracted from the measurement. Comparing this plot with the result from the low-pass filter in Example 4.1.3, it could be verified that the high-pass filter has separated the noise in the input signal successfully.

4.1.5 Complementary Filter with PI

Wiener's theory demands that the signal be noise-like in character. This is not the case in most autonomous navigation problems. Because the signal usually has at least some deterministic structure, so it is often not reasonable to assume it to be completely random.

Thus, the complementary filtering method is needed for this situation. Figure 17 shows the structure of the complementary filter. Clearly, the signal $s(t)$ is not affected by the filter gain $G(t)$ in any way. On the other hand, the two noise inputs are modified by the complementary transfer function $G(s)$ and $[1 - G(s)]$ [17].

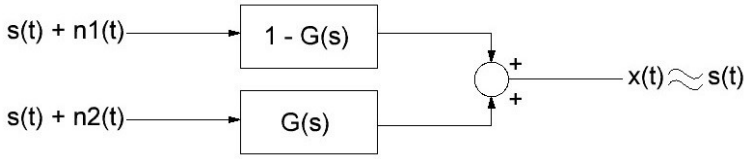
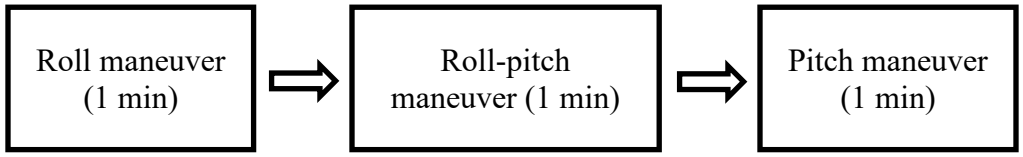


Figure 17: Complementary filter

The following example shows a simple application example of complementary filter. The background setting is that one wants to measure the horizontal altitude of a helicopter with the accelerometers and gyros. A test signal is applied to oscillate the navigation sensor with a sinusoidal wave of 0.2 Hz frequency and ±30° amplitude. The complementary filter is used to extract altitude angle rate noise from the measurement residual (altitude angle measurement from gyro minus altitude angle measurement from the accelerometer).

The procedure for the test was set as follows [16]:



The data collected from the test were accelerations and angular velocities along and about all three axes with a sampling rate of $0.01\ s$.

Gyro process model:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi/cos\theta & cos\phi/cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

where $\dot{\phi}, \dot{\theta}, \dot{\psi}$ are Euler angle rates. p, q, r are the angular rates from gyros.

Accelerometer process model:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} 0 & w & -v \\ -w & 0 & u \\ v & -u & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + g \begin{bmatrix} sin\theta \\ -cos\theta sin\phi \\ -cos\theta cos\phi \end{bmatrix}$$

where $a_x,\ a_y,\ a_z$ are accelerations along the x, y, z axis. u, v, w are the velocities along each axis in body frame. p, q, r are the angular rates about each axis in body frame. g is the gravity constant.

In this case, system moving at constant velocity $\left(\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = 0 \right)$, thus $\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = 0.$

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = g \begin{bmatrix} sin\theta \\ -cos\theta sin\phi \\ -cos\theta cos\phi \end{bmatrix};\ \begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{bmatrix} sin^{-1}\left(\frac{-a_y}{g cos\theta}\right) \\ sin^{-1}\left(\frac{a_x}{g}\right) \end{bmatrix}$$

Filter expression in the frequency domain:

$$\phi_m = [1 - G(s)]\left(\frac{1}{s}\dot{\phi}_g\right) + G(s)\phi_a$$

where $G(s) = \frac{sK_P + K_I}{s^2 + sK_P + K_I}$, ϕ_m is the estimate of altitude. ϕ_g is the gyro measurement, ϕ_a is the measurement of the accelerometer.

Transfer function of the PI controller in the frequency domain:

$$K_P + \frac{K_I}{s} = \frac{sK_P + K_I}{s}$$

where $K_I = 0.01,\ K_P = \sqrt{2} \times 0.1$

Discrete form of the PI controller in the time domain:

$$\frac{y_{k+1}}{u_{k+1}} = \frac{0.1415 - 0.1414z^{-1}}{1 - z^{-1}}$$

where y_k is the estimate of gyro noise. u_k is the combination of gyro noise plus accelerometer noise.

Table 6: Measurement parameters.

Measurement (Attitude)	Discrete sonar measurement data	m
Measurement Interval	0.01	sec
Simulation Time	415	sec

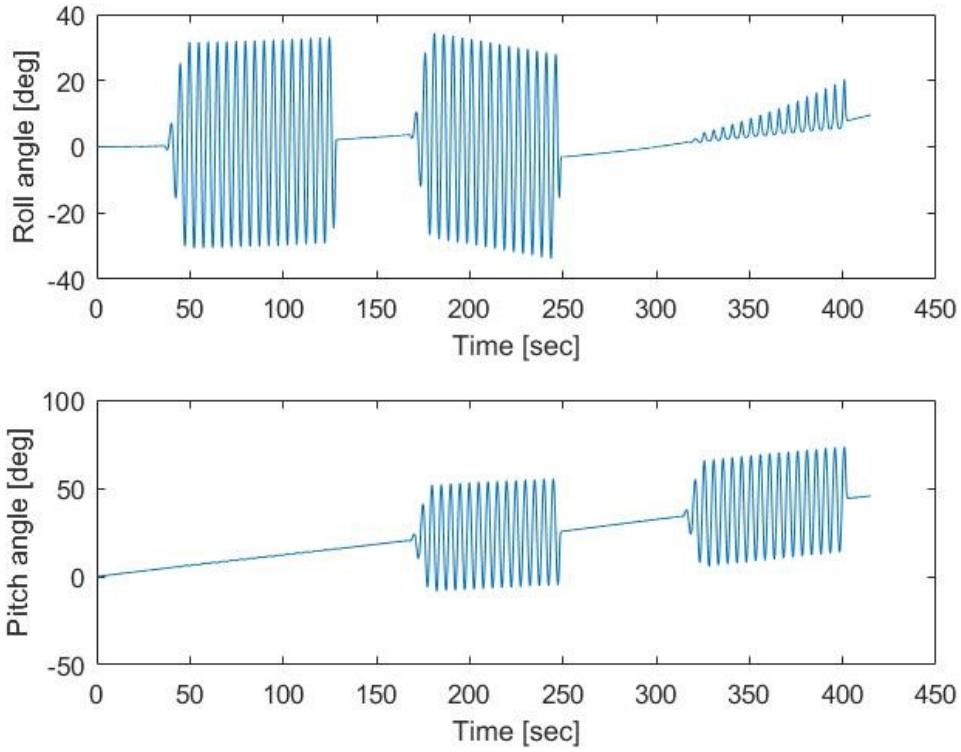


Figure 18: Euler angle measurement of gyro sensors.

Figure 18 shows the attitude obtained by numerical integration of the angular velocity. The data capture dynamic trend well but due to the error accumulation, it drifts off from the true value. Looking from the perspective of time, a gyro is relatively accurate for short period measurement but inaccurate for long period variation.

Figure 19 shows the Euler angle measurement from accelerometers. The sinusoidal pattern of the maneuver is well represented in the roll and pitch angle plot. Also, it comes back to 0 unlike the plot from the gyro. No error accumulation is seen. However, the amplitude is $\pm 9°$ which is far below the actual value which is $\pm 30°$.

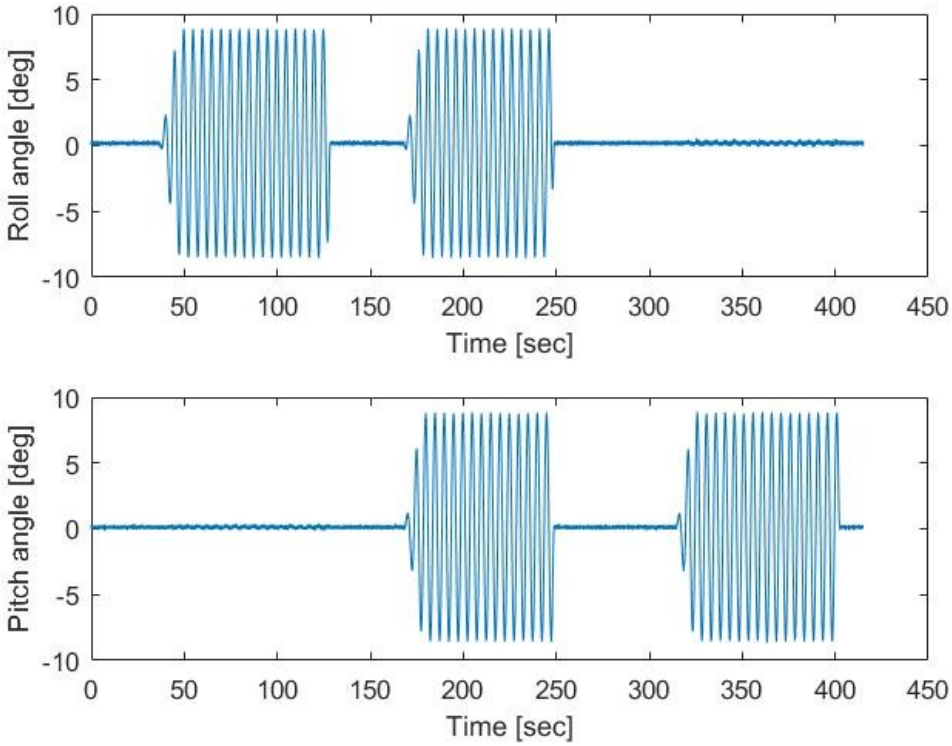


Figure 19: Euler angle measurement of accelerometers.

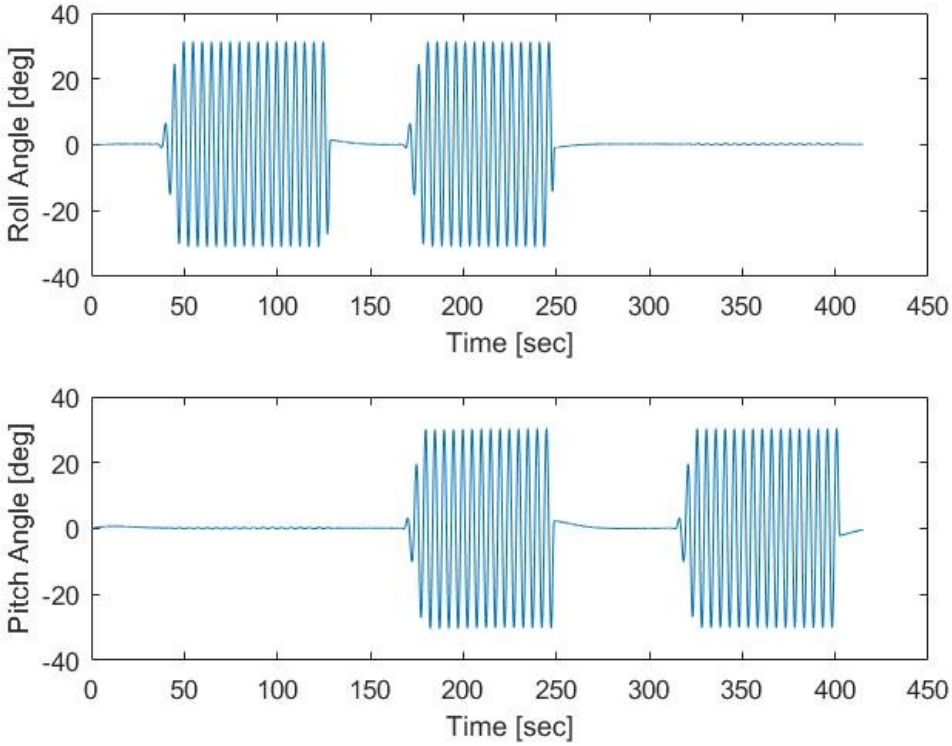


Figure 20: Filter estimates of the altitude Euler angle.

It could be seen in Figure 20 that the results have correct trend and amplitude, which are far better than that from accelerometers or gyros and not inferior to the Kalman filter estimates in Example 4.3.5 as well. Moreover, since it does not have a feedback structure like an extended Kalman filter, there is no worry for divergence.

4.2 Basic Kalman Filter Examples

4.2.1 Kalman Filter for Scalar Gauss-Markov Process

The following example shows an application example of using the Kalman filter to estimate a Gauss-Markov process. Suppose there is a sequence of noisy measurements of a stationary Gauss-Markov process. One wish to process these via a Kalman filter and obtain an optimal estimate of the process [17].

Table 7: Truth model and the Kalman filter model.

Truth Model (Autocorrelation)	$R_x(\tau) = 1 \cdot e^{-\lvert\tau\rvert}$	sec
State Transition Matrix (A)	0.9802	/
State to Measurement Matrix (H)	1	/
Covariance of the State Transitions Noise (Q)	0.03921	/
Covariance of Measurement Noise (R)	1	/
Prior Estimate $\hat{x}_0^-$	0	/
Error Covariance of Prior Estimate P_0^-	1	/
Measurement Interval	0.02	sec
Simulation Time	1	sec

Figure 21 shows that despite the assumed very noisy measurement situation, the filter does a reasonably good job of tracking the actual process. In Figure 22 it could be seen that after about the first 20 steps, the filter settles down to a steady-state condition where the Kalman gain is about 0.165 and the standard deviation of the filter error is about 0.4.

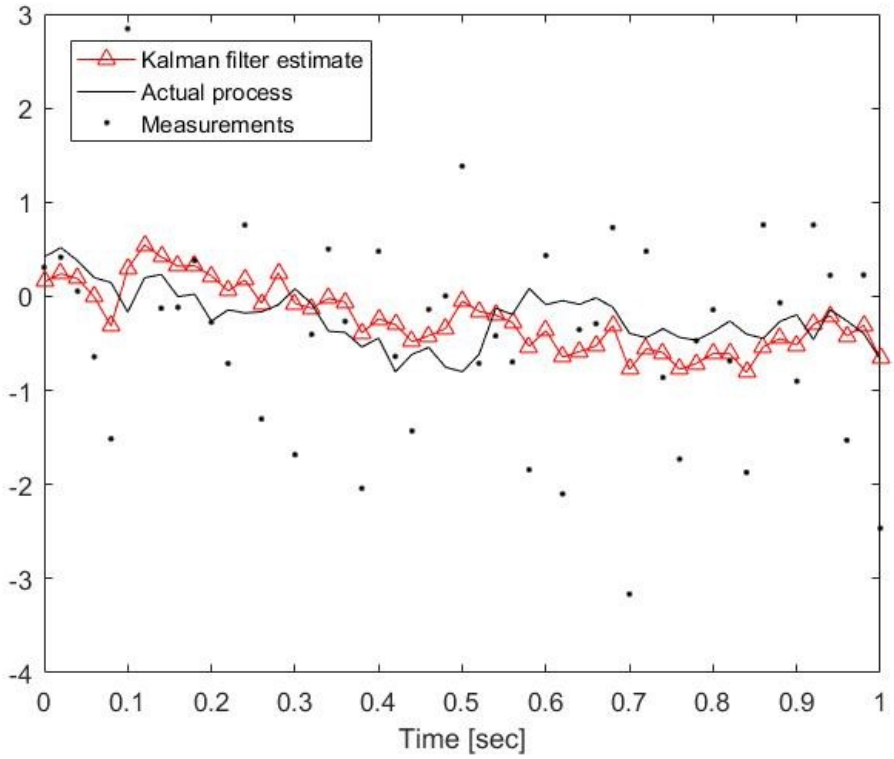


Figure 21: Kalman filtering for noisy Gauss-Markov process measurement.

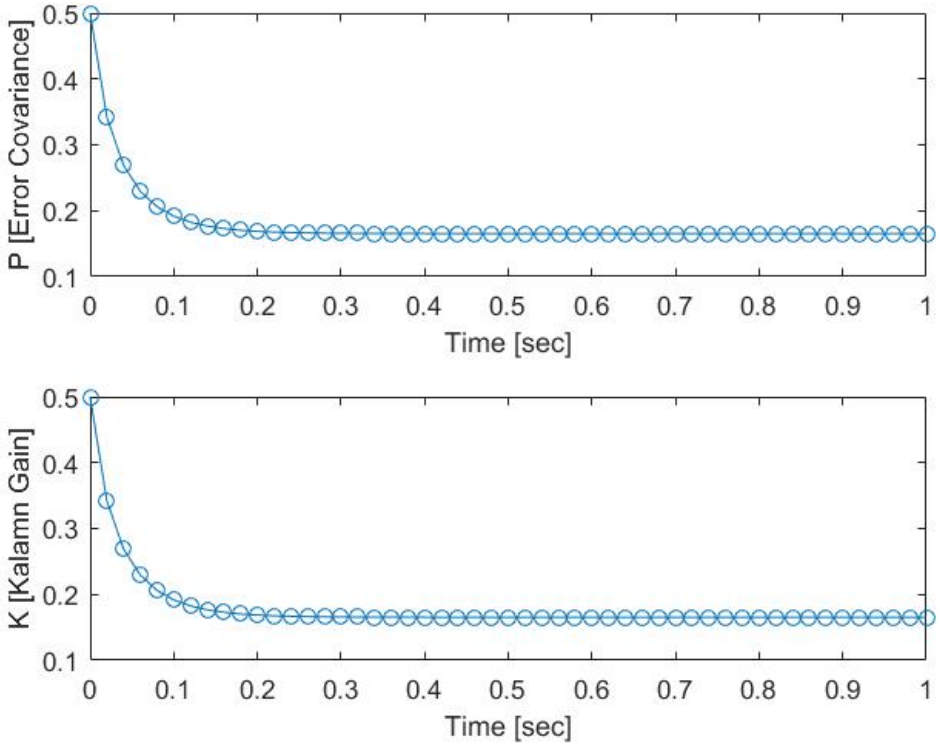


Figure 22: Kalman gain and error covariance for the estimates

4.2.2 Kalman Filter for Battery Voltage Measurement

The following example shows the estimation of a batter's voltage with the Kalman filter. The objective is the voltage value of a new battery. One gathers the voltage measurement data for a certain period and use the Kalman filter to suppress the noise [16].

Table 8: Truth model and the Kalman filter model.

Truth Model (Voltage)	14.4	V
State Transition Matrix (A)	1	/
State to Measurement Matrix (H)	1	/
Covariance of the State Transitions Noise (Q)	0	V^2
Covariance of Measurement Noise (R)	16	V^2
Prior Estimate $\hat{x}_0^-$	14	V
Error Covariance of Prior Estimate P_0^-	6	V^2
Measurement Interval	0.2	sec
Simulation Time	10	sec

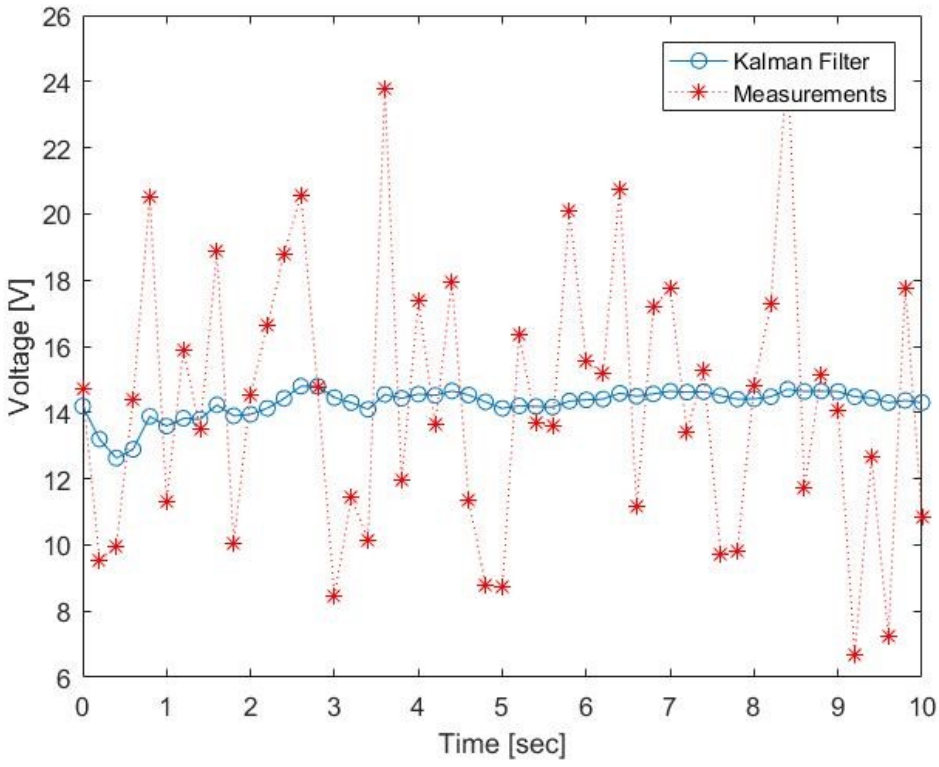


Figure 23: Kalman filtering for noisy battery voltage measurement.

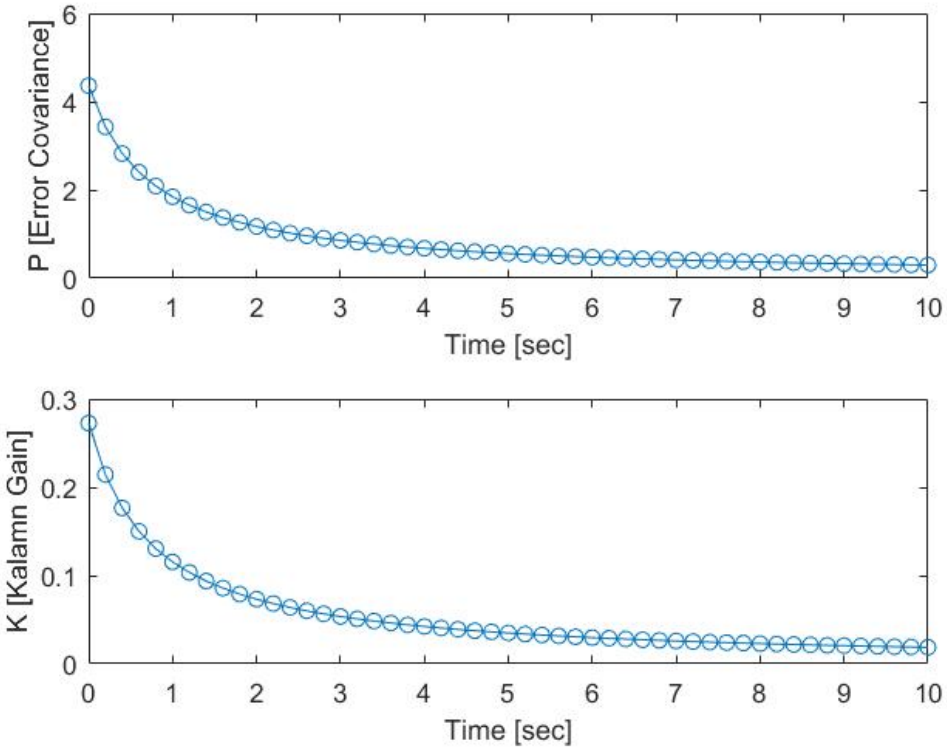


Figure 24: Kalman gain and error covariance for the estimates

Figure 23 shows that despite the assumed very noisy measurement situation, the filter does a reasonably good job of estimating the true battery voltage. The value of the error covariance in Figure 24 is decreasing all the way to the end. The rate of decrease was fast in the beginning but slowed down later in the end, almost stopped decreasing, which means the estimate is very much close to the actual value of the battery voltage.

4.3 Applications of Discrete Kalman Filtering

4.3.1 Estimating Velocity with Position

The following example shows the application example of using the Kalman filter to estimate the unknown parameters. Assuming data of the velocity measurement of a high-speed train performance test was lost due to accident. One wants to use the Kalman filter to estimate the velocity from the corresponding position measurement [16].

Discrete process model:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_k + \begin{bmatrix} \Delta t \\ 1 \end{bmatrix} w_k$$

where $w_k = \sqrt{3}\ m/s$ is the variance of white velocity noise and $\Delta t = 0.1\ s$ is the process time interval.

Measurement model:

$$z_k = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_k + v_k$$

where v_k is 1% of the distance (variance of measurement white noise).

Table 9: Kalman filter and measurement parameters.

State Transition Matrix (A)	$\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$	/
State to Measurement Matrix (H)	$[1 \quad 0]$	$[m \quad m/s]$
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} (\Delta t \cdot w_k)^2 & (\Delta t \cdot w_k) \cdot w_k \\ (\Delta t \cdot w_k) \cdot w_k & (w_k)^2 \end{bmatrix}$	/
Covariance of Measurement Noise (R)	$(1\%\ \text{of distance})^2$	/
Prior Estimate ($\hat{x}_0^-$)	$\begin{bmatrix} 0 \\ 80 \end{bmatrix}$	$\begin{bmatrix} m \\ m/s \end{bmatrix}$
Error Covariance of Prior Estimate (P_0^-)	$\begin{bmatrix} 0.03 & 0.3 \\ 0.3 & 3 \end{bmatrix}$	/
Measurement Interval (Δt)	0.1	Sec
Simulation Time	10	Sec

Figure 25 and 26 shows the filter estimate of the train's position and velocity. The velocity estimation is relatively correct in spite of the absence of the corresponding measurement. This is because the Kalman filter minimizes the trace of the mean-square error matrix of the state variables rather than a single scalar. But when the measurement unknown states are much larger or smaller compared to the other states with known measurement, Kalman filter might be failed to estimate them correctly.

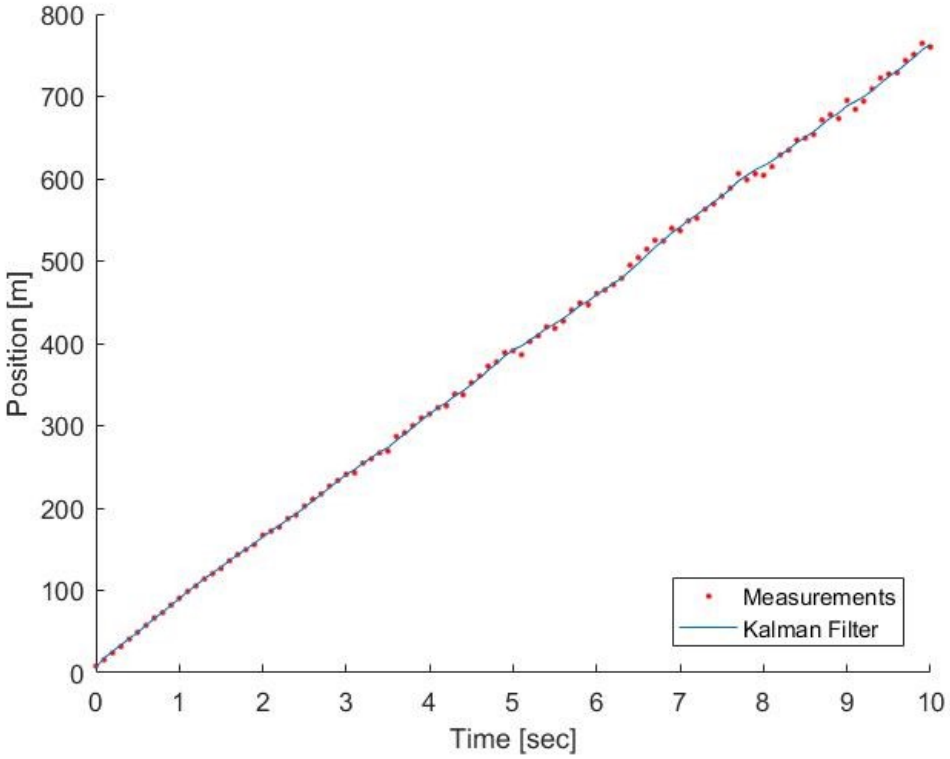


Figure 25: Position estimation based on velocity measurement of the train.

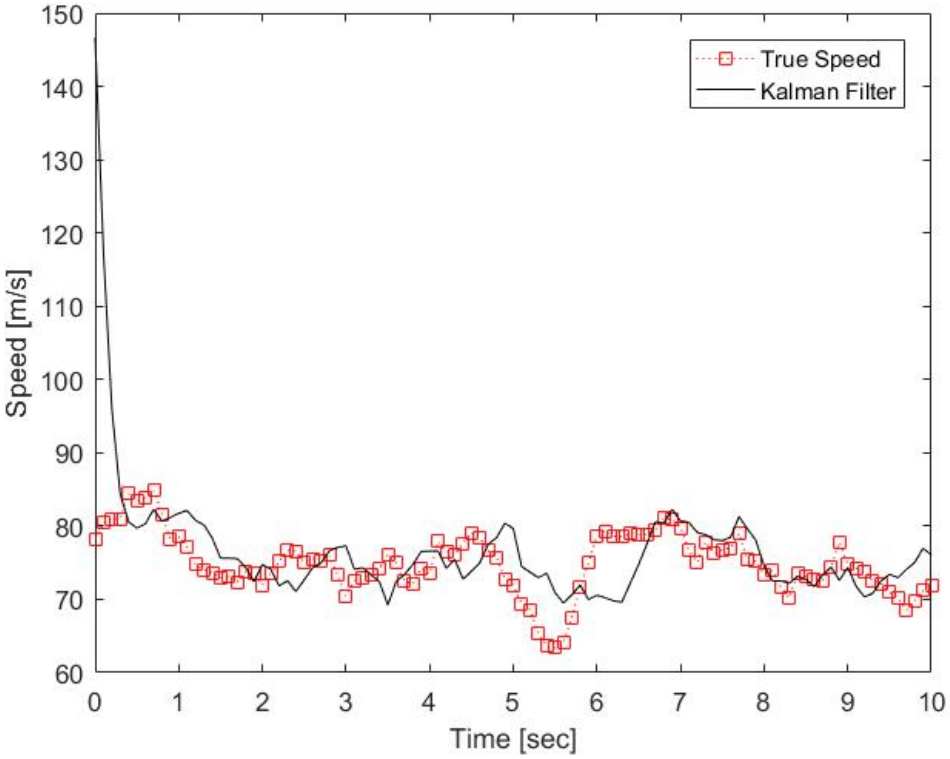


Figure 26: Velocity estimation from position measurement of the train.

4.3.2 Estimating Position with Velocity

This example has the same background setting as Example 4.3.1 except the lost data is position measurement. Assuming that one wants to use the given velocity measurement to estimate the missing position with Kalman filter [16].

The parameter values are the same as Example 4.3.1 except the measurement matrix becomes H = [0 1]. Since the train position is the unknown value.

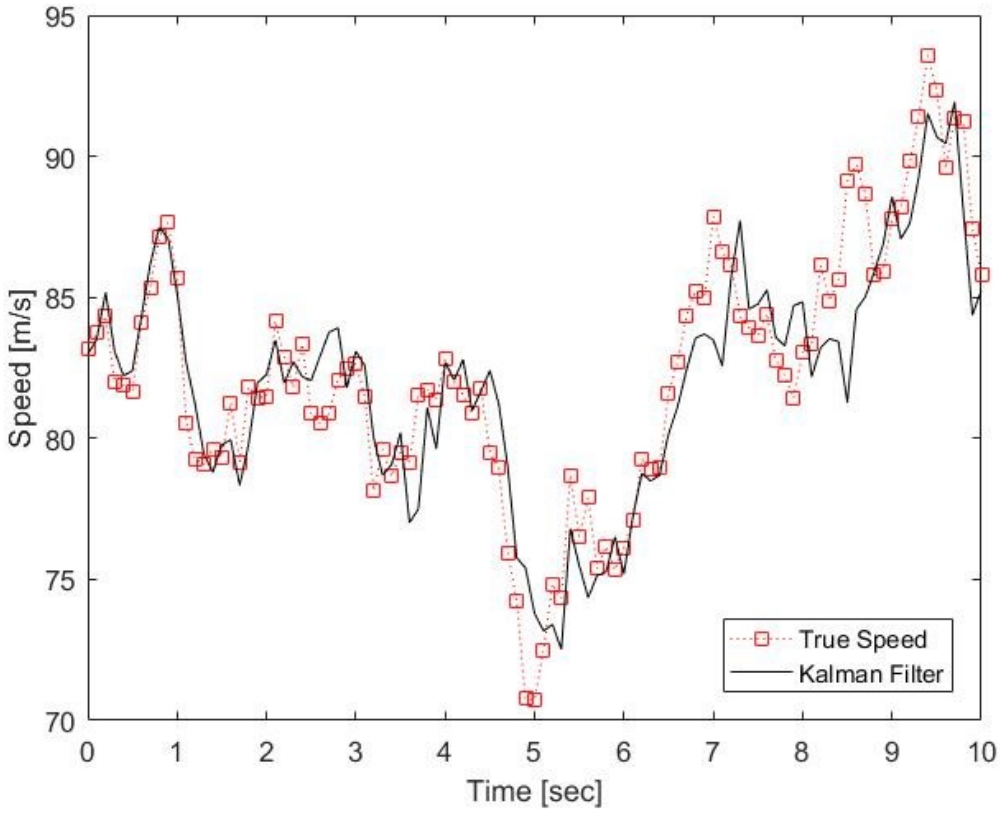


Figure 27: Kalman filtering for noisy train velocity measurement.

Figure 28 shows that both the train position and velocity estimate of the Kalman filter is quite correct despite the absence of the corresponding measurement, which is also because Kalman filter minimizes the trace of the mean-square error matrix of the state variables rather than a single scalar.

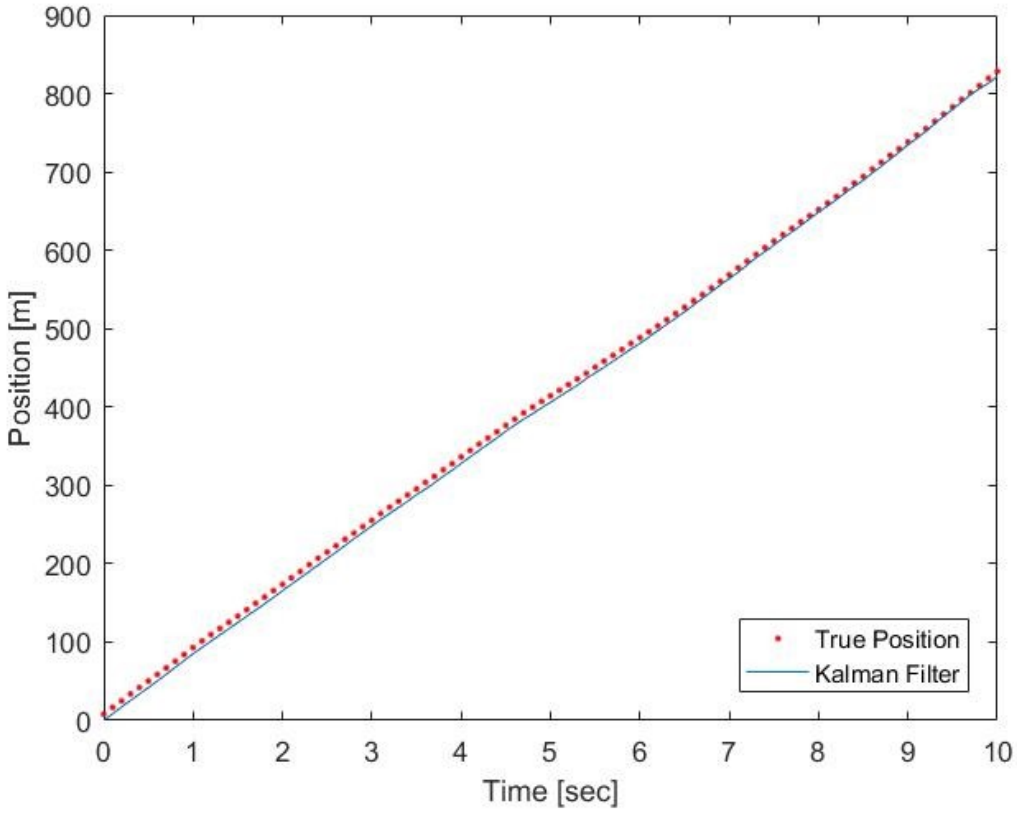


Figure 28: Position estimation from velocity measurement of the train.

4.3.3 Estimating Velocity with Sonar

The following example assumes a sonar sensor on a helicopter generates the measurement data of vertical distance to the ground surface for a certain period. One wants to use the Kalman filter to remove the noise due to vibration of the helicopter, condition of the ground surface and etc, and estimate the corresponding vertical velocity [16].

Discrete process model:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_k + \begin{bmatrix} \Delta t \\ 1 \end{bmatrix} w_k$$

where $w_k = \sqrt{3}\ m/s$ (variance of white velocity noise), $\Delta t = 0.1\ s$ (process time span).

Measurement model:

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_k + v_k$$

where $v_k = \sqrt{10}\ m$ (variance of measurement white noise).

Table 10: Kalman filter and measurement parameters.

