View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by DigitalCommons@CalPoly

TOWARDS AUTONOMOUS LOCALIZATION OF AN UNDERWATER DRONE

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Nathan Sfard

June 2018

© 2018

Nathan Sfard

ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE:	Towards Autonomous Localization of an Underwater Drone
AUTHOR:	Nathan Sfard
DATE SUBMITTED:	June 2018
COMMITTEE CHAIR:	Lynne Slivovsky, Ph.D. Professor of Computer Engineering
COMMITTEE MEMBER:	John Seng, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Xiao-Hua Yu, Ph.D. Professor of Electrical Engineering

ABSTRACT

Towards Autonomous Localization of an Underwater Drone

Nathan Sfard

Autonomous vehicle navigation is a complex and challenging task. Land and aerial vehicles often use highly accurate GPS sensors to localize themselves in their environments. These sensors are ineffective in underwater environments due to signal attenuation. Autonomous underwater vehicles utilize one or more of the following approaches for successful localization and navigation: inertial/dead-reckoning, acoustic signals, and geophysical data. This thesis examines autonomous localization in a simulated environment for an OpenROV Underwater Drone using a Kalman Filter. This filter performs state estimation for a dead reckoning system exhibiting an additive error in location measurements. We evaluate the accuracy of this Kalman Filter by analyzing the effect each parameter has on accuracy, then choosing the best combination of parameter values to assess the overall accuracy of the Kalman Filter. We find that the two parameters with the greatest effects on the system are the constant acceleration and the measurement uncertainty of the system. We find the filter employing the best combination of parameters can greatly reduce measurement error and improve accuracy under typical operating conditions.

ACKNOWLEDGMENTS

Thanks to:

- My parents and grandparents, for supporting me throughout my college experience.
- Dr. Lynne Slivovsky, for guiding and intriguing me through the Capstone and Thesis projects.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Autonomous vehicles have gained considerable popularity over the years and their capabilities are continuously expanding as computers become capable of processing complex tasks at faster speeds and lower power. Some modern day examples of these autonomous vehicle advancements include the realization of self-driving cars and self-stabilizing quadcopters. Although incredible work has been accomplished for surface and air vehicles, not as much work has been performed for underwater vehicles and even less for underwater vehicles in unbounded and poor visibility environments, like the ocean. The reason autonomous vehicles are less popular in the ocean is because the ocean poses different challenges to the core problem of autonomous vehicles: autonomous navigation. Many industries and institutions have successfully avoided these core problems and deployed AUVs into the ocean by using state-of-the-art, high precision sensors and computers on-board. AUVs are used for many different practical and scientific applications in the areas of marine geoscience [21], marine biology, underwater mining [15], and the military. Some specific applications include mapping the seafloor [21], surveying changes to an ecosystem, following tagged marine animals [6], or, in the case of military applications, collecting intelligence or removing sea mines. Although autonomous navigation can be achieved with state-of-the-art equipment, the challenges associated with using typical sensors and computers can greatly affect the navigation accuracy of an AUV and, in response, compensations must be implemented.

Navigation accuracy is defined to be how accurate a vehicle can arrive from one point to another [13]. In order to be accurate and useful, an AUV must be able to navigate from one point to another while also staying as close as possible to a preset trajectory and dynamically avoiding obstacles along the way. To successfully accomplish accuracy and utility while autonomously navigating, an AUV must first determine its location, then identify all the objects around it, and finally plan a path accordingly toward its destination. These underlying and necessary tasks are commonly referred to as autonomous localization, feature extraction, and path planning, respectively.

[13] defines autonomous localization as how accurate a vehicle can locate itself with respect to a map or its initial position. On land, this problem is easily solved with a global positioning system (GPS); however, radio frequencies quickly attenuate underwater and therefore GPS cannot be used. The only signals that do not attenuate underwater are acoustic signals but these signals exhibit extremely high latency.

Navigation systems often rely on a feature detection system to perform localization as well. Feature detection augment localization by dynamically generating maps and using these maps to verify the AUV's location. Feature detection is also necessary to detect objects for avoidance protocols. The challenges present in underwater feature detection include overcoming poor visibility and fusing camera and sonar data.

Once autonomous localization and feature detection have been accomplished, a path planning algorithm can be developed. A path planning algorithm must send control inputs to the AUV so that the AUV follows a preset trajectory, updating the trajectory as necessary to avoid obstacles. The challenges associated with a path planning algorithm include correctly processing output from the autonomous localization and feature detection systems, accurately controlling the AUV, and experimenting with potentially many different path planning strategies. Once an AUV can plan a path, the goal of autonomous navigation can be realized. A general trajectory will be provided, like the one seen in Figure 1.1, and the AUV will be able to follow the trajectory accurately.

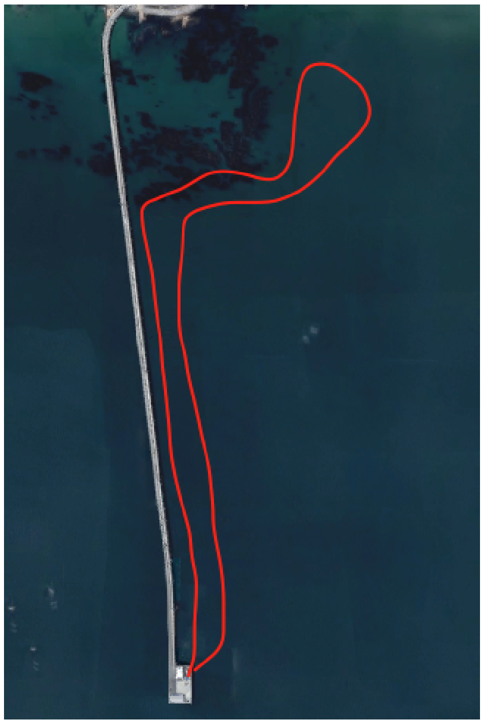

Figure 1.1: AUV Input Trajectory (location shown is the Cal Poly Pier)

This paper focuses specifically on the task of underwater autonomous localization. Our contributions include creating a framework to validate the accuracy of a localization scheme, implementing a constant acceleration Kalman Filter to estimate the pose of an AUV, suggesting methods of testing autonomous localization in a non-simulation environment, and identifying the tasks necessary to augment the Kalman Filter in order to produce more accurate results. The rest of this paper is structured as follows: Chapter 2 discusses background and related works in autonomous underwater localization, Chapter 3 goes over the kalman filter design, Chapter 4 covers the experimental setup, Chapter 5 presents the obtained results, and Chapter 6 includes potential future work and our concluding remarks.

Chapter 2

BACKGROUND AND RELATED WORK

AUVs are broadly used to either survey a general area or interact with a specific object. In both cases, the area in question may be deep underwater (e.g. surveying a hydrothermal vent on the seafloor) or just beneath the surface (e.g. surveying groves of seagrass or polar ice) [21]. When performing a specific task, an AUV only needs to navigate to the destination in question, perform the task, and optionally return to the initial location. In the case of surveying a general area, researchers will often deploy an AUV on a path closely resembling a lawn mower [12] or sawtooth [21] pattern. All these use cases greatly depend on the accuracy of an autonomous localization scheme and can also be a deciding factor in which scheme will be the most accurate.

The problem of autonomous underwater localization has undergone immense research and can be divided into three broad categories [13, 16]. The first approach is dead reckoning, which involves using an inertial measurement unit (IMU) – consisting of a number of accelerometers and gyroscopes, a compass, and a depth sensor – to calculate a change in position and orientation. The problem with dead reckoning is that measurement errors are additive, so the difference between the actual and measured position and orientation increases – or drifts – with time; even extremely accurate IMUs are susceptible to drift when the mission spans a great enough length of time. Work from Woodman states the standard deviation in position drift due to accelerometer white noise is proportional to the time the system is running raised to the power of $\frac{3}{2}$; Woodman proves this statement by deriving the following equation [20]:

$$\sigma_p(t) = \sigma_a * t^{\frac{3}{2}} * \sqrt{\frac{\delta t}{3}} \tag{2.1}$$

In Equation 2.1, $\sigma_p(t)$ is the standard deviation in position drift at time t, σ_a is the standard deviation in accelerometer noise, and δt is the amount of time between accelerometer measurements. Because of this inherent positional drift, the dead reckoning approach is often paired with a state estimation algorithm to alleviate the issue of drift. The state estimation algorithm first predicts the current state and then updates that prediction. The algorithm predicts the current state based on the last state and known control inputs, and process noise. The algorithm then updates that state prediction based on the observed position and orientation, taking into account measurement error and noise, in order to determine the true state of the AUV [13]. The Kalman Filter is the fastest known state estimator assuming the system is Markovian, linear, and the uncertainties are Gaussian [16]. Although the ocean is essentially nonlinear, the Kalman Filter can be extended (EKF) to account for the nonlinearity.

The second type employs acoustic transponders and modems placed at known locations to assist an AUV with the task of localization. In this solution, a beacon has a fixed location on the seafloor or is on the surface utilizing GPS and relays this information via acoustic signals to an AUV in order to calculate a relative position. The long baseline (LBL) approach places beacons along the seafloor to triangulate a position [19] and other approaches place beacons on buoys and vessels [13, 16]. The main drawback with the acoustic transponder and modem solution is that the mission area is limited to within the beacon network. An extension of this approach is to place beacons on multiple AUVs so they can cooperatively locate themselves [10], although this requires multiple vehicles. The idea is that a team of AUVs will be deployed and each dead reckon until an AUV periodically surfaces, gets an accurate GPS location, and shares the information with the rest of the AUVs. This idea is further developed in [11] in which one AUV served as the leader, equipped with extremely high precision sensors, and this leader AUV relayed its trusted position via acoustic signals to the other, less precise worker AUVs.

The third approach involves using geophysical data from the AUVs surroundings to determine the position of the AUV. This solution aims to solve the localization problem the same way animals do, by using sensory information to identify landmarks. This sensory information is often in the form of optical and sonar data but other data can be used, such as magnetic forces, temperature, water quality, etc. The sensory information is input into some feature detection algorithm and a map is constructed and updated with every sensor reading. The constructed map can additionally be compared to a given map to increase accuracy. On every iteration of this map generating algorithm, the AUV's location is also calculated within the map using the distances of all the landmarks. This technique is commonly referred to as simultaneous localization and mapping (SLAM) [13, 16]. The problem with this solution is that sonar and optical data are often noisy underwater, especially in the ocean, and, in order for this solution to be accurate, there must be lots of landmarks nearby.

A combination of the solutions discussed is the best solution [13, 16]. An example would be using dead reckoning with state estimation until a landmark or a beacon is found, and then restarting the state estimation algorithm. A common approach is to use a state estimation algorithm to estimate additive errors in dead reckoning as well as errors in SLAM due to time delays. An implementation of this approach, an EKF-SLAM algorithm, can be found in [3, 4, 12, 2, 5]. In [10] a team of AUVs use an EKF-SLAM algorithm to estimate a position and a map and then each AUV sends this information acoustically to the rest of the team so that every AUV can create the same detailed map. In this paper we attempt to solve the localization problem using the Kalman Filter to estimate state while dead reckoning. Although other state estimators yield far better results, including particle filters [16] and neural networks [8], these approaches require orders of magnitude more computing resources and are therefore unable to perform in real time using the typically limited on-board computer.

