A Localization Solution for an Autonomous Vehicle in an Urban Environment

Jonathan Michael Webster

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

Dr. Charles F. Reinholtz, Chairman
Alumni Distinguished Professor of Mechanical Engineering and Engineering Education

Dr. Alfred L. Wicks, Co-Chairman
Associate Professor of Mechanical Engineering

Dr. Dennis W. Hong
Assistant Professor of Mechanical Engineering

December 3, 2007
Blacksburg, Virginia

A Localization Solution for an Autonomous Ground Vehicle in an Urban Environment

Jonathan Michael Webster

ABSTRACT

Localization is an essential part of any autonomous vehicle. In a simple setting, the localization problem is almost trivial, and can be solved sufficiently using simple dead reckoning or an off-the-shelf GPS with differential corrections. However, as the surroundings become more complex, so does the localization problem. The urban environment is a prime example of a situation in which a vehicle's surroundings complicate the problem of position estimation. The urban setting is marked by tall structures, overpasses, and tunnels. Each of these can corrupt GPS satellite signals, or completely obscure them, making it impossible to rely on GPS alone. Dead reckoning is still a useful tool in this environment, but as is always the case, measurement and modeling errors inherent in dead reckoning systems will cause the position solution to drift as the vehicle travels eventually leading to a solution that is completely diverged from the true position of the vehicle.

The most widely implemented method of combining the absolute and relative position measurements provided by GPS and dead reckoning sensors is the Extended Kalman Filter (EKF). The implementation discussed in this paper uses two Kalman Filters to track two completely separate position solutions. It uses GPS/INS and odometry to track the Absolute Position of the vehicle in the Global frame, and simultaneously uses odometry alone to compute the vehicle's position in an arbitrary Local frame. The vehicle is then able to use the Absolute position estimate to navigate on the global scale, i.e. navigate toward globally referenced checkpoints, and use the Relative position estimate to make local navigation decisions, i.e. navigating around obstacles and following lanes.

This localization solution was used on team VictorTango's 2007 DARPA Urban Challenge entry, Odin. Odin successfully completed the Urban Challenge and placed third overall.

Acknowledgments

This thesis, as well as my entire college career, would not have been possible without the help of several people. First, I would like to thank my family for their love and prayers throughout the years. Thank you for always expecting the best of me and never letting me settle for less. I would also like to thank Danielle Boltersdorf for her love and support as I worked on this project, and for spending late nights in the lab with me to help me keep my sanity.

I would like to especially thank Joseph Putney and Eric Bonnini for inviting me into their highly exclusive study group. This was a turning point for me in my undergraduate career and without them I may not have made it through the undergraduate ME curriculum, much less the graduate one. Thank you to Brett Leedy for his leadership on the Grand Challenge project and for making me the new team "expert" on everything I asked him about. Thank you to Andrew Bacha and Ruel Faruque for all of their LabVIEW advise over the years. Thank you to the rest of the members of the 2005 Grand Challenge team and team Victor Tango for making the smartest, cleanest, most well designed vehicles I could ever hope to work on and for making the experience as a whole so rewarding.

Finally I would like to thank my committee members. Thank you to Dr. Reinholtz for giving me the chance to get involved in such an exciting field of research. Thank you to Dr. Wicks for continuing to challenge me day to day throughout my graduate career. Thank you to Dr. Hong for his advise on this thesis and for having such confidence in me and in our Urban Challenge team in general. Working with all of these people has made me a better engineer, and knowing them has made me a better

Contents

Acronyms

AGV Autonomous Ground Vehicle

APF Absolute Position Filter

CCD Charge Coupled Device

CEP Circular Error Probable

DARPA Defense Advanced Research Projects Agency

DBW Drive-by-Wire

DRAC Drivable Area Coverage

DOP Dynamic Obstacle Predictor

DUC DARPA Urban Challenge

ECU Electronic Control Unit

EKF Extended Kalman Filter

GDOP Geometric Dilution of Precision

GPS Global Positioning System

GUI Graphical User Interface

IMU Inertial Measurement Unit

INS Inertial Navigation System

JAUS Joint Architecture for Unmanned Systems

LAN Local Area Network

LIDAR Light Detecting and Ranging

MDF Mission Definition File

MOUT Military Operations on Urban Terrain

NQE National Qualifying Event

OC Object Classification

RLP Report Lane Position

RNDF Route Network Definition File

RPF Relative Position Filter

SBAS Satellite-Based Augmentation System

SFM Structure From Motion

SPAN Sychronized Position Attitude & Navigation

UAV Unmanned Aerial Vehicle

UFE Urban Challenge Final Event

UGV Unmanned Ground Vehicle

UTM Universal Transverse Mercador

VI Virtual Instrument

VM Virtual Machine

List of Figures

Chapter 1

Introduction

This thesis presents the localization solution developed for team *V*ictor *T*ango's 2007 DARPA Urban Challenge entry, Odin. This work was funded by the Defense Advanced Research Projects Agency (DARPA), and although it was designed specifically for Odin and for the Urban Challenge, the algorithm can be applied to any autonomous platform.