State Transition Matrix (A)	$\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$	/
State to Measurement Matrix (H)	$[1 \quad 0]$	[m m/s]
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} (\Delta t \cdot w_k)^2 & (\Delta t \cdot w_k) \cdot w_k \\ (\Delta t \cdot w_k) \cdot w_k & (w_k)^2 \end{bmatrix}$	/
Covariance of Measurement Noise (R)	10	m^2
Prior Estimate ($\hat{x}_0^-$)	$\begin{bmatrix} 0 \\ 20 \end{bmatrix}$	$\begin{bmatrix} m \\ m/s \end{bmatrix}$
Error Covariance of Prior Estimate (P_0^-)	$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$	/
Measurement Interval (Δt)	0.1	sec
Simulation Time	50	sec

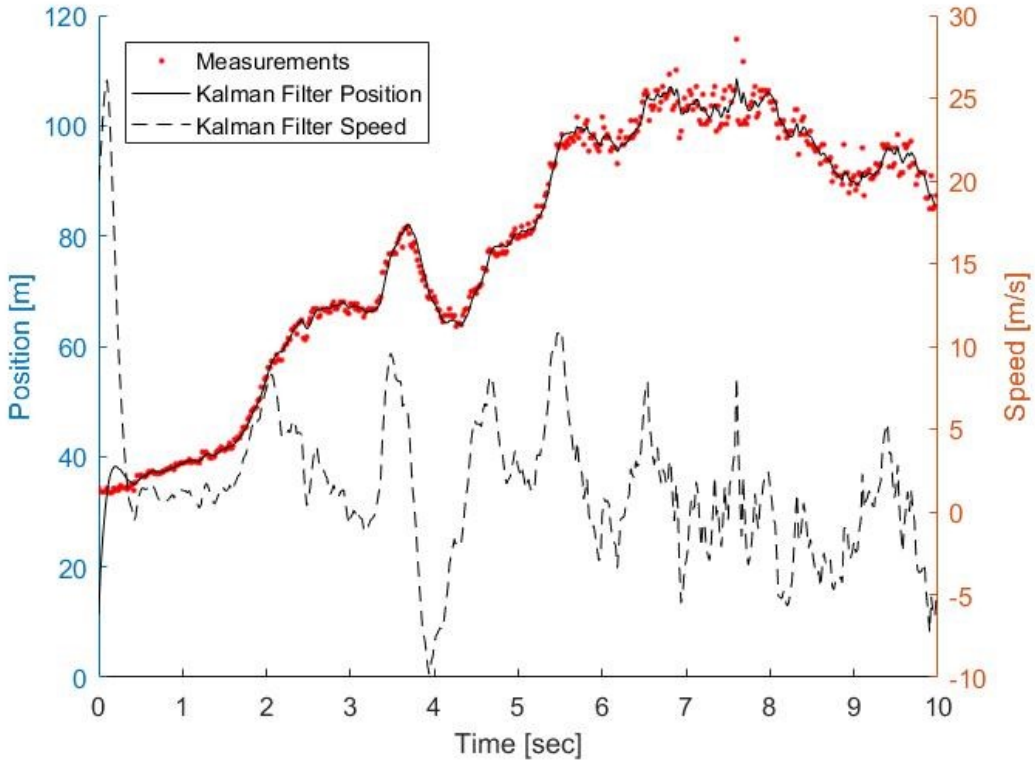


Figure 29: Position and velocity estimation from the sonar measurement.

Fig 29 shows that the solid and dash lines indicate the estimated value of position and velocity, respectively. Comparing the rate of change in the distance and the estimated velocity, it could be judged that the estimated velocity is reasonable.

4.3.4 Tracking an Object in an Image

The following example shows an example of tracking an object from images with the Kalman filter. The situation is given as a ball moving diagonally from the top right to the bottom left. It is assumed that the velocity of the ball is constant with a variance of 1 m/s. One wants to use the Kalman filter to estimate the true position from noisy position measurement [16].

Discrete process model:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_k + \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ w_x \\ 0 \\ w_y \end{bmatrix}_k$$

where variance of velocity noise $w_k = 1\ m/s$ and process time interval $\Delta t = 1\ s$.

Measurement model:

$$z_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_k + v_k$$

where the variance of measurement white noise is $v_k = 15\ m$.

Table 11: Kalman filter and measurement parameters.

State Transition Matrix (A)	$\begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$	/
State to Measurement Matrix (H)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} (\Delta t \cdot w_k)^2 & (\Delta t \cdot w_k) \cdot w_k & 0 & 0 \\ (\Delta t \cdot w_k) \cdot w_k & (w_k)^2 & 0 & 0 \\ 0 & 0 & (\Delta t \cdot w_k)^2 & (\Delta t \cdot w_k) \cdot w_k \\ 0 & 0 & (\Delta t \cdot w_k) \cdot w_k & (w_k)^2 \end{bmatrix}$	/
Covariance of Measurement Noise (R)	$\begin{bmatrix} 225 & 0 \\ 0 & 225 \end{bmatrix}$	m^2

Prior Estimate ($\hat{x}_0^-$)	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} m \\ m/s \\ m \\ m/s \end{bmatrix}$
Error Covariance of Prior Estimate (P_0^-)	$\begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$	/
Measurement Interval (Δt)	1	sec
Simulation Time	24	sec

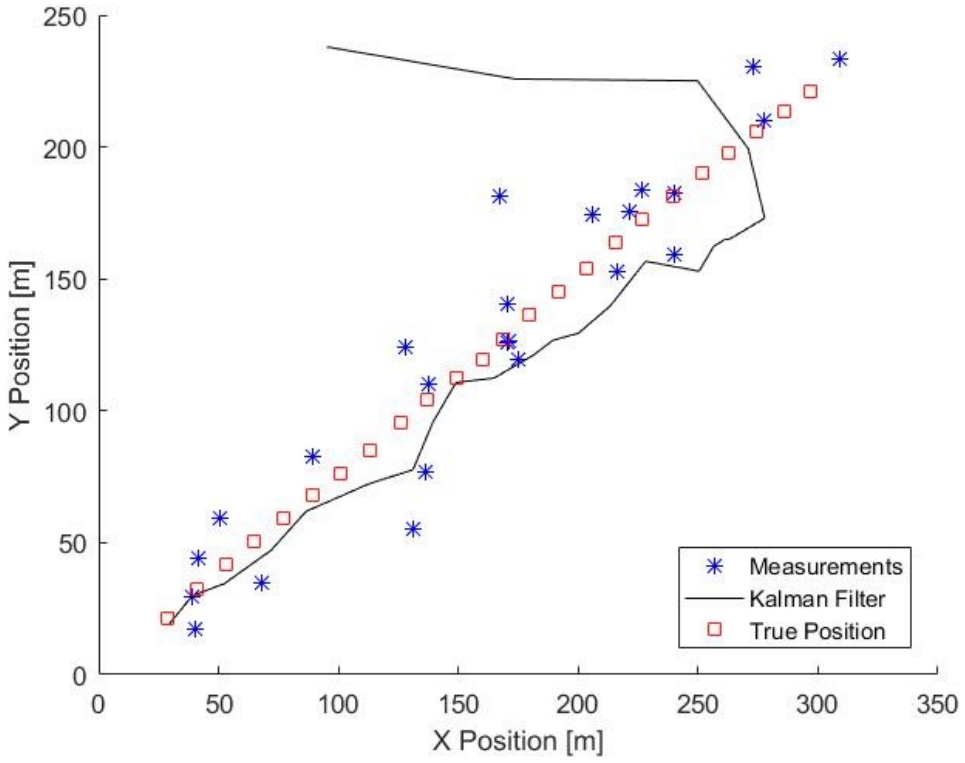


Figure 30: Measurement, truth and Kalman filter estimate of the ball position.

Figure 30 shows that estimates from the Kalman filter are lying closer to the diagonal from top right to bottom left, which indicates the Kalman filter removed the measurement noise effectively and returned the position close to the true value despite the very inaccurate prior estimation. By using different magnitudes of Q in the Kalman filter and comparing the estimates with the true position in the plot, it could be seen in Figure 31 that the estimate from the Kalman filter with Q of large magnitude does not fit the true nature of the system

well. The best Q value for this moving ball system is around 1.

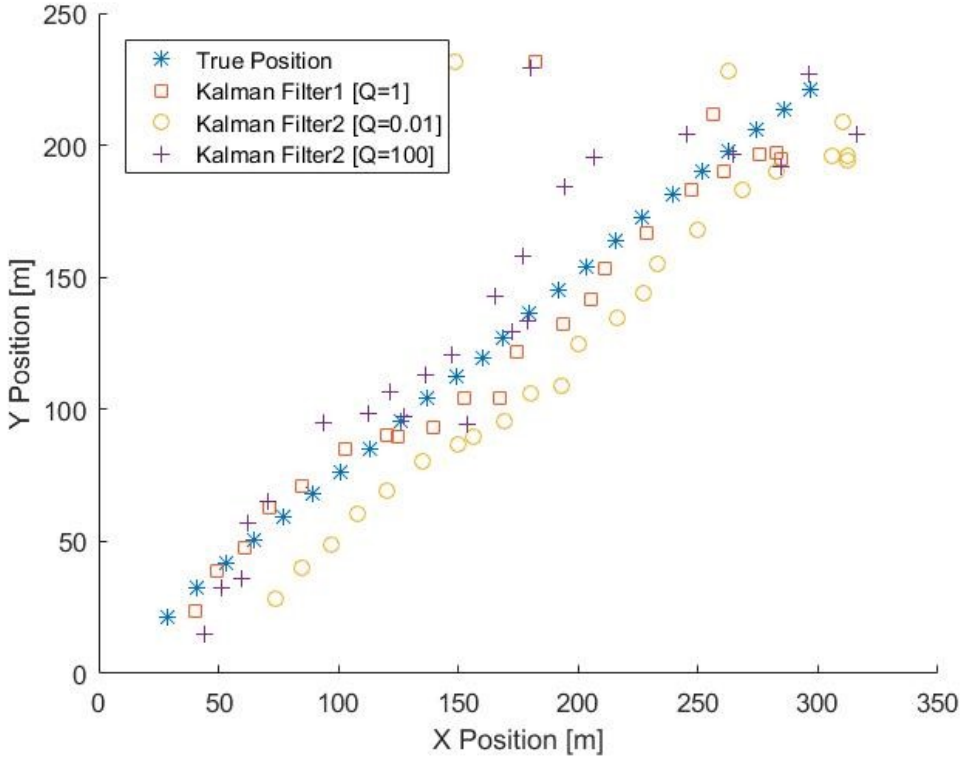
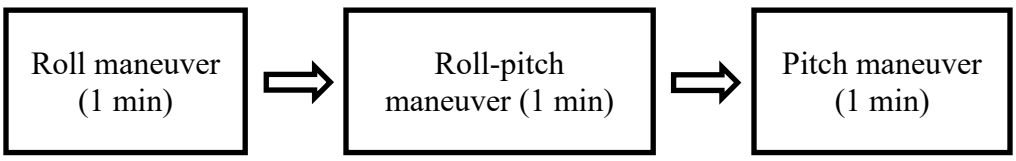


Figure 31: Kalman filter estimates with different q values.

4.3.5 Usual Kalman Filter for Attitude Reference System

The following example has the same background setting as Example 4.1.5, one wants to measure the horizontal attitude of a helicopter with the accelerometers and gyros. A test signal is applied to oscillate the navigation sensors with a sinusoidal wave of 0.2 Hz frequency and $\pm 30°$ amplitude. The Kalman filter is used to get the optimal attitude estimate from the two measurements. The procedure for the test is set as follows [16]:



The data collected from the test were accelerations and angular velocities along and about all three axes with a sampling rate of $0.01\ s$.

Gyro process model:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\begin{bmatrix}1 & sin\phi tan\theta & cos\phi tan\theta\\ 0 & cos\phi & -sin\phi\\ 0 & sin\phi/cos\theta & cos\phi/cos\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

where $\dot{\phi},\ \dot{\theta},\dot{\psi}$ are Euler angle rates. p,q,r are the angular rates from the gyros.

Accelerometer process model:

$$\begin{bmatrix}a_x\\ a_y\\ a_z\end{bmatrix}=\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix}+\begin{bmatrix}0 & w & -v\\ -w & 0 & u\\ v & -u & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}+g\begin{bmatrix}sin\theta\\ -cos\theta sin\phi\\ -cos\theta cos\phi\end{bmatrix}$$

where $a_x,\ a_y,\ a_z$ are accelerations along the x,y,z axis. u,v,w are the velocities along each axis in body frame. p,q,r are the angular rates about each axis in body frame.

In this case, system moving with constant velocity $\left(\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix}=\begin{bmatrix}p\\ q\\ r\end{bmatrix}=0\right)$, thus $\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix}=\begin{bmatrix}u\\ v\\ w\end{bmatrix}=0$.

$$\begin{bmatrix}a_x\\ a_y\\ a_z\end{bmatrix}=g\begin{bmatrix}sin\theta\\ -cos\theta sin\phi\\ -cos\theta cos\phi\end{bmatrix};\ \begin{bmatrix}\phi\\ \theta\end{bmatrix}=\begin{bmatrix}sin^{-1}\left(\frac{-a_y}{gcos\theta}\right)\\ sin^{-1}\left(\frac{a_x}{g}\right)\end{bmatrix}$$

Discrete gyro process model (linearized via quaternion along the gyro trajectory):

$$\begin{bmatrix}q_1\\ q_2\\ q_3\\ q_4\end{bmatrix}_{k+1}=\left(I+\frac{1}{2}\Delta t\begin{bmatrix}0 & -p & -q & -r\\ p & 0 & r & -q\\ q & -r & 0 & p\\ r & q & -p & 0\end{bmatrix}\right)\begin{bmatrix}q_1\\ q_2\\ q_3\\ q_4\end{bmatrix}_k+w_k$$

where $\begin{bmatrix}q_1\\ q_2\\ q_3\\ q_4\end{bmatrix}=\begin{bmatrix}cos\frac{\phi}{2}cos\frac{\theta}{2}cos\frac{\psi}{2}+sin\frac{\phi}{2}sin\frac{\theta}{2}sin\frac{\psi}{2}\\ sin\frac{\phi}{2}cos\frac{\theta}{2}cos\frac{\psi}{2}-cos\frac{\phi}{2}sin\frac{\theta}{2}sin\frac{\psi}{2}\\ cos\frac{\phi}{2}sin\frac{\theta}{2}cos\frac{\psi}{2}+sin\frac{\phi}{2}cos\frac{\theta}{2}sin\frac{\psi}{2}\\ cos\frac{\phi}{2}cos\frac{\theta}{2}sin\frac{\psi}{2}-sin\frac{\phi}{2}sin\frac{\theta}{2}cos\frac{\psi}{2}\end{bmatrix}$

Measurement model (Data from accelerometers):

$$\begin{bmatrix}z_1\\ z_2\\ z_3\\ z_4\end{bmatrix}_k=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}q_1\\ q_2\\ q_3\\ q_4\end{bmatrix}_k+v_k$$

Table 12: Kalman filter and measurement parameters.

State Transition Matrix (A)	$I + \frac{1}{2}\Delta t \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}$	/
State to Measurement Matrix (H)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} 0.0001 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix}$	/
Covariance of Measurement Noise (R)	$\begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$	/
Prior Estimate ($\hat{x}_0^-$)	$[0 \quad 0 \quad 0 \quad 0]^T$	/
Error Covariance of Prior Estimate (P_0^-)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	/
Measurement Interval (Δt)	0.01	Sec
Sample Number	41500	/

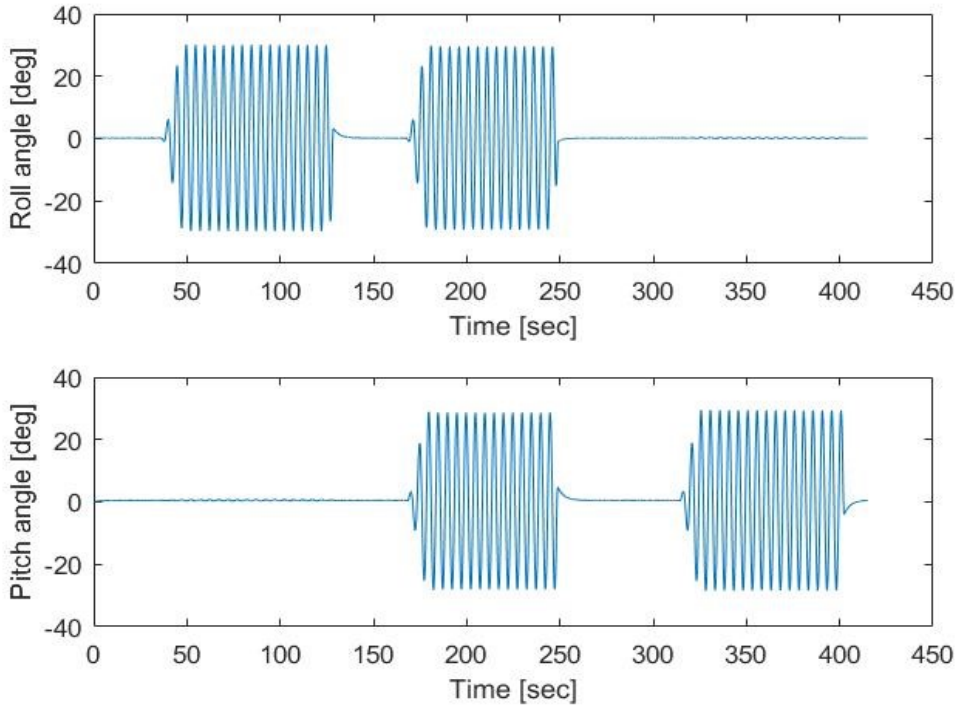


Figure 32: Kalman filter estimate of the attitude Euler angle.

Figure 32 shows the Kalman filter estimate of the roll and pitch angles. The oscillation with $\pm 30°$ amplitude is estimated accurately. And both the accumulated error of the gyros and the inaccuracy of the accelerometers are eliminated. The advantages of each sensor have been blended well.

4.3.6 Divergence and Error Analysis

Assuming a random walk process is incorrectly modelled as a random constant, which will cause the divergence issue. Then the Kalman filter is used for system error analysis [17].

Table 13: Truth model and the Kalman filter model.

Correct Process Model	$\dot{x} = u(t)$	/
Incorrect Process Model	$x = \text{constant}, \text{where } x \sim N(0,1)$	/
State Transition Matrix (A)	1	/
State to Measurement Matrix (H)	1	/
Covariance of the State Transitions Noise (Q) for Correct Model	1	/
Covariance of the State Transitions Noise (Q) for Incorrect Model	0	/
Covariance of Measurement Noise (R)	0.1	/
Prior Estimate $\hat{x}_0^-$	0	/
Error Covariance of Prior Estimate P_0^-	1	/
Measurement Interval	1	sec
Simulation Time	50	sec

The results are shown along with the sampling process in Figure 33. Estimates of the correctly modelled filter follow the random walk quite accurately. On the other hand, the incorrectly modelled filter does very poorly after the first few steps. We consider the random-walk model as the truth model and the random-constant model as the suboptimal system and then cycle the suboptimal gains through the truth model to get the resulting RMS error. The resulting RMS error of optimal gains and suboptimal gains are shown in Figure 34. It can be seen that divergence does occur in this situation.

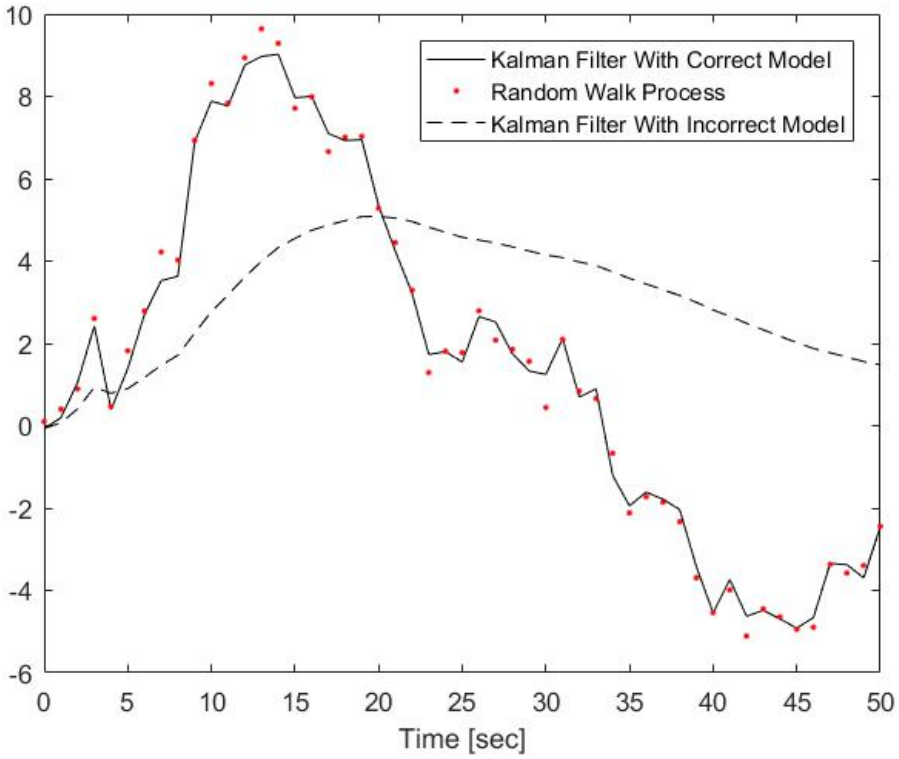


Figure 33: Kalman filter estimates of the process with the correct and incorrect model.

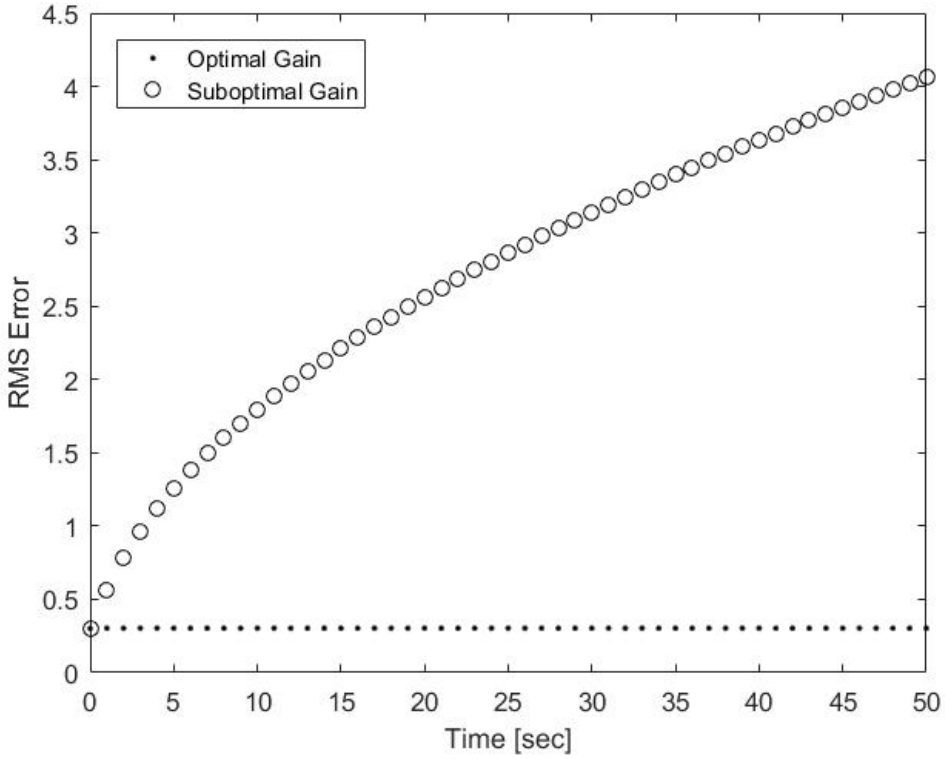


Figure 34: Root-mean-square error of the Kalman filter estimate.

4.3.7 Estimating a Constant with Alternative Kalman Filter

Consider a basic physics experiment that is intended to measure the gravitational constant g. A mass is released at $\mathrm{t} = 0$ in a vertical-evacuated column and multiple-exposure photographs of the falling mass are taken at 0.05 sec intervals beginning at $\mathrm{t} = 0.05$ sec. Assuming there is no prior knowledge about the gravity constant, hence the alternative Kalman filter is used for the infinite error covariance of prior estimate situation [16].

Table 14: Truth model and the Kalman filter model.

Truth Model	$\mathrm{x} = 9.8$	m/s^2
Process Model	x is constant where $\mathrm{x} \in (-\infty, +\infty)$	m/s^2
State Transition Matrix (A)	1	/
State to Measurement Matrix (H)	$0.5t^2$	/
Covariance of the State Transitions Noise (Q) for Correct model	0	m^2/s^4
Covariance of Measurement Noise (R)	0.0001	m^2
Prior Estimate $\hat{x}_0^-$	0	m/s^2
Error Covariance of Prior Estimate P_0^-	∞	m^2/s^4
Measurement Interval	0.05	sec
Simulation Time	1	sec

The acceleration estimate is shown along with the position measurement in Figure 35. The dash line is the position measurement and the solid line is the gravity estimation. It can be seen that the estimates of the gravity constant correctly converged to the correct value after about 10 steps despite the infinite large uncertainty of initial estimate.

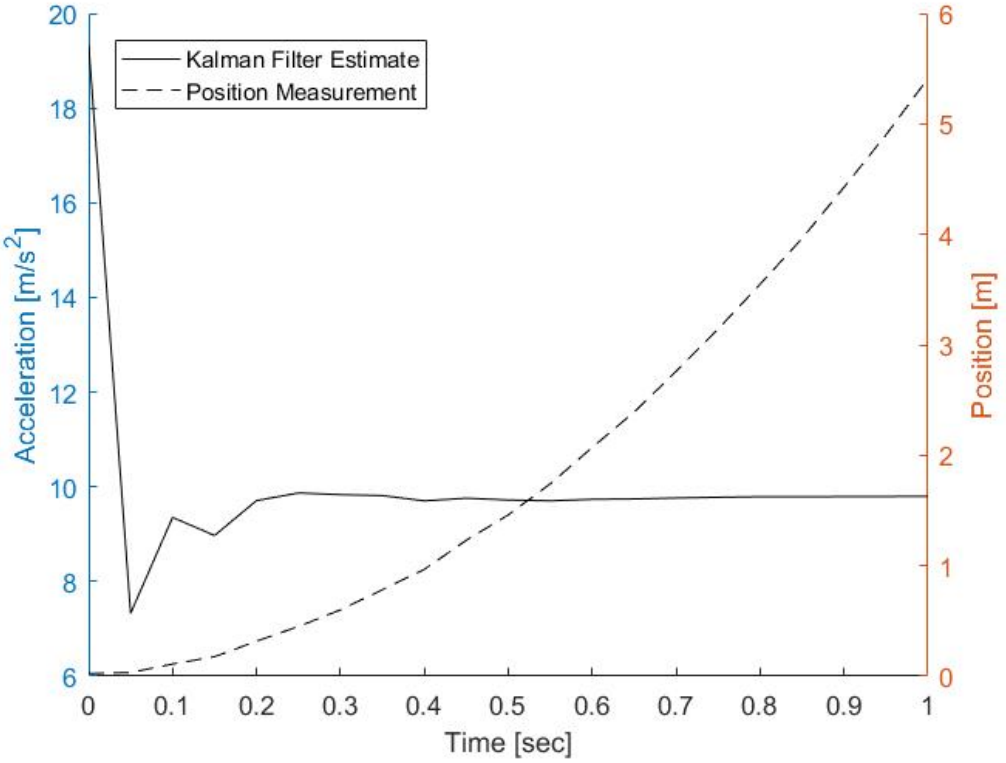


Figure 35: Position measurement and Kalman filter estimate of gravity constant.

4.4 Discrete Smoothing

4.4.1 Discrete Fixed-Interval Smoothing

The following example shows the application of discrete fixed-interval smoothing to smooth the Kalman filter estimate of a Gauss-Markov process [17].

Table 15: Truth model and Kalman filter model.

Truth Model (autocorrelation)	$R_x(\tau) = 1 \cdot e^{-\lvert\tau\rvert}$	sec
State Transition Matrix (A)	0.9802	/
State to Measurement Matrix (H)	1	/
Covariance of the State Transitions Noise (Q)	0.03921	/
Covariance of Measurement Noise (R)	1	/
Prior Estimate $\hat{x}_0^-$	0	/
Error Covariance of Prior Estimate P_0^-	1	/
Measurement Interval	0.2	sec
Simulation Time	20	sec

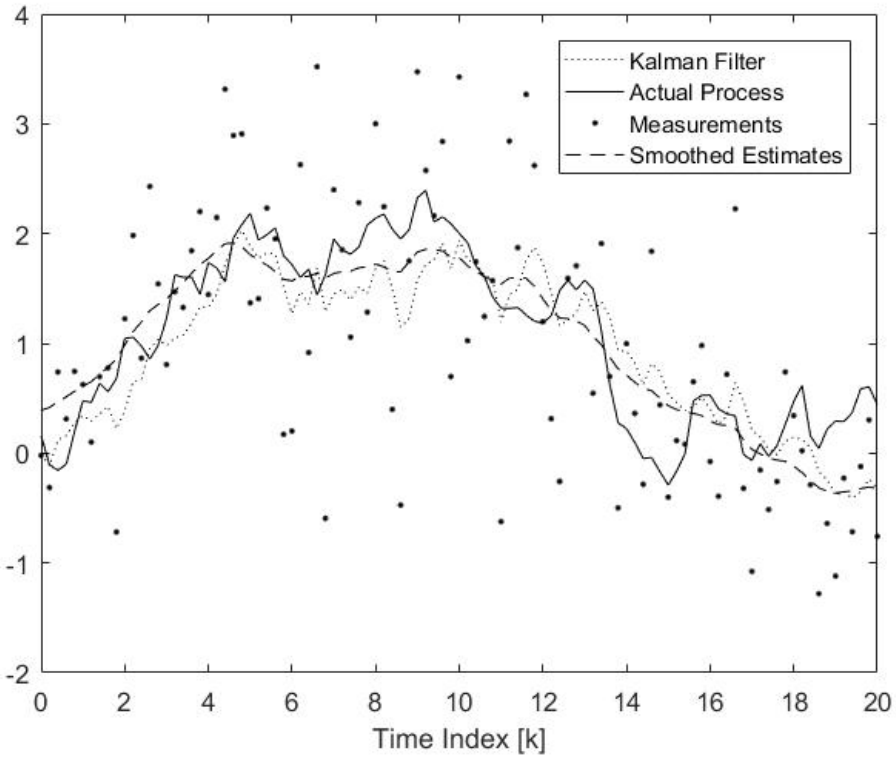


Figure 36: Smoothed and original filter estimate of the Gauss-Markov process.

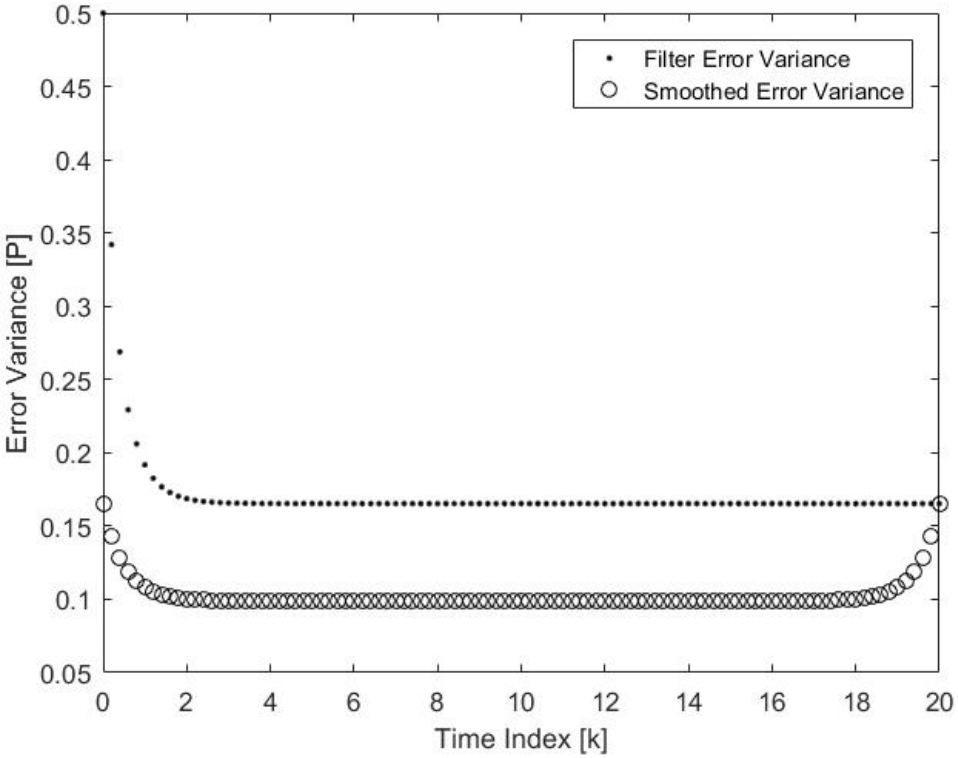


Figure 37: Error variance of Kalman filter and smoothed estimates.

It is evident from the plot that the smoothed plot is smoother than the filter plot. Because the smoother uses both past and future data in the makeup of its estimate. Both the filter and smoother converge to a steady-state condition after about 15 steps. The steady-state values are:

$$P_{filter} \approx 0.1653 \ (\sigma_{filter} \approx 0.406)$$

$$P_{smoother} \approx 0.099 \ (\sigma_{smoother} \approx 0.315)$$

In addition, the smoothed error covariance is minimum in the middle and then gets larger as either point is approached. This indicates that the best situation for estimation occurs in the interior region where there is an abundance of measurement data in either direction from the point of estimation.

4.4.2 Discrete Fixed-Point Smoothing

This example is using discrete fixed-point smoothing to smooth the Kalman filter estimate at a single point. The process and simulation parameter setting are the same as used in Example 4.4.1. The objective of smoothing is the optimal estimate of x_0 [17].

It is indicated from the simulation data in table 16 that the estimate and error covariance solution of fixed-point smoothing and fixed-interval smoothing of x_0 converged after about 40 steps.

Table 16: Fixed-point and fixed-interval smoothing results.

step	Fixed point Smoothing estimate of x_0	Fixed interval smoothing estimate	step
36	-0.457806600333161	**-0.457645193234919**	**0**
37	-0.457500581869915	-0.432766089044032	1
38	-0.457494762975732	-0.348221166585109	4
39	-0.457455567891364	-0.286574203624869	3
40	-0.457699840529226	-0.260729811462950	4
41	-0.457666319760824	-0.238567315909382	5
42	-0.457526638154054	-0.231425350970056	6
43	**-0.457537622977624**	-0.252257032130613	7

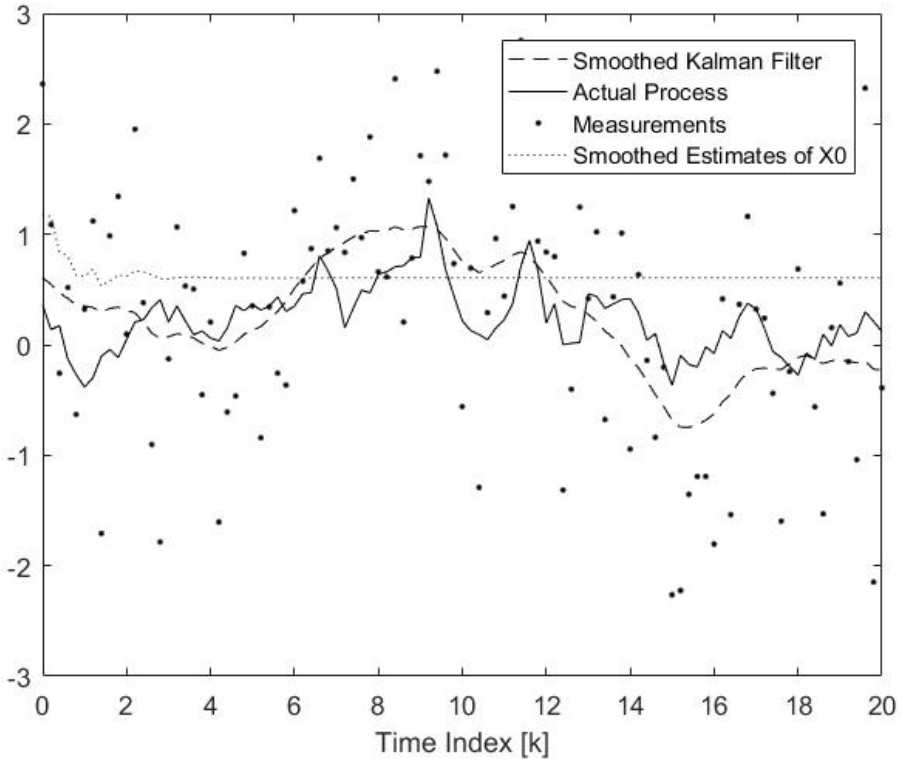


Figure 38: Fixed-point smoothed estimate and fixed-interval smoothed estimate.

4.4.3 Simple Fixed-Lag Smoothing

This example shows the application of simple fixed-lag smoothing to smooth the Kalman filter estimate of a Gauss-Markov process. The process and simulation parameters setting are the same as used in Example 4.4.1. The backward sweep window is three [17].

It could be seen from Figure 39 that the smoothed plot is smoother than the normal Kalman filter plot. Although the effect is not as good as the fixed interval smoothing. Because the fixed-lag smoothing uses a limited size of backward sweep while the fix-interval smoothing uses all available data.