Chapter 3

KALMAN FILTER DESIGN

Before any solution can be carefully designed, it is necessary to provide a proper model of an AUV [13, 16]. The model in [17] describes two reference frames: the inertial frame, in which the earth is fixed, and the local frame, in which the AUV is fixed. Considering a local frame, an AUV is modelled as a free body having six degrees of freedom (DOF) called surge, sway, heave, roll, pitch, and yaw corresponding to movement along the x-axis (forward and backward), movement along the y-axis (side to side), movement along the z-axis (up and down), rotation about the x-axis, rotation about the y-axis, and rotation about the z-axis, respectively [17, 18]. The six degrees of freedom, representing a vehicle's position and orientation with respect to a local frame, are grouped into the single state variable that a localization algorithm aims to estimate and is commonly referred to as the vehicle's pose. A diagram visualizing an AUV's pose within two reference frames can be seen in Figure 3.1.

The Kalman Filter is a learning algorithm that considers process and measurement uncertainties to model a Markovian, linear system in which measurements exhibit gaussian noise. The algorithm operates in a cycle of two distinct stages: predict and update, shown in Figure 3.2. The algorithm first attempts to predict the current state and uncertainty from the last state and uncertainty. The algorithm then uses the measurements obtained at the current timestep to update the current state and uncertainty estimations. These two stages of the algorithm are then repeated over and over again for every discrete timestep in the running system. Note, however, that the update stage does not necessarily need to occur at every timestep, allowing this algorithm to continue to predict state in the presence of low frequency measurements or slow processors. The rest of this chapter further details the Kalman Filter imple-

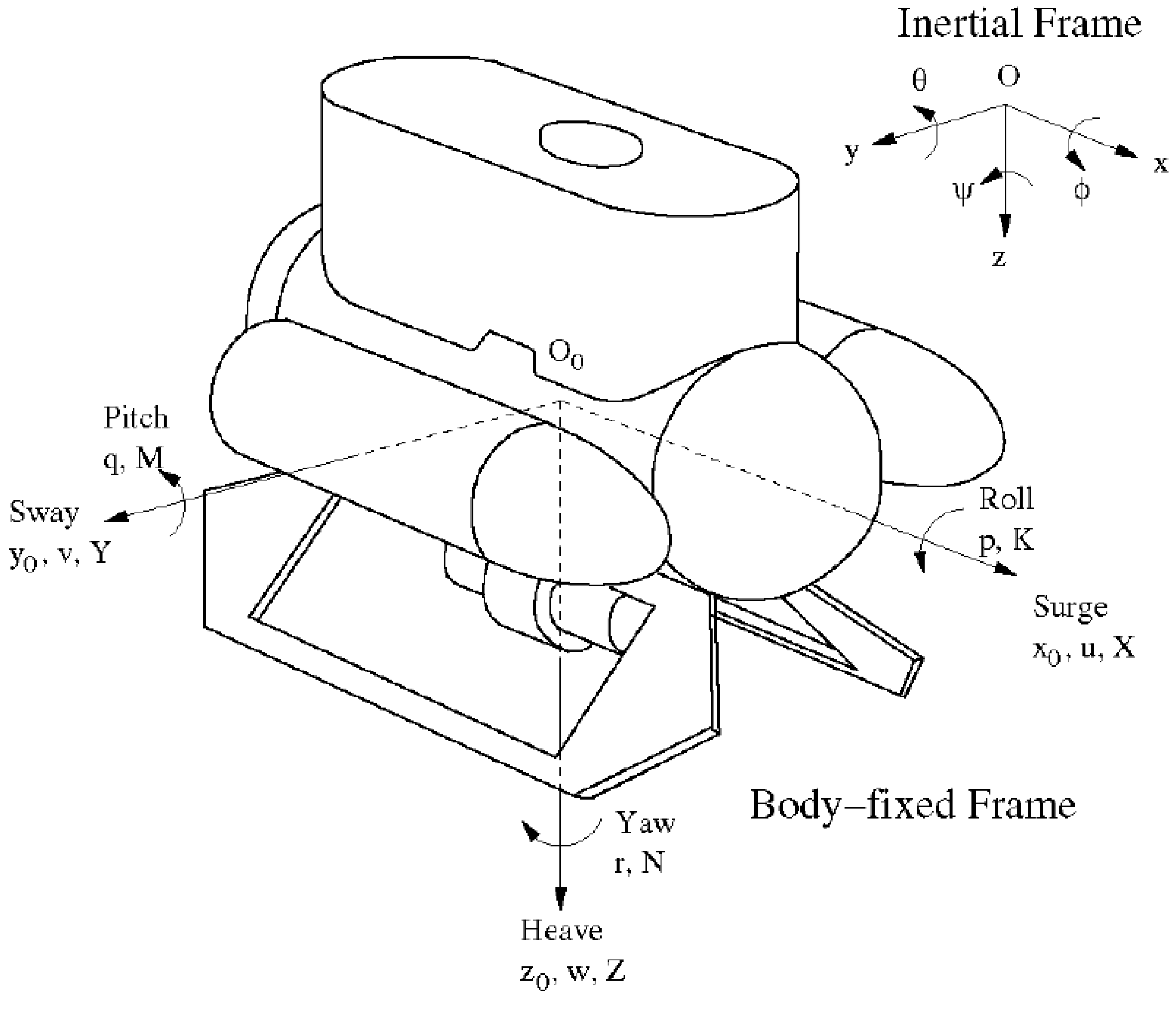

Figure 3.1: Six degrees of freedom in inertial and local frame [17]

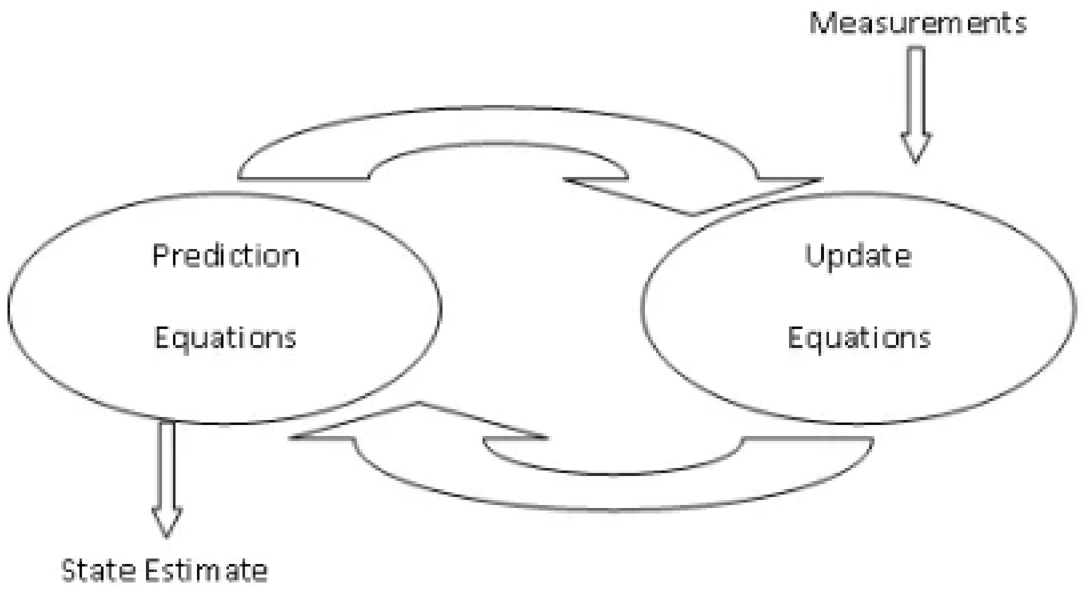

Figure 3.2: Kalman Filter Flow [9]

mented for this project, first stating the assumptions of our system, then presenting the system, predict, and update equations, and finally deriving datasheet variables.

3.1 Assumptions

In order to define and implement the Kalman Filter equations, we need to first correctly model our system and state our assumptions. To correctly model our system, we use the six degree-of-freedom (6-DOF) model mentioned at the beginning of this chapter, specifically using the body-fixed frame for measurements and the Earth-fixed frame for pose estimation. Using the body-fixed frame enables us to measure an AUV's current pose as a difference from the AUV's last pose. Using the Earth-fixed frame enables us to estimate an AUV's pose in relation to its initial pose, which leads us to our first assumption: *the initial pose of the AUV is known and correct.* We need the initial pose to be known and correct so our system has some reference point from which to begin estimating and it is also quite common to assume the initial pose is known and correct since AUVs are commonly deployed at the surface where GPS can be used.

The second assumption involves the characteristics of some of the IMU sensors. Specifically, the depth sensor and compass exhibit some noise but the error associated with this noise is not additive, therefore we assume *the depth sensor and compass are adequately accurate* and don't need to be corrected. This assumption means that the Kalman Filter does not need to include a change in z position or yaw angle (using the 6-DOF body-fixed frame) as a part of its overall state. To further simplify our system, we also assume that *the roll and pitch angles of the AUV are fixed.* This assumption simplifies the system to a 2-DOF model instead of a 6-DOF model. Using the 2-DOF model enables us to take raw x and y measurements from the accelerometer without having to include the gyroscope measurements, eliminating the need to convert body-fixed x and y measurements to Earth-fixed x and y measurements. Not considering the gyroscope also eliminates the inherent drift in orientation that occurs from noisy gyroscope measurements. Note that, although we chose to assume orientation is fixed, the filter implemented in this paper can easily be augmented to correct x and y measurements for an Earth-fixed frame (by multiplying the measurement with the cosine and sine of the yaw angle, respectively) and estimate orientation (by adding orientation to the filter's state and using gyroscope measurements to update the state).

This work focuses on a theoretical approach. Therefore, the last two assumptions are that *there are no control inputs for the filter to account for* and *the AUV exhibits constant acceleration.* When designing a system for a physical AUV, evaluating the effect of control inputs on the system is necessary in order to accomplish autonomous localization and involves measuring the actual effect that control inputs have on an AUV. Furthermore, assuming the AUV moves with constant acceleration is necessary since the basic Kalman Filter works best when the system it estimates is linear and Markovian. Accounting for the non-linearity of the ocean is possible, though, by extending the kalman filter (EKF) and would enable the filter to work without assuming constant acceleration. Implementing an EKF, however, is left as another future project.

To summarize, we assume that the initial pose of the AUV is known and correct, the system only needs to estimate raw x and y measurements from the accelerometer, control inputs are not used, and the AUV operates at constant acceleration. Next, we present our System and Kalman Filter equations taking into account the mentioned assumptions.

3.2 System Equations

Every control system must be described by two equations. The first equation, Equation 3.1, is the state equation. This equation calculates the system's next state based on the current state, x_k, and the inputs to the system, u_k. The second equation is the system's output equation (Equation 3.2). This equation calculates the system's current output in terms of the system's current state and inputs. In our system the output, y_k, is a vector including only position in the x and y direction.

$$x_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ x'_{k+1} \\ y'_{k+1} \\ x''x_{k+1} \\ y''_{k+1} \end{bmatrix} = Ax_k + Bu_k \tag{3.1}$$

$$y_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix} = Cx_k + Du_k \tag{3.2}$$

In both equations x_k is the state vector, which contains all the variables necessary to represent the system's state at timestep k. In our system, there are six variables that represent our state: x and y position, x and y velocity, and x and y acceleration, denoted by x, y, x', y', $x'\prime$, and $y'\prime$, respectively. Equation 3.1 shows the system's state vector.

Although every control system uses the exact same general equations to describe the system, the A, B, C, and D matrices differ from system to system. In our system we assume no control inputs so B and D will be 0. The A matrix, called the state transition matrix, is a $6X6$ matrix representing how each variable in the state vector relates to every other variable in the state vector including itself. Since we assume our system exhibits constant acceleration and fixed orientation, we can use the basic kinematic equations relating position, velocity, and acceleration to fill in the A matrix, seen in Equation 3.3.