1.1 Thesis Overview

Localization is a fundamental element of any autonomous vehicle. There are many techniques for localizing a vehicle, and each technique has unique advantages and disadvantages. A logical solution when attempting to develop a robust localization algorithm is to blend different techniques in a way that takes advantage of the strengths of the individual methods, and mitigates their weaknesses. Since the early 1960's, the tool that engineers have chosen to accomplish this has been the Kalman filter. The Kalman filter provides

a means of optimizing measurements to achieve the best position estimate. This thesis will describe how the Kalman filter can be used to take advantage of different localization techniques to provide a robust localization solution and overcome challenges presented by even the most difficult environments.

Chapter One describes the motivation for the development of this localization software. This chapter discusses the DARPA Urban Challenge, and explains why it is necessary to have more sophisticated localization to be successful in this competition. Chapter Two will then describe the platform on which this localization solution has been implemented. Chapter Two will also present the vehicle's software architecture to explain how the localization software will interact with other software modules.

Chapter Three will begin by clearly defining the localization problem, and will then describe various methods available for autonomous vehicle applications. For each method, this chapter will discuss strengths and weaknesses, as well as sources of error. Chapter Four will present the linear Kalman filter and the Extended Kalman filter. These two chapters give the background necessary for the discussion of the localization solution presented in this thesis.

Chapters Five and Six will describe the measurements used in this localization solution and the position filters developed to blend the measurements. These chapters will follow the development of the software from modeling to implementation and testing.

Finally, Chapter Seven will discuss the conclusions reached after observing the performance of the localization software during the Urban Challenge, and give suggestions for future work.

1.2 The DARPA Urban Challenge

The DARPA Urban Challenge (DUC) is the third in a series of autonomous vehicle races sponsored by DARPA, the first being the 2004 DARPA Grand Challenge. The previous two challenges have both been set in a rugged desert environment. The challenge was to design an Autonomous Ground Vehicle (AGV) that could navigate itself through more than 130 miles of rough desert terrain. The inspiration for this competition was a 2001 congressional mandate that set the goal of having one third of the operational ground combat vehicles in the Armed Forces to be unmanned by 2015 [5]. The competitions were therefore a means of stimulating industry and academia to further the state of the art of AGV.

The DUC is the next phase in the advancement of AGV's. In the 2005 DARPA Grand Challenge, five teams proved that the problem of long distance autonomous navigation through rough terrain was well within the capabilities of the unmanned systems community. The DUC now addresses the problem of navigating through crowded city streets, interacting with moving traffic, avoiding dynamic obstacles and obeying the rules of the road. This problem requires far more intelligence from an AGV since it must not only perceive the world around it, but correctly interpreting what it perceives to make the appropriate decision for it's current situation. This situational awareness is what sets the DUC apart from the previous challenges.

Entry into the DUC was achieved by following one of two competition tracks, Track A or Track B. To participate in Track A, teams were required to submit a proposal to DARPA to be eligible for up to $1,000,000 in technology development funding awards. The teams selected for Track A were then required to meet a series of milestone events that evaluated the progress of the team and guaranteed adequate progress toward the goal of completing the DUC. Teams not selected for Track A were still permitted to participate in the challenge through Track B. The primary distinction between Track A teams and Track

B teams is funding. Track B teams were still required to meet a series of milestone events, including a site visit and the National Qualifying Event (NQE). Track B teams however did not receive any funding from DARPA to aid them in completing the challenge.

At the Urban Challenge Final Event (UFE), teams were given a Route Network Definition File (RNDF). The RNDF describes the entire network of roads and zones through which the vehicles must navigate during the UFE. Individual Mission Definition File (MDF)s were used to define which areas within the RNDF the vehicle was required to travel to for a particular mission. In the end each vehicle had approximately 60 miles to travel and 6 hours to complete all of the missions. Six teams completed all three missions in under six hours, with Tartan racing, Stanford, and team Victor Tango comprising the top three.