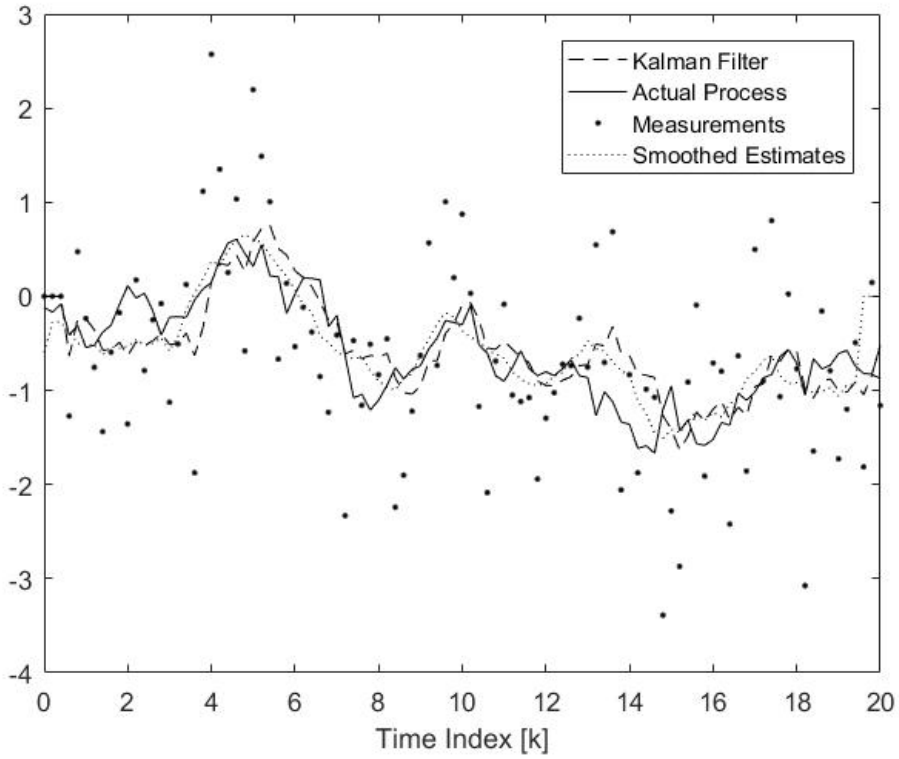


Figure 39: Fixed-lag smoothed estimate and Kalman filter estimate plots.

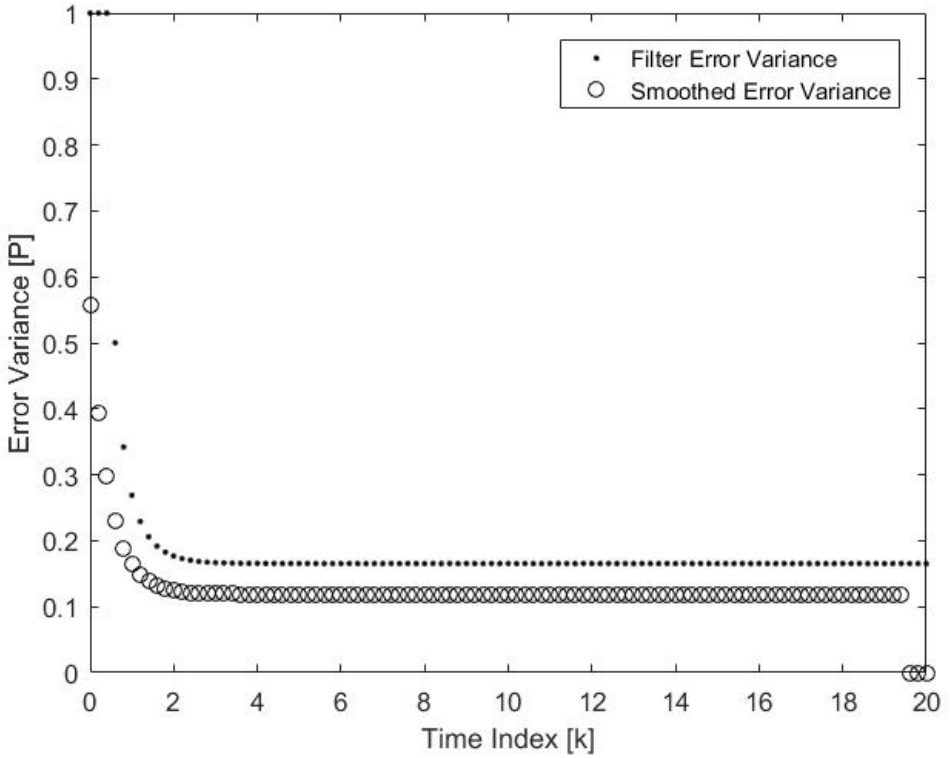


Figure 40: Error covariance of fixed-lag smoothed estimate and Kalman filter estimate.

The steady-state error covariance values of Kalman filter, fixed-lag smoothing, and fixed-interval smoothing are:

$$P_{filter} \approx 0.1653 \ (\sigma_{filter} \approx 0.406)$$

$$P_{fixed-lag} \approx 0.1189 \ (\sigma_{smoother} \approx 0.3448)$$

$$P_{fixed-interval} \approx 0.099 \ (\sigma_{smoother} \approx 0.315 \ Example \ 4.1)$$

which is exactly matches what we see from Figure 36 and Figure 39.

4.4.4 Forward-Backward Filter Approach to Smoothing

The following example is using simple forward-backward smoothing to smooth Kalman filter estimate of a Gauss-Markov process. The process and simulation parameter settings are the same as used in Example 4.4.1. The fixed-interval smoothing is also applied to the process and the result is used as a control group [17].

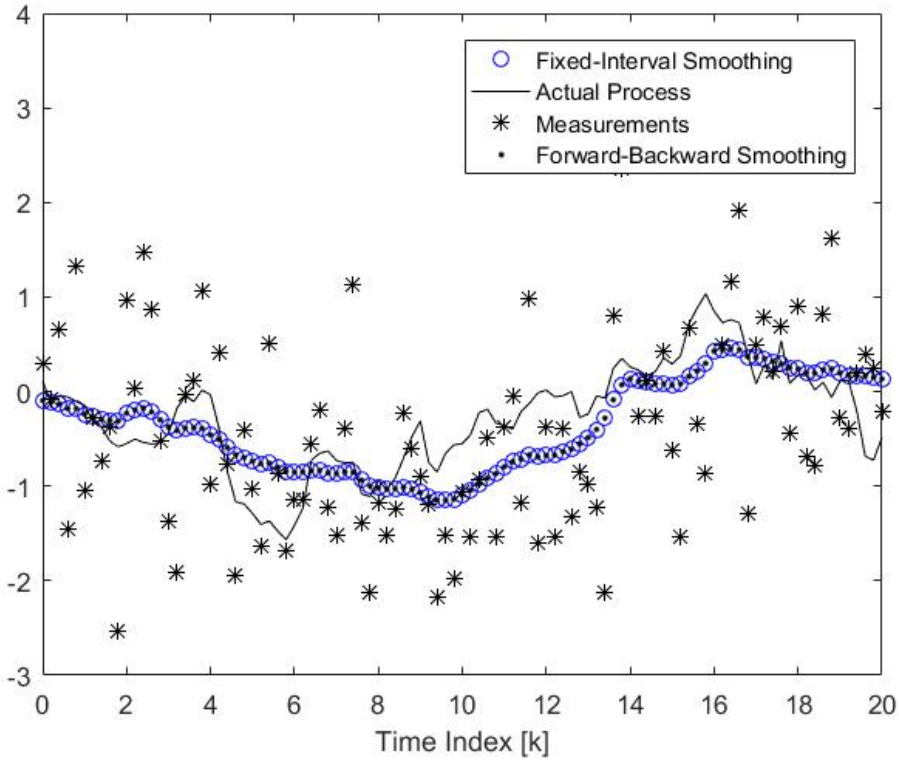


Figure 41: Forward-backward smoothing and fixed-interval smoothing.

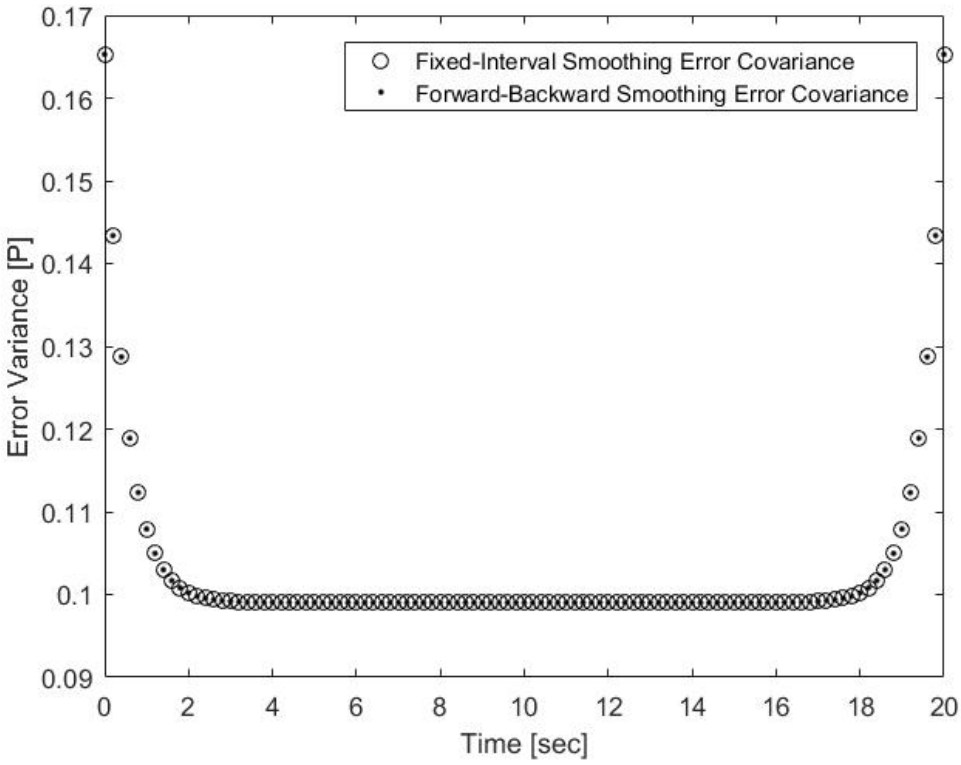


Figure 42: Error covariance of forward-backward and fixed-interval smoothing.

It could be seen in Figure 41 that the black points (forward-backward smoothing) and blue circles (fixed-interval smoothing) matches perfectly, which means they achieve identical results. It is also quite clear in Figure 42 that the black points (forward-backward smoothing) and black circles (fixed-interval smoothing) matches perfectly, which means they achieve the same results.

4.5 Linearization and Additional Topics on Applied Kalman Filtering

4.5.1 Linearized Kalman Filter for Distance Measuring Equipment

A simplified situation is assumed that the aircraft and the two DME stations are all in a horizontal plane. The coordinates of the two DME stations are assumed to be known, and the aircraft coordinates are unknown and to be estimated with the Kalman filter. The measurement matrix is linearized around a nominal trajectory [16].

Discrete process model:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_k + \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ w_x \\ 0 \\ w_y \end{bmatrix}_k$$

where the variance of velocity noise $w_k = 2\ m/s$ simulation time interval $\Delta t = 0.05\ s$

Nominal trajectory:

$$x_2 = x_4 = V_0\ (a\ constant\ velocity)$$

$$x_1 = x_3 = V_0 t$$

Measurement model:

$$z = \begin{bmatrix} \sqrt{(x - s1_x)^2 - \left(y - s1_y\right)^2} \\ \sqrt{(x - s2_x)^2 - \left(y - s2_y\right)^2} \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

where (x, y) is target coordinates; $(s1_x, s1_y), (s2_x, s2_y)$ are coordinates of DME station.

Linearized measurement model:

$$z_k = \begin{bmatrix} \dfrac{(x_1 - s1_x)}{\sqrt{(x_1 - s1_x)^2 + \left(x_3 - s1_y\right)^2}} & 0 & \dfrac{\left(x_3 - s1_y\right)}{\sqrt{(x_1 - s1_x)^2 + \left(x_3 - s1_y\right)^2}} & 0 \\ \dfrac{(x_1 - s2_x)}{\sqrt{(x_1 - s2_x)^2 + \left(x_3 - s2_y\right)^2}} & 0 & \dfrac{\left(x_3 - s2_y\right)}{\sqrt{(x_1 - s2_x)^2 + \left(x_3 - s2_y\right)^2}} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_k + \begin{bmatrix} v_x \\ v_y \end{bmatrix}_k$$

where the variance of measurement white noise $v_k = 1\%$ of measurement.

Table 17: Kalman filter and measurement parameters.

State Transition Matrix (A)	$\begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$	/
State to Measurement Matrix (H)	$\begin{bmatrix} \frac{(x_1 - 400)}{d_1} & 0 & \frac{(x_1 - 1600)}{d_1} & 0 \\ \frac{(x_1 - 1800)}{d_2} & 0 & \frac{(x_1 - 400)}{d_2} & 0 \end{bmatrix}$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} a^2 & a \cdot w_k & 0 & 0 \\ a \cdot w_k & {w_k}^2 & 0 & 0 \\ 0 & 0 & a^2 & a \cdot w_k \\ 0 & 0 & a \cdot w_k & {w_k}^2 \end{bmatrix}$	/
Covariance of Measurement Noise (R)	$\begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix}$	m^2
Prior Estimate ($\hat{x}_0^-$)	$[0 \quad 100 \quad 0 \quad 100]^T$	/
Error Covariance of Prior Estimate (P_0^-)	Q (covariance of state transition noise)	/
Measurement Interval (Δt)	0.05	sec
Simulation Time	20	sec

where $d_1 = \sqrt{(x_1 - 400)^2 + (x_3 - 1600)^2}$; $d_2 = \sqrt{(x_1 - 1800)^2 + (x_3 - 400)^2}$;

$a = \Delta t \cdot w_k$; $r = (1\%[\text{normal position} - \text{DME measurement}])^2$;

Figure 43 shows the noisy measurements from two DME stations. Since the noise is proportional to the distance, it could be seen that the variance is small in the middle and strengthen at both ends. It is shown from Figure 44 that the Kalman filter estimate (dot line) follows the actual trajectory (solid line) instead of nominal trajectory (dash line) quite well but indicates an increasing divergence near the end because the variance of both the process and measurement noise becomes larger.

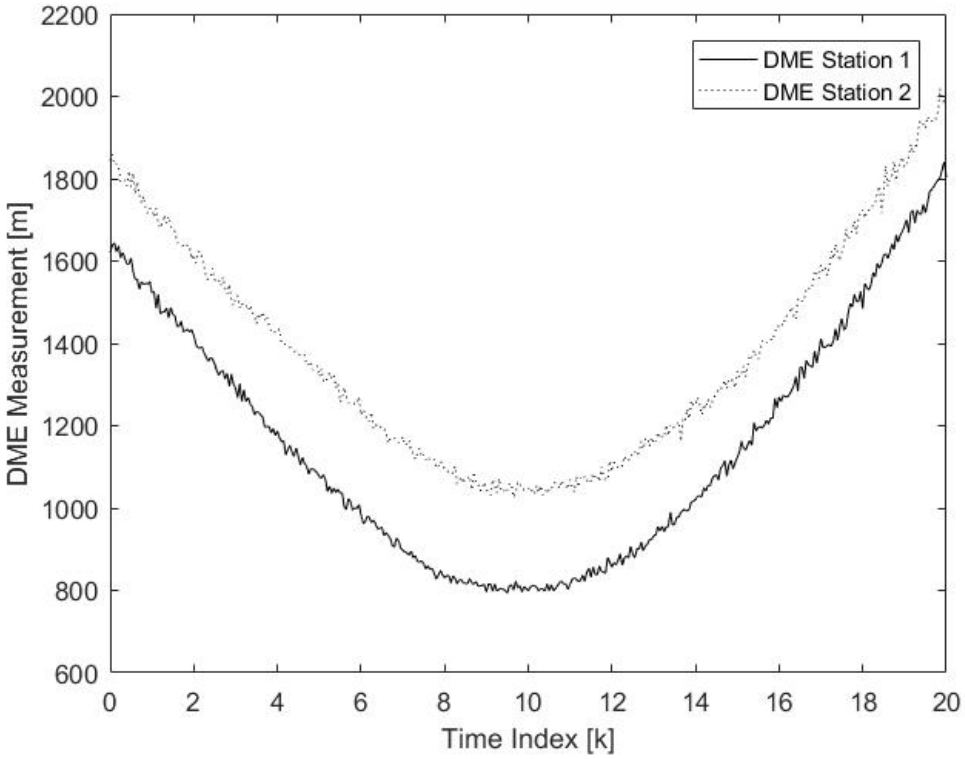


Figure 43: Distance measurements of the two DME stations.

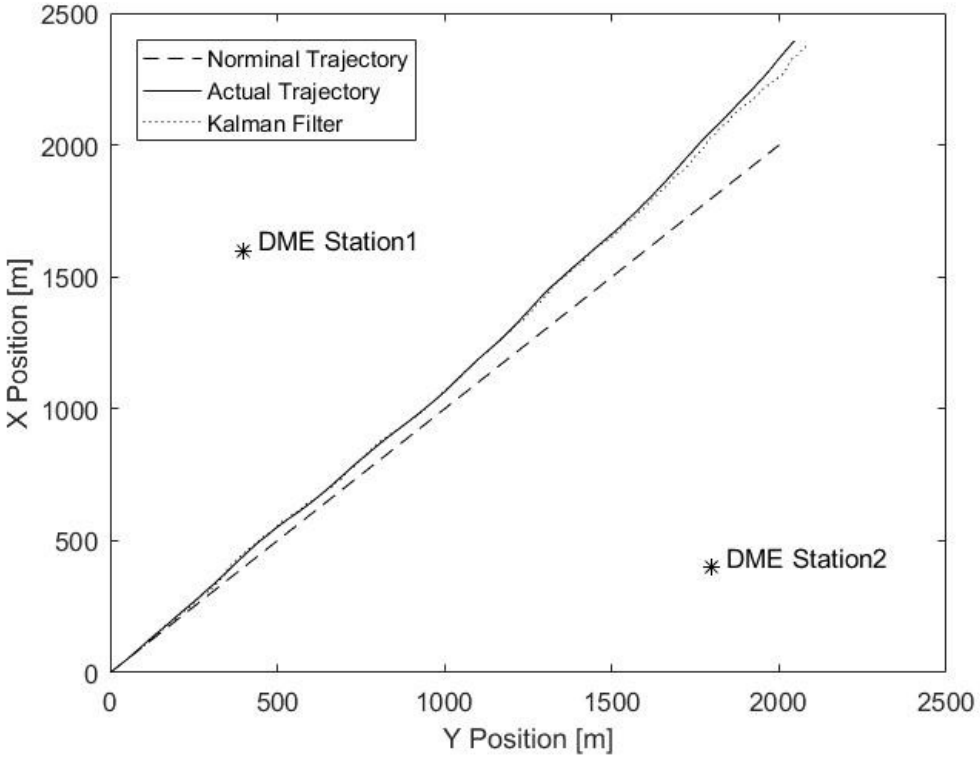


Figure 44: Nominal value, actual value and Kalman filter estimate of target coordinates.

4.5.2 Linearized Kalman Filter for Near-Earth Space Vehicle

The following example shows a near-earth space vehicle in a nearly circular orbit. It is desired to estimate the vehicle's position and velocity based on a sequence of angular measurements made with a horizon sensor. The state transition and measurement matrix are linearized around a nominal trajectory [17].

Nominal trajectory:

$$r^* = R_0 \ (a\ constant\ radius);\ \theta^* = \omega_0 t \ (\ \omega_0 = \sqrt{\frac{K}{R_0^3}}\ )$$

where $K = gR_e^2;\ g = 9.8\ m/s^2$ is gravitational constant; $R_e = 6.3 \times 10^6\ m$ is the radius of the Earth; $R_0 = 6.8 \times 10^6\ m$ is the radius of space vehicle orbit.

Continuous process model:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 x_4^2 - \dfrac{K}{x_1^2} \\ x_4 \\ -\dfrac{2x_2 x_4}{x_1} \end{bmatrix} + \begin{bmatrix} 0 \\ u_r(t) \\ 0 \\ \dfrac{u_\theta(t)}{x_1} \end{bmatrix}$$

where u_r, u_θ are random white forcing functions.

Discrete-linearized process model:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 3\omega_0^2 & 1 & 0 & 2R_0\omega_0 \\ 0 & 0 & 1 & \Delta t \\ 0 & \dfrac{-2w_0}{R_0} & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_k + \begin{bmatrix} 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ w_x \\ 0 \\ w_y \end{bmatrix}_k$$

where variance of white velocity noise $w_x = 5 \times 10^{-7}\ m/s$ variance of white polar angle rate noise) $w_y = 1 \times 10^{-9}\ radian/s$ simulation time interval $\Delta t = 0.5\ s$.

Measurement model:

$$z = \begin{bmatrix} sin^{-1}\left(\dfrac{R_e}{r}\right) \\ \alpha_0 - \theta \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

where α_0 is the angle between the local vertical and a known reference line; $R_e = 6.3 \times 10^6\ m$ is the average radius of the Earth

Linearized measurement model:

$$z_k = \begin{bmatrix} -\frac{R_e}{\sqrt{R_0^2 - R_e^2}} & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_k + \begin{bmatrix} v_x \\ v_y \end{bmatrix}_k$$

where $v_x = v_y = 4 \times 10^{-6}\ radian$ (variance of measurement white noise)

Table 18: Kalman filter and measurement parameters.

State Transition Matrix (A)	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 3\omega_0^2 & 1 & 0 & 2R_0\omega_0 \\ 0 & 0 & 1 & \Delta t \\ 0 & \frac{-2w_0}{R_0} & 0 & 1 \end{bmatrix}$	/
State to Measurement Matrix (H)	$\begin{bmatrix} -\frac{R_e}{\sqrt{R_0^2 - R_e^2}} & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} a^2 & a \cdot w_x & 0 & 0 \\ a \cdot w_x & (w_x)^2 & 0 & 0 \\ 0 & 0 & b^2 & b \cdot w_y \\ 0 & 0 & b \cdot w_y & (w_y)^2 \end{bmatrix}$	/
Covariance of Measurement Noise (R)	$\begin{bmatrix} 4 \times 10^{-6} & 0 \\ 0 & 4 \times 10^{-6} \end{bmatrix}$	$^{\circ 2}$
Prior Estimate ($\hat{x}_0^-$)	$[0 \quad 0 \quad 0 \quad 0]^T$	/
Error Covariance of Prior Estimate (P_0^-)	Q (covariance of state transition noise)	/
Measurement Interval (Δt)	0.5	sec
Simulation Time	50	sec

where $a = \Delta t \cdot w_x;\ b = \Delta t \cdot w_y$

The results show that the Kalman filter estimate follows the actual trajectory well in most state variables but a little poorer for the polar angle rate. The reason probably is this quantity is too small compared to the other state variables and they are lumped in one measurement state variable and lack of a direct measurement of the polar angle rate.

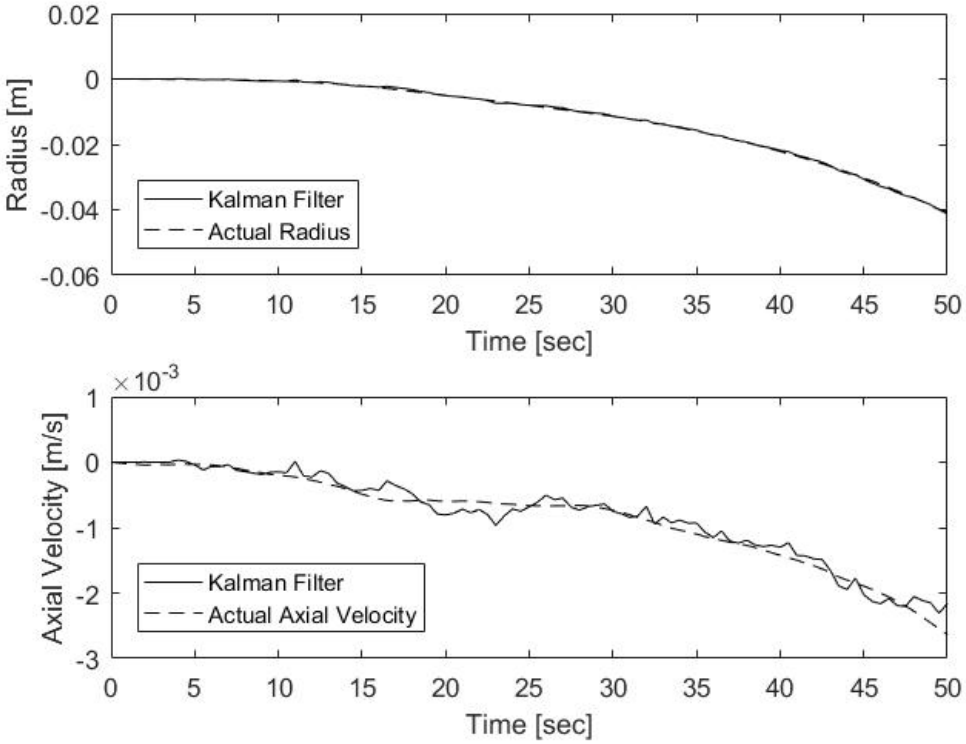


Figure 45: Kalman filter estimate of axial velocity and radius.

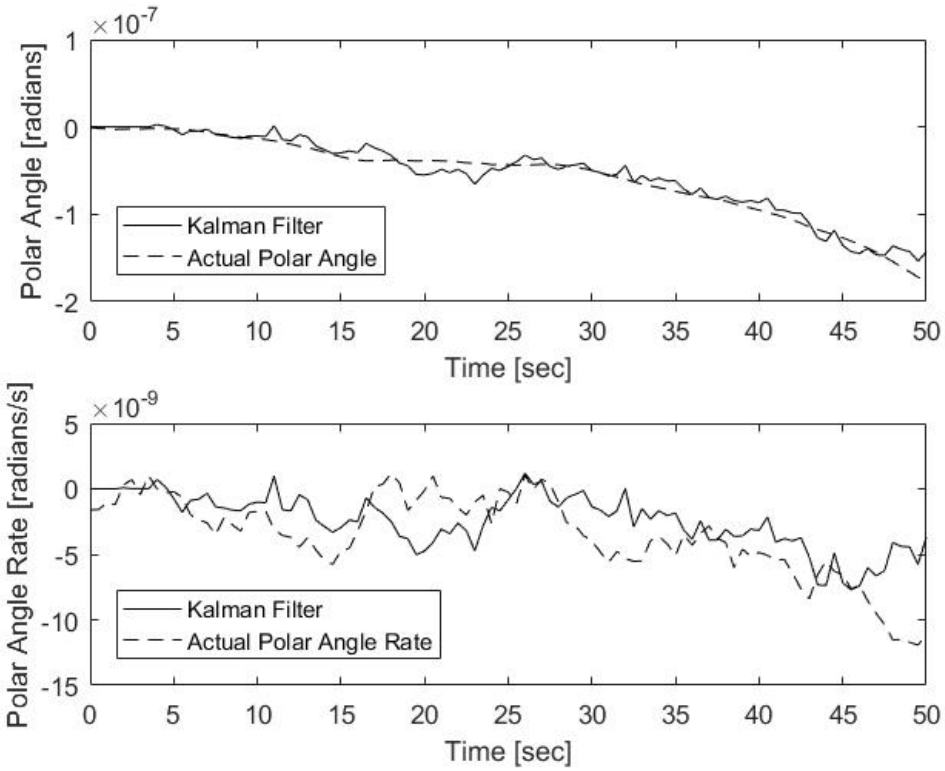


Figure 46: Kalman filter estimate of polar angle rate and polar angle.

4.5.3 Extended Kalman Filter for Radar Tracking

This example is using the extended Kalman filter to estimate the position and altitude of an aircraft with measured slant range data from a radar station. It is assumed that the aircraft is moving in 2-dimension with constant speed and altitude [16].

Discrete process model:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & dt & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_k + \begin{bmatrix} 0 & \Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ w_x \\ w_y \end{bmatrix}_k$$

where $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} horizontal\ distance \\ horizontal\ velocity \\ altitude \end{bmatrix}$; $w_x = 0.1\ m/s$ (standard deviation of white velocity noise); $w_y = 3\ m$ (standard deviation of white altitude noise); $\Delta t = 0.05\ s$ (process time interval)

Measurement model:

$$z = \sqrt{x_1^2 - x_3^2} + v$$

where v is measurement noise

Linearized measurement model:

$$z_k = \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2 + x_3^2}} & 0 & \frac{x_3}{\sqrt{x_1^2 + x_3^2}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_k + v_k$$

where $v_k = 1\%$ of measurement (variance of measurement white noise)

Table 19: Kalman filter and measurement parameters.

State Transition Matrix (A)	$\begin{bmatrix} 1 & dt & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	/
State to Measurement Matrix (H)	$\begin{bmatrix} \frac{x_1}{\sqrt{x_1^2 + x_3^2}} & 0 & \frac{x_3}{\sqrt{x_1^2 + x_3^2}} \end{bmatrix}$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} (\Delta t \cdot w_x)^2 & (\Delta t \cdot w_x) \cdot w_x & 0 \\ (\Delta t \cdot w_x) \cdot w_x & (w_x)^2 & 0 \\ 0 & 0 & (w_y)^2 \end{bmatrix}$	/

Covariance of Measurement Noise (R)	100	m^2
First Estimate ($\hat{x}_0$)	$\begin{bmatrix} 0 \\ 100 \\ 1000 \end{bmatrix}$	$\begin{bmatrix} m \\ m/s \\ m \end{bmatrix}$
Error Covariance of First Estimate (P_0)	Q (covariance of the state transition noise)	/
Measurement Interval (Δt)	0.05	sec
Simulation Time	20	sec

It is shown from the results that the Kalman filter estimates of position and attitude follow the actual trajectory well, but the deviation between the estimate and actual velocity gets quite large after a few steps because of the variance of the random walk process forcing noise and measurement noise are getting larger. Another possible reason is the absence of corresponding higher-order measurement.

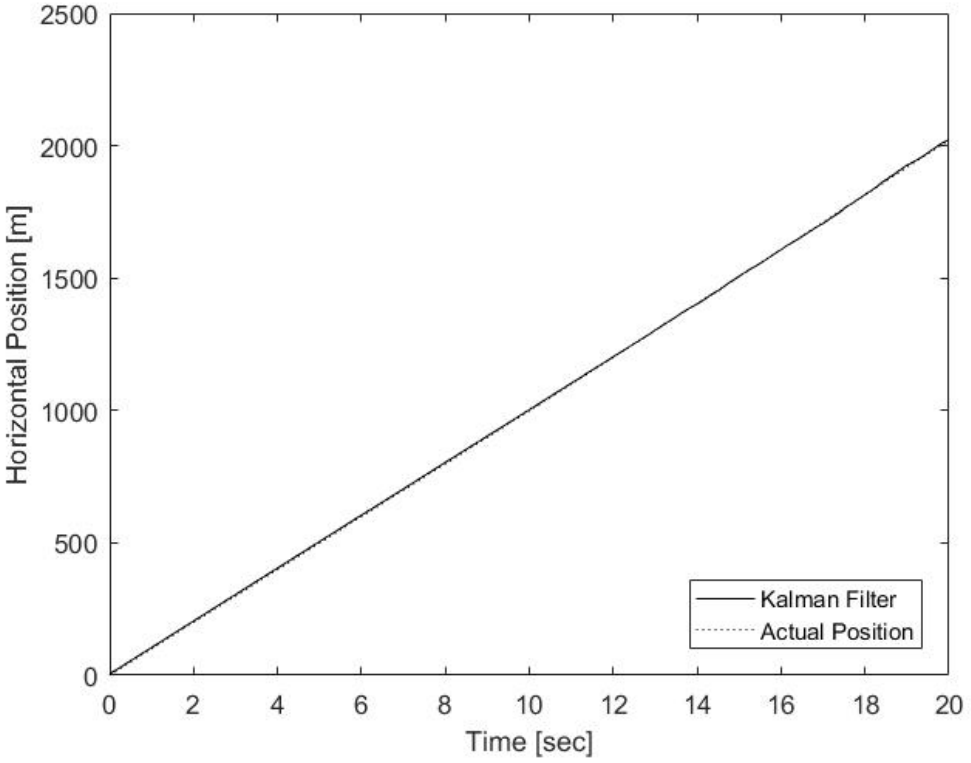


Figure 47: Kalman filter estimate of horizontal position.

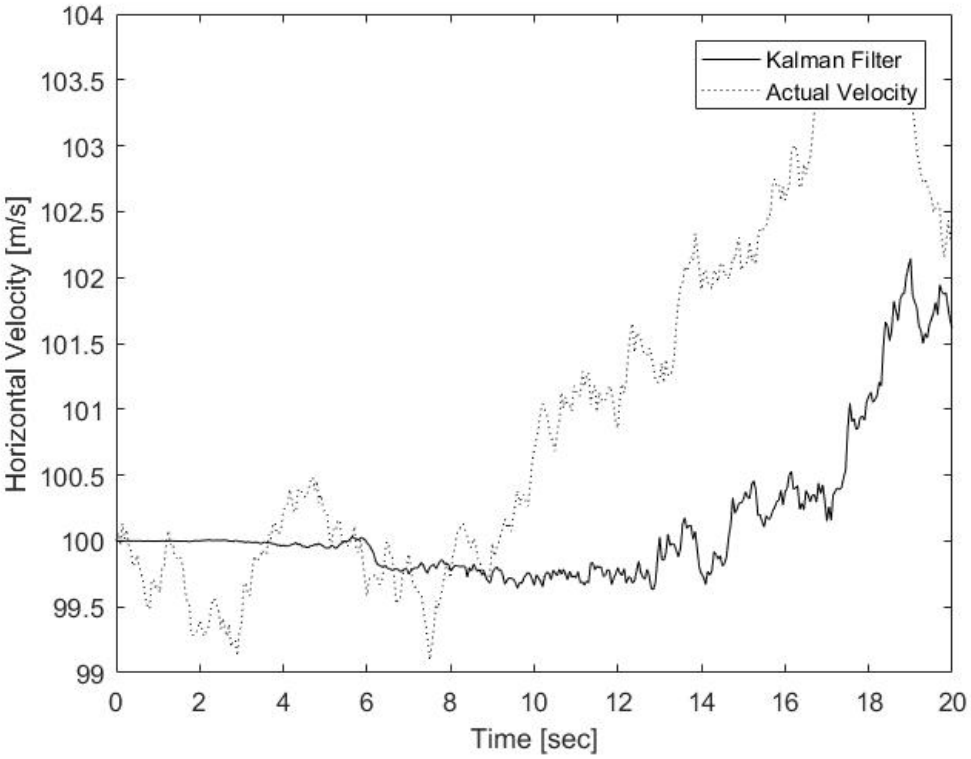


Figure 48: Kalman filter estimate of horizontal velocity.

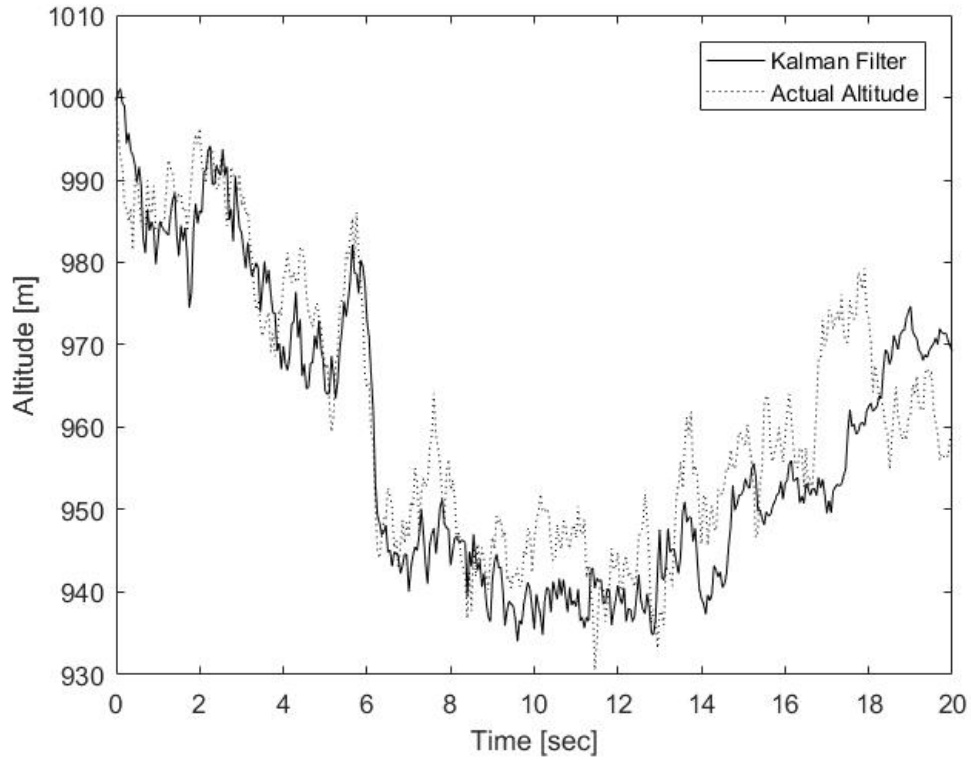


Figure 49: Kalman filter estimate of altitude.

4.5.4 Extended Kalman Filter for Altitude Reference System

Same as Example 4.3.5, one wants to measure the horizontal attitude of a helicopter with the accelerometers and gyros. A test signal is applied to oscillate the navigation sensors with a sinusoidal wave of 0.2 Hz frequency and ±30°amplitude. The extended Kalman filter is used to optimize the estimate of altitude. This simulation uses the same gyro and accelerometer process as Example 4.3.5. The gyro data is used in the process model and accelerometer data is used as a measurement [16].

Continuous process model (gyro data):

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & sin\phi tan\theta & cos\phi tan\theta\\ 0 & cos\phi & -sin\phi\\ 0 & sin\phi/cos\theta & cos\phi/cos\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + w$$

where u_r, u_θ are white random forcing functions; w is a random white forcing function.