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

The C matrix relates the state vector, x_k to the output vector, y_k. Since the output vector only contains the current position of the system (Equation 3.2), the C matrix need only extract those variables from the state vector. Thus, Equation 3.4 shows the C matrix.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

An important part of control systems is calculating the observability of the system. One can determine the observability of a system by calculating the rank of the system's observability matrix. The observability matrix is calculated using Equation 3.5, in which n is the number of state variables. Using Matlab, we found the system has a full rank (i.e. rank is equal to the number of state variables) so the system is completely observable.

$$O = \begin{bmatrix} A \\ C * A \\ C * A^2 \\ \dots \\ C * A^{n-1} \end{bmatrix} \tag{3.5}$$

3.3 Predict Equations

As mentioned earlier, the Kalman Filter operates as a constant cycle of predict and update stages. The following equations represent the predict stage of the Kalman Filter:

$$\hat{x_k} = F * x_{k-1} + B * u_{k-1} \tag{3.6}$$

$$\hat{P_k} = Q + F * P_{k-1} * F^T \tag{3.7}$$

In Equations 3.6 and 3.7, the hat above a variable denotes the variable is an estimate, therefore the predict equations are yielding an estimate for x_k and P_k, respectively. x_k is the same state vector from Equation 3.1. The state variable is initialized to a known value, i.e. x_0 should be initialized so the x and y accelerations are the constant acceleration of the system and the x and y velocities and positions are the known initial velocities and positions.

P_k is a $6X6$ matrix representing the system's process uncertainty or, in other words, how confident the filter is with its state estimate. P_k includes the subscript k to indicate that it is updated with every filter cycle, thus the uncertainty is dynamic. P_0 is typically initialized as a diagonal matrix where each term in the diagonal is the square of the standard deviation in error for each state variable [9]. Thus, P_0 takes the form seen in Equation 3.8.

$$P_0 = \begin{bmatrix} \sigma_{px}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{py}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{vx}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{vy}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{ax}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{ay}^2 \end{bmatrix} \tag{3.8}$$

The F term is the state transition matrix and is identical to the A term in Equation 3.1.Note that assuming a fixed orientation also means that the position drift in the x direction is independent of a position drift in the y direction and visa-versa. The matrix in Equation 3.9 shows the F matrix used in our predict equations.

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

The next part of the equation, denoted by $B * u_{k-1}$, is the part of the estimate that takes the control inputs into account. Since we assume there are no control inputs, this term will always be 0. Finally, Q is the covariance of the process noise, i.e. how noise in the process from unknown sources affects each variable in the state vector over one timestep. This $6X6$ matrix is similar to the P_k matrix except the Q matrix is constant. To calculate Q, we again use the basic kinematic relationships to see how noise in the system's acceleration will affect the x and y acceleration, velocity, and position and then multiply those relationships by the square of the standard deviation of the noise in acceleration measurements (σ_α^2). The matrices described in Equation 3.10 and Equation 3.11 are used to calculate the Q matrix (Equation 3.12), where Δt is the length of the timestep.

$$G = \begin{bmatrix} \frac{1}{2}\Delta t^2 & \frac{1}{2}\Delta t^2 & \Delta t & \Delta t & 1 & 1 \end{bmatrix}^T \tag{3.10}$$

$$Q = G * G^T * \sigma_\alpha^2 \tag{3.11}$$

$$Q = \begin{bmatrix} \frac{1}{4}\Delta t^4 & \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^2 & \frac{1}{2}\Delta t^2 \\ \frac{1}{4}\Delta t^4 & \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^2 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^3 & \Delta t^2 & \Delta t^2 & \Delta t & \Delta t \\ \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^3 & \Delta t^2 & \Delta t^2 & \Delta t & \Delta t \\ \frac{1}{2}\Delta t^2 & \frac{1}{2}\Delta t^2 & \Delta t & \Delta t & 1 & 1 \\ \frac{1}{2}\Delta t^2 & \frac{1}{2}\Delta t^2 & \Delta t & \Delta t & 1 & 1 \end{bmatrix} * \sigma_\alpha^2 \tag{3.12}$$

In summary, the predict equations calculate an estimate for the current state, $\hat{x_k}$ and current process uncertainty, $\hat{P_k}$. The current state is estimated by multiplying the state transition matrix, F, by the (corrected) last state, x_k. The current process uncertainty is estimated by multiplying the (corrected) last process uncertainty, P_k by the transition matrix, F, squared and adding the covariance of the process noise, Q.

3.4 Update Equations

The second stage of the Kalman Filter is the update or correction stage, modeled by Equations 3.13, 3.14, 3.15, and 3.16.

$$x_k = \hat{x_k} + K_k * (z_k - H * \hat{x_k}) \tag{3.13}$$

$$P_k = \hat{P_k} - K_k * S_k * K_k^T \tag{3.14}$$

$$K_k = \hat{P_k} * H^T * S_k^{-1} \tag{3.15}$$

$$S_k = H * \hat{P_k} * H^T + R \tag{3.16}$$

Similar to the predict equations, the main objective of the update equations is calculating the current state, x_k, and the process uncertainty, P_k. These equations use measurements at the current timestep and the corresponding measurement noise to correct the estimates from the predict stage. z_k is a vector containing all the measurements taken at timestep k. In our system, we are only measuring x and y accelerations from an accelerometer so this vector contains two elements. H is a $2X6$ matrix that translates how each measurement affects each variable in the state vector. Since our system only measures x and y acceleration, the H matrix will only map these measurements to the x and y acceleration variables in the state vector, respectively. Equation 3.17 represents the H matrix.

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

The R term is a $2X2$ matrix representing the measurement uncertainty, or noise, from the x-axis and y-axis accelerometers. This matrix is just a diagonal matrix consisting of the square of the standard deviation for each sensor. The R matrix used in our system is represented in Equation 3.18.

$$R = \begin{bmatrix} \sigma_\alpha^2 & 0 \\ 0 & \sigma_\alpha^2 \end{bmatrix} \tag{3.18}$$

Finally, the K_k and S_k matrices are intermediate terms. The whole set of update equations are employed to correct the estimates calculated in the predict stage. The state vector, x_k, is updated for the current timestep by taking into account the state estimate for the current timestep, $\hat{x_k}$, the measurements at the current timestep, z_k, and the constant measurement noise, R. Similarly, the process uncertainty for the current timestep, P_k is updated by taking into account the process uncertainty estimate for the current timestep, $\hat{P_k}$, and the constant measurement noise, R. In the next chapter we describe how to obtain the σ_α used in the predict and update equations.

Chapter 4

EXPERIMENTAL DESIGN

Our goal is to implement an autonomous localization algorithm using accelerometer readings from an actual OpenROV and we use simulations to evaluate the algorithm. In order to test the accuracy of a localization algorithm, including the Kalman Filter implemented in this paper, we develop an experimental testbench that simulates noisy measurements from the AUV's accelerometers and processes those measurements through a given filter. In addition to simulating measurement noise and the Kalman Filter's response, we also test the effect of tuning certain parameters in our system, specifically acceleration, initial velocity, normal vs. uniform noise, initial process uncertainty (P), measurement uncertainty (R), and process uncertainty covariance (Q).

4.1 Experimental Testbench

The experimental testbench, implemented in Python 2.7, consists of two components: a simulator and a filter. The simulator is the main test script, responsible for initializing the filter, simulating noisy measurements, passing these measurements to the filter, and reporting position errors from both the measurements and the filter. The filter is a Python class that implements the interface defined by another abstract class; this interface includes one method to initialize the filter and another method that accepts noisy measurements and produces corrections. It is designed to be modular so that it can be adapted to test additional localiztion algorithms in the future.

The simulator conducts an experiment by running 100 simulations, using the simulation number as the seed to the random number generator so every simulation is repeatable. Each simulation produces 10 minutes of uniformly distributed noisy measurements for both x and y accelerations (recall that the modeled accelerometer produces measurements at a rate of 100Hz). At each timestep the current set of measurements is passed to the filter and the current position is calculated by integrating the measured acceleration. Every 60 seconds, the simulator will additionally report the error in the calculated position as well as the error in the filtered position, which is calculated as the difference from the true position. The simulator is also configurable, so one can change the number of simulations, the duration of each simulation, the x and y acceleration, the initial x and y velocities, the data rate, whether the noise is normally or uniformly distributed, and the standard deviation of the measurement noise.

Python's random number generator is used to simulate noisy measurements since simulating actual ocean physics is a difficult task. Most current underwater simulators, like UWSim [14], are great for systems that are concerned only with visualizing water beneath the surface or simulating physics between two underwater objects but lack the ability to simulate hydrodynamic forces on an object. A Matlab tool called Marine Systems Simulator (MSS) [7] could potentially be used, however, collecting and simulating actual hydrodynamic forces is left as a future project.

The σ used for the random number generator and in the filter equations is the standard deviation in accelerometer noise, which can be calculated using the IMU's datasheet. The IMU we model, and sold by OpenROV, is the Bosch BNO055 Intelligent 9-axis absolute orientation sensor [1]. The variables from the sensor's datasheet we are concerned with, noise density and data rate, are listed in Table 4.1. To calculate σ_α for the accelerometer, we substituted known values for the units of the noise density. Specifically, we multiplied the noise density by the gravitational constant, g, multiplied by μ (10^{-6}), and divided by the square root of the data rate, which is 100Hz. Using the typical noise density, σ_α is calculated to be 1.4715e-4 ms^{-2}. Using

Parameter	Value	Units
Output Noise Density (typical)	150	$\mu g/\sqrt{Hz}$
Output Noise Density (worst)	190	$\mu g/\sqrt{Hz}$
Data Output Rate	100	Hz

Table 4.1: Datasheet Variables [1]

equation 2.1, we find that the standard deviation in acceleration for this accelerometer yields a standard deviation in position of slightly more than 1.8 meters over one hour (recall that the standard deviation in position grows proportionally to $t^{\frac{3}{2}}$).

Another important aspect of the filter is that it must maintain its own state in order to produce correct estimates and to be decoupled from the simulator. The Kalman Filter implemented in this paper maintains state by saving the estimates and corrections of the state vector and process uncertainty matrix. This Kalman Filter is initialized with the constant x and y accelerations and the initial x and y velocities used in the simulator. The rest of the parameters are initialized independent of the simulator and how these parameters are tuned are analyzed in the next section.

4.2 Parameter Tuning

We tune the following parameters to see the effect on the filter's ability to correct position error: acceleration, initial velocity, sensor noise, initial process uncertainty (P), measurement uncertainty (R), and process uncertainty covariance (Q). Acceleration, initial velocity, and sensor noise, are all tuned within the simulator and the rest are tuned within the filter. Each parameter is varied individually while keeping the other parameters constant at their default value. Each of these variations is then tested across 100 simulations. We describe our rationale for varying parameter values and summarize them in Table 4.2.

The Kalman Filter we implemented assumes a constant acceleration, however, we would like to test the effect that a zero and non-zero constant acceleration has on the filter. We decided to test values between 0 and $0.1ms^{-2}$ for both x and y directions to start with. The top speed of the OpenROV Underwater Drone is just above $1m/s$ so an acceleration of $.1ms^{-2}$ is well above its capabilities. We also decided to keep both x and y acceleration at $0ms^{-2}$ when testing the effect of other parameters since the acceleration value theoretically affects the system greatly and we would like to isolate each parameter's effect.

We wanted to test the effect that velocity may have on the zero acceleration system so we tested the system with x and y velocities between 0 and $1m/s$. Note that our Kalman Filter equations guarantee that x and y measurements are independent, so we therefore do not need to test the effect of varying acceleration or velocity in one direction on the other direction.