Discrete-linearized process model:

$$\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}_{k+1} = (I + dt \cdot F)\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}_k + w_k$$

Where $w_k = [0.01;\ 0.01;\ \sqrt{0.1}]\ radian$ (variance of white angular noise);

$\Delta t = 0.01\ s$ (process time interval);

$$F = \begin{bmatrix} q\cos(\phi)\tan(\theta) - rsin(\phi)\tan(\theta) & q\sin(\phi)\sec(\theta)^2 - rcos(\phi)\sec(\theta)^2 & 0\\ -q\sin(\phi) - \cos(\phi) - rsin(\phi)\tan(\theta) & 0 & 0\\ q\cos(\phi)\sec(\theta) - rsin(\phi)\sec(\theta) & q\sin(\phi)\sec(\theta)\tan(\theta) + rcos(\phi)\sec(\theta)\tan(\theta) & 1\end{bmatrix}$$

Measurement model (accelerometer data):

$$z_k = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}_k + \begin{bmatrix}v_\phi\\ v_\theta\end{bmatrix}_k$$

where $v_\phi = v_\theta = 2.45\ radian$ (variance of measurement white noise)

Table 20: Kalman filter and measurement parameters.

State Transition Matrix (A)	$I + dt \cdot F$ (F is state transition matrix)	/
State to Measurement Matrix (H)	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} 0.0001 & 0 & 0 \\ 0 & 0.0001 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$	$^{\circ 2}$
Covariance of Measurement Noise (R)	$\begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix}$	$^{\circ 2}$
First Estimate ($\hat{x}_0$)	$[0 \quad 0 \quad 0]^T$	$^{\circ}$
Error Covariance of Prior Estimate (P_0)	Q (covariance of the state transition noise)	$^{\circ 2}$
Measurement Interval (Δt)	0.01	sec
Simulation Time	415	sec

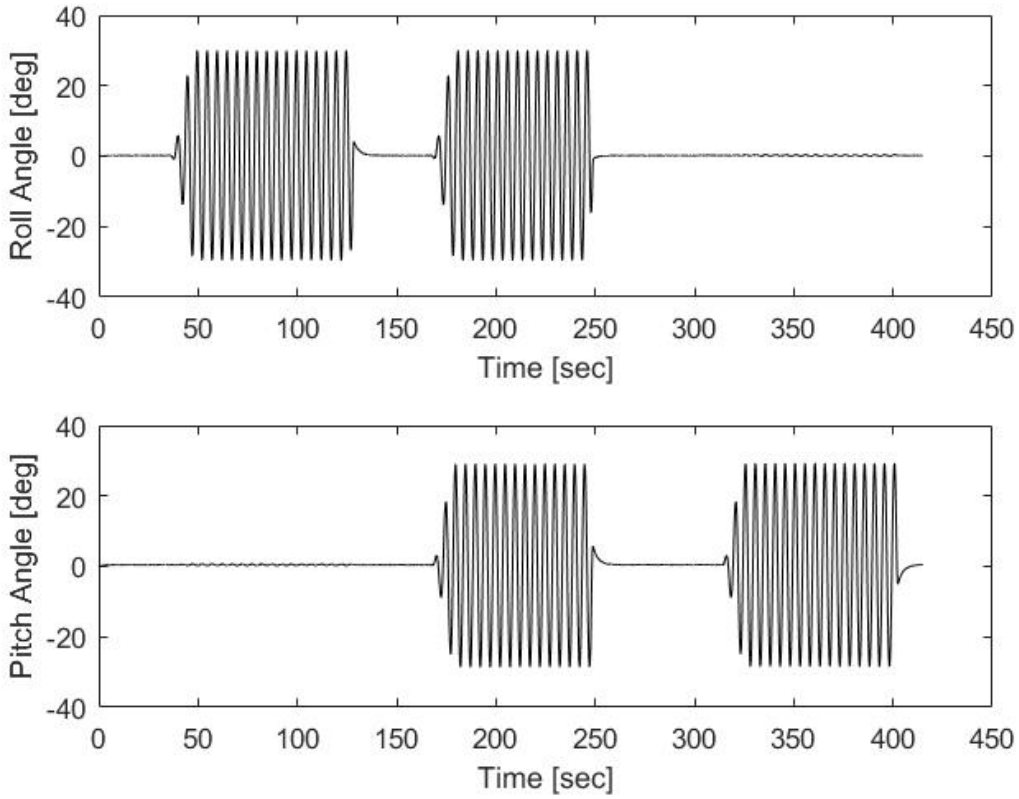


Figure 50: Extended Kalman filter estimate of attitude Euler angle.

Figure 50 shows an identical result as the usual Kalman filter in Example 4.3.5. The oscillation of the roll and pitch angles with ±30° amplitude is estimated accurately. And both the accumulated error of the gyros and the inaccuracy of the accelerometers are

eliminated. The advantages of each sensor have been blended well.

The first difference between the extended Kalman filter and linearized Kalman filter is linearization based on a nominal trajectory or filter estimation. The second deference is the predicted measurement and x projection in the filter. The extended Kalman filter always tracks the total estimation not the incremental part of the predicted z [17]. And its x projection needs to use the discrete original nonlinear function of the process. The Kalman filter that linearized around a nominal trajectory estimates the incremental part, so it can use either discrete original nonlinear function or the linearized ones like $H \cdot x$ and $A \cdot x$.

4.5.5 Unscented Kalman Filter for Radar Tracking (white noise)

The background setting is the same as Example 4.5.3. But in this simulation, one wants to use the unscented Kalman filter to estimate the position and altitude of an aircraft with measured slant range data from a radar station. It is assumed that the aircraft is moving in 2-dimension with constant speed and altitude [16].

Both extended Kalman filter and unscented Kalman filter are used to estimating the same process as used in Example 4.5.3 to compare the difference between them. Most parameters are unchanged except the covariance of first estimate becomes P = [0.1 1 0; 1 10 0; 0 0 9] for the matrix positive definite issue.

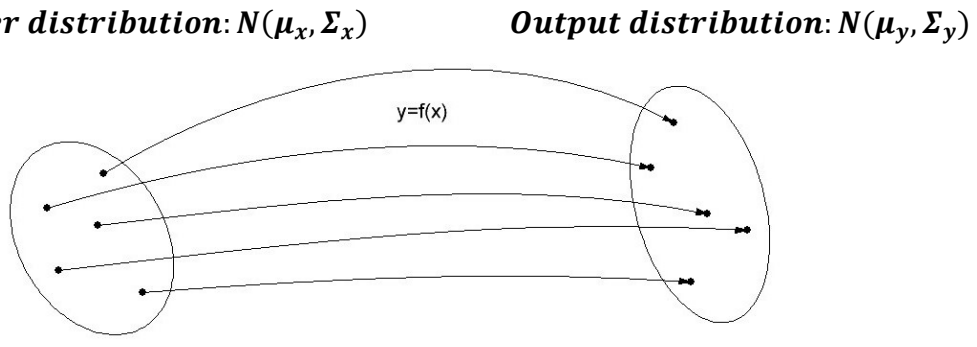


Figure 51: Unscented transform

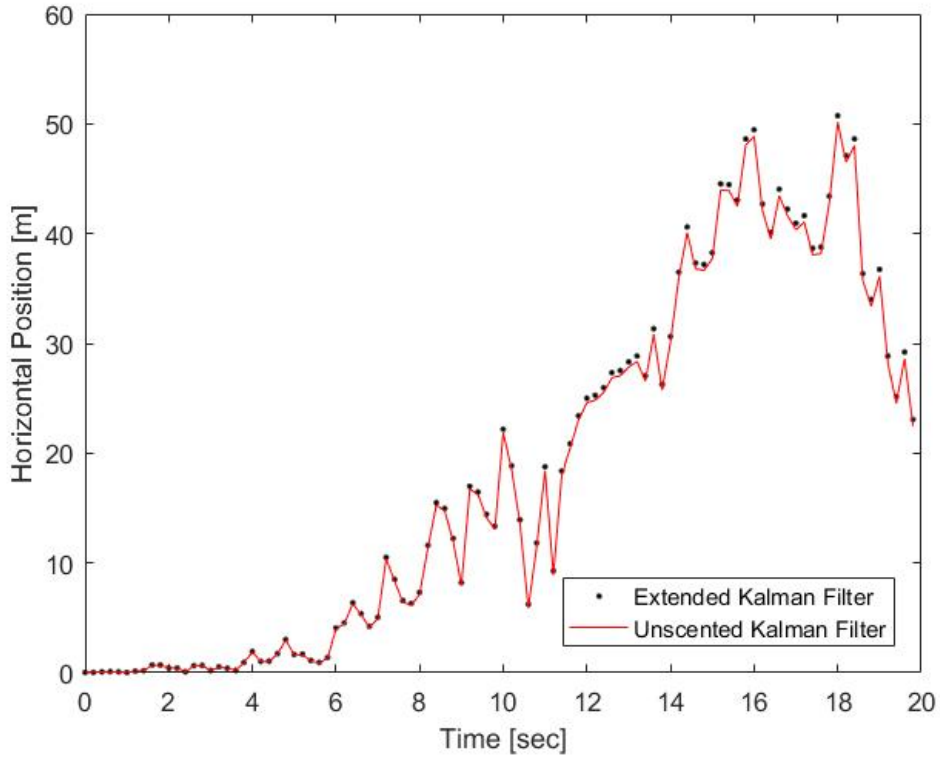


Figure 52: Absolute difference value between estimate and true trajectory

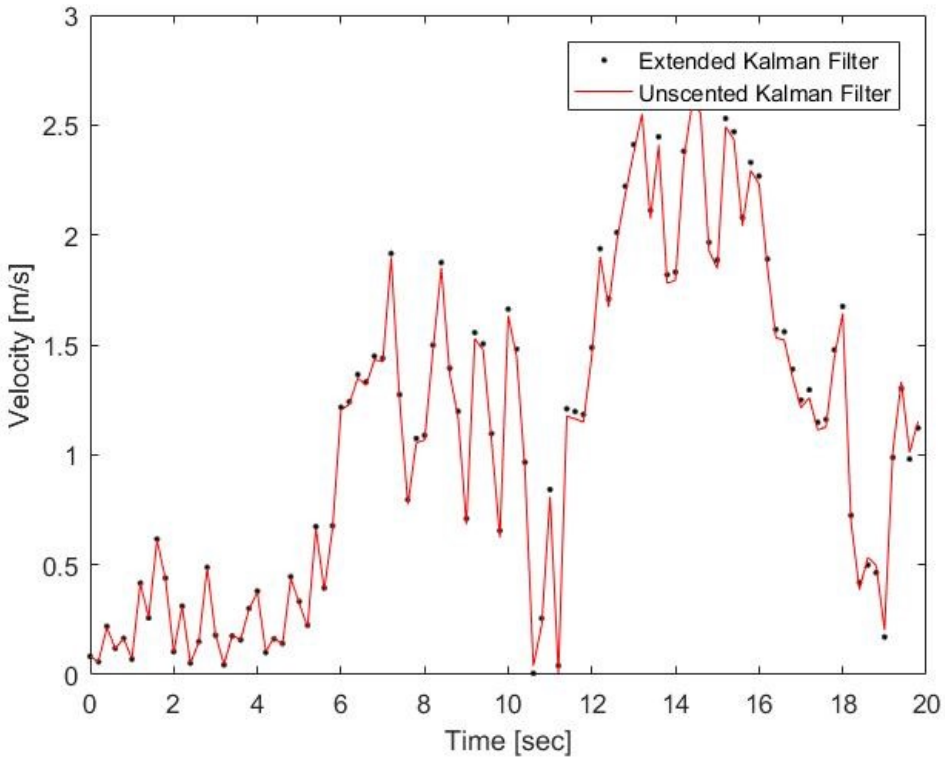


Figure 53: Absolute difference value between estimate and true trajectory.

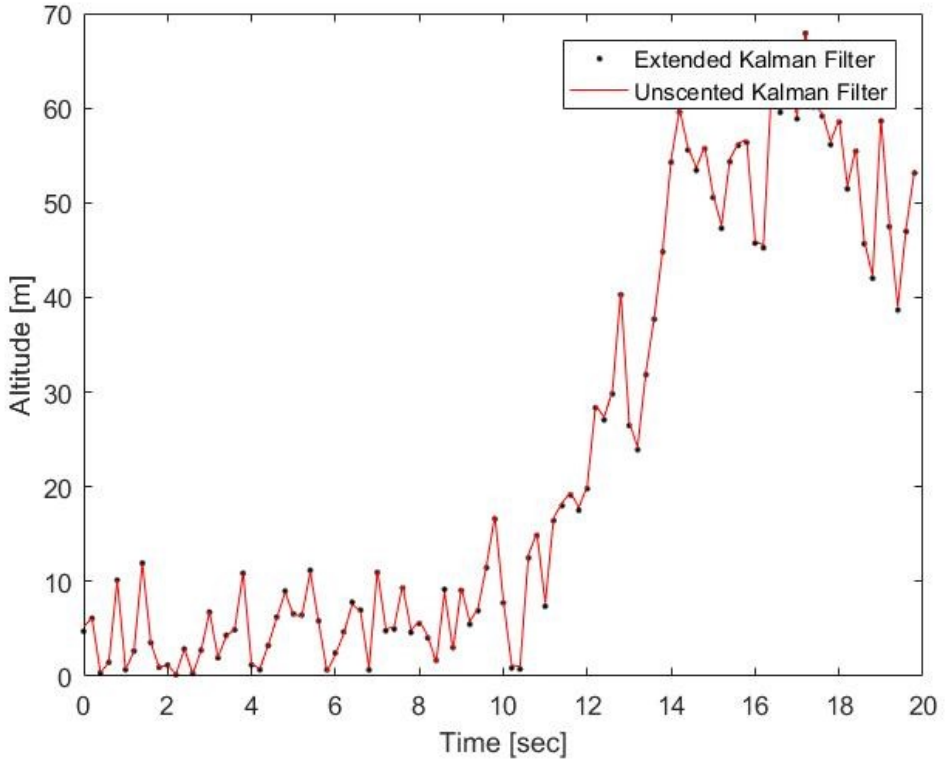


Figure 54: Absolute difference value between estimate and true trajectory.

The three plots Figure 52, 53 and 54 show the down-sampled absolute difference values between estimates of both filter output and true values, from which we can see that the extended Kalman filter has a little higher estimate error. Because the unscented transform does a slightly better job of estimating the mean and variance of a random variable that has been subject to a nonlinear transformation over a linearized approach.

The unscented Kalman filter provides an alternative way when the Jacobian matrix is hard to compute or there is a possibility for divergence in an extended Kalman filter. Hence, its application to the Kalman filter concern strictly the process model or the measurement model or maybe both.

4.5.6 Unscented Kalman Filter for Radar Tracking (coloured noise)

The background setting is the same as Example 4.5.3. But in this simulation, one wants to use the unscented Kalman filter to estimate the position and altitude of an aircraft with measured slant range data from a radar station. It is assumed that the aircraft is moving in 2-dimension with constant speed and altitude. In addition to the white measurement noise, a coloured component (Markov noise) is added on the measurement and modelled with additional state variables [16].

Both extended Kalman filter and unscented Kalman filter is used to estimating the same process as used in Example 4.5.3 to compare the difference between them. Most parameters are unchanged except the covariance of first estimate becomes P = [0.1 1 0; 1 10 0; 0 0 9] for the matrix positive definite issue.

Discrete process model:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.9802 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_k + \begin{bmatrix} 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ w_x \\ w_y \\ w_m \end{bmatrix}_k$$

where $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} horizontal\ distance \\ horizontal\ velocity \\ attitude \\ Markov\ noise\ in\ measurement \end{bmatrix}$; $w_x = 0.1\ m/s$ (standard deviation of white velocity noise); $w_y = 3\ m$ (standard deviation of white attitude noise); $w_m = 19.80151\ m$ (standard deviation of Markov noise); $\Delta t = 0.05\ s$ (process time interval)

Measurement model:

$$z = \sqrt{x_1^2 - x_3^2} + x_4 + v$$

where v is white measurement noise, x_4 is Markov measurement noise.

Linearized measurement model:

$$z_k = \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2 + x_3^2}} & 0 & \frac{x_3}{\sqrt{x_1^2 + x_3^2}} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_k + v_k$$

where $v_k = 1\%$ of measurement (standard deviation of measurement white noise).

Table 21: Kalman filter and measurement parameters.

State Transition Matrix (A)	$\begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.9802 \end{bmatrix}$	/
State to Measurement Matrix (H)	$\begin{bmatrix} \frac{x_1}{\sqrt{x_1^2 + x_3^2}} & 0 & \frac{x_3}{\sqrt{x_1^2 + x_3^2}} & 1 \end{bmatrix}$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} (\Delta t \cdot w_x)^2 & (\Delta t \cdot w_x) \cdot w_x & 0 & 0 \\ (\Delta t \cdot w_x) \cdot w_x & (w_x)^2 & 0 & 0 \\ 0 & 0 & (w_y)^2 & 0 \\ 0 & 0 & 0 & (w_m)^2 \end{bmatrix}$	/
Covariance of White Measurement Noise (R)	$(1\% \text{ of measurement})^2$	m^2
First Estimate ($\hat{x}_0$)	$\begin{bmatrix} 0 \\ 100 \\ 1000 \\ 0 \end{bmatrix}$	$\begin{bmatrix} m \\ m/s \\ m \\ m \end{bmatrix}$
Error Covariance of First Estimate (P_0)	Q (covariance of the state transition noise)	/
Measurement time Interval (Δt)	0.05	sec
Simulation Time	20	sec

The three plots Figure 55, 56 and 57 show the down-sampled absolute difference values between estimate results of both filter output and true values, from which we can see that the extended Kalman filter has a noticeable higher estimate error. Because the unscented transform does a better job of estimating the mean and variance of a random variable, which is an important advantage when the measurement noise is coloured.

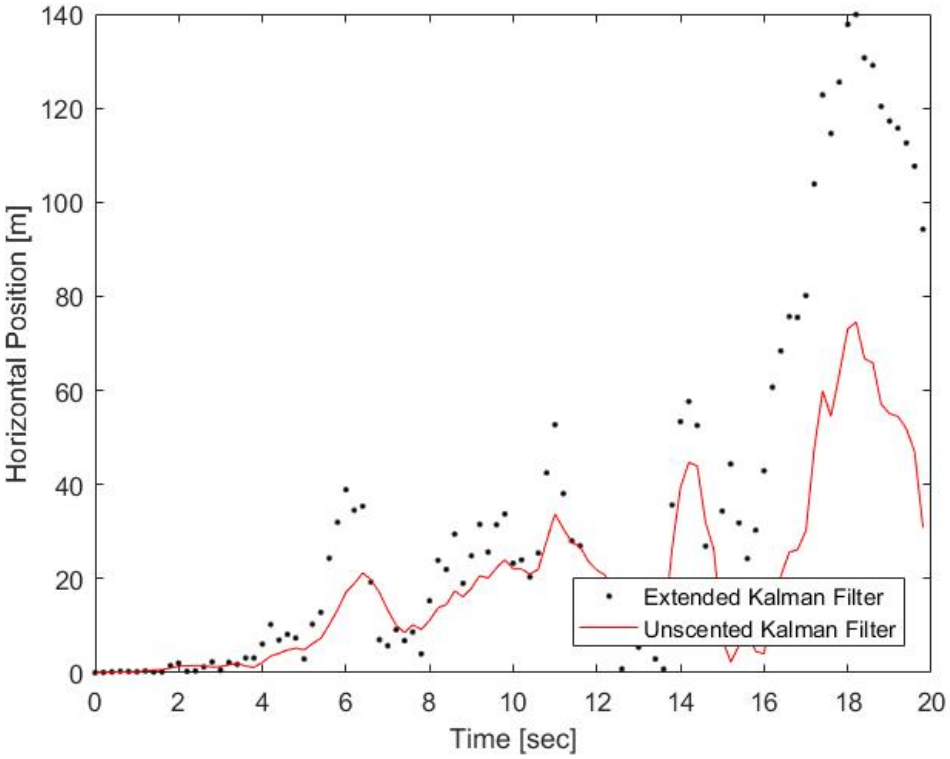


Figure 55: Absolute difference value between estimate and true trajectory

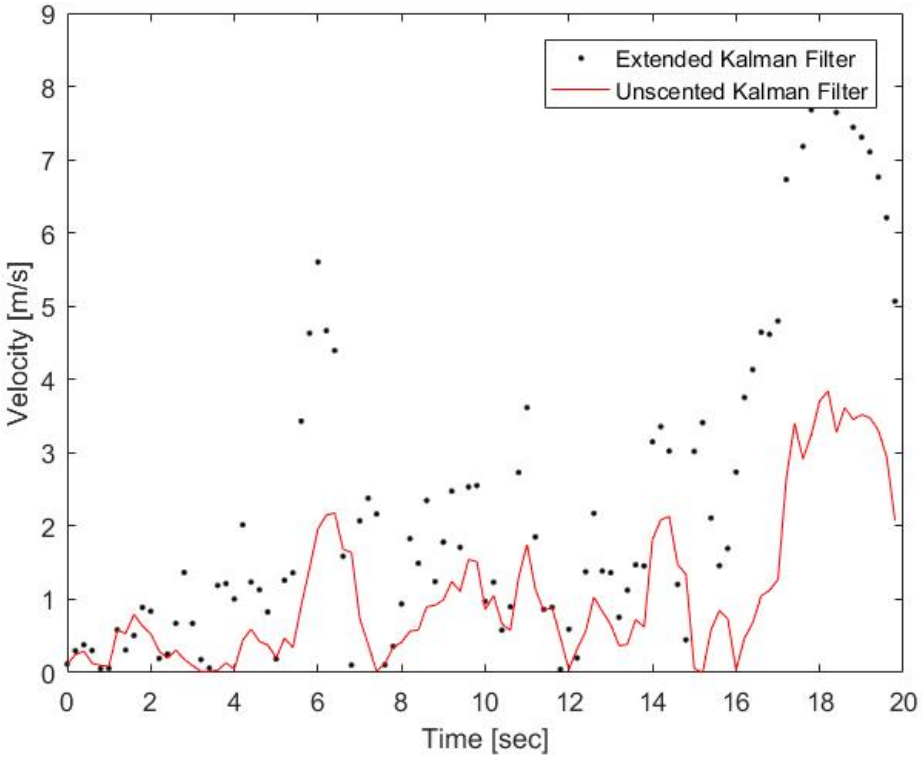


Figure 56: Absolute difference value between estimate and true trajectory.

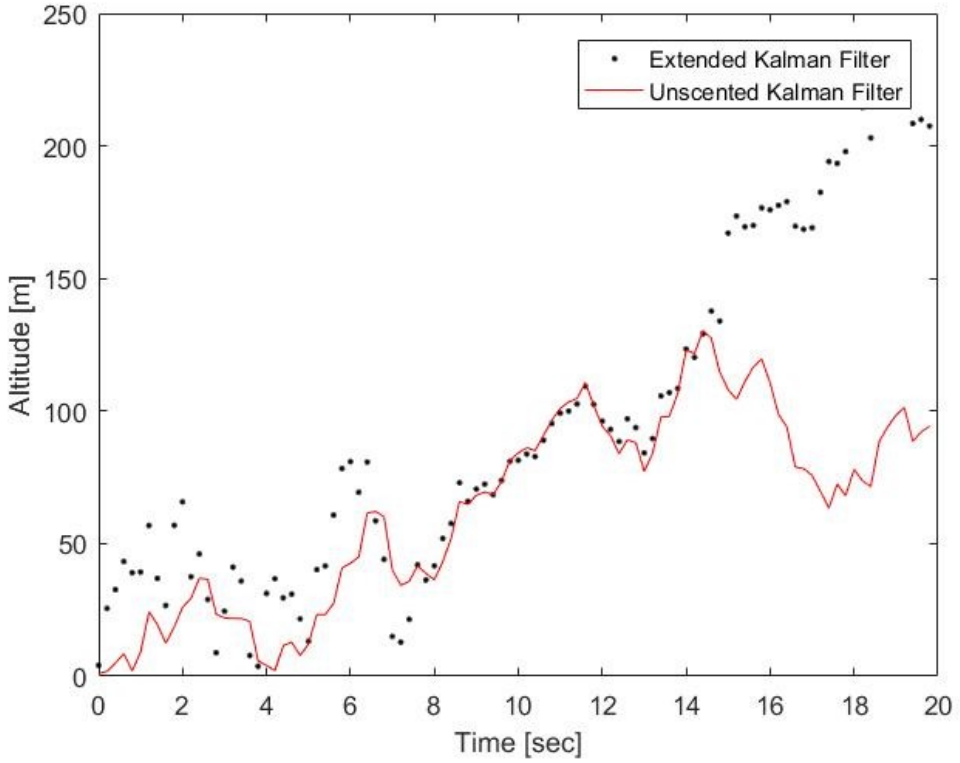


Figure 57: Absolute difference value between estimate and true trajectory.

4.5.7 Unscented Kalman Filter for Attitude Reference System

The background setting is the same as Example 4.5.4. One wants to measure the horizontal attitude of a helicopter with the accelerometers and gyros. A test signal is applied to oscillate the navigation sensor with a sinusoidal wave of 0.2 Hz frequency and ±30° amplitude. The unscented Kalman filter is used to optimize the estimate of the altitude. Both extended Kalman filter and unscented Kalman filter are used to estimating the same process as used in Example 4.5.4 to compare the difference between them. All parameters are unchanged in this simulation [16].

The plots Figures 58 and 59 show that the unscented Kalman filter and extended Kalman filter achieved a similar result. Both the amplitude and trend match true values.

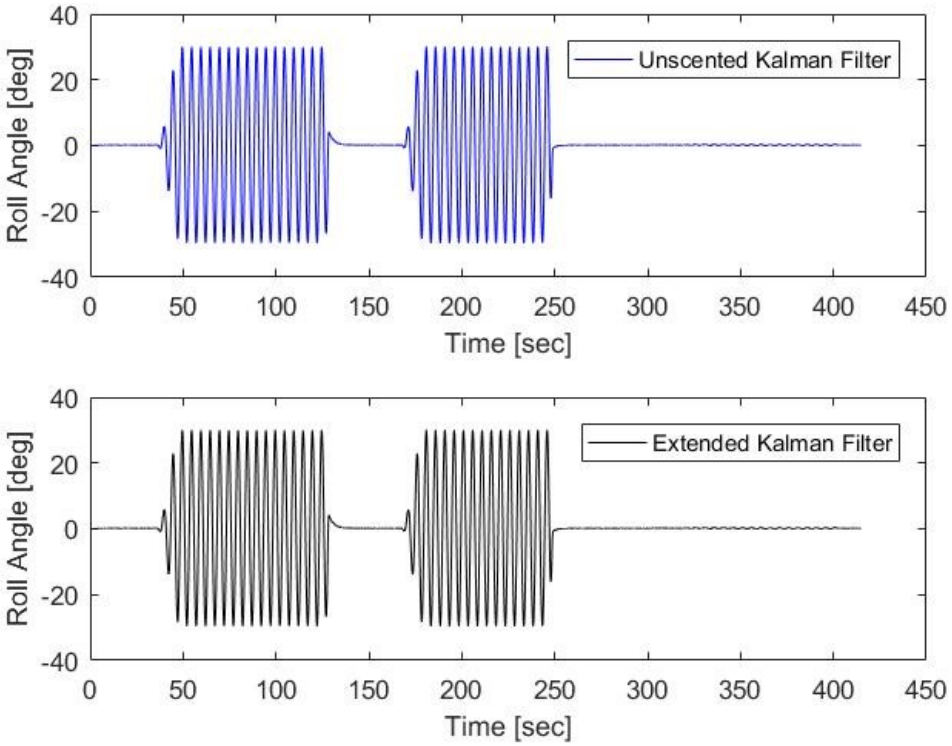


Figure 58: Roll angle estimate of unscented and extended Kalman filter.

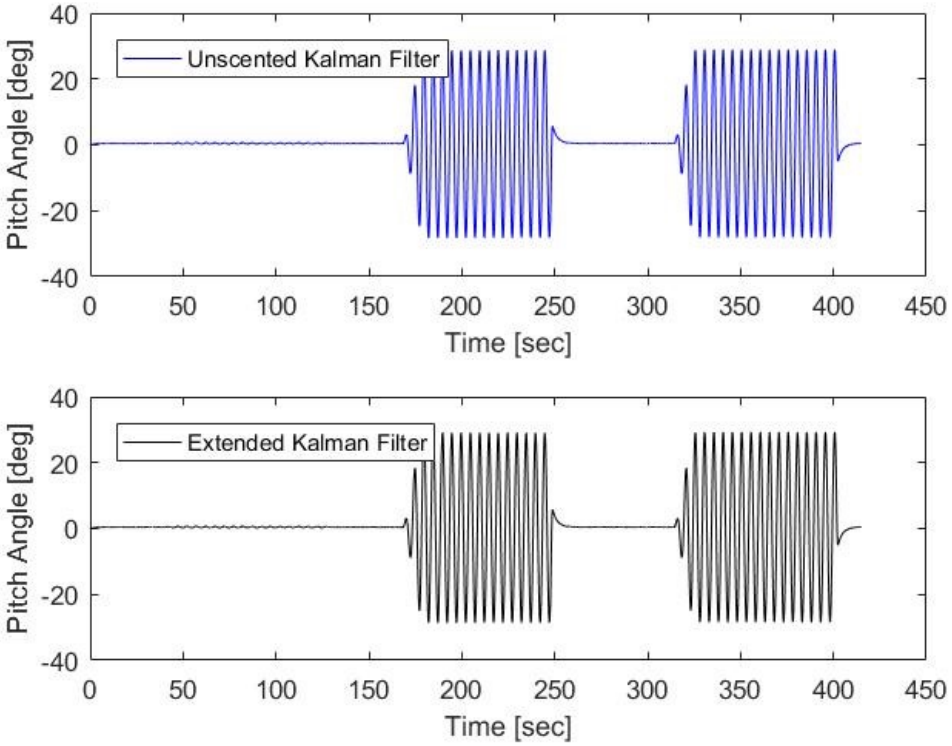


Figure 59: Pitch angle estimate of unscented and extended Kalman filter.

Unscented Kalman filter is a nonlinear Kalman filter based on the strategy preferring the modelling of the probability distribution of a given nonlinear function rather than the function itself. Instead of finding an approximation of the nonlinear function in a linearized form, it takes the approach of computing the mean and covariance of the function in an approximate manner.

The unscented Kalman filter is a good substitute for an extended Kalman filter when the linearized model from Jacobian is unstable or hard to be obtained. Also, it is practical because the implementation of the algorithm is not so complicated compared to that of an extended Kalman filter.

4.5.8 Adaptive Kalman Filter

Same as Example 4.3.7, an elementary physics experiment that is intended to measure the gravitational constant g. A mass is released at $t = 0$ in a vertical-evacuated column and multiple-exposure photographs of the falling mass are taken at 0.05 sec intervals beginning at $t = 0.05$ sec. Assuming there is no prior knowledge about the gravity constant and there are 3 possible values of α in the state to measurement transition matrix $H = \alpha \cdot t^2$, where α could be 1, 1/2 or 1/3. So, the adaptive Kalman filter is used to estimate the gravity constant and parameter α [17].

Table 22: Truth model and Kalman filter model.

Truth Model	$x = 9.8$	m/s^2
Process Model	x is constant, where $x \in (-10000, +1000)$	m/s^2
State Transition Matrix (A)	1	/
State to Measurement Matrix (H)	$\alpha \cdot t^2$	/
Distribution of α	$\left[1 \quad \frac{1}{2} \quad \frac{1}{3}\right]$	/
Corresponding Possibility Density of α	[33% 33% 33%]	/
Covariance of the State Transitions Noise (Q)	0	m^2/s^4

Covariance of Measurement Noise (R)	0.0001	m^2
Prior Estimate $\hat{x}_0^-$	0	m/s^2
Error Covariance of Prior Estimate P_0^-	96*	m^2/s^4
Measurement Time Interval	0.05	sec
Simulation Time	1	sec

The value of error variance of prior estimate P_0^- has been changed from infinite to 96, because from several sample run of this simulation, it turns out that the Magill Adaptive Kalman filter needs an accurate P_0^- to calculate the weight factors and fallow the actual trajectory. A very inaccurate P_0^- (the extreme case is $P_0^- = \infty$) will make the adaptive filter choose the wrong α and hence fail to fallow the actual trajectory.

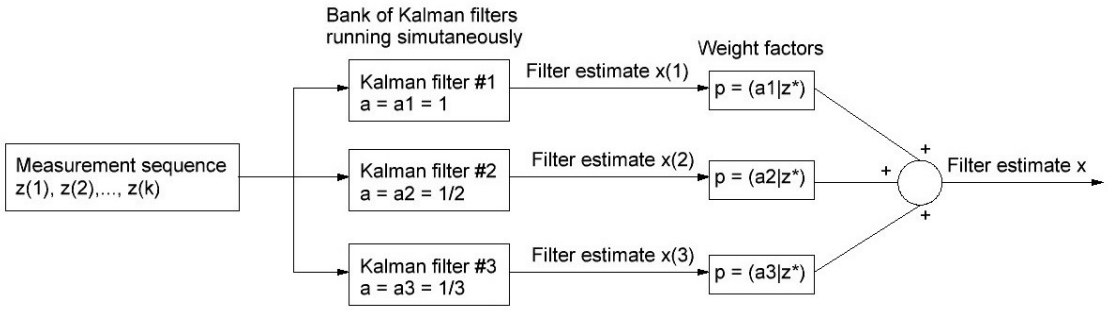


Figure 60: Weighted sum of Kalman filter estimates.

The gravity constant estimate is shown along with the actual movement in Figure 61. The estimate converges to the correct value (9.8) after a few steps. It could be seen in Figure 62 that the filter estimate converges to the accurate system parameter (1/2) at the end of the simulation. The accuracy of the initial prior estimate and its error variance is very important for the adaptive Kalman filter converges to the right parameter value. Because the solution for the adaptive Kalman filter with infinite initial prior estimate error has not been found.

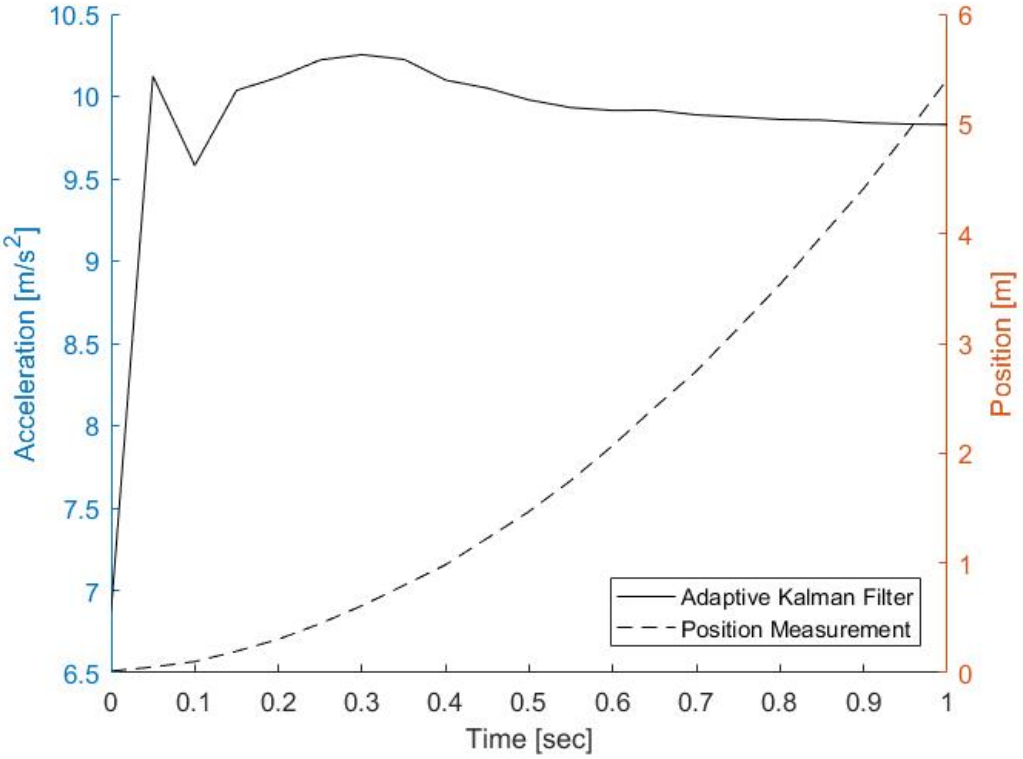


Figure 61: Gravity constant estimate and actual displacement.

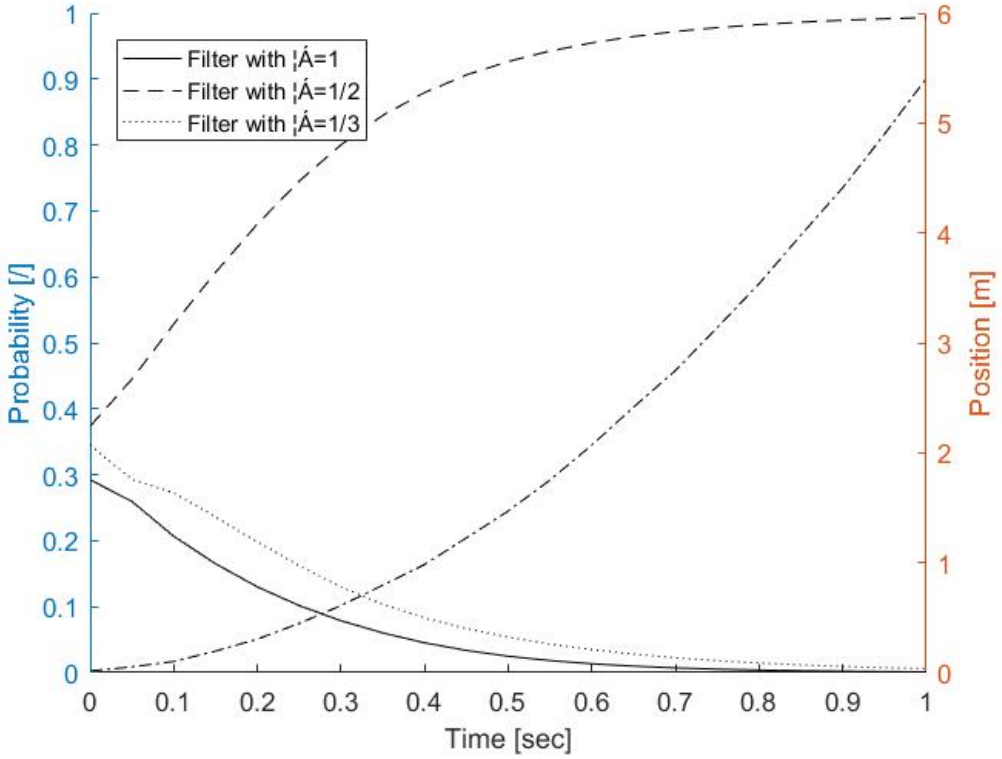


Figure 62: Weight factors of the parallel Kalman filters.

4.5.9 Delayed-State Kalman Filter

Consider a scalar Gauss-Markov process with an autocorrelation function $R_x(\tau) = e^{-|\tau|}$ and continuous noisy measurement of it with Gaussian white noise. Suppose the delayed-state Kalman filter is used to make filter recursive cycle be relatively slow in order not to impose hardship on the system computer. The continuous Kalman filter, Delayed-State Kalman filter and usual Kalman filter are used to propagate the error covariance of the estimate and compare their accuracies [17].

Table 23: Delayed-state Kalman filter model.

Process Model	$\dot{\mathrm{x}} = -\mathrm{x} + \sqrt{2}\mathrm{u(t)};\ z = \mathrm{x} + \mathrm{v}(t)$	/
State Transition Matrix (A)	$\begin{bmatrix} 1 & 0.0951626 \\ 0 & 0.9048374 \end{bmatrix}$	/
State to Measurement Matrix (H)	$\mathrm{H} = [10\ 0], \mathrm{J} = [-10\ 0]$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} 0.00061892 & 0.0090559 \\ 0.0090559 & 0.1812692 \end{bmatrix}$	/
Covariance of Measurement Noise (R)	0.2	/
Error Covariance of Prior Estimate P_0^-	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	/
Measurement Interval	0.1	sec
Simulation Time	1	sec

Table 24: Usual Kalman filter model.

Process Model	$\dot{\mathrm{x}} = -\mathrm{x} + \sqrt{2}\mathrm{u(t)};\ z = \mathrm{x} + \mathrm{v}(t)$	/
State Transition Matrix (A)	0.9048374	/
State to Measurement Matrix (H)	1	/
Covariance of the State Transitions Noise (Q)	0.1812692	/
Covariance of Measurement Noise (R)	0.2	/
Error Covariance of Prior Estimate P_0^-	1	/
Measurement Interval	0.1	Sec
Simulation Time	1	Sec

Table 25: Continuous Kalman filter model.

Process Model	$\dot{\mathrm{x}} = -\mathrm{x} + 2z;\ \dot{z} = 50\mathrm{x} + \mathrm{z}$	/
Error Covariance P	$\dfrac{0.450248e^{-\sqrt{101}t} + 0.549752e^{\sqrt{101}t}}{-2.037345e^{-\sqrt{101}t} + 3.037345e^{\sqrt{101}t}}$	/

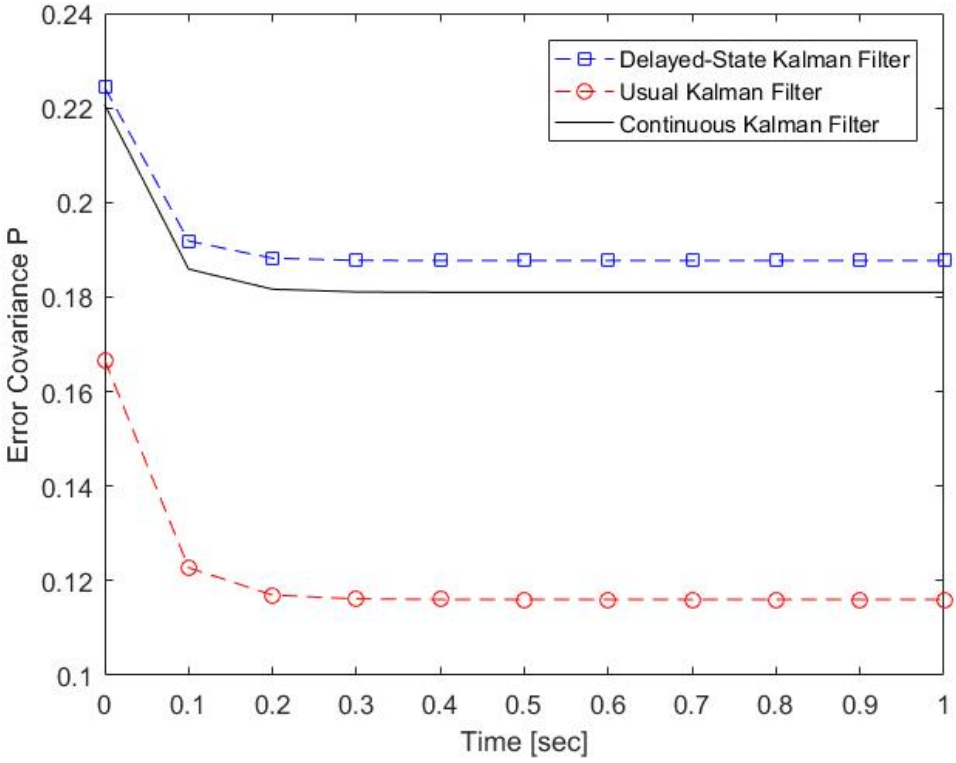


Figure 63: Estimate error covariance of the three filters.

Table 26: Estimate error covariance of the three filters.

step	Delayed state Kalman filter	Usual Kalman filter with only one state variable	Continuous Kaman filter
1	0.224321948970811	0.166666666666667	0.220695670582449
2	0.191835904810728	0.122738804322829	0.185897714312996
3	0.188189555898923	0.116970973185464	0.181647273165949
4	0.187749299089200	0.116149049358585	0.181084485028937
5	0.187695687427977	0.116030597543824	0.181009195418577
6	0.187689152181513	0.116013499226864	0.180999109354589
7	0.187688355436325	0.116011030538908	0.180997757941075
8	0.187688258299568	0.116010674092994	0.180997576863155
9	0.187688246456927	0.116010622626665	0.180997552600173
10	0.187688245013104	0.116010615195566	0.180997549349128
11	0.187688244837078	0.116010614122607	0.180997548913514

The results show that the P computation of the simplified one-state model yields an overly optimistic estimate of the filter's accuracy. However, despite the modelling errors, the one-state filter's performance is nearly as good as the optimal two-state filter, which uses the delayed-state measurement model.

4.5.10 Schmidt-Kalman Filter

The following example shows a comparison between Schmidt-Kalman Filter and the normal Kalman filter. Consider a situation where one wishes to estimate a random-walk process whose initial value is a random variable described as N (0,1). The measurement noise in this situation will be assumed to be the sum of a unity variance Markov component plus a white component with a variance of 2 units. The optimal Kalman filter, Schmidt-Kalman filter, Kalman filter with bumped-up-R and model with Markov state-ignored are used to propagate the error covariance of the estimate and compare their accuracies [17].

Table 27: Usual Kalman filter model and delayed-state model.

Process Model	$\dot{\mathrm{x}} = -\mathrm{x} + \sqrt{2}\mathrm{u}(\mathrm{t});\ z = \mathrm{x} + \mathrm{v}(t)$	/
State Transition Matrix (A)	$\begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}$	/
State to Measurement Matrix (H)	$[1 \quad 0]$	/
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} 0.25 & 0 \\ 0 & 0.75 \end{bmatrix}$	/
Covariance of Measurement Noise (R)	2	/
Error Covariance of Prior Estimate P_0^-	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	/
Measurement Interval	0.05	sec
Simulation Time	1	sec

Table 28: Markov noise ignored model and bumped-up-r model.

Process Model	$\dot{\mathrm{x}} = -\mathrm{x} + \sqrt{2}\mathrm{u}(\mathrm{t});\ z = \mathrm{x} + \mathrm{v}(t)$	/
State Transition Matrix (A)	1	/
State to Measurement Matrix (H)	1	/
Covariance of the State Transitions Noise (Q)	0.25	/
Covariance of Measurement Noise (R)	Bumped-up model: 3 Markov-ignored model: 2	/
Error Covariance of Prior Estimate P_0^-	1	/
Measurement Interval	0.05	sec
Simulation Time	1	sec

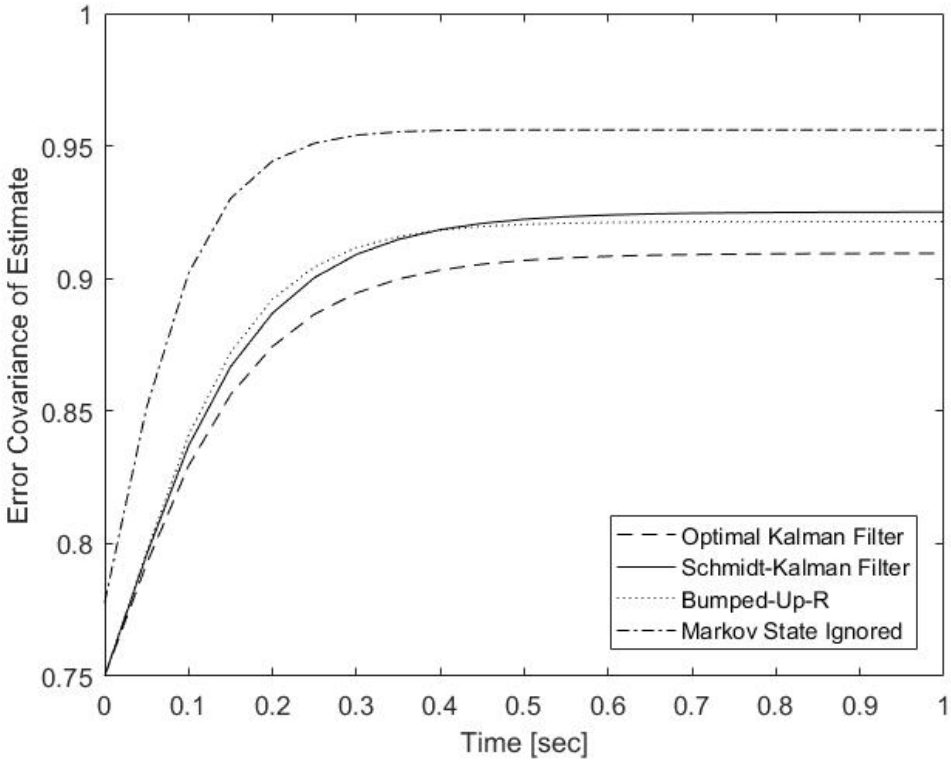


Figure 64: Error covariance of four Kalman filter estimates.

As could be expected, the optimal filter is the best; the Schmidt-Kalman filter and the filter with R bumped up have similar accuracy and are close to the optimal filter; the filter with Markov noise completely ignored is a poor fourth. Some degree of suboptimality results in accounting for the discarded state with a Schmidt-Kalman filter (or by simply bumping up R), but this is much better than ignoring them completely.

4.6 The Global Positioning System

4.6.1 Kalman Filter for GPS with Pseudo range Measurement

Consider a situation that there are an observer and four GPS satellites. Both the observer and satellites are assumed to be stationary for simplification. It is desired to estimate the observer's position and velocity based on a sequence of GPS pseudo-range measurements. The measurement matrix is linearized around the nominal trajectory [17].

Discrete process model:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}_k + \begin{bmatrix} w_{ep} \\ w_{ev} \\ w_{np} \\ w_{nv} \\ w_a \\ w_{at} \\ w_{cb} \\ w_{cd} \end{bmatrix}_k$$

where $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} East\ position\ error \\ East\ velocity \\ North\ position\ error \\ North\ velocity \\ altitude\ error \\ altitude\ rate \\ clock\ bias\ state \\ clock\ drift\ state \end{bmatrix}; \begin{bmatrix} w_{ep} \\ w_{ev} \\ w_{np} \\ w_{nv} \\ w_a \\ w_{at} \\ w_{cb} \\ w_{cd} \end{bmatrix} = \begin{bmatrix} 3.154\ m \\ 5.462\ m/s \\ 3.154\ m \\ 5.462\ m/s \\ 3.154\ m \\ 5.462\ m/s \\ 0.9625 \times 10^{-9}\ sec \\ 1.257 \times 10^{-9}\ sec/s \end{bmatrix}; \Delta t = 1\ s$

Measurement model:

$$z = \sqrt{(x_1 - o_x)^2 + \left(x_3 - o_y\right)^2 + (x_5 - o_z)^2} + cx_7$$

where $[o_x, o_y, o_y] = [0,0,0]$ is observer location; $c = 299792458\ m/s$ is speed of light.

Linearized measurement model (around nominal point [0, 0, 0]):

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}_k = \begin{bmatrix} h_x^{(1)} & 0 & h_y^{(1)} & 0 & h_z^{(1)} & 0 & c & 0 \\ h_x^{(2)} & 0 & h_y^{(2)} & 0 & h_z^{(2)} & 0 & c & 0 \\ h_x^{(3)} & 0 & h_y^{(3)} & 0 & h_z^{(3)} & 0 & c & 0 \\ h_x^{(4)} & 0 & h_y^{(4)} & 0 & h_z^{(4)} & 0 & c & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}_k + v_k$$

where $v_k = [15m \quad 15m \quad 15m \quad 15m]^T$ is variance of measurement white noise.

$$\begin{bmatrix} h_x^{(i)} \\ h_y^{(i)} \\ h_z^{(i)} \end{bmatrix} = \begin{bmatrix} \frac{\partial \psi_i}{\partial x} \\ \frac{\partial \psi_i}{\partial y} \\ \frac{\partial \psi_i}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{-(x_1 - o_x)}{\sqrt{(x_1 - o_x)^2 + \left(x_3 - o_y\right)^2 + (x_5 - o_z)^2}} \\ \frac{-(x_3 - o_y)}{\sqrt{(x_1 - o_x)^2 + \left(x_3 - o_y\right)^2 + (x_5 - o_z)^2}} \\ \frac{-(x_5 - o_z)}{\sqrt{(x_1 - o_x)^2 + \left(x_3 - o_y\right)^2 + (x_5 - o_z)^2}} \end{bmatrix} for\ i = 1, \dots, 4$$

Table 6.1: Kalman filter and simulation parameters.

State Transition Matrix (A)	$\begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
State to Measurement Matrix (H)	$\begin{bmatrix} h_x^{(1)} & 0 & h_y^{(1)} & 0 & h_z^{(1)} & 0 & c & 0 \\ h_x^{(2)} & 0 & h_y^{(2)} & 0 & h_z^{(2)} & 0 & c & 0 \\ h_x^{(3)} & 0 & h_y^{(3)} & 0 & h_z^{(3)} & 0 & c & 0 \\ h_x^{(4)} & 0 & h_y^{(4)} & 0 & h_z^{(4)} & 0 & c & 0 \end{bmatrix}$
Covariance of the State Transitions Noise (Q)	$\begin{bmatrix} c_p & c_{pv} & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{pv} & c_v & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_p & c_{pv} & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{pv} & c_v & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_p & c_{pv} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{pv} & c_v & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_b & c_{bd} \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{bd} & c_d \end{bmatrix}$
Covariance of Measurement Noise (R)	$\begin{bmatrix} 225 & 0 & 0 & 0 \\ 0 & 225 & 0 & 0 \\ 0 & 0 & 225 & 0 \\ 0 & 0 & 0 & 225 \end{bmatrix}$
First Estimate ($\hat{x}_0$)	$[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$
Error Covariance of Prior Estimate (P_0)	Q (covariance of the state transition noise)
Measurement time Interval (Δt)	1s
Simulation Time	100s

where $dt = 1s$; $\begin{bmatrix} h_x^{(i)} \\ h_y^{(i)} \\ h_z^{(i)} \\ c \end{bmatrix} = \begin{bmatrix} -0.9371 \\ -0.2449 \\ -0.2487 \\ 299792458 \end{bmatrix}$; $\begin{bmatrix} c_p \\ c_{pv} \\ c_v \\ c_b \\ c_{bd} \\ c_d \end{bmatrix} = \begin{bmatrix} 9.945 \\ 14.9175 \\ 29.835 \\ 0.9264 \times 10^{-18} \\ 0.7896 \times 10^{-18} \\ 1.579 \times 10^{-18} \end{bmatrix}$

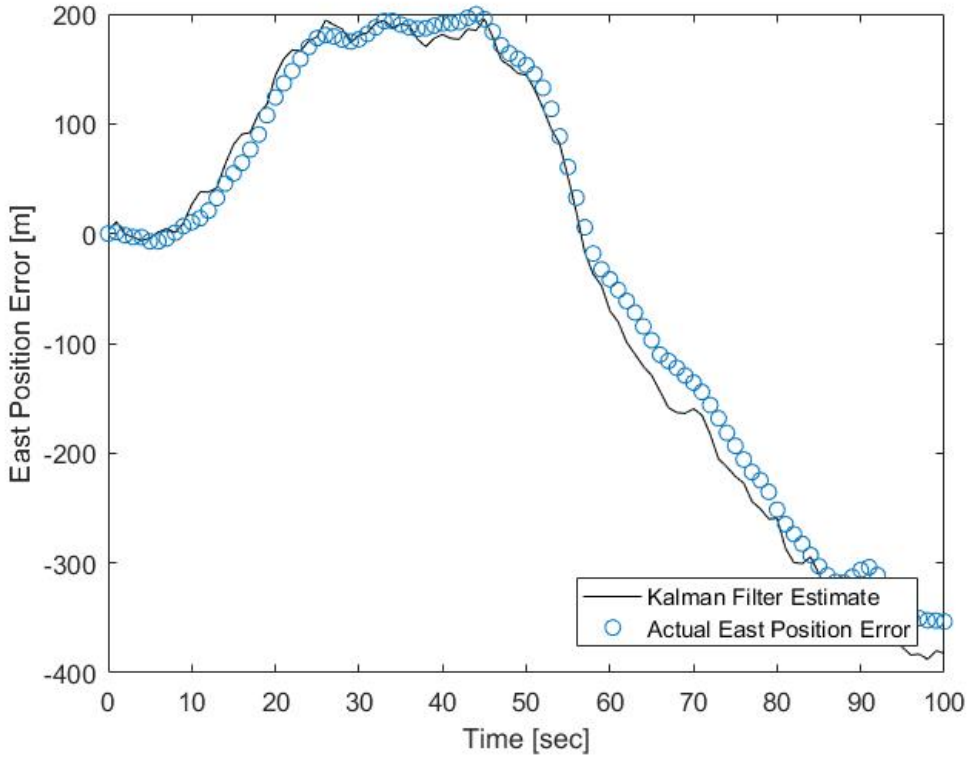


Figure 65: Kalman filter estimate and actual east position error.

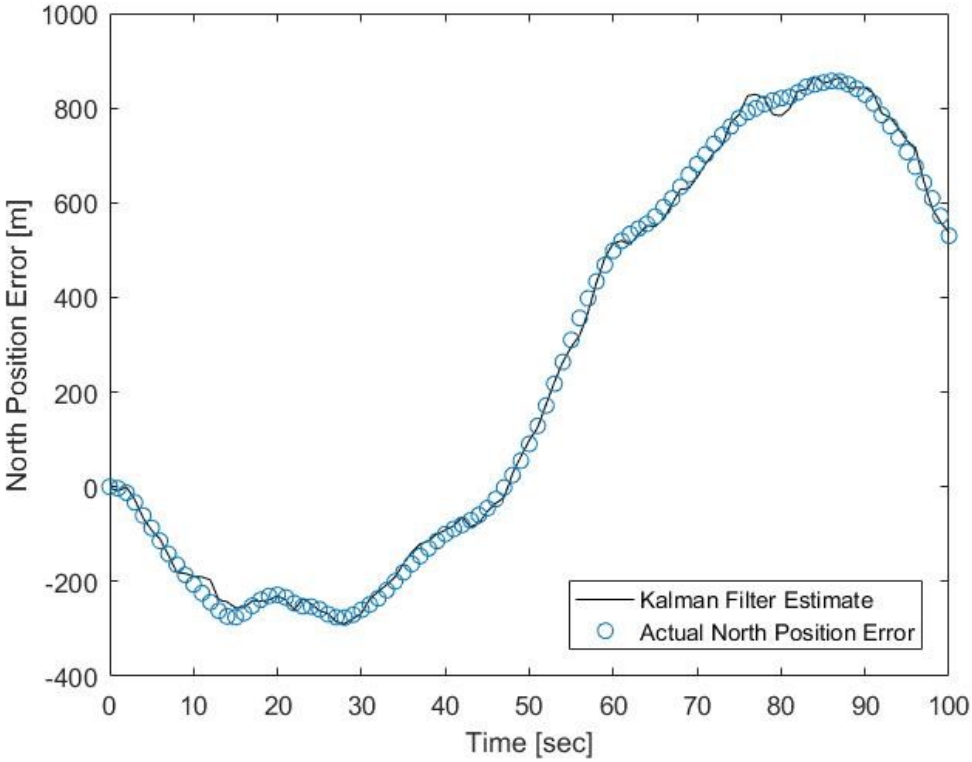


Figure 66: Kalman filter estimate and actual north position error.

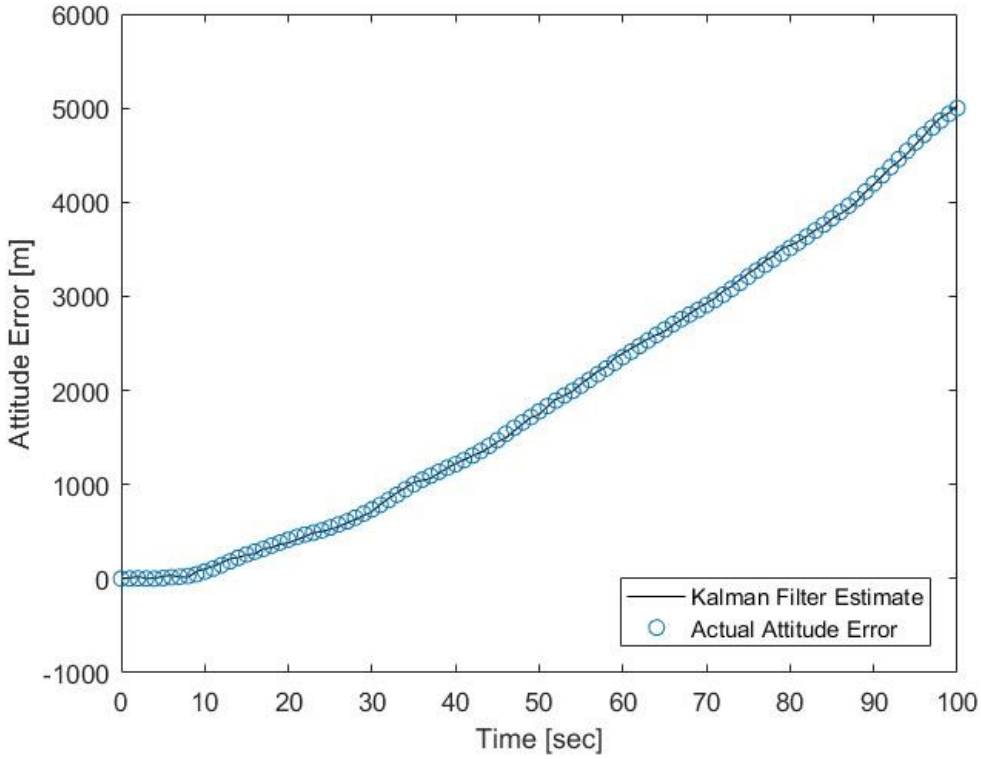


Figure 67: Kalman filter estimate and actual altitude error.

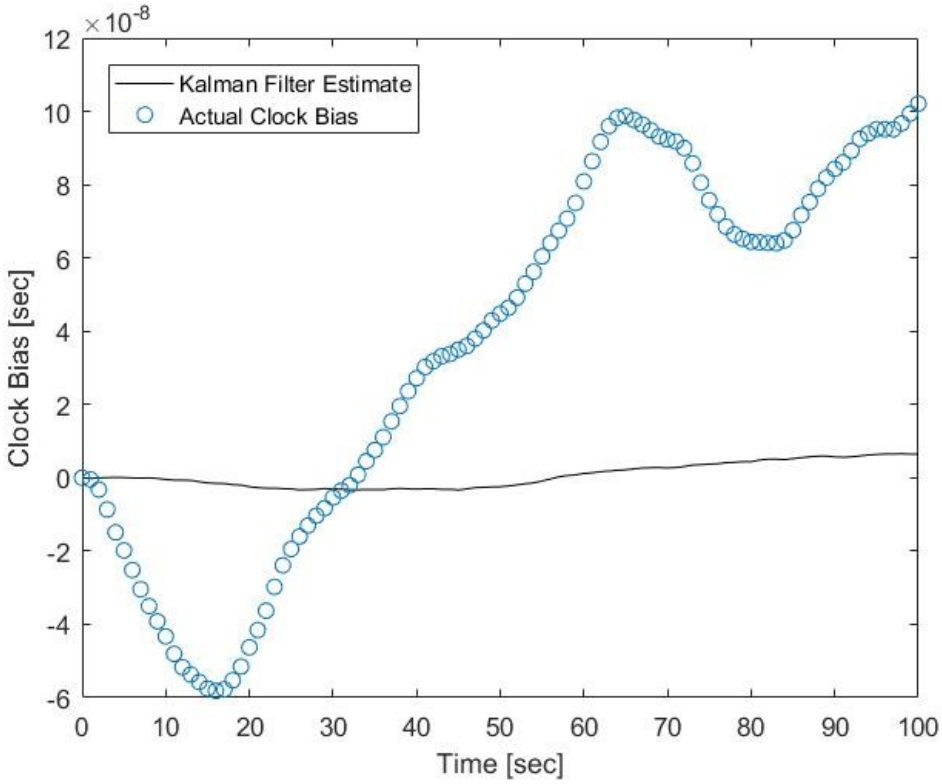


Figure 68: Kalman filter estimate and actual clock bias.

It could be in the result figures that the Kalman filter estimates of the position errors (East, West, Attitude) follow the actual value quite well, but does not follow the correct clock bias value. The possible reason for this issue is that all the state variables lumped into one measurement and the quantity of clock bias error is much smaller compared to the other three position errors in the combination, which makes the gain for clock bias too small to follow the actual trend.

Chapter 5: Simultaneously Localization and Mapping

In autonomous navigation, robotic mapping and odometry for virtual reality or augmented reality, simultaneous localization and mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it. [6] [18] [19] Creating an environment map using multi-sensor information is considered important for a lot of tasks in autonomous navigation (e.g. object recognition). In particular, multi-laser systems or a laser scanner linked with radar sensors, sonar, or video cameras ought to be considered. The system improvement with the multi-sensor and sensor fusion procedure would make the recognizing procedure more effective and the overall system more robust [20]. SLAM algorithms are tailored to the available resources, hence not aimed at perfection, but at operational compliance [20]. Published approaches are employed in self-driving vehicles, unmanned aerial vehicles, autonomous underwater vehicles, planetary rovers, newer domestic robots and even inside the human body [21]. From the probabilistic perspective, there are two main forms of the SLAM problem. One is known as the online SLAM problem which is estimating the posterior over the momentary pose along with the map [22]. The second SLAM problem is called the full SLAM problem which is to calculate a posterior over the entire path along with the map, instead of just the current pose.

This chapter specifically addresses the problem of feature-based online SLAM, where the environment is modelled as a discrete set of features, each described by a few of continuous state variables. The standard solution is to use a Bayesian approach which is modelling the joint probability distribution over possible robot trajectories and maps [23]. There are currently two popular approaches to modelling this distribution. The first is to

linearize and represent the joint probability with a single high-dimensional Gaussian. This is the approach taken by EKF SLAM [4] and its variants [6]. The second is to use a Rao-Blackwellised particle filter, representing the robot's trajectory using a set of particles and conditioning the map on the robot's trajectory. Examples of this approach include Fast SLAM [24] and others (e.g. [25]).

In this chapter, we present an improved Fast SLAM strategy, which combines the strengths of both approaches. We model the particles as the robot pose mean of a Gaussian distribution and keep the error covariance of robot pose estimation (P) and feature position estimation (P) in Fast SLAM 2.0 propagating individually as normal EKF SLAM instead of setting it to zero after sampling in each iteration. The result is an improved Fast SLAM algorithm which is robust to linearization errors and data association ambiguities on a local scale and does not have the overconfident issue in Fast SLAM 2.0.

5.1 Extended Kalman Filter (EKF) SLAM

EKF SLAM is an iterative least-square on-line SLAM method which uses linearized Kalman filter to optimize the state estimation by minimizing the error between prediction and measurement [26]. It is optimal in the statistical sense when the noise distribution is Gaussian. While on the other hand, when applied to the highly nonlinear system, the accumulation of the random linearization error could be very large.

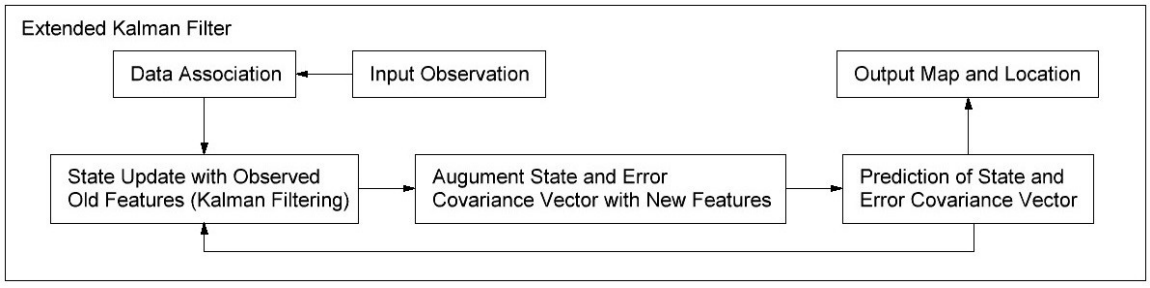


Figure 69: Extended Kalman filter

The major shortcoming of EKF SLAM is its limitation to the first-order accuracy of

propagated the mean and covariance as a result of first-order truncated Taylor series linearization technique [27]. The EKF SLAM is prone to failure where significant vehicle uncertainty induces the linearization errors, or where significant clutter induces ambiguity in data association. The second issue is problematic because the standard EKF SLAM requires that accurate data association decisions be made, via using the most likely hypothesis. Once a wrong association is made, the error will be accumulated in the filter estimate and may cause the filter to fail catastrophically [23].

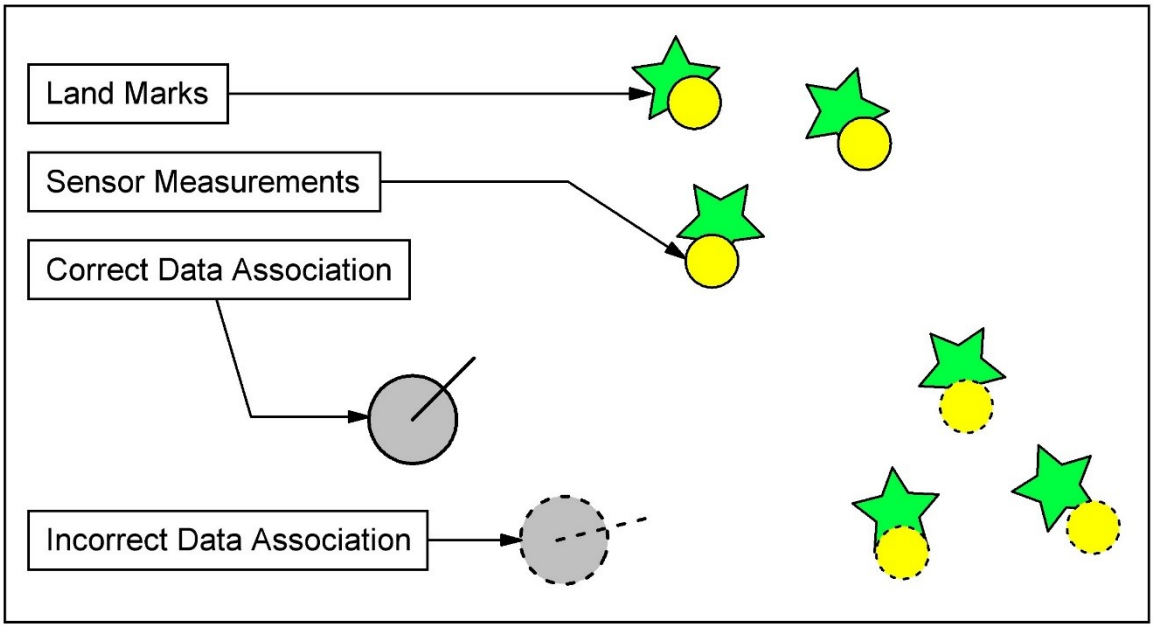


Figure 70: Example of data association error caused by scene ambiguity

As could be seen in Figure 70, the scene ambiguity will lead large errors in filter estimate since the filer will give the same weight for every estimate of the robot pose and combine all the filter estimate as the final output. Even a single wrong data association is made, the introduced error could be very large.

For remembering long-term uncertainty, the EKF SLAM is more superior than the Fast SLAM. By using a continuous Gaussian representation, the pose distribution does not degrade purely as a function of trajectory length as in the Fast SLAM. Linearization errors

are still a concern, but these are far less severe than Fast SLAM's particle diversity problems [23].

5.2 Graph-Based Least Squared SLAM

Graph-based least squared SLAM is a nonlinear least square full SLAM method. It uses a graph to represent the problem. Every node in the graph corresponds to a pose of the robot during mapping and every edge between two nodes corresponds to a spatial constraint between them. The essence of graph-based SLAM is building the graph and find a node configuration that minimizes the error introduced by the constraints

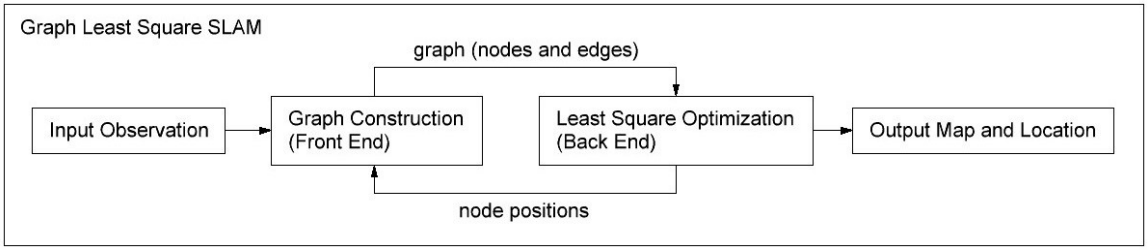


Figure 71: Graph least square SLAM

Updating the covariance in the graph is much cheaper than EKF SLAM, but graph SLAM needs additional inference when recovering the map and the path, whereas EKF maintains its best estimate of the map and robot pose all the time.

As a full SLAM method, graph SLAM calculates posteriors over whole robot paths, while EKF SLAM is an incremental algorithm that only maintains a posterior over the momentary pose of the robot. Hence graph SLAM is best suited for problems of building a map with fixed-size data set. Because graph SLAM has access to the full data in map building, it could apply improved linearization and data association techniques, which makes graph SLAM tends to produce maps that are superior in accuracy to maps build by EKF SLAM. On the other hand, graph SLAM needs extra linear-increased memory requirement for its graph data and separate computational phase for graph building. It also

requires inference when computing the data association probabilities while it could be easily obtained in EKF SLAM.

5.3 Fast SLAM

Fast SLAM uses the particle filter to estimate the robot's current position, which does not suffer from the two types of error that caused by linearization and scene ambiguity in EKF SLAM [28]. Hence it is potentially more accurate than EKF SLAM.

- Sample the particles using the proposal distribution.

$$x_t^{[j]} = \pi(x_t | \cdots)$$

- Compute the importance weights.

$$w_t^{[j]} = \frac{target(x_t^{[j]})}{proposal(x_t^{[j]})}$$

- Resampling: Draw sample i with probability $w_t^{[i]}$ and repeat J times.

Figure 72: Particle filter algorithm

The Particle filter is a non-parametric recursive Bayes filter in which the posterior is represented by a set of weighted samples. It could model arbitrary distributions and works well in the low-dimensional spaces [29].

- Each particle is a pose hypothesis
- Proposal is the motion model

$$x_t^{[j]} \sim p(x_t | x_{t-1}, u_t) \quad \text{Fast SLAM 1.0}$$

$$x_t^{[j]} \sim p(x_t | x_{t-1}, u_t, z_t) \quad \text{Fast SLAM 2.0}$$

- Correction via the observation model

$$w_t^{[j]} = \frac{target}{proposal} \propto p(z_t | x_t, m)$$

$$\approx |2\pi Q|^{-\frac{1}{2}} exp\{-\frac{1}{2}(z_t - \hat{z}^{[j]})^T Q^{-1}(z_t - \hat{z}^{[j]})\}$$

Figure 73: Monte Carlo localization

Fast SLAM generates a set of particles stand for the robot's pose, each particle assumes its pose is completely accurate, then builds the map with an extended Kalman filter. Fast

SLAM 1.0 uses the input control signal to create a proposal area directly and sample one random position as the prediction while Fast SLAM 2.0 uses observed features to optimize the proposal distribution [30]. After having the proposal, the weight factor of this particle is calculated based on the Gaussian probability changes between pre-proposal and post-proposal. Then sample from the particles based on their normalized weight factors. The less accurate particles tend to die out due to the resampling and be replaced by more trusted particles.