Sensor noise describes how the simulated measurements are randomly generated; either uniformly or normally with 0 mean and the standard deviation of the measurement noise. The reason the type of distribution is tested is because the Kalman Filter works best for Gaussian (normal) noise, however the noise exhibited by the accelerometer in question is uniformly distributed. Because the sensor is uniformly distributed, we choose the uniform distribution to be the default distribution when testing other parameters.

The initial process uncertainty describes how fast the filter can converge on the true position from the noisy measurements. Recall that the process uncertainty is often initialized as a diagonal matrix with each diagonal element initialized to the corresponding state variable's variance in process noise. These values can only be achieved through experimentation, so we tested with values ranging from σ^2 up to 1000. We also tested the effect of having a low uncertainty in acceleration but a high

Parameter	Default Value	Variations	Units
x Acceleration	0	$0, 1e{-}6, 1e{-}3, 1e{-}1$	ms^{-2}
y Acceleration	0	$1e{-}1, 1e{-}3, 1e-6, 0$	
x Velocity	0	$0, 1e{-}3, 1e{-}1, 1$	m/s
y Velocity	0	$1, 1e{-}1, 1e{-}3, 0$	
Sensor Noise	Uniform	$Uniform, Normal$	N/A
P_a (see Figure 4.1)	σ	$\sigma^2, \sigma, 0.01, 1, 1000, \sigma, 1000$	m
P_v	σ	$\sigma^2, \sigma, 0.01, 1, 1000, 1, 1$	m
P_p	σ	$\sigma^2, \sigma, 0.01, 1, 1000, 1000, \sigma$	m
σ_α^2 in R	σ^2	$\sigma^4, \sigma^2, \sigma, 0.01, 1, 1000$	m
scalar on Q	1	$\sigma^2, \sigma, 0.01, 1, 1000$	N/A

Table 4.2: Parameter Tuning Variations and Default Values

uncertainty in position and visa-versa. The default initial process uncertainty for all the variables was set to σ since this value closely resembles the true variance for each variable.

The measurement uncertainty is supposed to be set to σ^2, however, varying this value will vary how much the system trusts the measurements it receives. We tested this variable using values ranging from σ^4 to 1000 and kept the default at the theoretical value, σ^2.

The process uncertainty covariance describes how noise in one state variable relates to noise in the other state variables. This variable is initialized using the basic kinematic equations and, when testing, we maintain this relationship and only test the effect that a scalar on all the values has on the overall system. We test scalars ranging from σ^2 to 1000, using a scalar of 1 as the default.

$$P = \begin{bmatrix} P_a & 0 & 0 & 0 & 0 & 0 \\ 0 & P_a & 0 & 0 & 0 & 0 \\ 0 & 0 & P_v & 0 & 0 & 0 \\ 0 & 0 & 0 & P_v & 0 & 0 \\ 0 & 0 & 0 & 0 & P_p & 0 \\ 0 & 0 & 0 & 0 & 0 & P_p \end{bmatrix}$$

Figure 4.1: Model used to tune the initial process uncertainty matrix

Chapter 5

RESULTS

The main goal of the simulation experiments is to evaluate the effect each parameter has on the overall accuracy of the filter. The overall accuracy of the filter is highly influenced by the accuracy of the measurements, which means the filter's performance varies between improving and worsening the measured position depending on the generated random numbers. Because of this variability, each experiment consists of 100 simulations and compares how the mean measurement and filter error across all 100 simulations changes over time. We additionally use the simulation number as the seed to our random number generator to ensure each simulation is unique and repeatable. We first evaluate the effect of each of the previously stated parameters and then we evaluate what we consider the *best* permutation of these parameters.

5.1 Results of Parameter Tuning

To determine the effect one parameter has on the overall system, we run a single experiment for each value we would like to test while keeping the other parameters constant at an optimal value. We then compare the mean filter error of each experiment with the measurement mean error. Recall that, in our system, the white noise from the y-axis accelerometer does not affect the uncertainty in acceleration readings from the x-axis accelerometer and visa-versa. This means that we only need to evaluate the filter error along one axis to perform an overall evaluation of this Kalman Filter. The sections below describe our findings when tuning acceleration, velocity, type of sensor noise, initial process uncertainty, measurement uncertainty, and process uncertainty covariance.

5.1.1 Acceleration

The initial experiment tested values of $0.000001ms^{-2}$, $0.001ms^{-2}$, and $0.1ms^{-2}$ for acceleration (Figure 5.1). Figure 5.1 plots error, in meters from true position, against time in seconds. That figure shows that when using filtering accelerations up to and including $0.001ms^{-2}$, the filter's error closely matches the measurement error. An acceleration of $0.1ms^{-2}$ resulted in a significant increase in filter error. In order to better evaluate the behavior of the smaller acceleration values, as well as to find the acceleration that causes the filter to perform worse than measurements, we ran this experiment again, using values closer to $0.001ms^{-2}$.

Figure 5.2 is the result of this second experiment. We can see that the filter's error is slightly less than the measurement error for values up to and including $0.001ms^{-2}$ but the filter begins to perform worse at $0.005ms^{-2}$. One can also see that, although all the filter errors increase with respect to time, the filter error associated with an acceleration of $0.005ms^{-2}$ seems to increase at a higher rate than the measurement error or other filter errors. This idea that a higher acceleration causes a higher rate of change in filter error is further proved by the $0.1ms^{-2}$ acceleration series in Figure 5.1 and can best be explained by the fact that position also changes at a higher rate as acceleration increases. This higher rate of change in position therefore degrades the accuracy of the filter being evaluated.

5.1.2 Initial Velocity

When tuning the system's initial velocity, acceleration is kept constant at $0ms^{-2}$. Therefore, we expect that the results of the initial velocity experiment would all be the same as the results of the $0ms^{-2}$ acceleration results. Figure 5.3, which shows the results of the initial velocity experiment, confirms are expectations since all the different velocity series in the graph are coincidental.

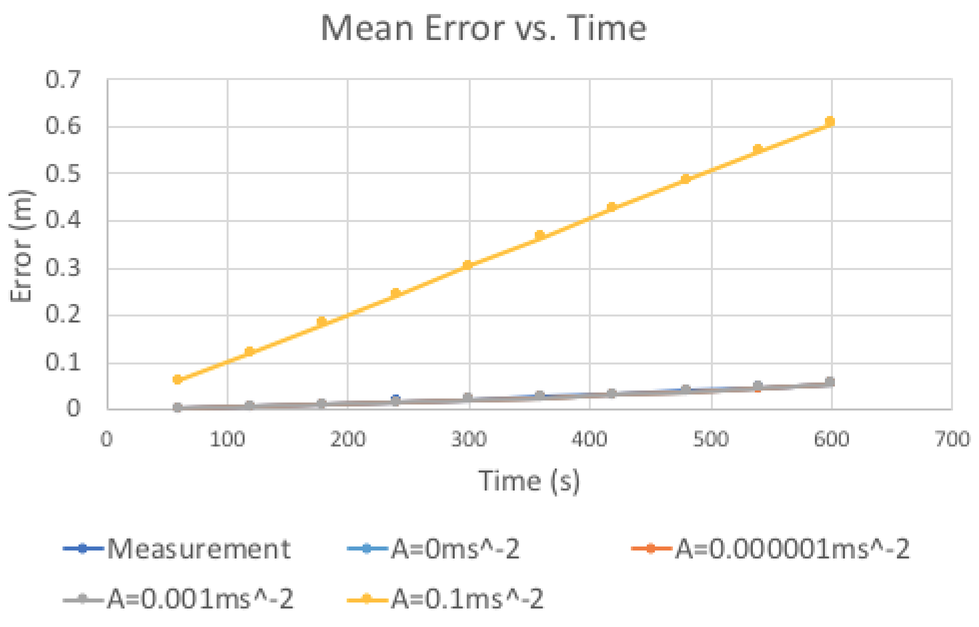

Figure 5.1: Results of Initial Acceleration Experiment

(All series coincide except $A = 0.1ms^{-2}$)

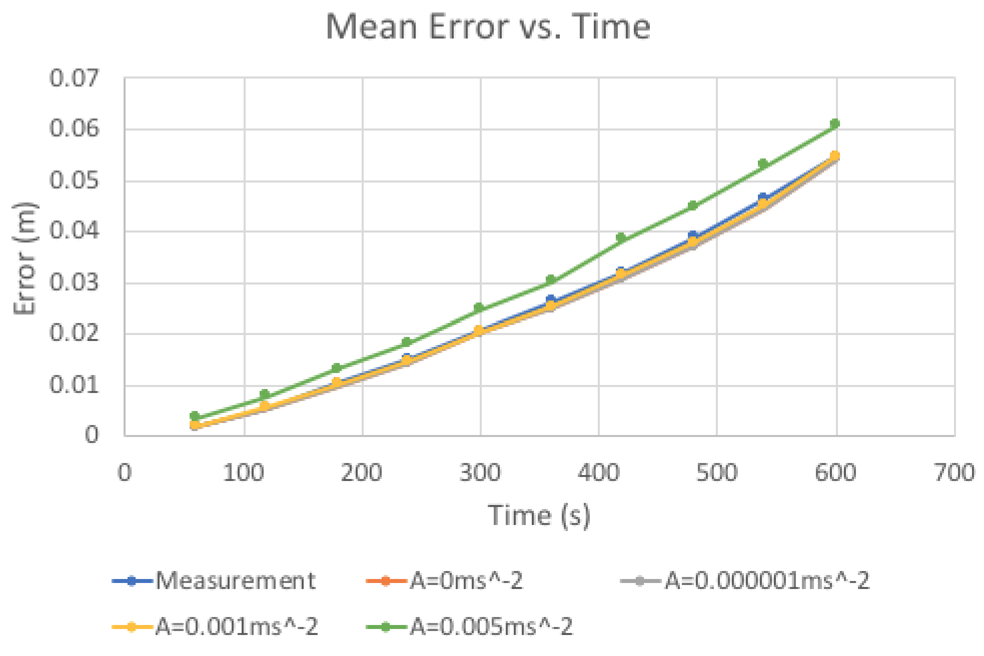

Figure 5.2: Results of Fine Tuned Acceleration Experiment

(All series except $A = 0.005ms^{-2}$ are mostly coincidental)

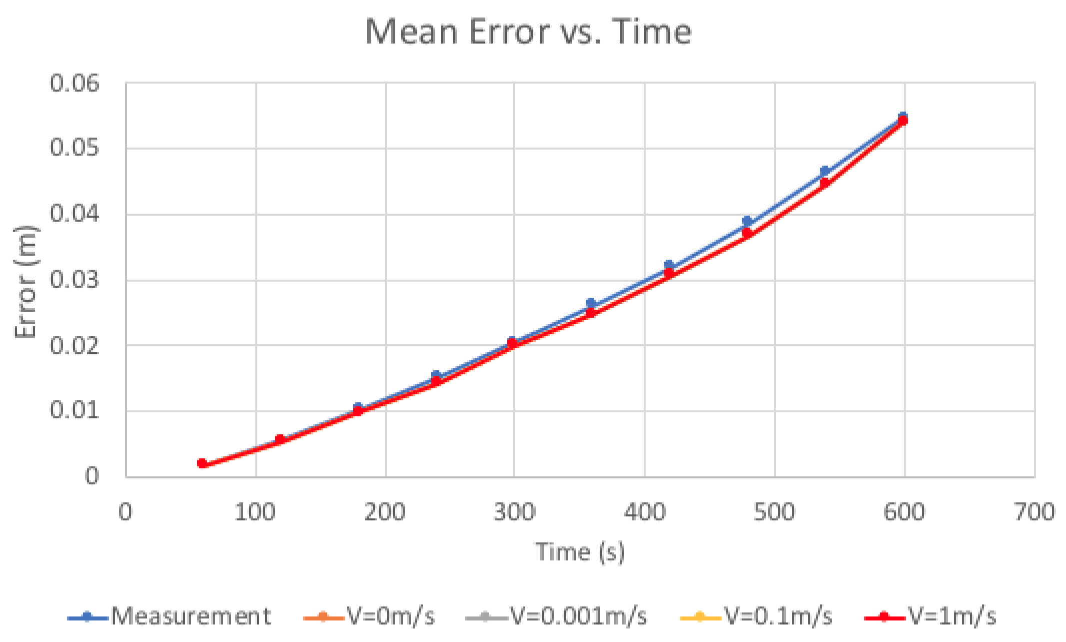

Figure 5.3: Results of Velocity Experiment

(All series coincide except for the Measurement series)

5.1.3 Type of Sensor Noise

The sensor we are modeling exhibits uniformly distributed white noise, however the Kalman Filter assumes Gaussian noise. Figure 5.4 shows how the type of sensor noise affects the cumulative position error. A normally distributed sensor noise results in a higher measurement error in position than a uniformly distributed sensor noise. This seems counter-intuitive since a normally distributed sensor noise should, in theory, imply that measurements are generally closer to the true value than a uniformly distributed sensor noise. However, a normally distributed sensor noise will still have some outliers that lie far away from the true value, whereas a uniformly distributed sensor noise is guaranteed to have an upper bound on the distance from the noisy measurement and the true value. At the 600 second mark in Figure 5.4, it can be seen that the difference between filter error and measurement error is greater when the sensor noise is normal than when the sensor noise is uniform.