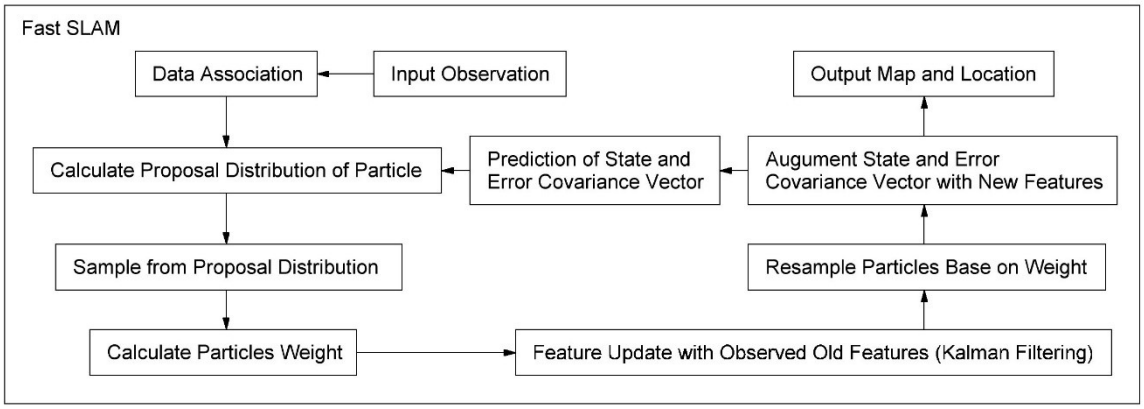


Figure 74: Fast SLAM 2.0

In contrast to the EKF SLAM, Fast SLAM with its particle filter could cope with non-linear robot motion models and does not suffer from linearization problems because it does not linearize the robot pose whereas other techniques approximate such models via linearized functions [31]. Referring to the experimental results from Sasiadek, J.Z., Monjazeb, A., Necsulescu, D., FastSLAM is very suitable for non-linear motions and under non-Gaussian Conditions [28].

Another advantage of the Fast SLAM is due to its weighted particle pose estimation scheme that makes it much more robust to association ambiguity in situations of cluttered scenes. Results from Sasiadek, J.Z., Monjazeb, A., Necsulescu, D. showed that EKF-SLAM fails to map a dense map with nearby landmarks. Robot can travel near the desire

path if Fast SLAM algorithm is implemented instead of EKF SLAM. Having hundreds of

1: **FastSLAM1.0_known_correspondence**$(z_t, c_t, u_t, \mathcal{X}_{t-1})$**:**

2: *for* $k = 1$ *to* N *do* // *loop over all particles*

3: *Let* $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots \right\rangle$ *be particle* k *in* $\mathcal{X}_{t-1}$

4: $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ // *sample pose*

5: $j = c_t$ // *observed feature*

6: *if feature* j *never seen before*

7: $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // *initialize mean*

8: $H = h'(\mu_{j,t}^{[k]}, x_t^{[k]})$ // *calculate Jacobian*

9: $\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ // *initialize covariance*

10: $w^{[k]} = p_0$ // *default importance weight*

11: *else*

11: *else*

12: $\langle \mu_{j,t}^{[k]}, \Sigma_{j,t}^{[k]} \rangle = \textit{EKF-Update}(\ldots)$ // *update landmark*

13: $w^{[k]} = |2\pi Q|^{-\frac{1}{2}}\ \exp\left\{ -\frac{1}{2} (z_t - \hat{z}^{[k]})^T Q^{-1}\ (z_t - \hat{z}^{[k]}) \right\}$

14: *endif*

15: *for all unobserved features* j' *do*

16: $\langle \mu_{j',t}^{[k]}, \Sigma_{j',t}^{[k]} \rangle = \langle \mu_{j',t-1}^{[k]}, \Sigma_{j',t-1}^{[k]} \rangle$ // *leave unchanged*

17: *endfor*

18: *endfor*

19: $\mathcal{X}_t = \text{resample}\left(\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, w^{[k]} \right\rangle_{k=1,\ldots,N} \right)$

20: *return* $\mathcal{X}_t$

Figure 75: Fast SLAM 1.0 algorithm

nearby landmarks in an environment and considering non-Gaussian noises, Fast SLAM algorithm may be a better choice to be implemented [27]. An improved version of Fast SLAM has been proposed by Lu, Y., Polotski, V. and Sasiadek, J.Z. The field experiment

results show that their algorithm is particularly useful for the localization of the outdoor mobile robots, in the environments with pronounced repeatability in geometrical characteristics [31] [30]. For data association, each particle can make independent decisions, hence Fast SLAM can maintain a probability distribution over all possible associations [24]. As more observations arrive, those particles which made poor association decisions in the past tend to be replaced in the resampling process, hence most particles tend to converge to the correct set of associations. For the purposes of data association, Fast SLAM automatically allows information to be integrated between observations at a single time step as JCBB does, and between multiple time steps as sliding-window methods do. The former occurs due to the landmarks being conditionally independent given the robot's trajectory, the latter due to each particle's memory of past associations. Furthermore, Fast SLAM is simple to implement relative to complicated batch association algorithms [23]. The advantage of relative fast run time in Fast SLAM is due to the fully confidence of the robot pose estimation that avoids the inversion of large error covariance matrix of pose distribution (P) in EKF SLAM.

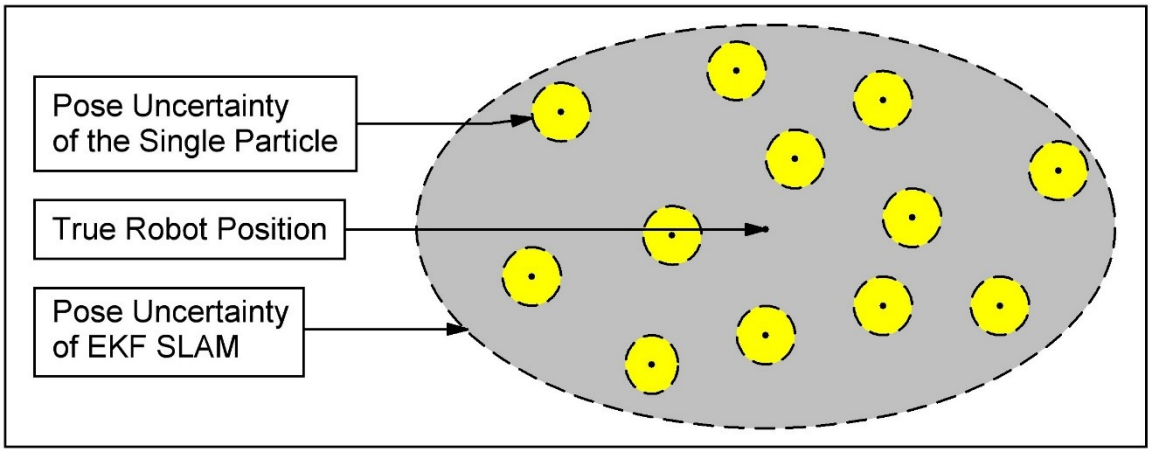


Figure 76: Overconfident issue of Fast SLAM

The disadvantage of Fast SLAM is its inability to maintain particle's diversity over long

periods of time [32]. According to A. Brooks and T. Bailey, the fundamental problem is that the particle filter is really operating in a very high dimensional state space, which is the space of robot's trajectories not the momentary poses. The number of particles needed is therefore exponential in the length of the trajectory. When a smaller number is used, the filter would underestimate the total uncertainties, and eventually become inconsistent. While it may still produce good maps, problems are encountered when the full uncertainty is required, for example when large loops need to be closed [23].

This problem could also be called the overconfident issue in Fast SLAM. As shown in Figure 73, Each particle is assumed to be the true robot position and its accuracy is judged by a Gaussian weight parameter. But the weight parameter itself is in fact not noise-free, hence in most of the situation even the most weighted particle would not have the exactly same coordinate as the true robot position and this difference is the overconfident error. The magnitude of this error depends on the noise of the measurement that is used to calculate to Gaussian probability. Besides it only has one step prediction error covariance of robot pose distribution in the filter optimize and sampling procedure, which would prevent Fast SLAM from fixing the overconfident issue. First problem of the incorrectly assumed small error covariance of pose distribution is that when the true distribution of robot pose gets large enough that the particles pose distribution area cannot cover it anymore, the overconfident error will not be fixed by the next sampling procedure and then be accumulated in the following estimates [33]. Furthermore, the incorrect small pose distribution value of the particles will lead to a larger weight factor for the particle's prediction rather than the measurement, which will also prevent the EKF refinement procedure from capping the overconfident error. Simply increasing the number of particles

has limited improvement effect on this issue, because the resample of particles will make them converge to a same pose assumption and continue being constrained by the incorrectly modeled low pose error variance assumption.

From the experiment of Sasiadek, J.Z., Monjazeb, A., Necsulescu, D., it is evident that the more difference is between observation and motion noises, the earlier the Fast SLAM diverges. Their study suggests that a compromise between observation noise and motion error is needed to overcome such issue. If the motion is noisy, but the sensor is almost noiseless, adding some noise to the sensor may bring the filter back to a desirable performance [26].

5.4 Hybrid SLAM

The hybrid localization and mapping strategy is proposed by Alex Brooks and Tim Bailey that uses of the Fast SLAM as a frontend and uses an EKF SLAM as a backend [34]. The Fast SLAM frontend is used to build local maps and can run for long enough to disambiguate cluttered data associations. Before the trajectory length becomes so long that the overconfident issue becomes problematic, one single Gaussian submap is computed from the Fast SLAM posterior [35]. This Gaussian submap is then fused into the global map. At the point of fusion, a data association must be made about the correspondence between local map features and global map features. However, the large local map could provide enough constraints to make the probability of a bad incorrect data association extremely low [36]. The experiment result shows that the Hybrid SLAM is robust to linearization errors and data association ambiguities on a local scale and can also close large loops on a global scale [23]. Further experiments conducted by Monjazeb et al show that Hybrid SLAM is able to handle hundreds of nearby landmarks with minimum data

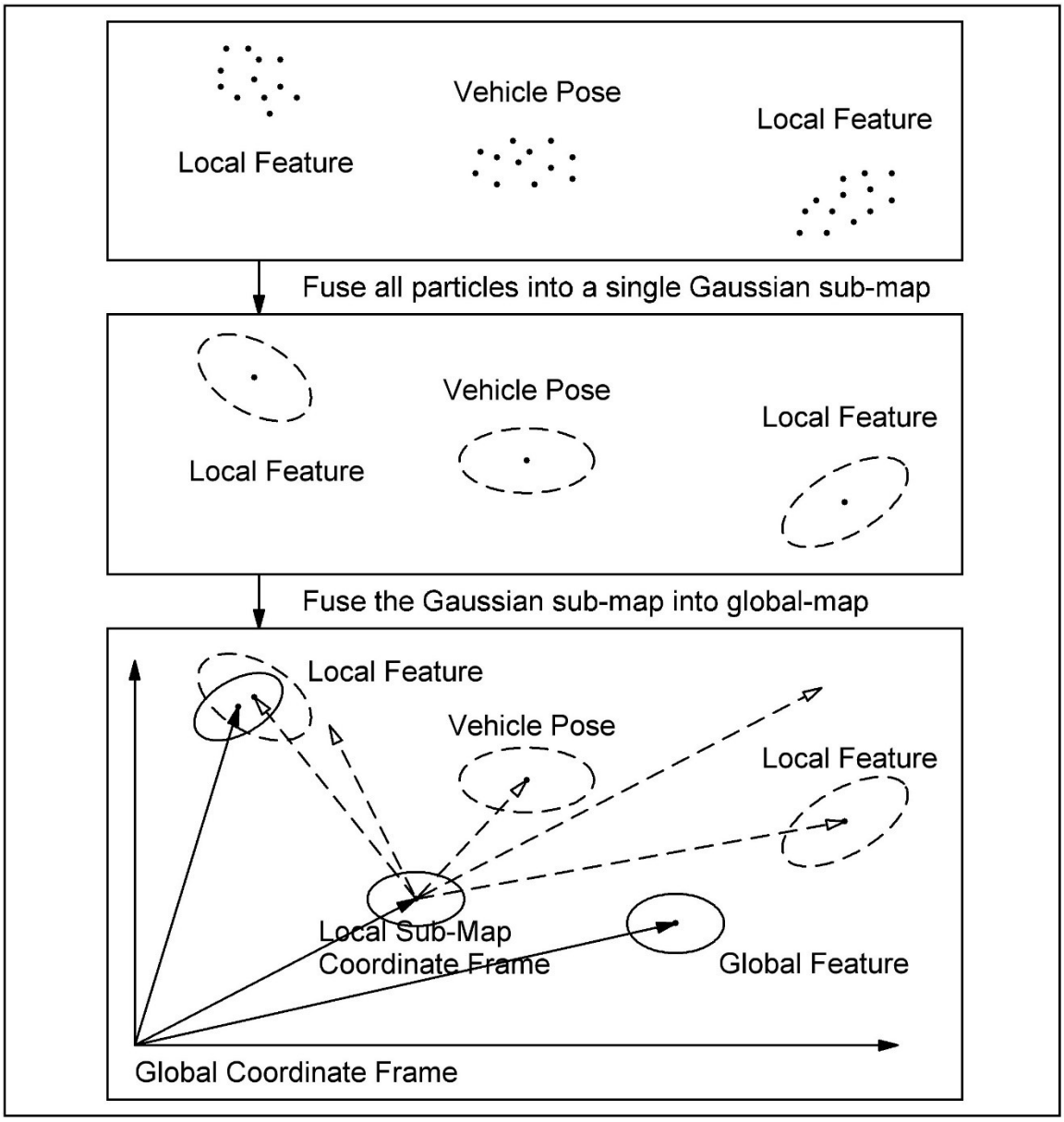


Figure 77: Hybrid SLAM

association ambiguity and a high level of accuracy in the path estimation when compared to the currently used algorithms [34] [33].

As shown in Figure 75, one disadvantage of Hybrid SLAM is that the submap fusion process in Hybrid SLAM includes a linearized state transition of error covariance matrix of robot pose distribution between the submap origin and previous robot pose in global map [37]. In general, linearization is applied in small scale to limit the linearization error.

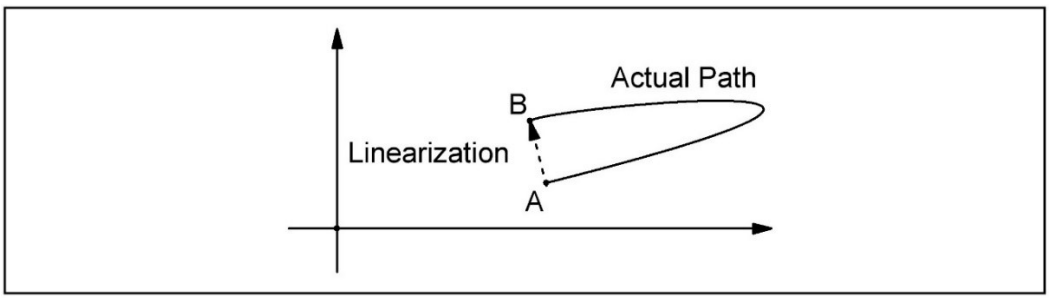


Figure 78: Linearization error

Hence, when the trajectory is long this linearization error could be very large. Sasiadek, J.Z., Monjazeb, A. and Necsulescu, D. use Unscented transform and second order Sterling polynomial linearization to suppress the system linearization error in the Hybrid SLAM. Simulation results show that this improvement strategy provides enough accuracy and stability for Hybrid SLAM in performance for double-loop scenarios in a non-domestic environment [38][37][36][35].

The other issue of Hybrid SLAM is that the overconfident error is reduced but not eliminated via the submap fusion process. Value of this error depends on the fusion frequency in Hybrid SLAM and proportion between the error covariance matrix of robot pose distribution (P) and covariance matrix of state transition noise (Q). To reduce the overconfident error, the fusion period must be as small as possible. On the other hand, for the robust estimation, the Fast SLAM frontend needs to be running for enough time to make the ambiguities die out. Sometimes the non-Gaussian distribution of robot pose would last for a long time, for example when entering an unknown area and it could exist until the loop closure happened [38]. In this situation, the process of submap fusing into a single Gaussian would eliminate the advantage of Fast SLAM front end that robust against to the ambiguous data association and of course would introduce noticeable errors caused by scene ambiguity just like the EKF SLAM.

5.5 Fast SLAM 3.0

Overconfident issue is the main problem of the Fast SLAM. The true particle position assumption is a low dimension model that works well in particle filter but will fail the EKF update part because EKF works in a high dimensional space which is the whole trajectory.

$$P_t = (1 - K_t)H\hat{P}_t$$

The error residual with a magnitude of P is from EKF update and ignored by the particle filter since it works in a much lower dimension. The importance weight is calculated from target and proposal distribution based on the assumption of accurate previous particle position and proposal distribution. But the proposal distribution is optimized by EKF which cannot remove the noise completely. This noise residual will accumulate in the particles and affect the importance weight increasingly. Which means the particle with highest Gaussian probability is not the true robot position. The overconfident robot pose distribution matrix (P) will limit the sampling range of the particles and in the pose refinement and feature update process incorrectly give more weight to the prediction of robot pose rather than the sensor measurement. Instead of using an additional EKF filter backend to fuse the submap from Fast SLAM frontend when the overconfident error is small like what Hybrid SLAM does. We propose some alternative solutions to solve the overconfident issue that the core thought is to increase the uncertainty of particles' pose distribution, which could make the sampling procedure cover the actual distribution area of robot pose and eliminate the overconfident error in the filter optimization steps.

A. Increasing the state transition noise (Q)

The simplest strategy to increase the robot pose uncertainty matrix (P) in Fast SLAM is to increase the covariance matrix of state transition noise (Q). Instead of using a constant

Q in the original Fast SLAM algorithm, we propose that the Q matrix could be increased a little in every filter iteration to compensate the overconfident error. The difficulty of this method is to find a good strategy of calculating the Q propagation value for each specific operation.

B. Keeping the Robot Pose Uncertainty (P)

Since the core problem is the low dimensional modeling issue in Fast SLAM 2.0**.** ***The fundamental solution is to come up with a higher dimensional model.*** Hence The second way of solving the overconfident issue is that by modeling each particle in Fast SLAM 3.0 stands for the local mean of a Gaussian robot pose distribution instead of true robot pose. Then since the proposal in Fast SLAM 2.0 is optimized by the EKF, the best method to account for the noise residual in the Gaussian probability weight parameter is to keep the covariance matrix of robot pose distribution (P) after sampling rather than setting it to zero.

We propose a modified Fast SLAM 2.0 method that keeps P matrix of each particle propagating as normal EKF SLAM instead of setting it to zero after the sampling, which means the particles work as simple extended Kalman filter. There are three changes have been made in the Fast SLAM 3.0 that based on this modeling assumption:

1. Keeping the error covariance matrix of the robot pose distribution (P) propagating as normal EKF SLAM and do not set it to zero after sampling, by which to increase the sampling area and get more accurate pose proposal estimate.

2. Do not apply resampling if there is no old features in the filter circle to refine the robot pose distribution, because the particle in Fast SLAM 3.0 stands for the mean of a Gaussian robot pose distribution and when there is no old features to refine the proposal, the best prediction is to use the mean value rather than resampling.

3. When adding the new feature. The error covariance matrix of robot pose distribution need to be included in the feature's error covariance matrix of its position distribution since the particle does not stand for the perfect robot pose anymore.

1: **FastSLAM 3.0_known_correspondence**$(z_t, c_t, u_t, \mathcal{X}_{t-1})$**:**

2: *for* $k = 1$ *to* N *do* // *loop over all particles*

3: *Let* $\left\langle x_{t-1}^{[k]}, \Sigma_{x,t-1}^{[k]} \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots \right\rangle$ *be particle* k *in* $\mathcal{X}_{t-1}$

uncertainty of the particle k

4: $x_t^{[k]} \sim p(x_t, \Sigma_{x,t}^{[k]} \mid x_{t-1}^{[k]}, \Sigma_{x,t-1}^{[k]}, u_t, z_t)$ // *sample pose*

Do not sample if no old feature observed

5: $j = c_t$ // *observed feature*

6: *if feature* j *never seen before*

7: $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ // *initialize mean*

8: $H = h'(\mu_{j,t}^{[k]}, x_t^{[k]})$ // *calculate Jacobian*

9: $\Sigma_{j,t}^{[k]} = H^{-1} Q_t (H^{-1})^T + A^{-1} \Sigma_{x,t}^{[k]} (A^{-1})^T$ // *initialize covariance*

10: $w^{[k]} = p_0$ // *default importance weight*

11: *else*

include uncertainty of the particle k, where A is Jacobian of the particle state transition matrix

11: *else*

12: $\langle \mu_{j,t}^{[k]}, \Sigma_{j,t}^{[k]} \rangle = EKF\text{-}Update(\ldots)$ // *update landmark*

13: $w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} (z_t - \hat{z}^{[k]})^T Q^{-1} (z_t - \hat{z}^{[k]}) \right\}$

Step 11~20 is the same as Fast SLAM 2.0

14: *endif*

15: *for all unobserved features* j' *do*

16: $\langle \mu_{j',t}^{[k]}, \Sigma_{j',t}^{[k]} \rangle = \langle \mu_{j',t-1}^{[k]}, \Sigma_{j',t-1}^{[k]} \rangle$ // *leave unchanged*

17: *endfor*

18: *endfor*

19: $\mathcal{X}_t = \text{resample}\left(\left\langle x_t^{[k]}, \Sigma_{x,t}^{[k]} \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, w^{[k]} \right\rangle_{k=1,\ldots,N} \right)$

20: *return* $\mathcal{X}_t$

Figure 79: Fast SLAM 3.0 algorithm

As an example of robot pose distribution in a long trajectory operation shown in Fig. 76, The dash line is the error covariance of robot pose distribution in each filter. Solid line is the actual distribution of the robot pose. Compare to the actual robot pose distribution, the Fast SLAM 2.0 has overconfident robot pose distribution which only covers a small part

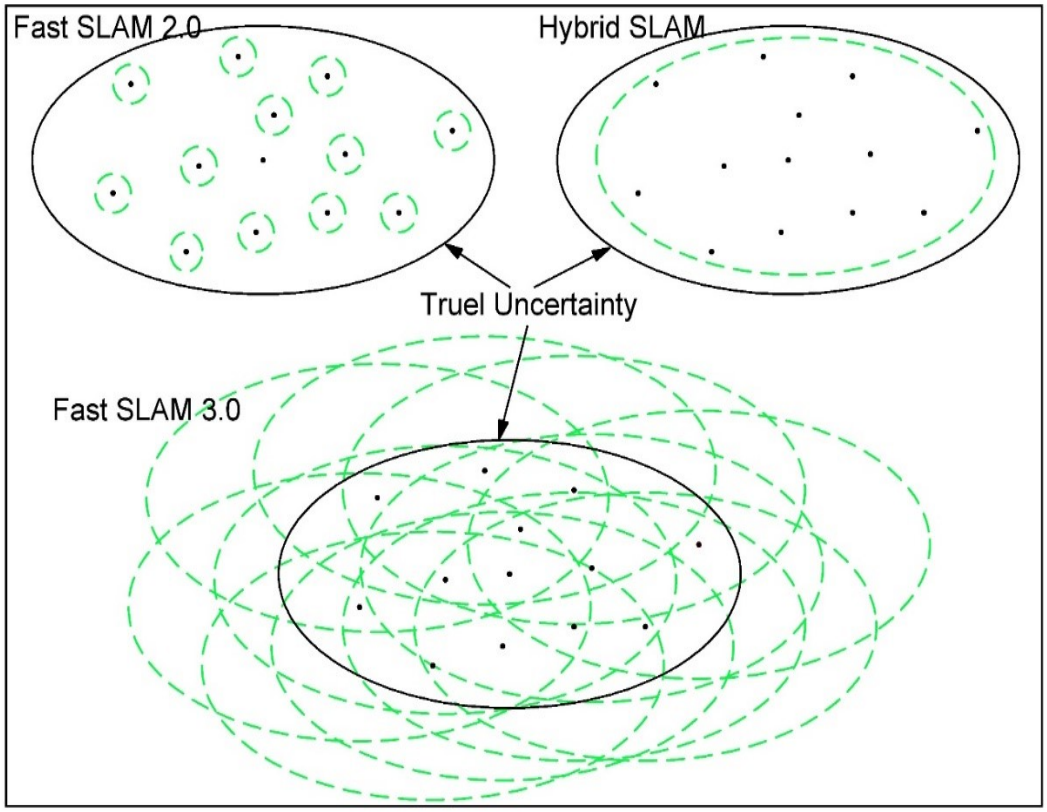


Figure 80: Error covariance P of different filters

of the actual pose distribution region while the Hybrid SLAM has the similar pose uncertainty region to the actual one. The Fast SLAM 3.0 has covered all of the robot pose uncertainty region for following the sampling process and each particle has the correct error covariance (P) value for pose proposal refinement, which is the similar situation as Fast SLAM 2.0 in its initial few steps when its overconfident error is small.

5.6 Simulation Environment and Parameters

A. Single fixed trajectories with random landmarks

Experiments compared the following filters in simulation:

1. Extended Kalman Filter SLAM [16] (EKF SLAM),
2. Fast SLAM v2.0 [16] (Fast SLAM),
3. Fast SLAM v2.0 with increased state transition noise Q
4. Fast SLAM v3.0

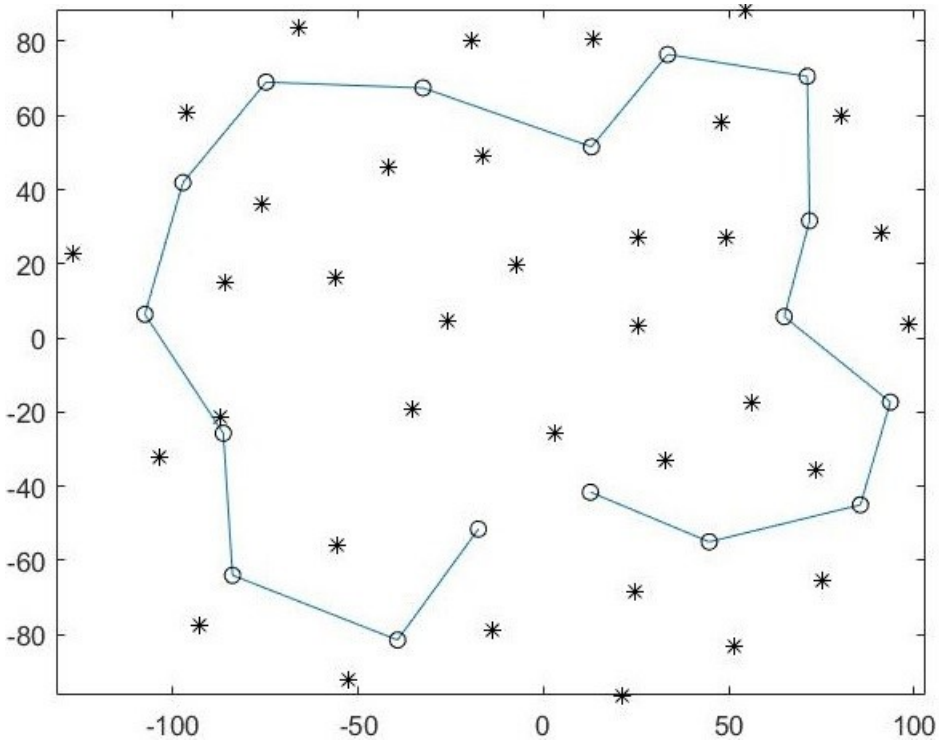


Figure 81: Position of landmarks and waypoints for the simulation A

This simulation focused on the filters' ability to both manage clutter and accurate pose estimation. This was done by having the filters operate over the long trajectory of approximately 640 m in one simulated environment, with a perfect sensor subject to no false alarms and non-detections. One single open-loop trajectory is used for the simulation and the perfect data association is assumed known. Because loop closure based on the accurate data association and 2D landmarks matching algorithm based on the accurate estimation of mean and distribution of landmarks. From the simulation of A. Brooks [23]

it could be seen that EKF has much higher possibility of closing large loops. The particles in Fast SLAM 3.0 has the similar distribution as EKF SLAM, so if we could prove Fast SLAM 3.0 has more accurate position mean value estimation, it is reasonable to assume Fast SLAM 3.0 has similar or even higher chances to close large coops in 2D environment.

The simulation environments consisted of one randomly generated simulated two-dimensional worlds of size $200\,m \times 200\,m$, which contains 35 uniformly distributed point features. The robot moved at $1\,m/s$ and up to $100°/s$, observing features using a $360°$ range-bearing sensor with a maximum range of $30\,m$. Four SLAM methods are tested in the same environment, which are Extended Kalman Filter SLAM, Fast SLAM 2.0, Fast SLAM 2.0 with increased Q and Fast SLAM 3.0. The prediction of state transition noise increased as $Q_{t+1} = Q_t + 0.00025Q_t$ in every prediction step of Fast SLAM 2.0 with increased Q.

Five test trials were executed for each type of SLAM. The absolute difference between ground truth and filter estimate is recorded for comparison. The weighted average of all particles' estimate of the Fast SLAM is used as its final estimate in the comparison. SLAM filters were running at the observation frequency of $5\,Hz$. Robot linear and rotational velocity were noisy, perturbed with standard deviations equal to $0.25\dot{x}$ and $0.35\dot{\theta}$ respectively, where $\dot{x}$ and $\dot{\theta}$ are commanded linear and rotational velocities. Observation noise standard deviations were set to $0.05\,m$ and $1°$ in range and bearing respectively. Identical worlds, odometry, and observations were used for all filters' simulation.

B. Multiple fixed trajectories with random landmarks

Experiments compared the following filters in simulation:

1. Extended Kalman Filter SLAM [16] (EKF SLAM),

2. Fast SLAM v2.0 [16] (Fast SLAM),
3. Fast SLAM v3.0

This simulation focused also on the filters' ability to both manage more random cluttered environment and accurate pose estimation. The simulation environments consisted of the same trajectory in 10 different randomly generated simulated two-dimensional worlds of size 200 m×200 m, which contains 56 uniformly distributed random point landmarks.

Most of the system parameters are the same as previous simulation except the landmarks are randomly regenerated in each trial and there are 100 particles in the Fast SLAM v2.0 and v3.0. Single test trial was executed for each type of SLAM in each map.

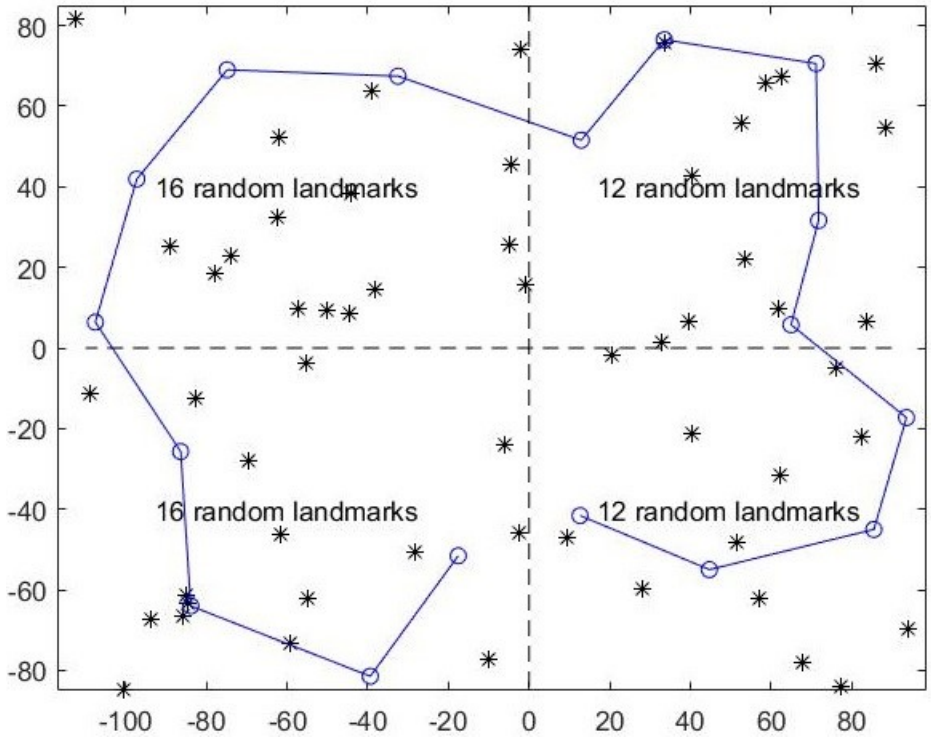


Figure 82: Position of landmarks and waypoints for the simulation B

The absolute difference between ground truth and filter estimate is recorded for comparison. The weighted average of all particles' estimate of the Fast SLAM is used as its final estimate for the comparison. Identical worlds, odometry, and observations were used for

all filters' simulation.

5.7 Results and Discussion

Figure 83 shows the results of EKF simulation repeated 5 times in environment setting A. It could be seen that the absolute difference between estimate and ground truth of EKF SLAM has high variance that the error increasing patterns in each trial is quite different from each other. There are two types of error that lead to this phenomenon, both of which are caused by the Gaussian distribution assumption of the EKF SLAM.

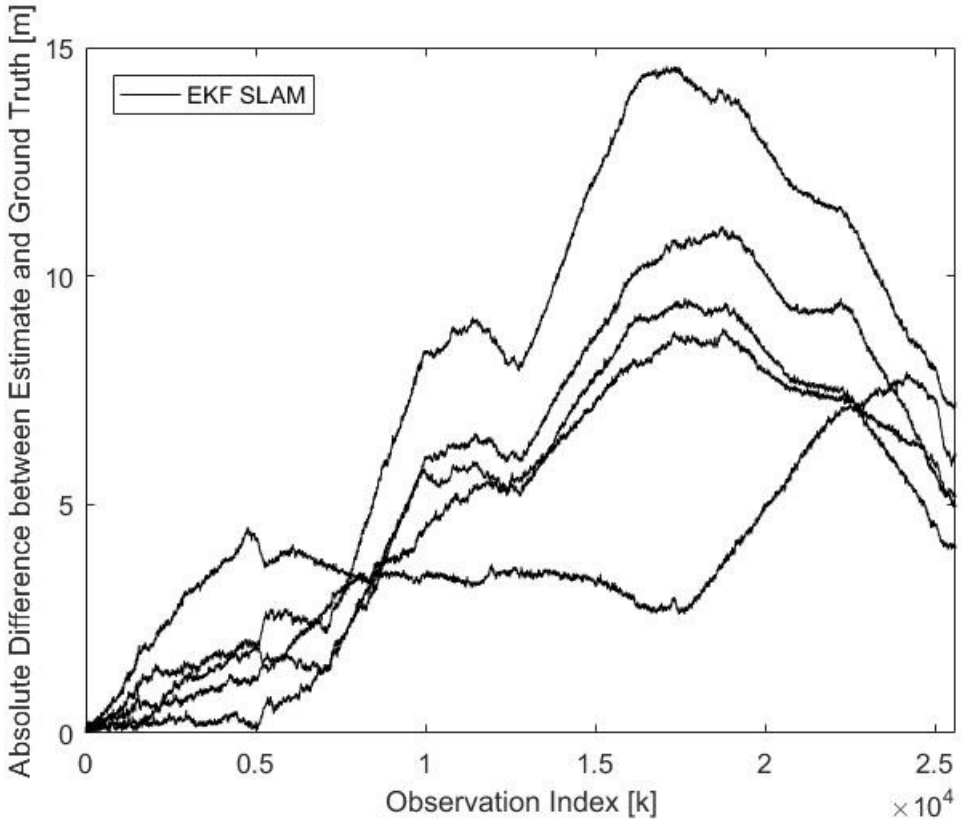


Figure 83: Simulation results of EKF SLAM

The first one is due to the scene ambiguous, which will generate multiple local maximum belief in the robot pose distribution as shown in Figure 70. The EKF SLAM uses the value from both right and wrong data association for its estimation which would introduce large errors. The second problem is incorrect mean value of robot pose distribution due to

linearization in EKF SLAM as talked in section two. When the measurement noise is Gaussian and the number of landmarks is very large, the linearization error is limited and the mean value could be very close to the true robot pose. But when the number of observation is relatively small in unit time as shown in Figure 83, the mean value of estimate could be close to the true robot pose as well but also could be far away from the true robot pose due to the linearization error, especially when the SLAM filter operates for the highly nonlinear robot and observation model over the long trajectory and covariance matrix of state transition noise (Q) or measurement noise (R) is large. Because the two types of error are both pure random so the error increasing patterns in each EKF SLAM trials could be very different from each other.