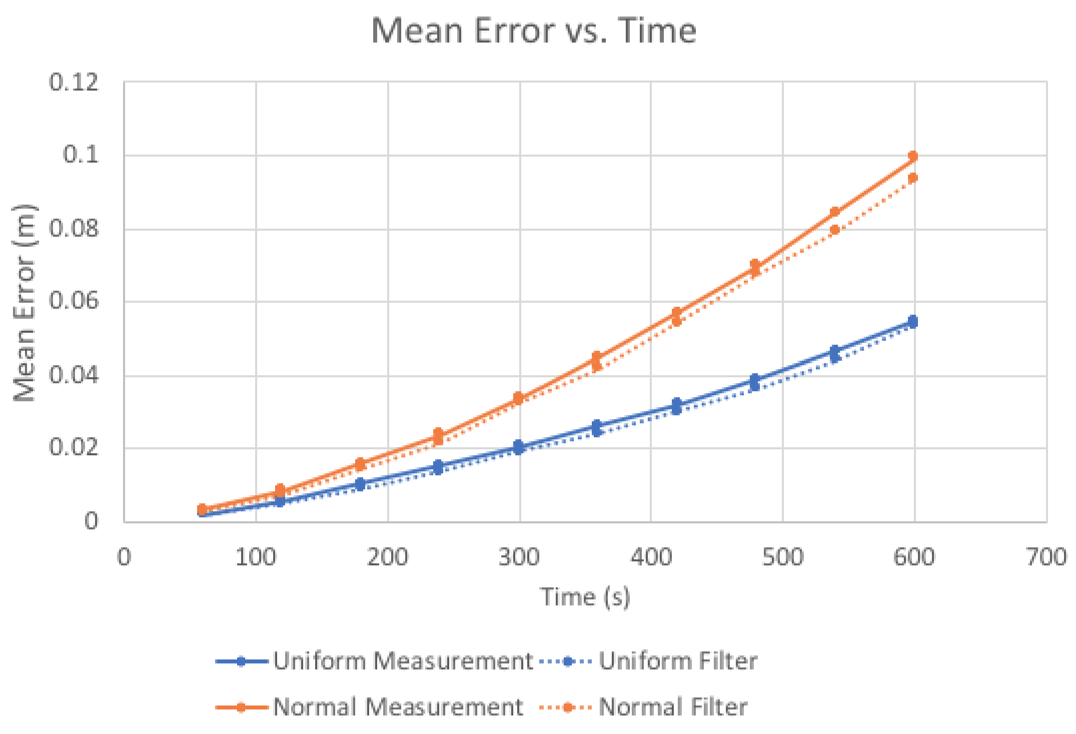

Figure 5.4: Results of Sensor Noise Type Experiment

5.1.4 Initial Process Uncertainty

Similar to the velocity experiment, the initial process uncertainty has no effect on the filter's accuracy as seen in Figure 5.5. All the experimental series are coincidental and are the same as the zero acceleration series in Figure 5.2. An explanation for why the initial process uncertainty has no effect on the system is that this variable is updated at every single timestep. This variable will quickly converge no matter how it is initialized and Kohanbash [9] states the initial value of this variable is not important.

5.1.5 Measurement Uncertainty

Varying measurement uncertainty yielded the most interesting results from these experiments. Figure 5.6 shows the results of our initial experiment, in which it can be

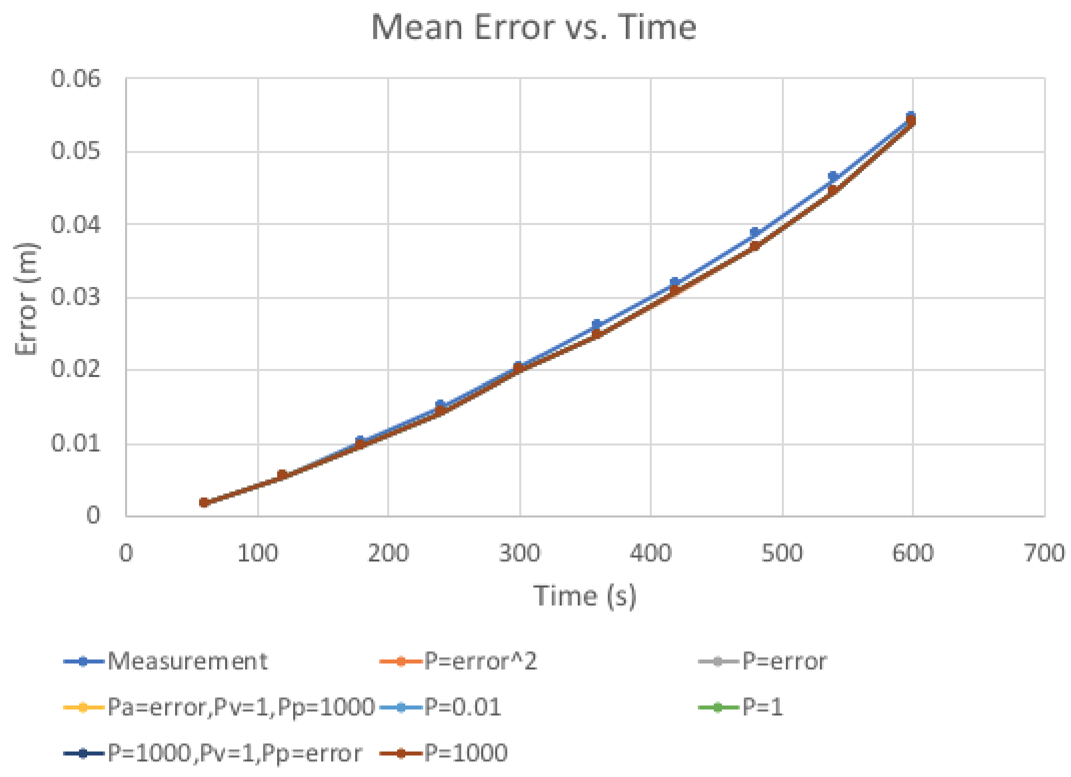

Figure 5.5: Results of Initial Process Uncertainty Experiment

(All series coincide except the Measurement series) seen that a low measurement uncertainty causes the filter error to closely match the measurement error. However, as measurement uncertainty increases, filter error decreases. A measurement uncertainty value of 1000 causes the filter to exhibit almost no error at all compared to the measurement error and, although the filter error still increases, the filter error increases at a much slower rate. These results make sense since a low measurement uncertainty means the filter, after every update, is certain that measurements are mostly correct and, therefore, don't need much correction. Conversely, a high measurement uncertainty means the filter is uncertain about the sensor measurements, so the filter assumes the system is running at constant acceleration and determines position using kinematic equations. Since we assume the acceleration in our system is constant, a high measurement uncertainty improves the filter's accuracy.

These results are expected so we want to further see the effect that measurement uncertainty has on the system in the presence of a higher rate of change. We decide to test the effect that measurement uncertainty has on the filter's accuracy when the constant acceleration is higher. Figure 5.7, Figure 5.8, and Figure 5.9 repeat the initial measurement uncertainty experiment for accelerations of $0.001ms^{-2}$, $0.005ms^{-2}$, and $0.01ms^{-2}$, respectively.

Figure 5.7 shows that, at an acceleration of $0.001ms^{-2}$, a measurement uncertainty of 1000 performs slightly worse than the same uncertainty at zero acceleration and all the other measurement uncertainty values perform about the same as the zero acceleration case. At $0.005ms^{-2}$ we notice, from Figure 5.8, that all the measurement uncertainty experiments start losing accuracy; a measurement uncertainty of 1 has roughly the same accuracy as using measurements alone, anything lower than 1 has worse accuracy, and anything higher than 1 still has better accuracy. Further increasing acceleration to $0.01ms^{-2}$ reveals that all measurement uncertainty values have higher filter errors than the measurement error. Although the filter errors are

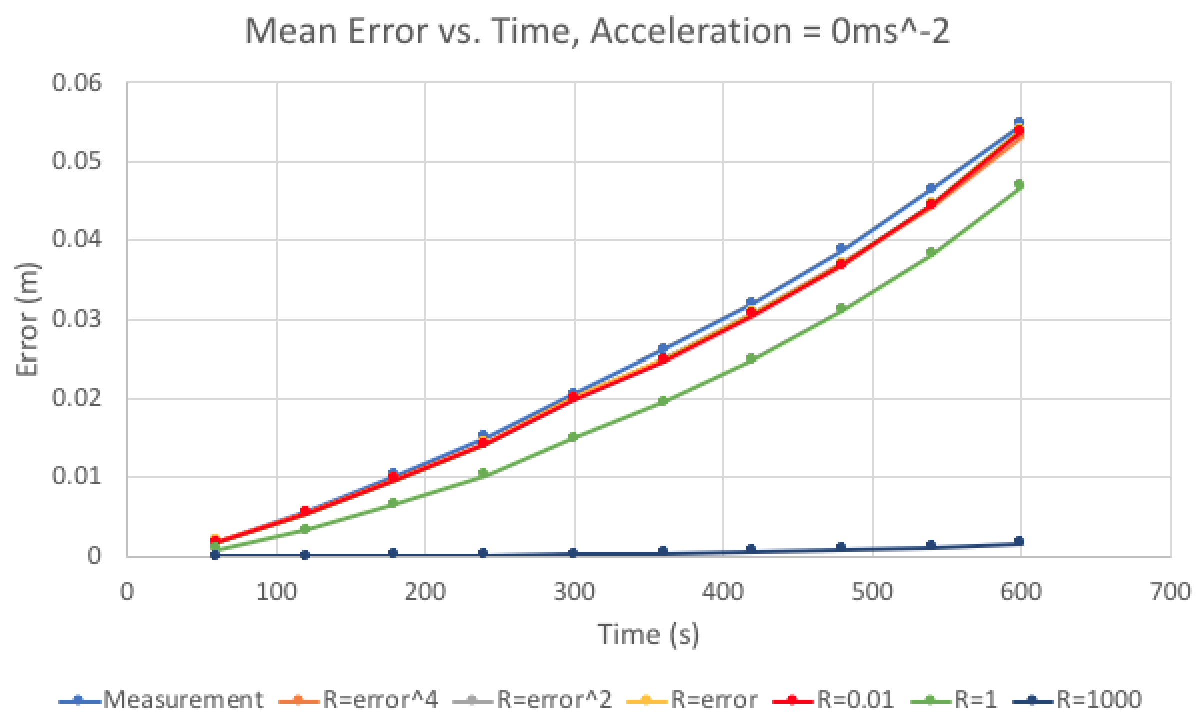

Figure 5.6: Results of Initial Measurement Uncertainty Experiment

(The series $R = error^4$, $R = error^2$, $R = error$, and $R = 0.01$ all coincide)

worse, it can be seen in Figure 5.9 that the measurement error grows at higher rate than the filter error for all measurement uncertainties. This difference in growth rate suggests that running the simulation for more than 10 minutes may result in lower filter errors than measurement errors. We therefore decided to evaluate our best parameter permutation with an hour long simulation instead of a 10 minute long simulation. The results of these experiments are detailed in Section 5.2 and can be seen in Figure 5.12, Figure 5.13, Figure 5.14, and Figure 5.15.

5.1.6 Process Uncertainty Covariance

Figure 5.10 illustrates the results from our initial process uncertainty covariance experiments. We can see from the graph that a lower process uncertainty covariance results in the filter exhibiting less error than higher process uncertainty covariance,

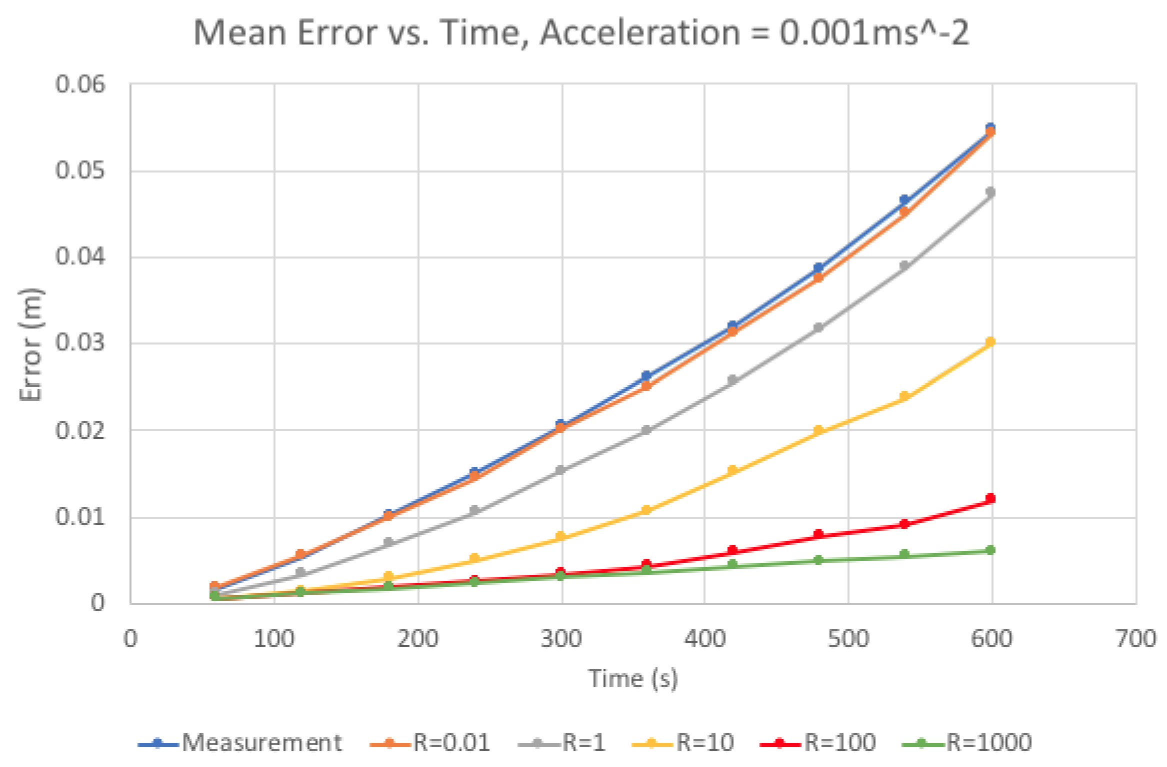

Figure 5.7: Results of Measurement Uncertainty Experiment at $0.001ms^{-2}$

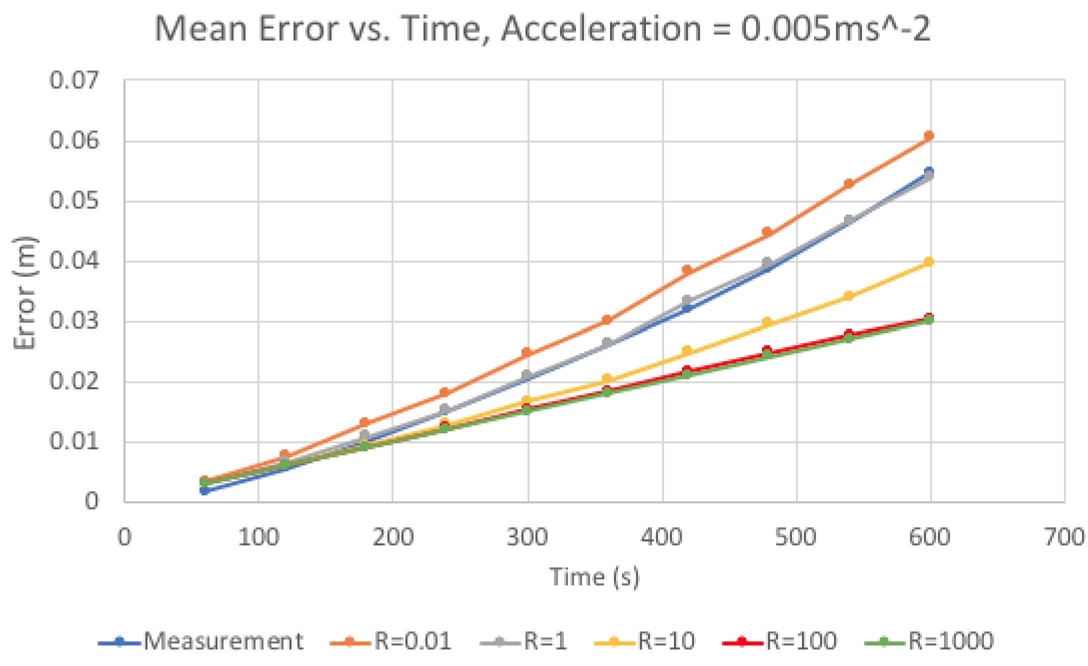

Figure 5.8: Results of Measurement Uncertainty Experiment at $0.005ms^{-2}$

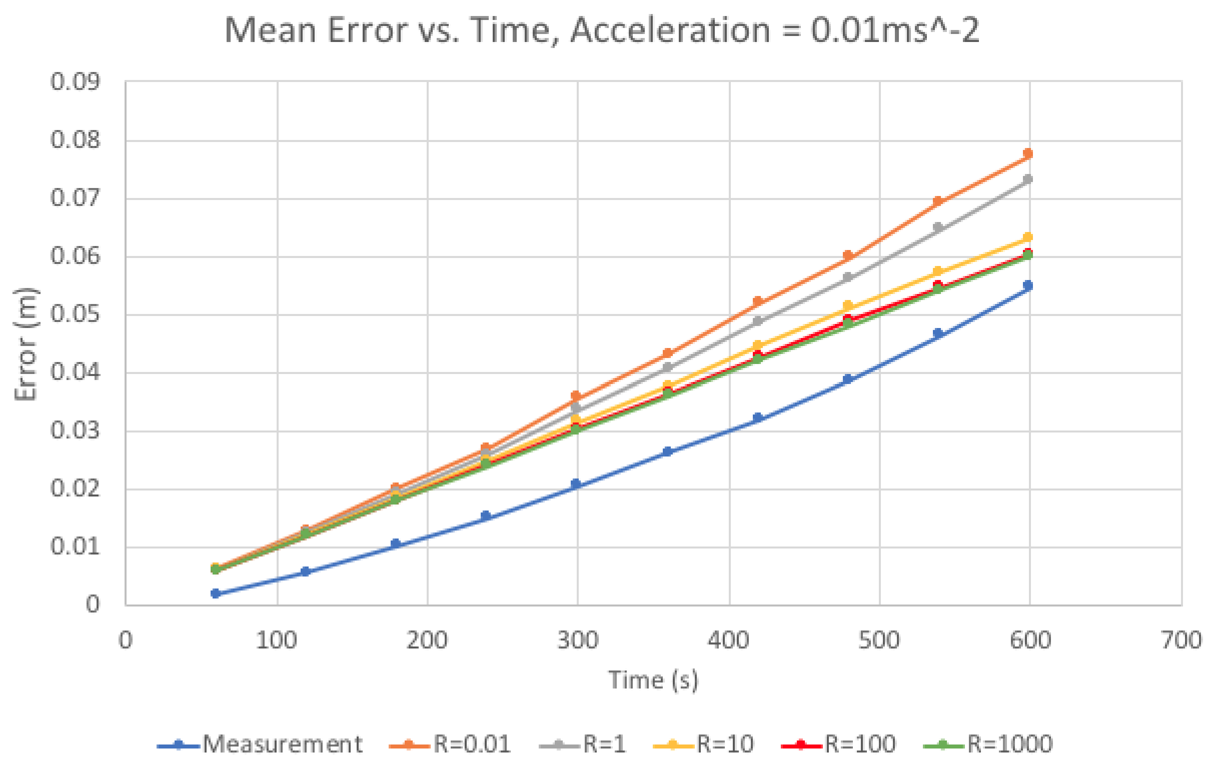

Figure 5.9: Results of Measurement Uncertainty Experiment at $0.01ms^{-2}$

although the highest covariance tested was still slightly more accurate than direct sensor measurements. A smaller covariance should result in less filter error because this covariance is added to the process uncertainty estimate at every timestep. Thus, a smaller covariance means the filter is more certain that the process will follow its state transition equations (the F matrix), whereas a larger covariance means the filter is unsure whether the process will follow the state transition equations or deviate from them.

Naturally, we then want to test the limit of how small the process uncertainty covariance can be while still decreasing the filter error. The smallest scale value we tested in the initial experiment was σ^2, so, for the secondary experiment, we tested with the following values: σ^2, σ^4, σ^{16}, and 0. Figure 5.11 illustrates the results of this secondary experiment, in which all the series are coincidental; thus, making the covariance smaller than σ^2 did not improve or worsen accuracy. The reason a covariance scale smaller than σ^2 didn't improve accuracy is most likely due to the fact that the standard deviation of the sensor's white noise we are modeling is relatively small and a covariance value less than the square of this standard deviation is much closer to 0 than the actual measurement error.