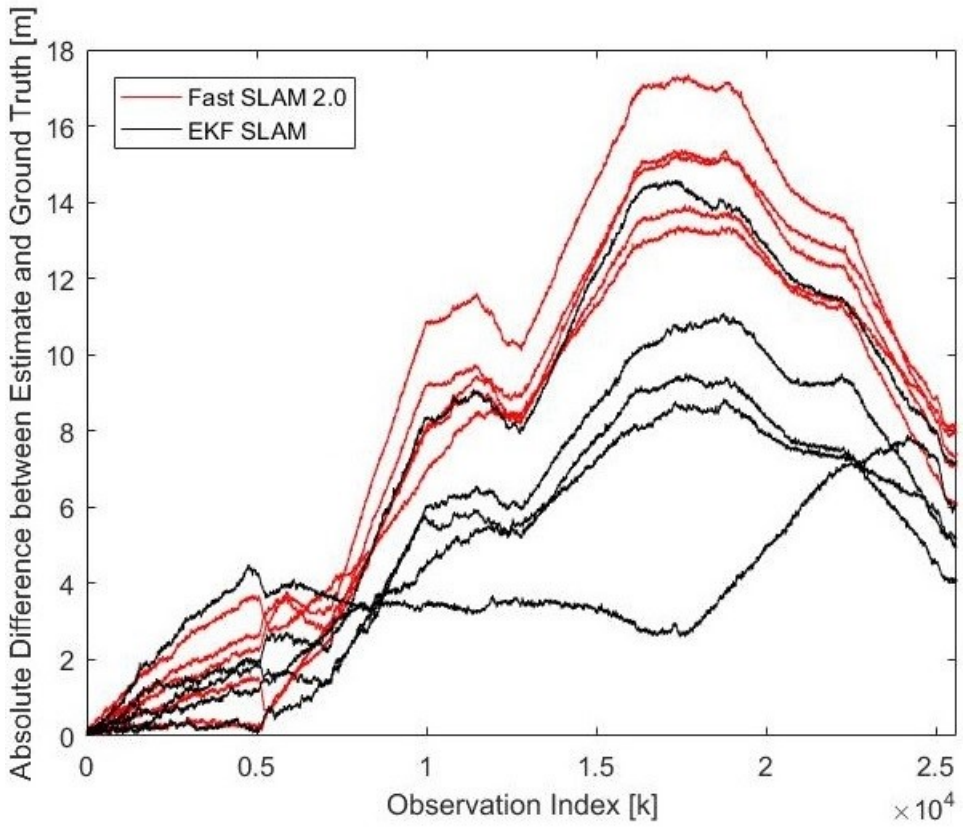


Figure 84: Simulation results of EKF SLAM and Fast SLAM 2.0

Fast SLAM is robust to scene ambiguity and its particle filter is not constrained by

Gaussian assumption so when the number of particles is enough, Fast SLAM is supposed to have a better performance compared to the EKF SLAM. While it could be seen in the Figure 84 that the 5 identical Fast SLAM 2.0 (red line) simulations has generally worse performance than the 5 identical EKF SLAM (black line) simulations, which probably because the overconfident error in Fast SLAM is trajectory based while the Gaussian related errors in EKF SLAM are pure random. Hence, for long trajectory operation, EKF SLAM could be better than Fast SLAM 2.0. From the observation index 0 to 0.7×10^4, the Fast SLAM 2.0 and EKF SLAM has the similar performance, which is due to robot pose uncertainty of EKF SLAM and the combined particles' pose uncertainty of Fast SLAM covered the similar robot pose distribution area. When the observation index exceeds value 0.7×10^4, the error's increasing rate of all Fast SLAM trials starts to grow uniformly which is because the particles' uncertainty value of distribution cannot cover the actual robot's pose uncertainty area anymore and hence the overconfident error started to grow. Fast SLAM 2.0 (red line) has similar error variance pattern between each trials, which is because the 5 trials of Fast SLAM 2.0 used the same path and the main error of Fast SLAM 2.0 in this simulation environment is caused by overconfidence issue which depends on the path that robot traveled. The error grew between the observation index value $0.7 \times 10^4 \sim 1.7 \times 10^4$ and declines after 1.7×10^4, which is because the robot moved on the opposite directions in the two observation spans.

As shown in Figure 85 that by using an increased covariance matrix of the state transition noise (Q) in the prediction step of each filtering iteration to compensate the overconfident error covariance of the robot position distribution (P), the max position estimate error is significantly reduced about 40% in the 5 identical trials of the Fast SLAM 2.0 with the

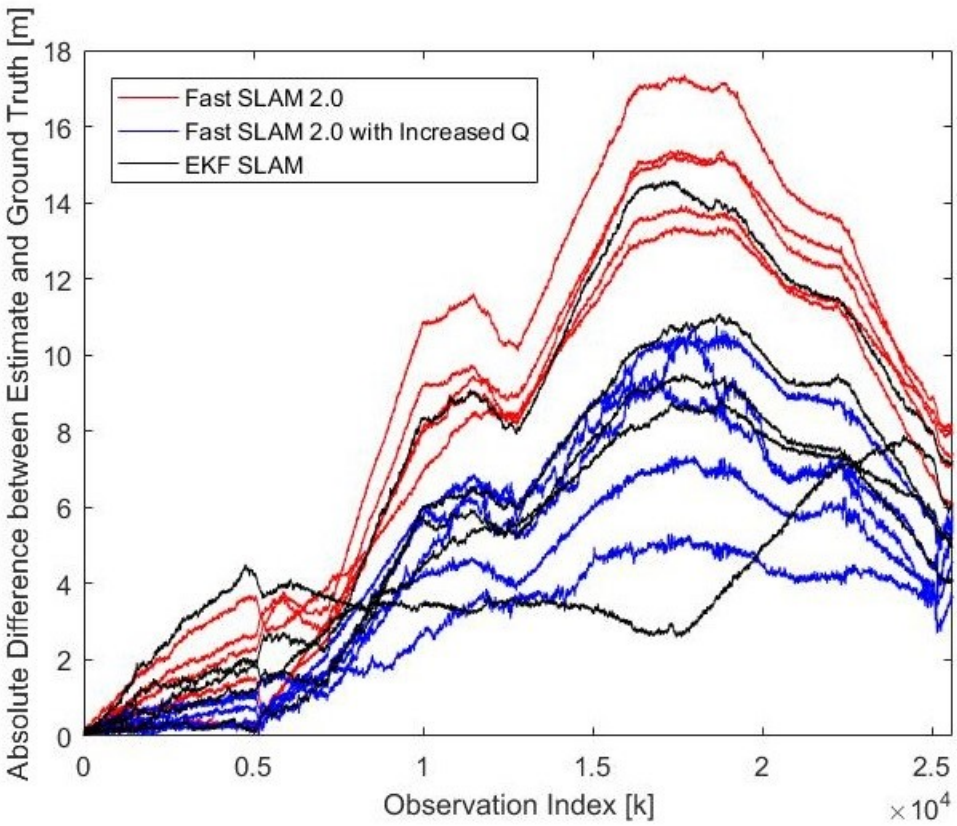


Figure 85: Results of EKF SLAM, Fast SLAM 2.0 and Fast SLAM 2.0 with increased Q

increased Q (blue line) compared to the original 5 Fast SLAM 2.0 (red line) simulations. However, it could also be seen that the error changes in both filters' simulation still have the similar patterns, which indicates that the overconfident error is decreased but still not being eliminated in the Fast SLAM 2.0 with increased Q. The reason is that the increased Q only affects the P matrix that used in the particle proposal optimization not the P matrix in landmarks update. Because the P matrix was set to zero after the sampling procedure.

On the other hand, it could be seen in Figure 86 that the 5 identical Fast SLAM 3.0 (green line) simulations has smallest position estimate error than the other SLAM methods, which reduced about 80% compared to Fast SLAM 2.0. By keeping the correct error covariance of robot pose distribution (P) and feature position distribution (P) in every iteration rather than setting robot pose uncertainty to zero in the original Fast SLAM, every

particle works as a fully functional extended Kalman filter.

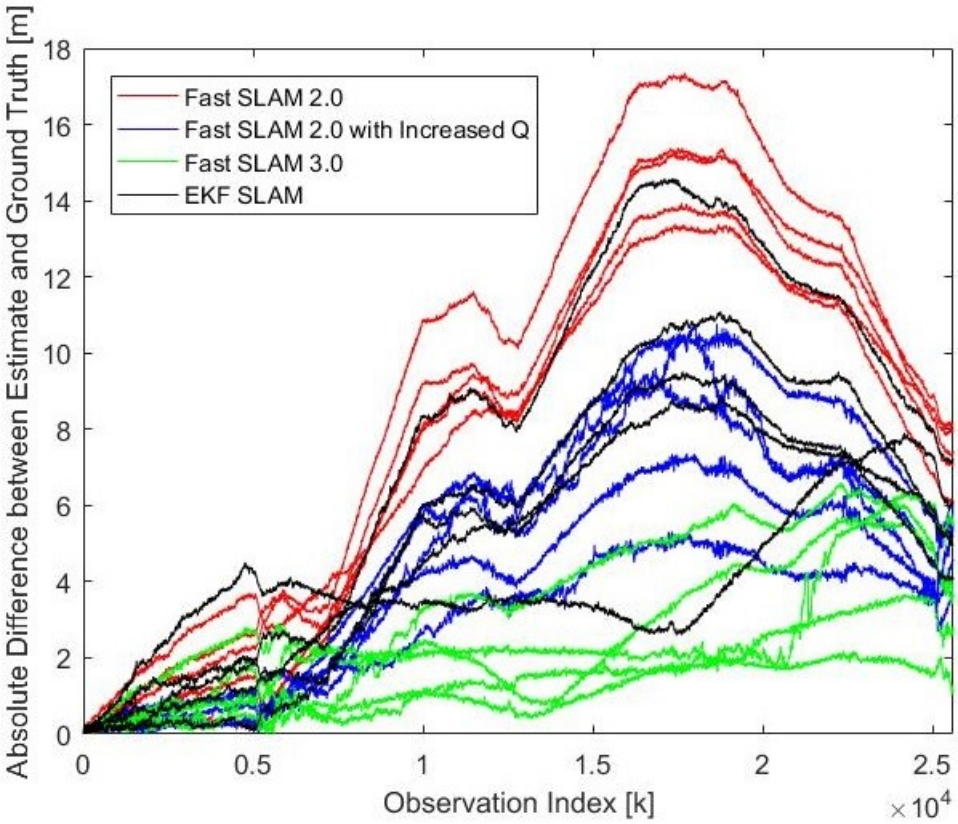


Figure 86: EKF SLAM, Fast SLAM 2.0, Fast SLAM 2.0 with increased Q, Fast SLAM 3.0

Hence the overconfident issue is completely solved theoretically. The simulation results also show that the error increasing pattern in each trial of the 5 Fast SLAM 3.0 simulations is quite different from those of Fast SLAM 2.0 and SLAM 2.0 with increased Q, which could be explained as the overconfident error is completely eliminated. Besides all the estimation error in Fast SLAM 3.0 from different simulation trials are much smaller than those of EKF SLAM due to the non-parametric estimation of particle filter. Its weighted sample proposal process that the accurate particles acquire more weight than the others is much more accurate than simply using the mean value from the linearized system in EKF SLAM.

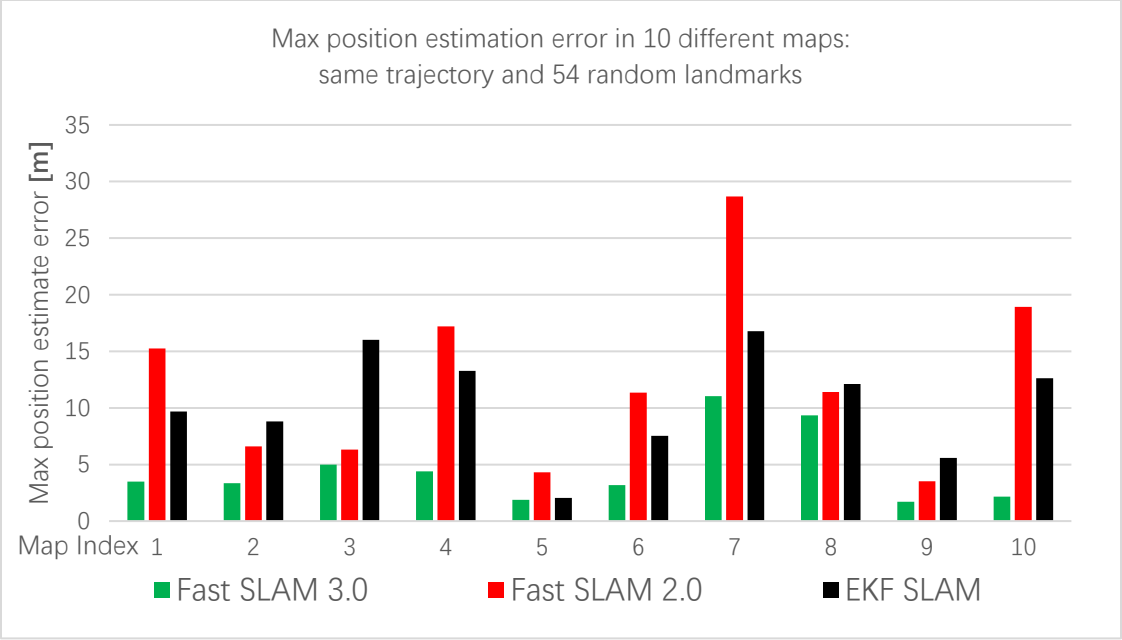


Figure 87: Max position estimation error of each SLAM in simulation B

It could be seen in Figure 87 that in simulation B, Fast SLAM 3.0 (green) has a general better performance than EKF SLAM and Fast SLAM 2.0 in all 10 different maps, which is because Fast SLAM 3.0 does not have the over confident issue in Fast SLAM 2.0 and also robust against error in the EKF SLAM that caused by linearization and scene ambiguity.

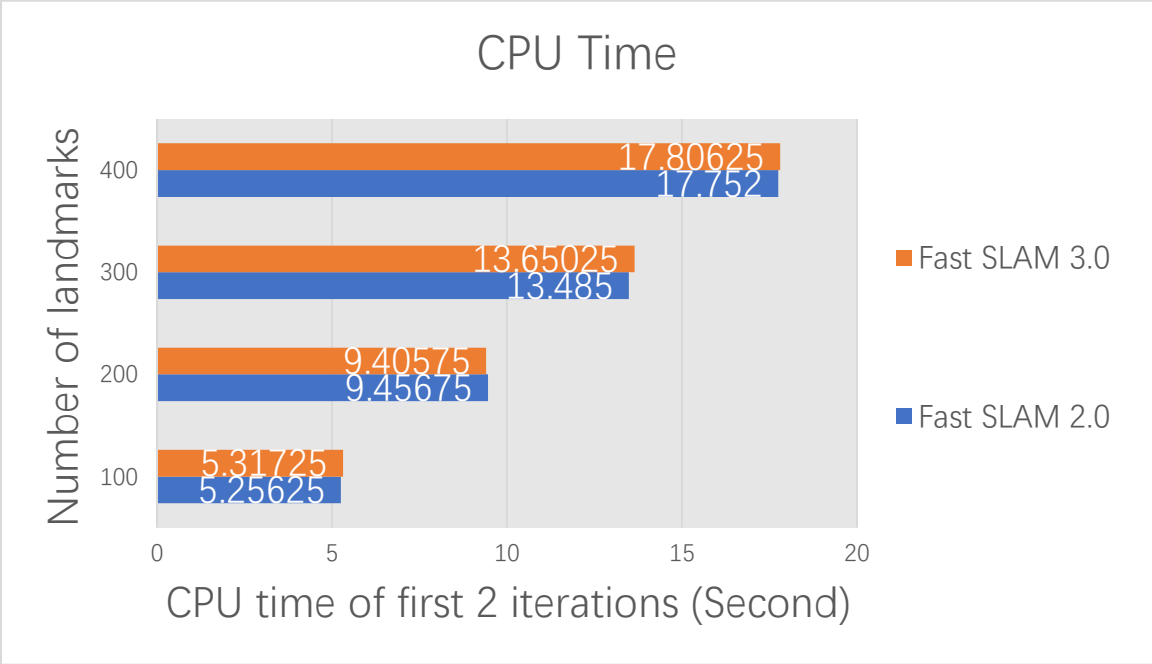


Figure 88: CPU time comparison between Fast SLAM 2.0 and 3.0

The Fast SLAM 3.0 algorithm is very likely to Fast SLAM 2.0. The core difference that may increase computational load is that in Fast SLAM 3.0 the proposal optimization and landmark update need consider the uncertainty of the particle. But since it is just a 2 by 2 matrix and only multiplication and addition are used, the Fast SLAM 3.0 only has a very small constant computational requirements increasing compared to the Fast SLAM 2.0 as shown in Figure 88 which makes it available for real time application.

All the previous MATLAB simulations based on the assumption of accurate system model which includes the accurate A, H, Q, R. Therefor a special case exists that the system model is underconfident about the robot and the filter algorithm is overconfident about the robot. The two errors will neutralize each other and make Fast SLAM 2.0 more accurate that Fast SLAM 3.0 as shown in Figure 89.

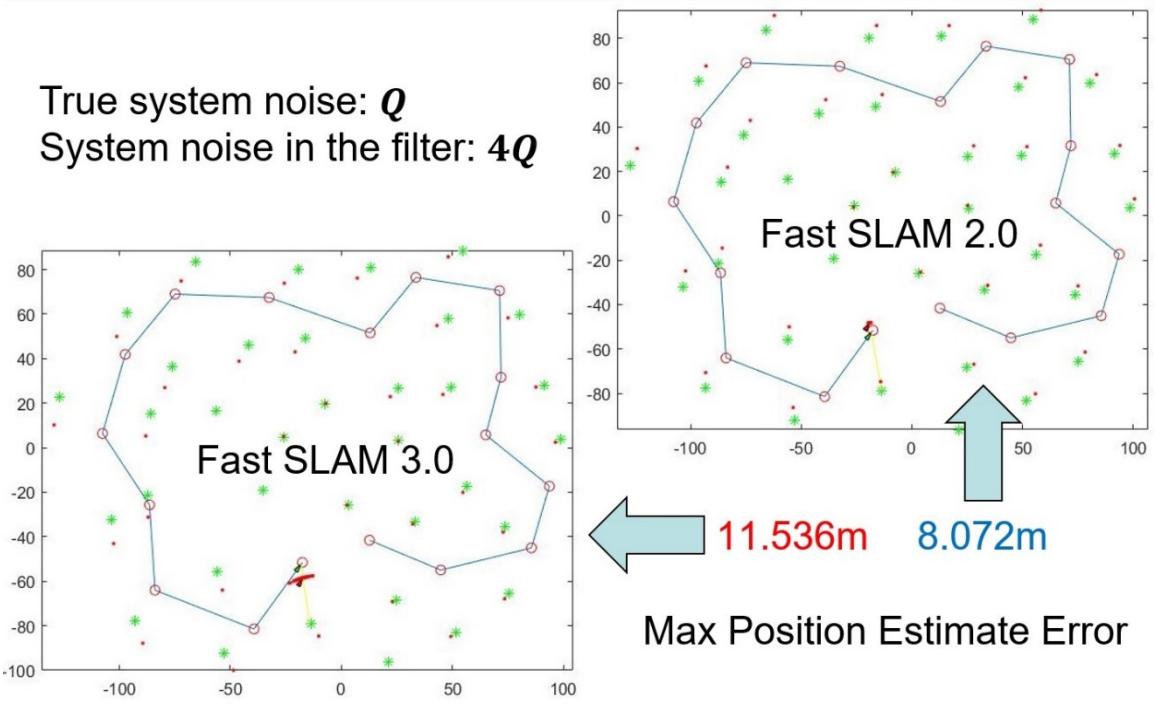


Figure 89: SLAM accuracy changes due to wrong modeling

Chapter 6: Conclusion

The true particle position assumption is a low dimension model that works well in particle filter. But this model is inappropriate for the EKF update part since EKF works in a high dimensional space which is the whole trajectory. While Fast SLAM 3.0 model the particle as EKF mean which is a high dimensional model that works well in both particle filter and EKF update. Because the particle filter will ignore the extra dimension of the model.

The Fast SLAM 3.0 models the particle as the robot pose mean of a Gaussian distribution and keeps the corresponding pose distribution matrix (P) of robot and features. This model allows the Fast SLAM 3.0 keeps both the advantages of EKF SLAM and Fast SLAM 2.0 without suffering from their most pronounced disadvantages. The Fast SLAM 3.0 does not have the overconfident issue of Fast SLAM 2.0 due to the lack of long-term pose uncertainty memory while keeps the robustness against scene ambiguity of Fast SLAM. On the other hand, The Fast SLAM 3.0 has the similar long-term pose uncertainty memory as the EKF SLAM while the robot pose linearization error is not exist in Fast SLAM 3.0 due to its non-parametric robot pose estimation.

The robot poses estimation of Fast SLAM 3.0 is compared to currently used algorithms, in particular, resulting in high quality of the produced map and reducing error in estimated position and robot heading. Results from the two simulations indicate that the proposed SLAM algorithm is able to handle a lot of nearby landmarks with minimum data association ambiguity and a high level of accuracy in the path estimation when compared to currently used algorithms.

It has also been found in the simulations that the accurate refinement of robot pose distribution proposal is important for both Fast SLAM v2.0 and v3.0. A good refinement

of the robot pose distribution proposal could cap the increasing of robot pose uncertainty, which could also relieve the overconfident issue in Fast SLAM 2.0. Since the sampling range in Fast SLAM 3.0 is getting larger and larger in each of its filtering iteration. The accurate refinement of proposal distribution from sensor measurement could limit increasing of sampling area, which could lead to a much more accurate posterior pose estimate and avoid potential filter divergence.

As an improvement of Fast SLAM 2.0, the Fast SLAM 3.0 could also be used in the Hybrid SLAM especially the for long distance operation or 3-D applications in the real world. The submap fusion scheme in Hybrid SLAM enable the system to accommodate the required computational ability and provide additional robustness against outliers.

Since Fast SLAM 2.0 and 3.0 have same EKF update structure, it can be expected that they have similar robustness against colored system noise. But because Fast SLAM 3.0 usually has larger P which gives more weight to the measurement in the EKF update, Fast SLAM 3.0 will have less degradation in the existence of colored system noise compared to Fast SLAM 2.0.

Particles' position distribution of Fast SLAM 2.0 shows very similar coordinate in the 5 trials as shown in the appendix because of the overconfident system model, while this feature does not exist in Fast SLAM 3.0 since the overconfident error has been completely removed.