5.2 Results of *Best* Parameter Permutation

When choosing which values to select for the best parameter permutation, we first address velocity and initial process uncertainty. From experimentation, we find that both velocity and initial process uncertainty have no effect on the system, so we leave these parameters at their default value. The process uncertainty covariance is found, through experimentation, to yield the same filter accuracy at scales between 0 and σ^2 and worse filter accuracy at higher scales, so we use σ^2 to scale the process uncertainty covariance. Finally, since the sensor we are modeling exhibits uniformly distributed

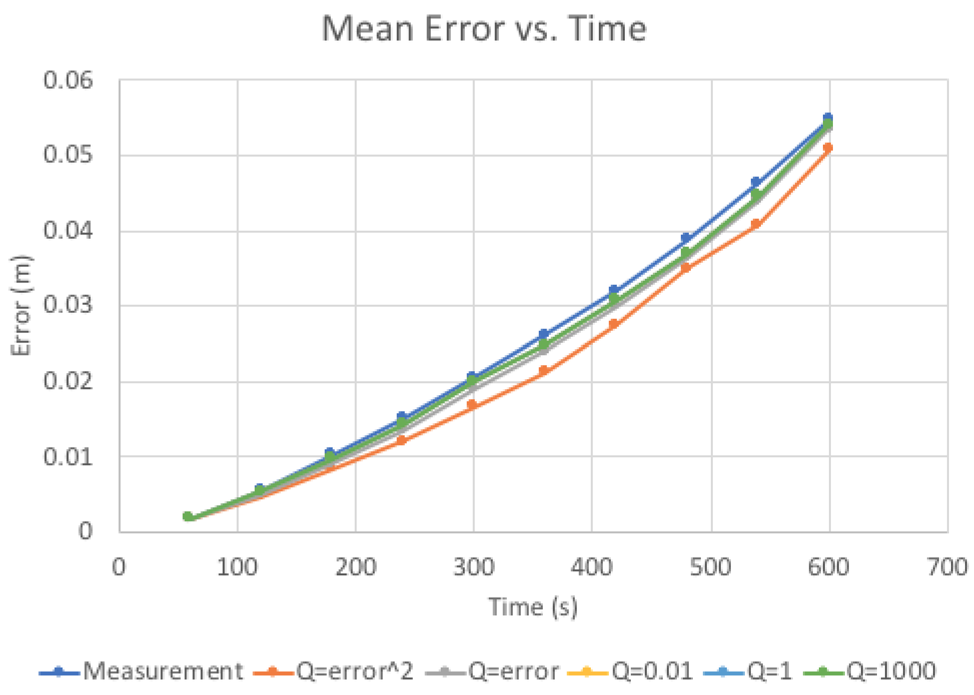

Figure 5.10: Results of Initial Process Uncertainty Covariance Experiment

(The series $Q = error$, $Q = 0.01$, $Q = 1$, and $Q = 1000$ are all coincidental)

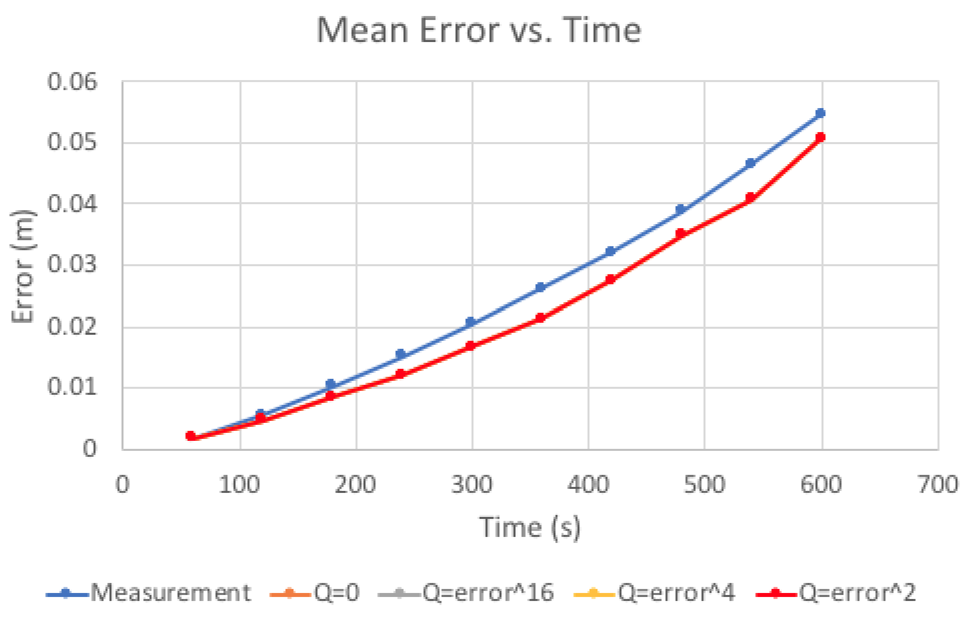

Figure 5.11: Results of Second Process Uncertainty Covariance Experiment

(All series are coincidental except the Measurement series) white noise, we use the uniformly distributed sensor noise as well.

The last two parameters – acceleration and measurement uncertainty – exhibited the most variability so picking which value is best was difficult. Therefore, we opt to continue experimentation with varying acceleration and measurement uncertainty values. We extend the same experiments to an hour instead of 10 minutes, we use a process uncertainty covariance scale of σ^2, and we run an additional experiment at an acceleration of $0.1ms^{-2}$ to see how a large measurement uncertainty responds to a much larger rate of change in position. Figure 5.12, Figure 5.13, Figure 5.14, and Figure 5.15 show the results of these final experiments for accelerations of $0.001ms^{-2}$, $0.005ms^{-2}$, $0.01ms^{-2}$, and $0.1ms^{-2}$, respectively.

From these experiments, we confirm our suspicion that the filter error, when using a high measurement uncertainty, will increase at a slower rate than the measurement error, which means that although the filter error was higher than the measurement error after 10 minutes, the filter error was actually lower than the measurement error after an hour. We can see from the graphs that, as acceleration increases, the filter's accuracy will still decrease regardless of the measurement uncertainty used. At $0.01ms^{-2}$ (Figure 5.14), which is a typical acceleration of the OpenROV Underwater Drone, a measurement uncertainty of 0.01 could not correct sensor measurements. However, a measurement uncertainty of 10 was able to reduce measurement error by about $0.1m$ and a measurement uncertainty of 1000 reduced measurement error by roughly $0.4m$. An interesting observation from these experiments is that the filter's error seems to grow linearly regardless of the measurement uncertainty or acceleration. This linear trend is the most clear in Figure 5.15, at an acceleration of $0.1ms^{-2}$.