Appendices

Appendix A Fast SLAM 2.0 trials 1-5

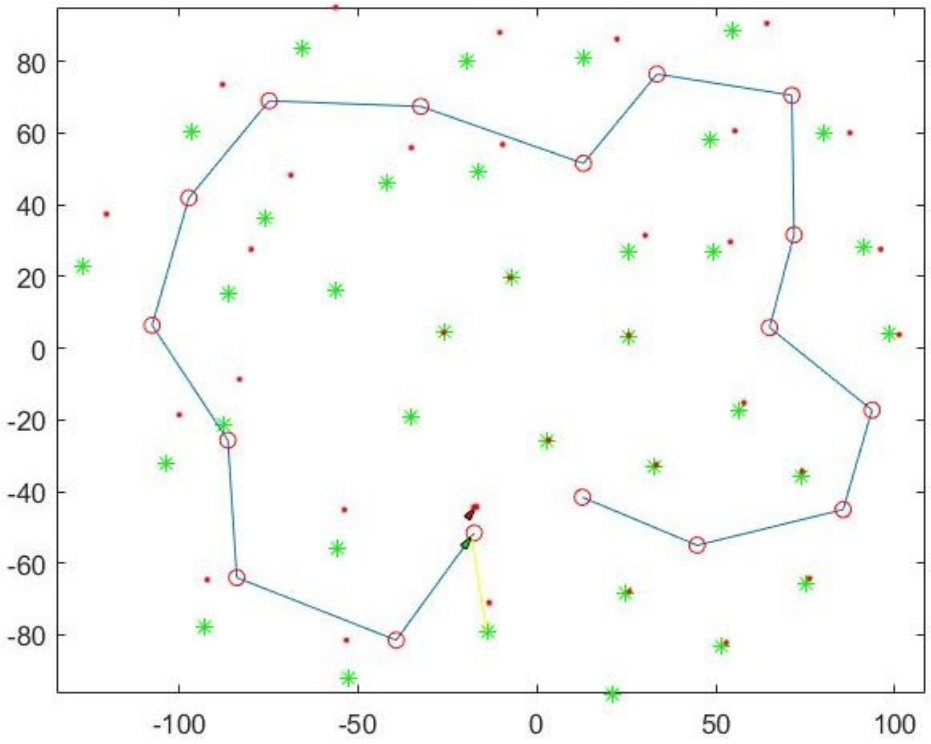


Figure 1-1: Fast SLAM 2.0 simulation trial 1

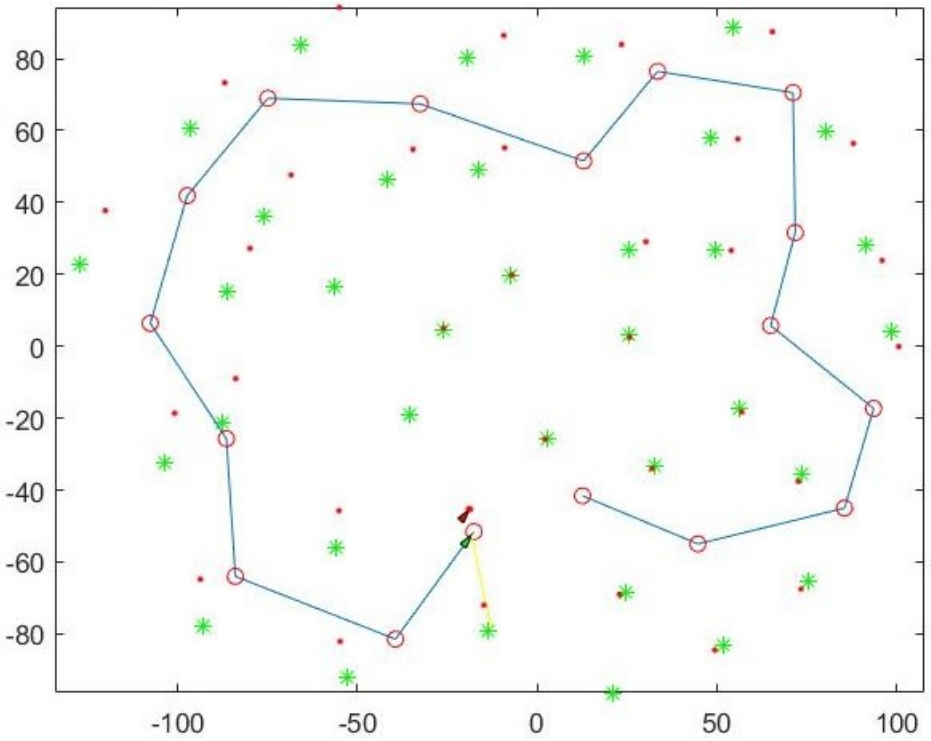


Figure 1-2: Fast SLAM 2.0 simulation trial 2

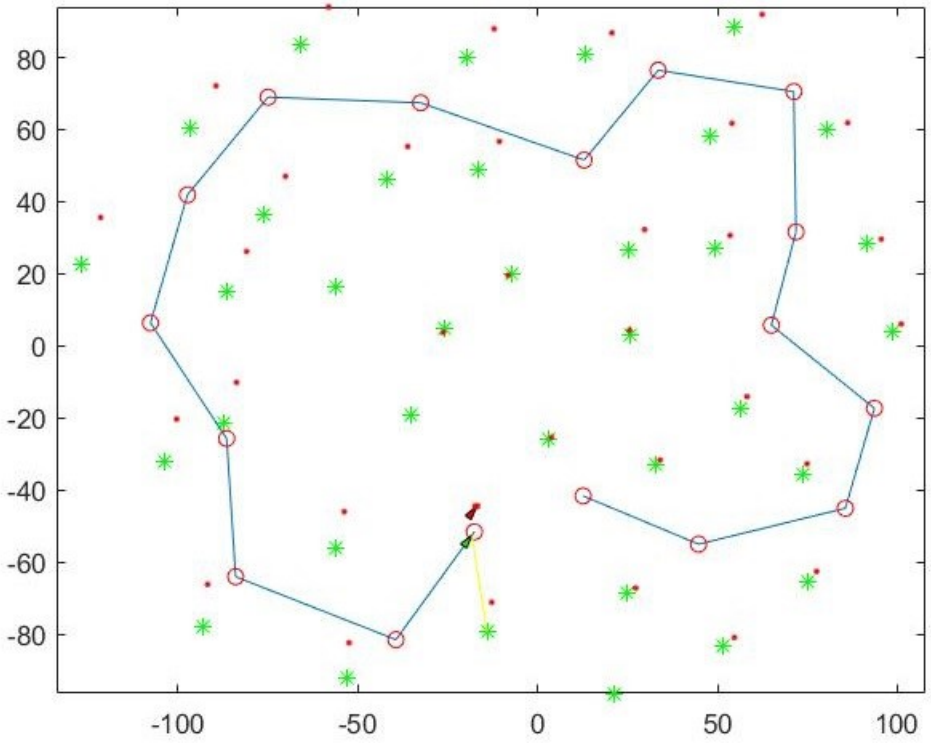


Figure 1-3: Fast SLAM 2.0 simulation trial 3

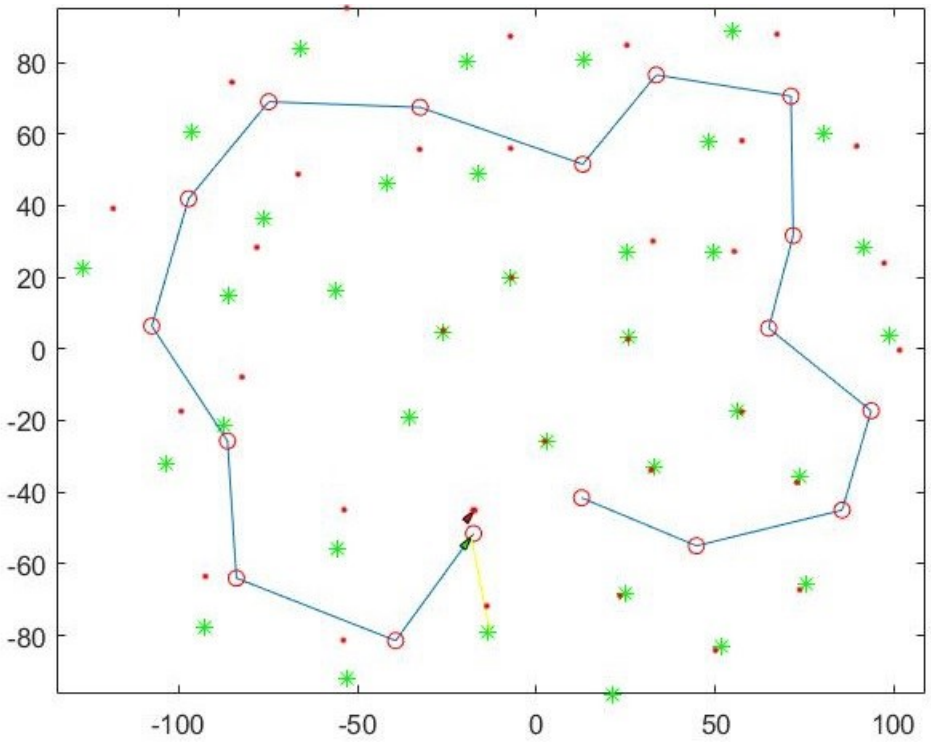


Figure 1-4: Fast SLAM 2.0 simulation trial 4

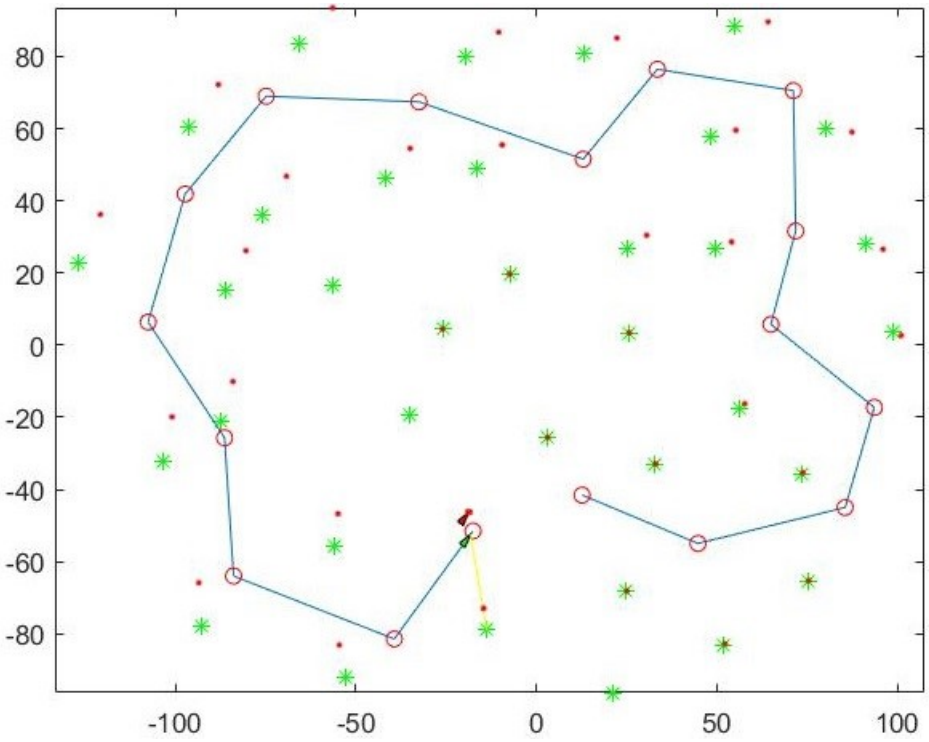


Figure 1-5: Fast SLAM 2.0 simulation trial 5

Appendix B Fast SLAM 2.0 with increased Q trials 1-5

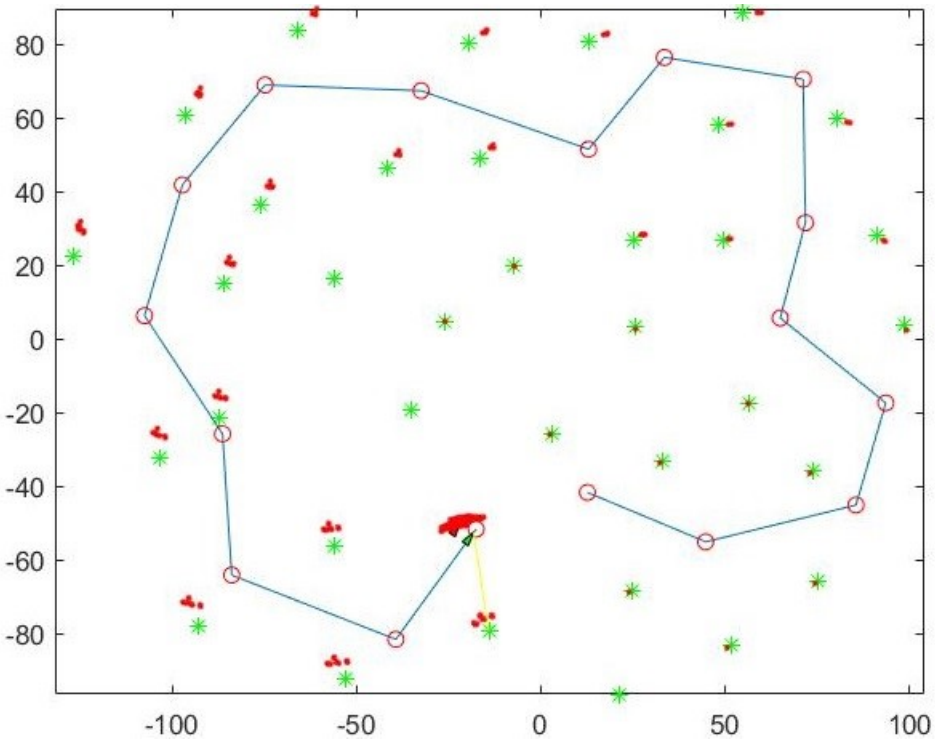


Figure 2-1: Fast SLAM 2.0 with increased Q simulation trial 1

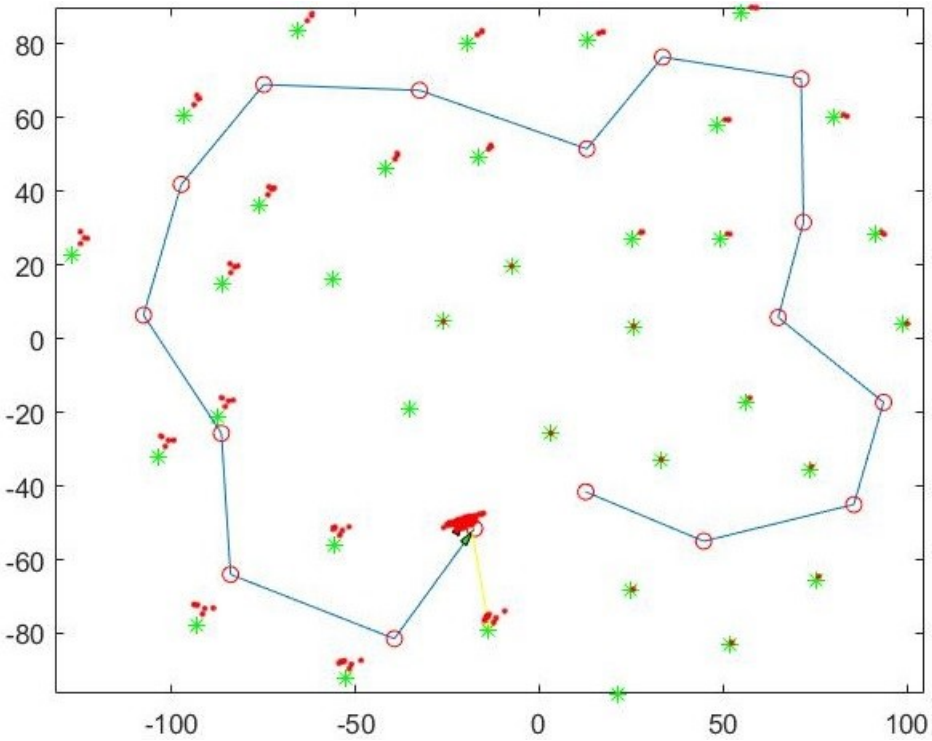


Figure 2-2: Fast SLAM 2.0 with increased Q simulation trial 2

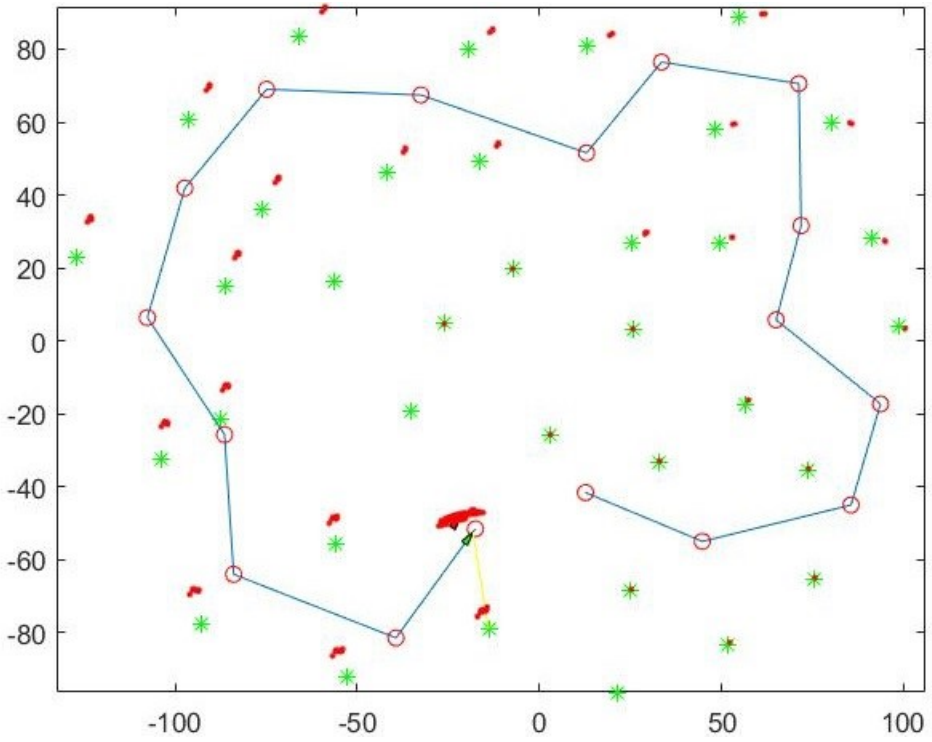


Figure 2-3: Fast SLAM 2.0 with increased Q simulation trial 3

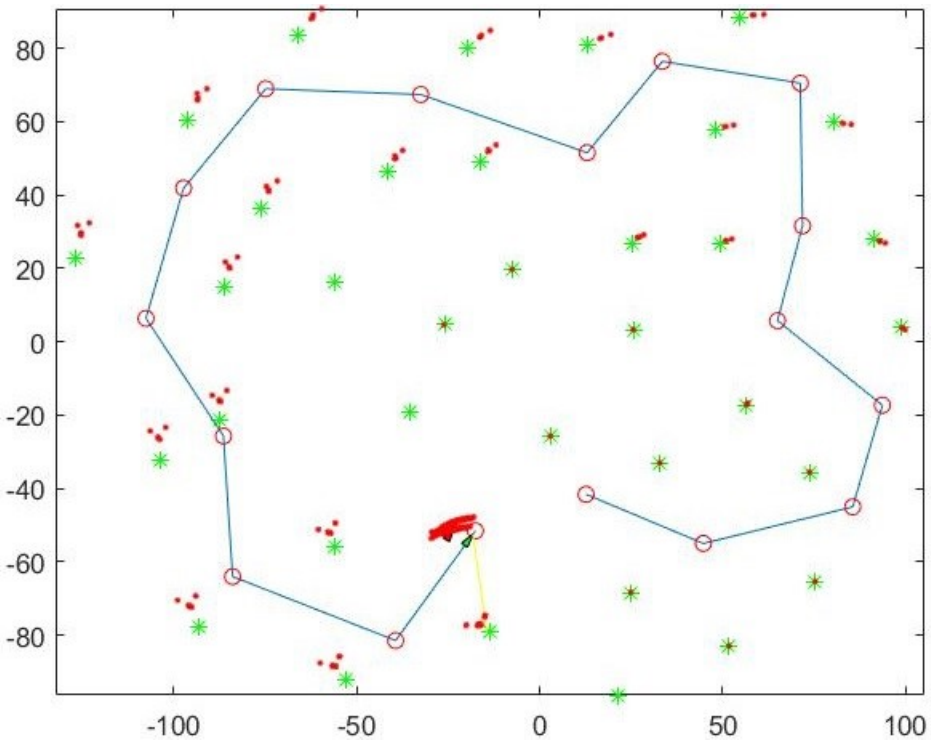


Figure 2-4: Fast SLAM 2.0 with increased Q simulation trial 4

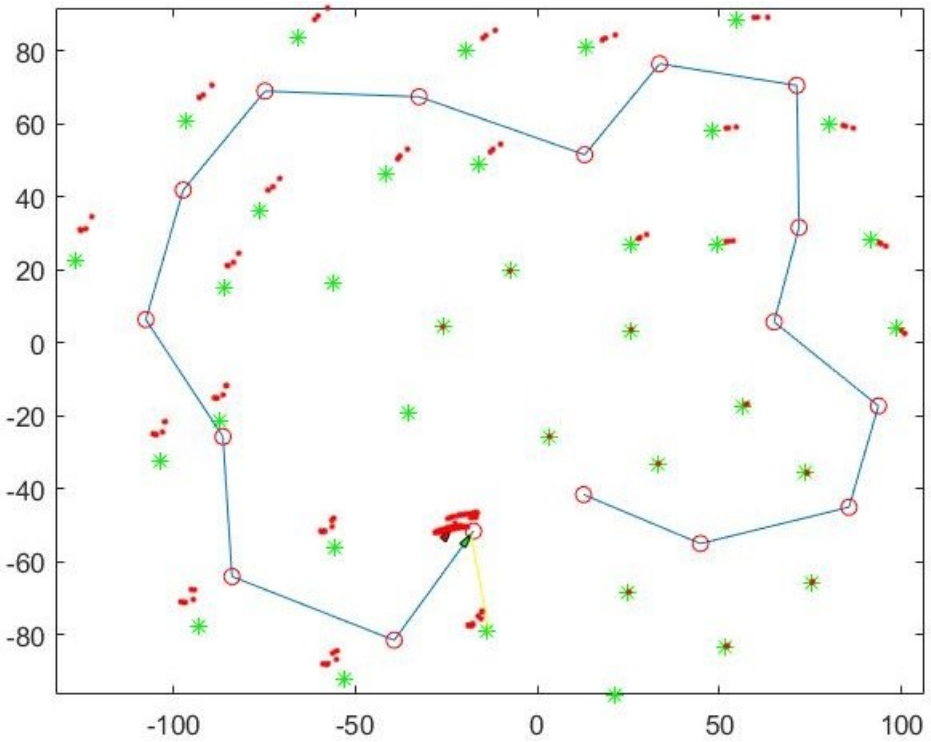


Figure 2-5: Fast SLAM 2.0 with increased Q simulation trial 5

Appendix C Fast SLAM 3.0 trials 1-5

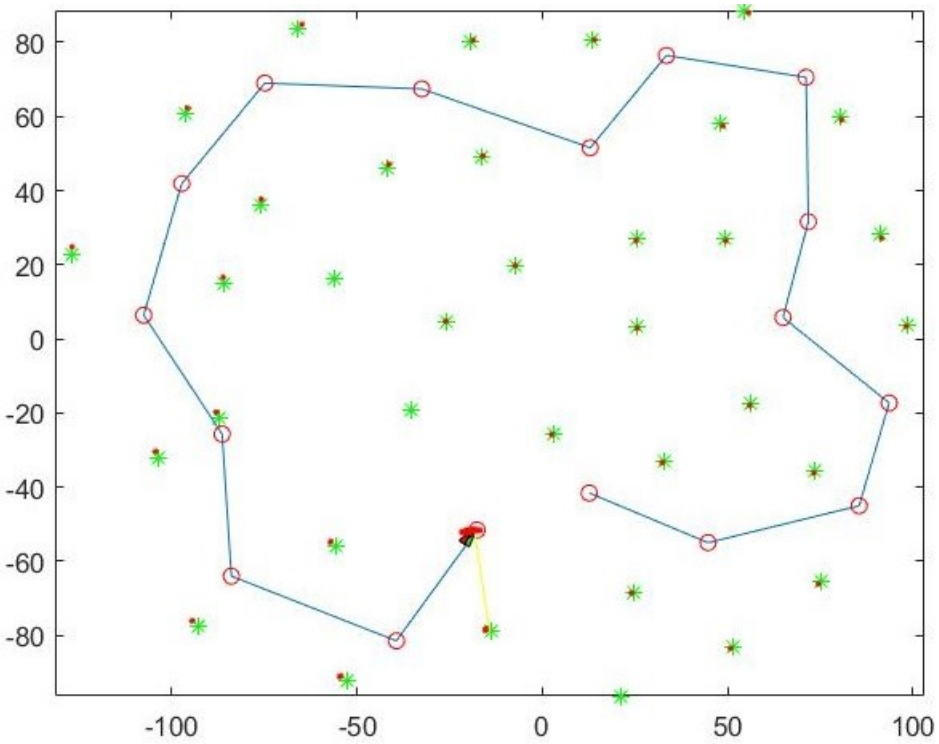


Figure 3-1: Fast SLAM 3.0 simulation trial 1

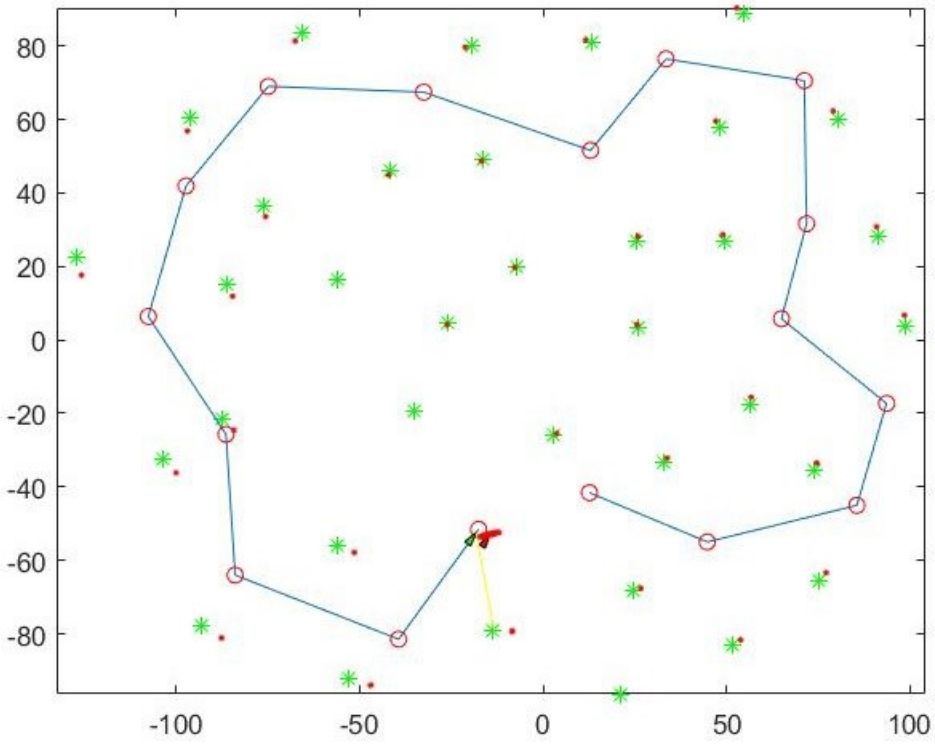


Figure 3-2: Fast SLAM 3.0 simulation trial 2

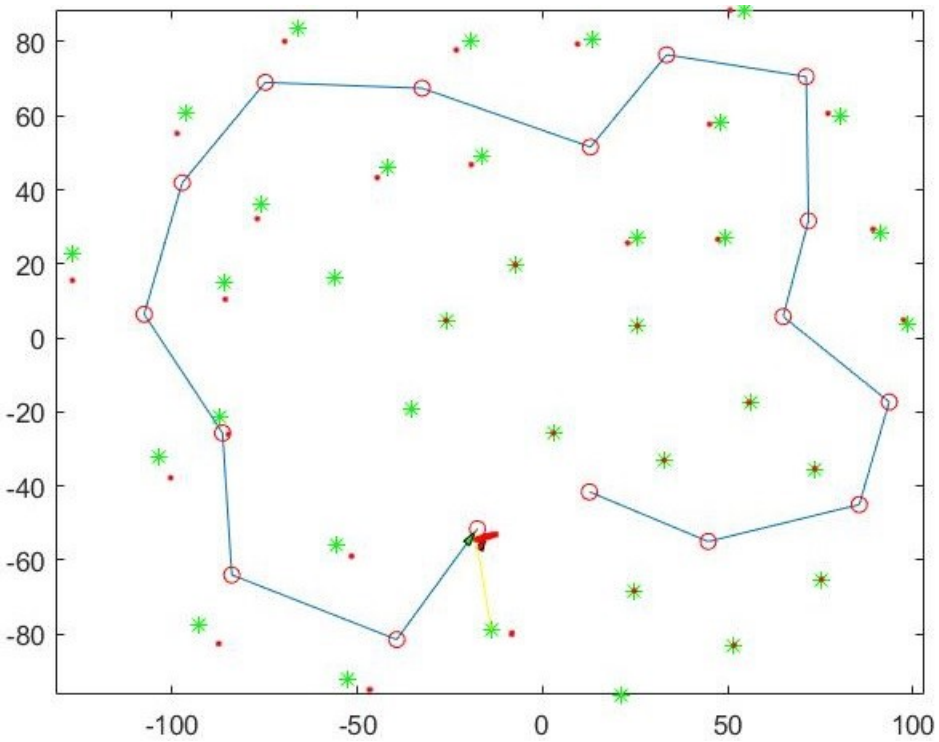


Figure 3-3: Fast SLAM 3.0 simulation trial 3

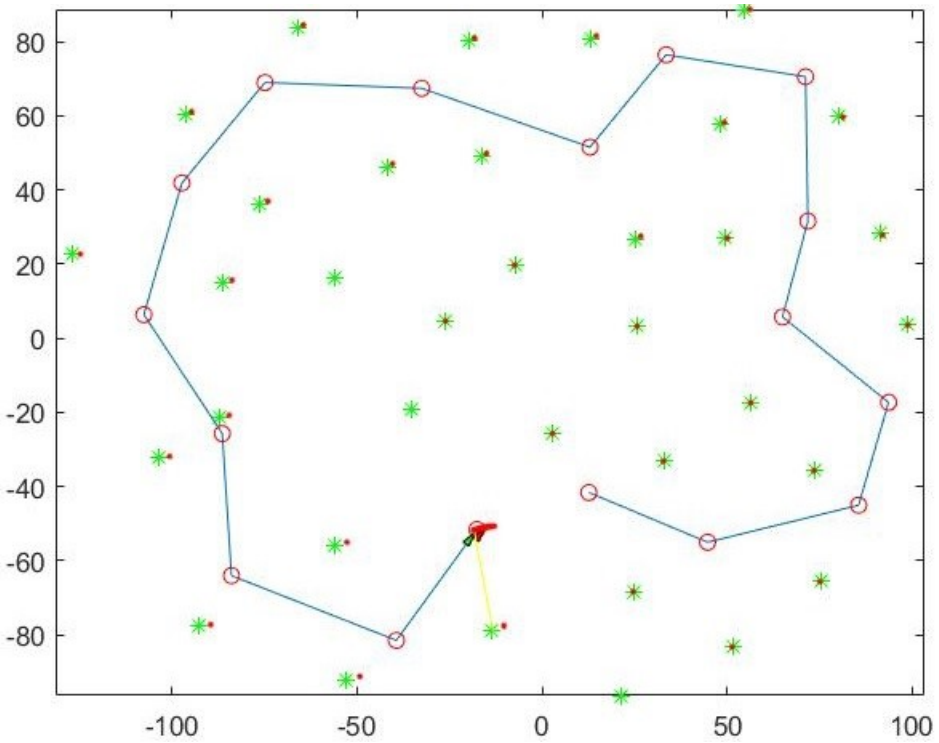


Figure 3-4: Fast SLAM 3.0 simulation trial 4

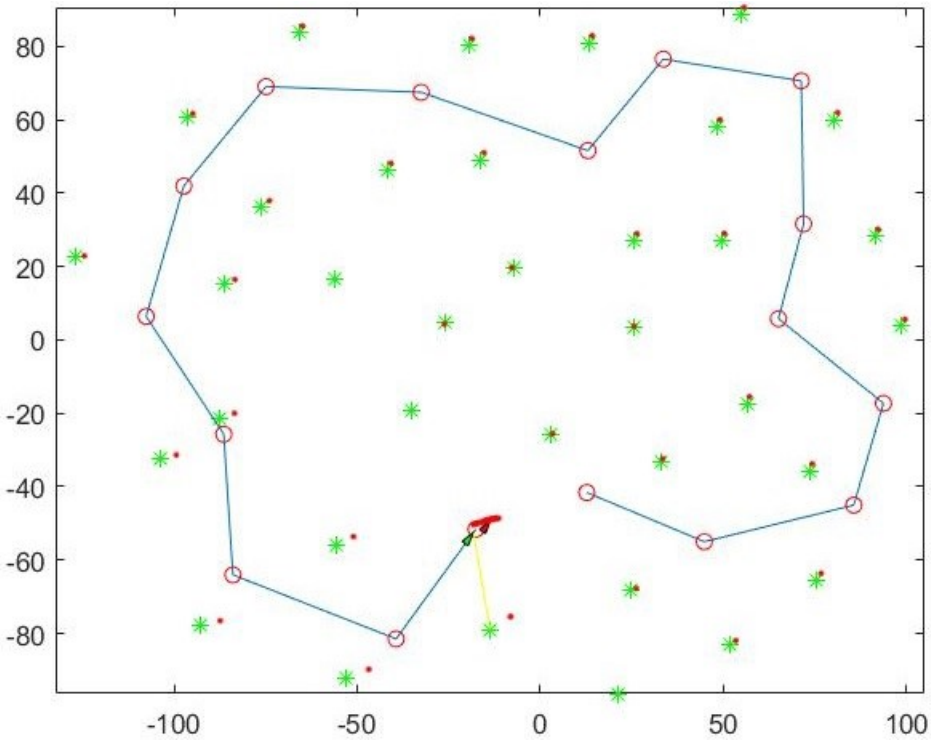


Figure 3-5: Fast SLAM 3.0 simulation trial 5

Appendix D EKF SLAM 2.0 trials 1-5

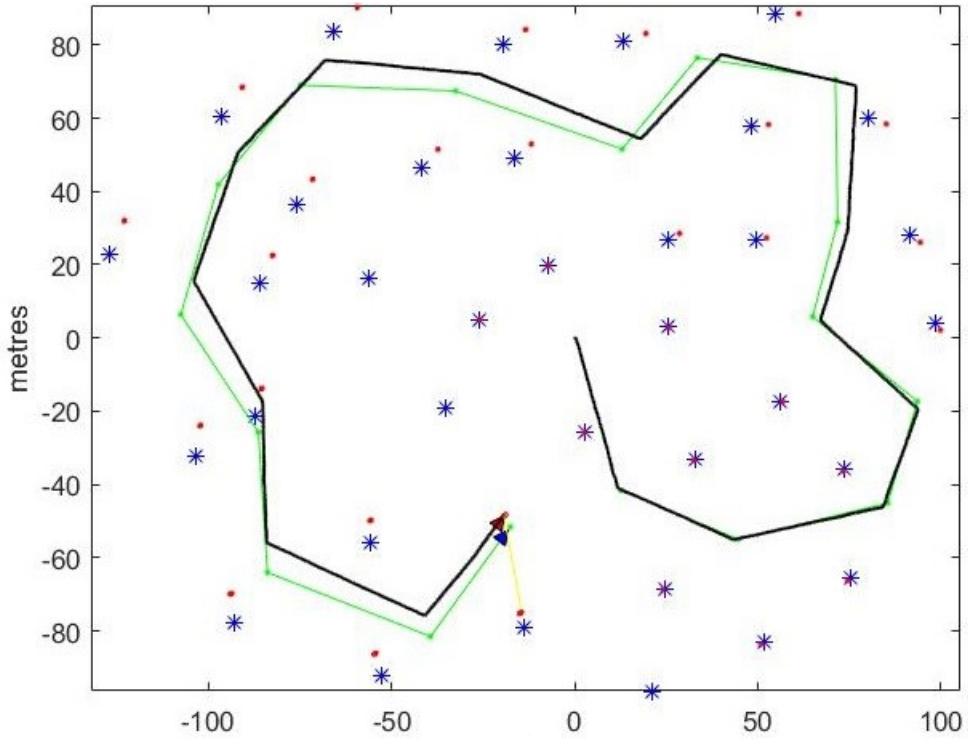


Figure 4-1: EKF SLAM simulation trial 1

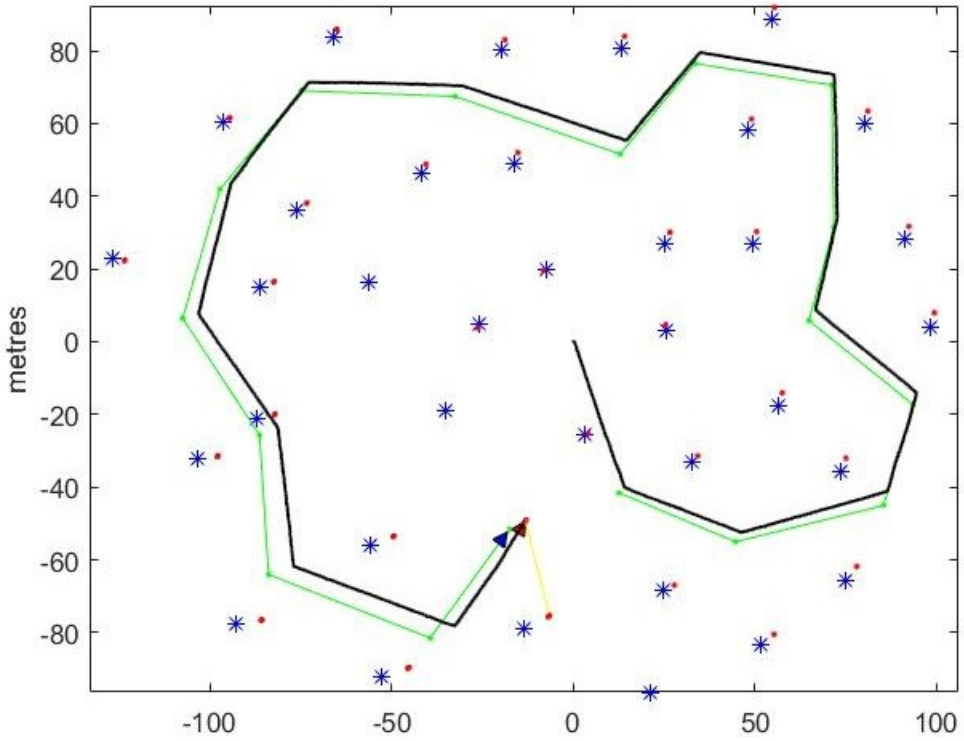


Figure 4-2: EKF SLAM simulation trial 2

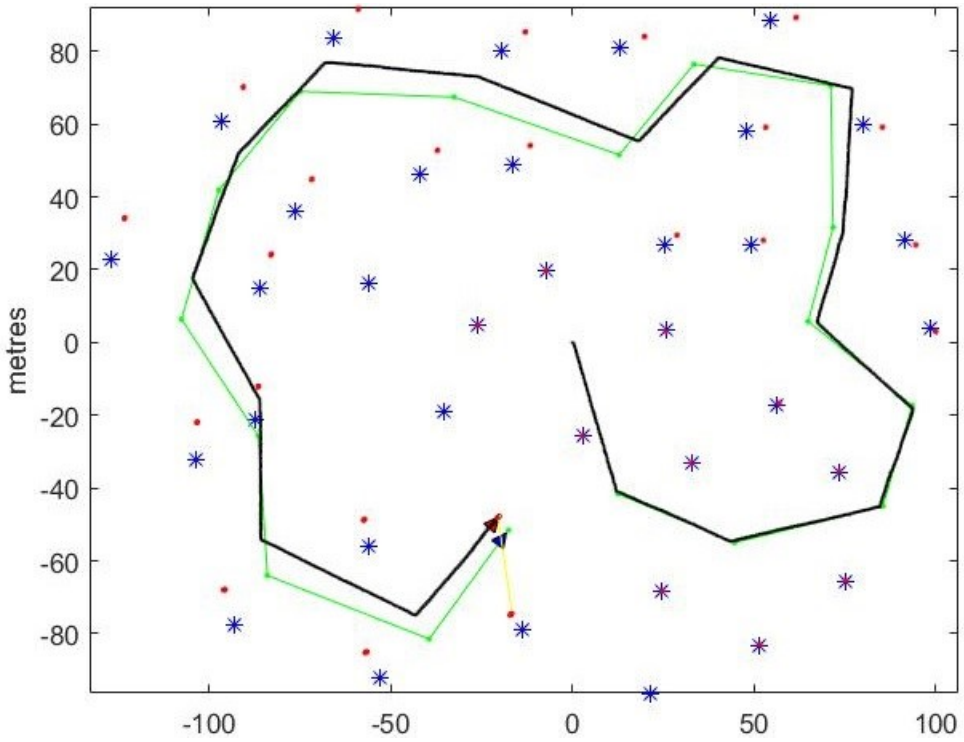


Figure 4-3: EKF SLAM simulation trial 3

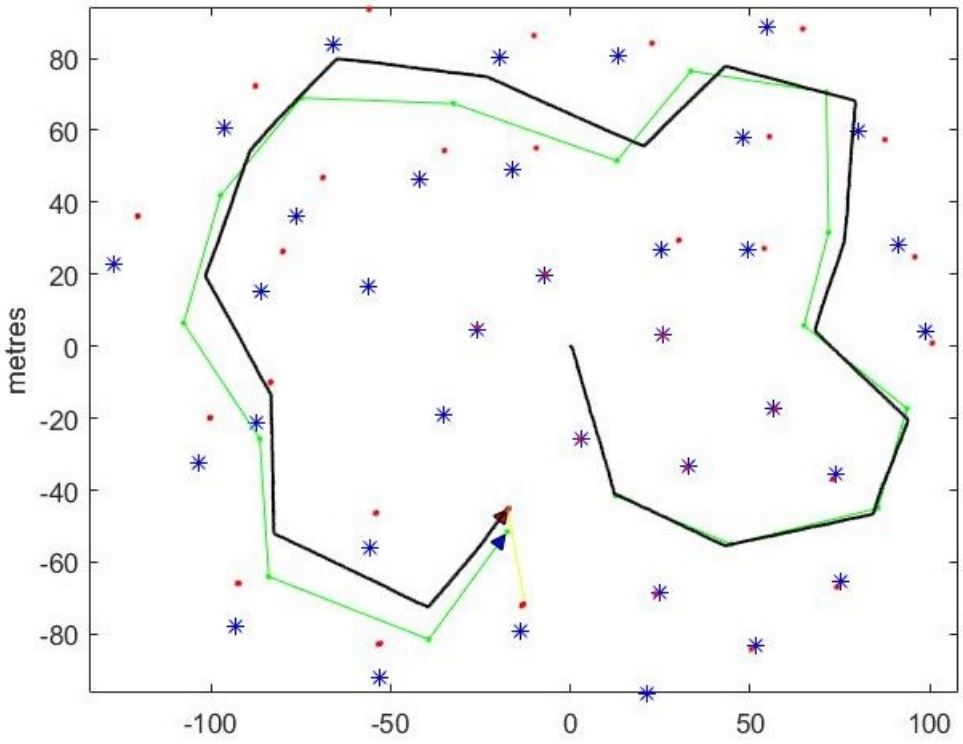


Figure 4-4: EKF SLAM simulation trial 4

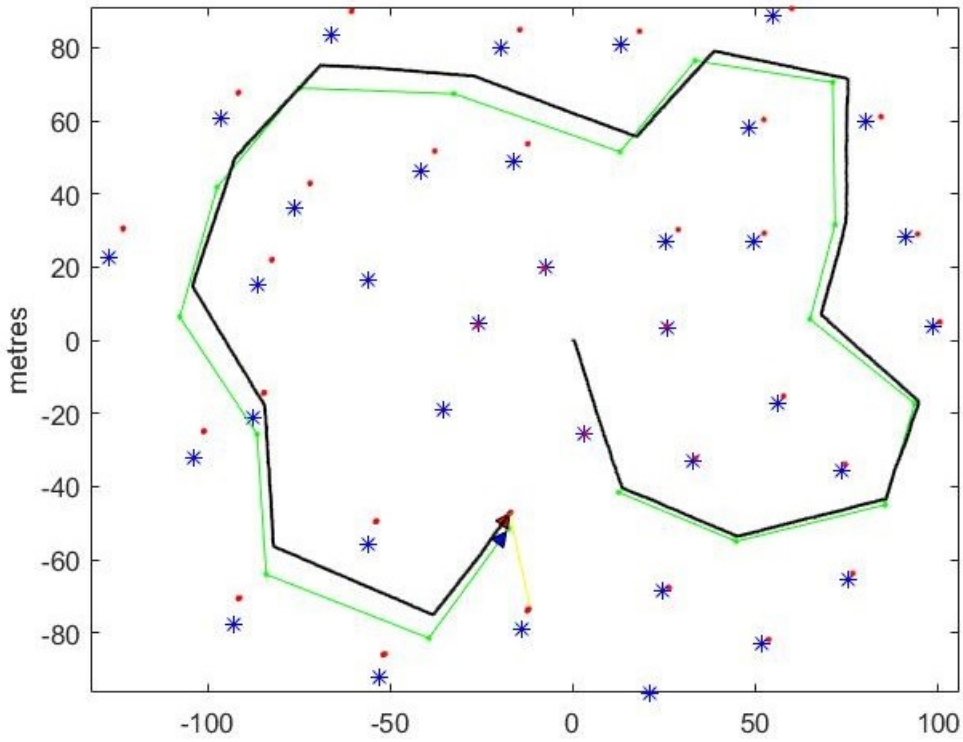


Figure 4-5: EKF SLAM simulation trial 5

Bibliography

[1] L. Taylor & Francis Group, Multi-Sensor Image Fusion and Its Application, 2006.

[2] D. Scaramuzza and F. Fraundorfer, "Visual Odometry [Tutorial]," *IEEE Robotics & Automation Magazine,* pp. 80 - 92, December 2011.

[3] D. Scaramuzza and F. Fraundorfer, "Visual Odometry: Part I - The First 30 Years and Fundamentals," *IEEE Robotics and Automation Magazine,* vol. 18, no. 4, 2011.

[4] T. Bailey and D. H. Whyte, "Simultaneous localisation and mapping (SLAM): Part I - the essential algorithms," *Robotics and Automation Magazine,* vol. 2, no. 13, pp. 99-110, 2006.

[5] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part II - Matching, robustness, optimization, and applications," *IEEE Robotics and Automation Magazine,* vol. 19, no. 2, 2012.

[6] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics & Automation Magazine,* vol. 3, no. 13, pp. 108-117, 2006.

[7] C. Forster, M. Pizzoli and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

[8] D. Nister, O. Naroditsky and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition*, 2004.

[9] D. Nister, "Preemptive ransac for live structure and motion," *Machine Vis. Applicat,* vol. 16, no. 5, p. 321–329, 2005.

[10] E. Rosten, G. Reitmayr and T. Drummond, "Improved ransac performance using simple, iterative minimal-set solvers," *University of Cambridge, Tech. Rep.,* 2010.

[11] O. Chum, J. Matas and J. Kittler, "Locally optimized ransac," *Proc. DAGM-Symp.,,* p. 236–243, 2003.

[12] C. Olson, L. Matthies, M. Schoppers and M. W. Maimone, "Robust stereo ego-motion for long distance navigation," in *Computer Vision and Pattern Recognition*, 2000.

[13] C. Olson, L. Matthies, M. Schoppers and M. Maimone, "Rover navigation using stereo ego-motion," *Robot Autonom,* vol. 43, no. 4, pp. 215-229, 2003.

[14] H. B. Mitchell, Image Fusion, Berlin Heidelberg: Springer, 2010.

[15] R. S. Blum and Z. Liu, Multi-Sensor Image Fusion and Its Application, CRC Press, 2006.

[16] P. Kim, Kalman Filter for Beginners with MATLAB Examples, A-JIN Publishing, 2010.

[17] R. G. Brown and P. Y. Hwang, Introduction to Random Signals and Applied Kalman Filtering, John Wiley & Sons, Inc., 1992.

[18] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics,* vol. 6, no. 32, p. 1309–1332, 2016.

[19] S. Perera, D. Barnes and D. Zelinsky, "Exploration: Simultaneous Localization and Mapping (SLAM)," *Computer Vision: A Reference Guide,* pp. 268-275, 2014.

[20] N. Aouf, A. Ollero and J. z. Sasiadek, "Airborne Simultaneous Localisation and Map Building (A-SLAM)," in *J Intell Robot Syst (2009)*, 2009.

[21] P. Mountney, D. et al. (Stoyanov, A. Davison and G.-Z. Yang, "Simultaneous Stereoscope Localization and Soft-Tissue Mapping for Minimal Invasive Surgery," *Lecture Notes in Computer Science,* vol. 1, no. 1, pp. 347-354, 2006.

[22] J. Z. Sasiadek, Y. Lu and V. Polotski, "Navigation of Autonomous Mobile Robots," *Robot Motion and Control, Springer Lecture Notes in Control and Information Sciences (LUNCIS),* vol. 360, pp. 187-208, 2007.

[23] A. Brooks and T. Bailey, "HybridSLAM: Combining FastSLAM and EKF-SLAM for Reliable Mapping," *Algorithmic Foundation of Robotics VIII,* vol. 57, pp. 647-661, 2009.

[24] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto and E. Nebot, "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research,* 2004.

[25] A. Eliazar and R. Parr, "DP-SLAM 2.0," in *IEEE Intl. Conf. on Robotics and Automation*, 2004.

[26] A. Monjazeb, J. Z. Sasiadek and D. Necsulescu, "Autonomous navigation among large number of nearby landmarks using FastSLAM and EKF-SLAM – a comparative study," in *16th IEEE International Conference on Mathematical Methods in Automation and Robotics (MMAR)*, Miedzyzdroje, Poland, 2011.

[27] J. Z. Sasiadek, A. Monjazeb and D. Necsulescu, "Autonomous navigation among

large number of nearby landmarks using FastSLAM and EKF-SLAM – a comparative study," in *IEEE 17-th International MED Conference on Control and Automation*, Thessaloniki, Greece, 2009.

[28] J. Z. Sasiadek, A. Monjazeb and D. Necsulescu, "Navigation of an Autonomous Mobile Robot using FastSLAM," *Journal of Intelligent Systems and Robotics,* 2008.

[29] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics, Massachusetts Institute of Technology, 2005.

[30] Y. Lu, V. Polotski and J. Z. Sasiadek, "Particle Filtering with Range Data Association for Mobile Robot Localization in Environments with Repetitive Geometry," in *Robot Motion and Control 2009. Lecture Notes in Control and Information Sciences*, London, 2009.

[31] Y. Lu, V. Polotski and J. Z. Sasiadek, "Mobile robot localization in environment with repetitive geometry, using particle filtering with range data association," in *Robot Motion and Control, Springer Lecture Notes in Control and Information Sciences (LUNCIS)*, 2009.

[32] T. Bailey, J. Nieto and E. Nebot, "Consistency of the FastSLAM algorithm," in *IEEE Intl. Conf. on Robotics and Automation*, 2006.

[33] A. Monjazeb, J. Z. Sasiadek and D. Necsulescu, "Autonomous Mobile Robot using Data Association Scheduling Method," in *10-th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Vienna, Austria, 2014.

[34] A. Monjazeb, J. Z. Sasiadek and D. Necsulescu, "HybridSLAM; A Robust Algorithm for Similtaneous Localization and Mapping," in *Proceedings of the 11-th*

International Conference on Informatics in Control, Automation and Robotics (ICINCO), Colmar, France, 2015.

[35] J. Z. Sasiadek, A. H. Monjazeb and D. Necsulescu, "Navigation of Autonomous Mobile Robots with Unscented HybridSLAM," *Informatics in Control, Automation and Robotics,* vol. 325, pp. 249-262, 2015.

[36] J. Z. Sasiadek, A. Monjazeb and D. Necsulescu, "Autonomous Mobile Robot Positioning using Unscented HybridSLAM," in *18th IEEE/IFAC International Conference on Methods and Models in Automation and Robotics*, Miedzyzdroje, Poland, 2013.

[37] A. Monjazeb, J. Z. Sasiadek and D. Necsulescu, "Some Aspects of Autonomous robot Navigation with Unscented HybridSLAM," in *9-th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Reykjavik, Iceland, 2013.

[38] A. Monjazeb, J. Z. Sasiadek and D. Necsulescu, "An Approach to Autonomous Navigation based on Unscented Hybrid SLAM," in *17-th IEEE/IFAC International Conference on Mathematical Methods in Automation and Robotics (MMAR)*, Miedzyzdroje, Poland, 2012.

[39] BusinessDictionary, "http://www.businessdictionary.com/definition/populism.html," 3 4 2017. [Online]. Available: http://www.businessdictionary.com/definition/populism.html.

[40] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine,* vol. 2, no. 13, p. 99–110, 2006.

[41] T. Bailey and H. D. Whyte, "Simultaneous localisation and mapping (SLAM): Part II - state of the art.," *Robotics and Automation Magazine,* vol. 3, no. 13, pp. 108-117, 2006.