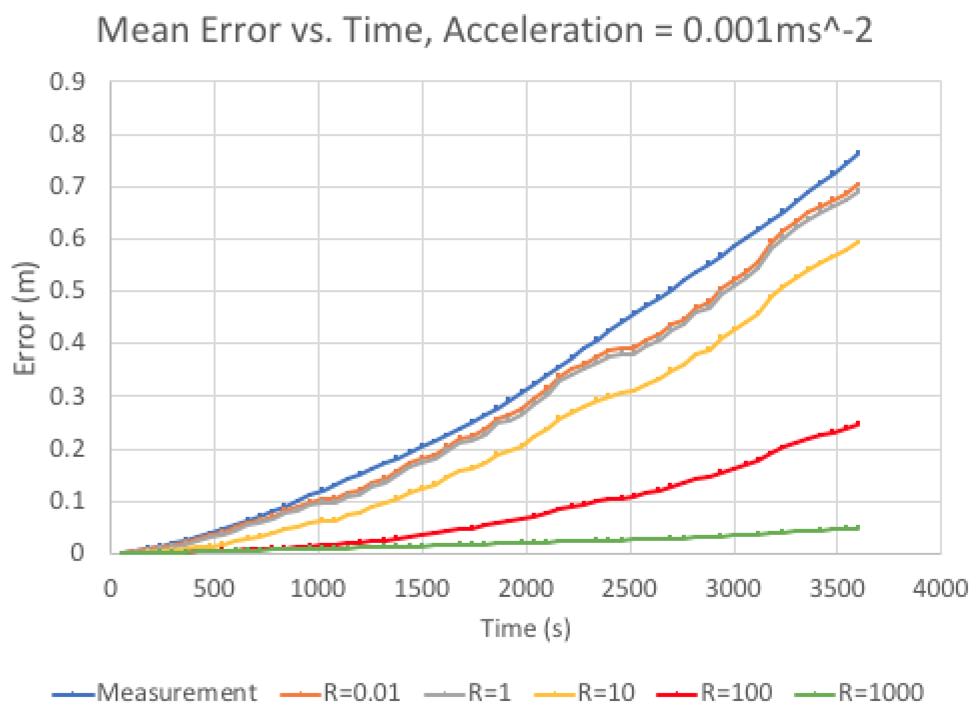

Figure 5.12: Results of Best Parameter Permutation at $0.001ms^{-2}$

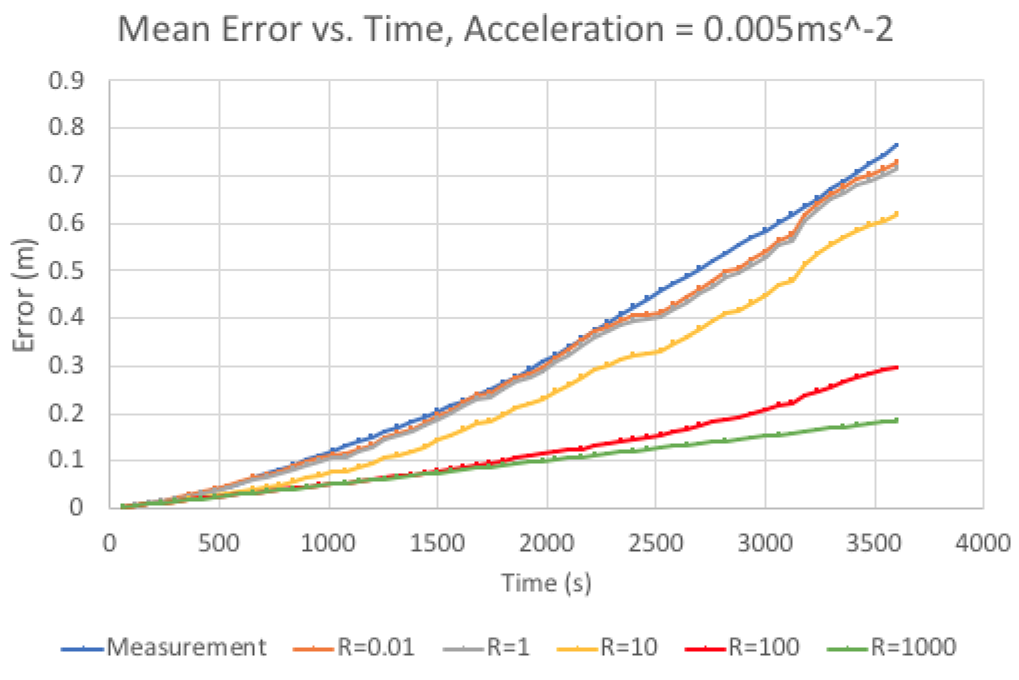

Figure 5.13: Results of Best Parameter Permutation at $0.005ms^{-2}$

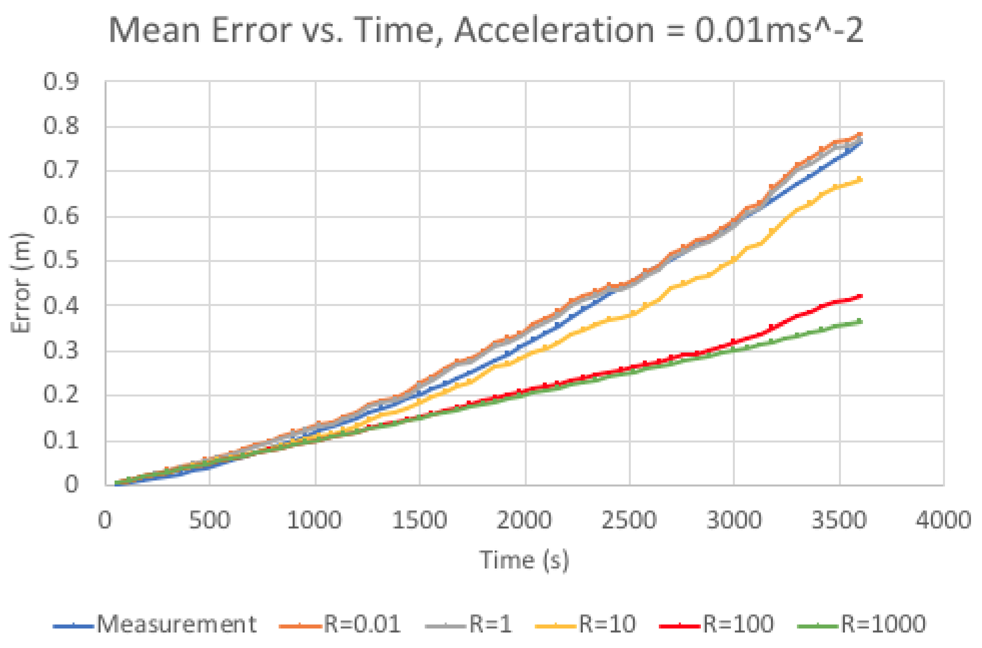

Figure 5.14: Results of Best Parameter Permutation at $0.01ms^{-2}$

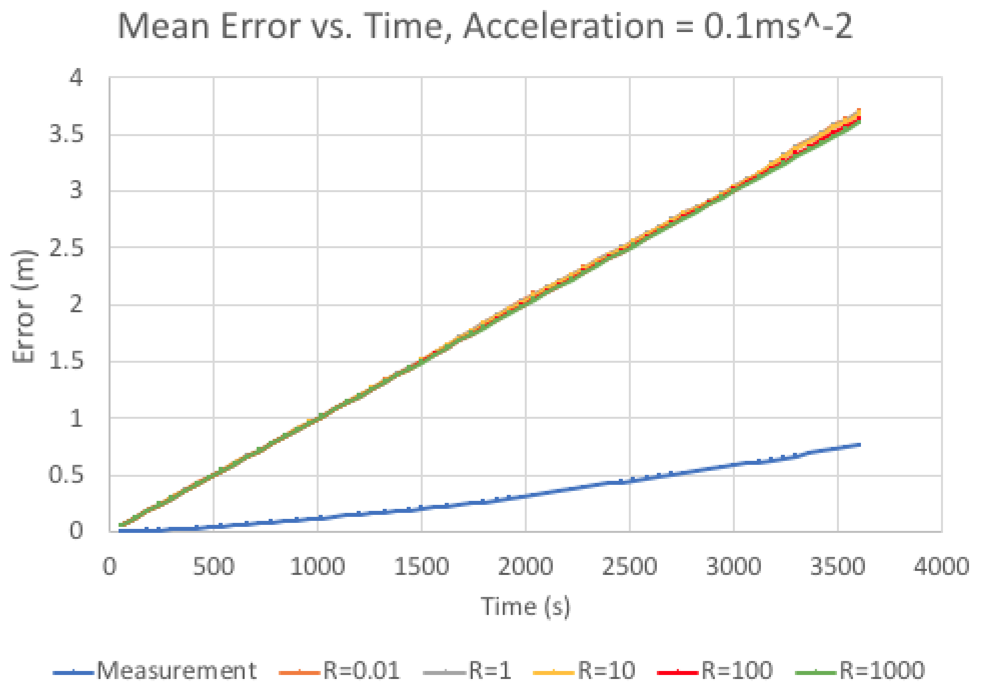

Figure 5.15: Results of Best Parameter Permutation at $0.1ms^{-2}$

(All series are coincidental except the Measurement series)

Chapter 6

CONCLUSION AND FUTURE WORK

We started by researching various solutions to the problem of autonomous underwater localization and found three general approaches: dead reckoning combined with state estimation, multi-AUV cooperative localization, and geophysical feature extraction combined with a SLAM algorithm. Our research also found that a combination of approaches is, in fact, the best approach. In this paper, we chose to start solving the problem of autonomous underwater localization by implementing a kalman filter to estimate the true state of a dead reckoning system by correcting for linear errors in accelerometer readings. We then tested the accuracy of this filter by comparing the filter error with the error associated with unfiltered dead reckoning.

We first tested the effect that various parameters in the filter have on the filter's overall accuracy and make a few findings. First, the velocity and initial process uncertainty of the system have no effect on accuracy. Second, measurement error when using a sensor with uniformly distributed noise was less than the measurement error when using a sensor with normally distributed noise, although the filter corrected normally distributed sensor noise better than uniformly distributed sensor noise. Third, a small process uncertainty covariance improved the accuracy of the filter, although anything smaller than σ^2 did not improve or worsen the accuracy.

Acceleration and measurement uncertainty had the largest effects on the filter's accuracy. A higher rate of change in position caused the accuracy to quickly degrade, even with a high measurement uncertainty. A high measurement uncertainty proved to greatly improve the accuracy of the filter since the system is linear but, as previously mentioned, too high of an acceleration would still cause even a very large measurement uncertainty to exhibit more filter error than dead reckoning alone.

Fortunately, the AUV we model will not typically accelerate faster than $0.01ms^{-2}$ so a high measurement uncertainty can enable the filter to sufficiently correct measurements.

The problem of autonomous underwater localization, specifically for the OpenROV Underwater Drone, is still far from solved. The following tasks represent the future work for this project and are necessary in order to achieve a complete, accurate, and deployable localization solution:

Extend the kalman filter - To deal with a nonlinear system (like the ocean) the kalman filter implemented in this paper must be extended. This extended kalman filter (EKF) replaces the linear F matrix with a Jacobian matrix that linearizes the nonlinear system.

Test the OpenROV vehicle dynamics - Test the dynamics associated with how control inputs translate into forces on the vehicle and how the vehicle dynamics respond to actual ocean conditions, specifically for the OpenROV Underwater Drone. This task will involve installing an IMU onto an OpenROV, running different testing scenarios while collecting and analyzing the data from the IMU, and finally augmenting the kalman filter to include these findings.

Combine the state estimation approach with a SLAM approach - Add features detected from sonar/camera data to the system state vector of the implemented filter and use these features to develop a SLAM algorithm, which utilizes the filter's estimation techniques. This task will require a working feature detection system but will additionally generate maps of the environment in addition to localizing position.

Implement an on-board filter - In order for the OpenROV to truly be autonomous, it must be able to run a state estimating filter on-board. This task involves reducing the complexity of the filter so that it may run to completion between each measurement reading and do so on a limited processor. This task will also involve implementing the simplified filter on an FPGA, Raspberry Pi, etc. and installing the hardware onto the OpenROV.

Perform real-world experiments - A major challenge of a real world experiment is that the true trajectory of an AUV is impossible to obtain, yet the true trajectory of an AUV is necessary to evaluate the accuracy of position sensors and an autonomous localization algorithm. We discuss two types of real world experiments: bounded and unbounded. A bounded experiment would be one conducted in a pool, where walls serve as physical boundaries and water conditions are predictable. In a bounded experiment, an exact trajectory is obtainable, although not perfect. One can devise methods, such as using fixed markers on the pool floor or surface, to provide an estimate of trajectory. An unbounded experiment in this application would take place in the ocean, where physical boundaries are limited and water conditions are highly unpredictable. In this situation, setting up markers to track trajectory requires an enormous amount of prior effort and is unreliable since markers are affected by underwater currents and an operator will have more difficulty navigating through these markers in the ocean than in a pool. To overcome the inability to track true trajectory in an unbounded experiment, another evaluation method is considered: tracking how much drift occurs. The problem we are trying to solve is the inherent drift in position found in dead reckoning alone. We can set up an experiment where an AUV follows a loosely defined trajectory in the ocean, starting at one point and ending at another. Since we know the true start and end locations, we can compare the true end location to the measured and filtered end location in order to validate an autonomous localization algorithm. The method of comparing only end locations can also be used in the bounded experiment, however this method will only yield meaningful results when the trajectory is fairly long, which can be achieved by moving in a circle or lawn mower shape within a bounded environment.

BIBLIOGRAPHY

[1] Bosch Sensortec. *Intelligent 9-axis absolute orientation sensor*, 11 2014. Rev. 1.2.

[2] E. Delgado and A. Barreiro. Sonar-based robot navigation using nonlinear-robust kalman filter. In *2001 European Control Conference (ECC)*, pages 1056–1061, Sept 2001.

[3] A. Elibol, N. Gracias, and R. Garcia. Augmented state-extended kalman filter combined framework for topology estimation in large-area underwater mapping. *Journal of Field Robotics*, 27(5):656–674, 2010.

[4] R. Eustice, O. Pizarro, and H. Singh. Visually augmented navigation for autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, 33(2):103–122, 2008.

[5] R. M. Eustice. Large-area visually augmented navigation for autonomous underwater vehicles. 2005.

[6] C. Forney, E. Manii, M. Farris, M. Moline, C. Lowe, and C. Clark. Tracking of a tagged leopard shark with an auv: Sensor calibration and state estimation. *2012 IEEE International Conference on Robotics and Automation*, 2012.

[7] T. Fossen and T. Perez. Marine systems simulator - mss, 2018.

[8] M. Haibo, Z. Liguo, and C. Yangzhou. Recurrent neural network for vehicle dead-reckoning. *Journal of Systems Engineering and Electronics*, 19(2):351–355, 2008.

[9] D. Kohanbash. Kalman filtering a practical implementation guide (with code!), 2014.

[10] D. Kroetsch and C. Clark. Towards gaussian multi-robot slam for underwater robotics. 08 2005.

[11] J. Li and J. Zhang. Research on the algorithm of multi-autonomous underwater vehicles navigation and localization based on the extended kalman filter. *2016 IEEE International Conference on Mechatronics and Automation*, 2016.

[12] P. A. Miller, J. A. Farrell, Y. Zhao, and V. Djapic. Autonomous underwater vehicle navigation. *IEEE Journal of Oceanic Engineering*, 35(3):663–678, 2010.

[13] L. Paull, S. Saeedi, M. Seto, and H. Li. Auv navigation and localization: A review. *IEEE Journal of Oceanic Engineering*, 39(1):131–149, 2014.

[14] M. Prats, J. Perez, J. J. Fernandez, and P. J. Sanz. An open source tool for simulation and supervision of underwater intervention missions. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[15] W. Senke. Applications of autonomous underwater vehicles (auvs) in ocean mining exploration. *2013 OCEANS - San Diego*, pages 1–3, 2013.

[16] L. Stutters, H. Liu, C. Tiltman, and D. Brown. Navigation technologies for autonomous underwater vehicles. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):581–589, 2008.

[17] W. Wang and C. M. Clark. Modeling and simulation of the videoray pro iii underwater vehicle. *OCEANS 2006 - Asia Pacific*, 2006.

[18] Wang, Wei. Autonomous control of a differential thrust micro rov. Master's thesis, 2007.

[19] L. Whitcomb, D. R. Yoerger, H. Singh, and J. Howland. Advances in underwater robot vehicles for deep ocean exploration: Navigation, control, and survey operations. *Robotics Research*, pages 439–448, 2000.

[20] O. J. Woodman. An introduction to inertial navigation, 2007.

[21] R. B. Wynn, V. A. Huvenne, T. P. Le Bas, B. J. Murton, D. P. Connelly, B. J. Bett, H. A. Ruhl, K. J. Morris, J. Peakall, and D. R. e. a. Parsons. Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience. *Marine Geology*, 352:451–468, 2014.