Figure 1.1: Boss, Junior, and Odin placed first, second, and third in the 2007 DARPA Urban Challenge.

Chapter 2

The Vehicle

The research for this thesis was done using Team VictorTango's entry for the DUC, Odin. The software discussed in this thesis is the localization software used by Odin in the competition. This chapter will discuss the following aspects of Odin's design: base vehicle, drive-by-wire, power systems, computing, sensors and software architecture.

2.1 The Base Platform

If anything was learned from the 2005 Grand Challenge about base vehicle design, it was that it is in a team's best interest to keep the design simple, and add as few components to the platform as possible. This means selecting a platform that has stock features that can be used in the conversion to autonomous operation. For example, the winning vehicle in the 2005 Grand Challenge, Stanley, which is based on a 2004 Volkswagen Touareg R5 TDI, used the vehicle's stock alternator to supply power to its computers and

instrumentation. In addition, the team took advantage of the Touareg's native brake and throttle-by-wire systems which greatly simplified the drive-by-wire conversion [12]. Using stock vehicle subsystems increases the reliability of the vehicle system as a whole because these subsystems have been rigorously tested to automotive industry standards.

With this in mind, team VictorTango chose the 2005 Ford Escape Hybrid as the base platform for Odin. The Escape is a compact SUV first introduced by Ford in 2001. The Escape provides enough rear storage ($0.75m^3$) to allow all of the vehicle's computers and electronics to be stored within the climate controlled interior of the vehicle. In addition, the Escape has an 11.5 meter turning radius making it more maneuverable than larger SUVs. The team chose the hybrid model because of its high voltage power system, and its natively Drive-by-Wire (DBW) throttle and shift control systems. Ford generously donated two of the 2005 Escapes to team VictorTango for use in the DUC. The purpose of the dual platforms was to allow the team to make progress in vehicle and software development simultaneously. Both vehicles were fully converted to autonomous operation, and have identical components and software. For the DUC competition, the second vehicle will served as a back-up vehicle for the entry vehicle.

2.2 Drive-By-Wire Conversion

To operate the vehicle via computer control, the throttle, braking, steering and shifting control systems had to be converted to DBW. As previously stated, the throttle and shifting controls are natively DBW; therefore no additional actuators were needed. Instead, automatic relays were added to the throttle and shifting controllers that replace the control signals sent by the accelerator pedal and shifter with simulated control signals. The system was designed so that the relays automatically return control of the vehicle to the human driver in case of an emergency.

The Escape's steering controls are not fully DBW, but they are electronically assisted rather than hydraulically. The steering assist motor provides enough torque to steer the vehicle without a human driver. The control signals sent from the steering wheel to the steering assist motor were replaced with simulated control signals, and automatic relays were installed to return control to a human driver in emergency situations.

The Escape's brake actuator is the only vehicle control system that is fully mechanical. As a result, a pedal-mounted linear actuator had to be added to the vehicle to electronically control the brakes. A brake controller was designed to control the linear actuator and switch between autonomous control and human control. The system was also designed such that additional brake effort can be added at any time by a human driver. A separate spring-actuated emergency brake was also added to fully stop the vehicle in emergency situations.

2.3 Vehicle Power System

All of Odin's electronics are powered by the Escape's stock power systems. The Escape's high-voltage power is supplied by a 300V sealed NiMH battery. A 2kW 300V to 52V DC-DC converter from V-Infinity was connected to the high-voltage system to convert the high-voltage to 48V. The DC-DC had to be slightly modified to output a nominal 48V instead of the nominal 52V it was originally designed to produce. This modification was done at V-Infinity before the team received the unit. The output of this DC-DC converter is then divided such that 80% of the power is sent to a Triplite UPS to provide redundant power to Odin's computers, and the remaining 20% of the power is sent to a 24V DC-DC converter that supplies a clean 24V to the vehicle's sensors. Any 12V systems onboard the vehicle are powered by the Escape's 12V battery located in the engine compartment.

2.4 Computing Systems

The computing systems on board Odin are one of the more uniquely designed systems on the vehicle. Odin is equipped with two HP Proliant DL140 rack mounted servers and a National Instruments cRIO-9012 real time controller. The RIO controls all low level vehicle functions such as actuating the brake, throttle and shifter as well as monitoring vehicle systems through the vehicle's CAN bus. The RIO is also cleverly mounted underneath the dashboard in the space where the glove box used to be as shown in Figure 2.1.

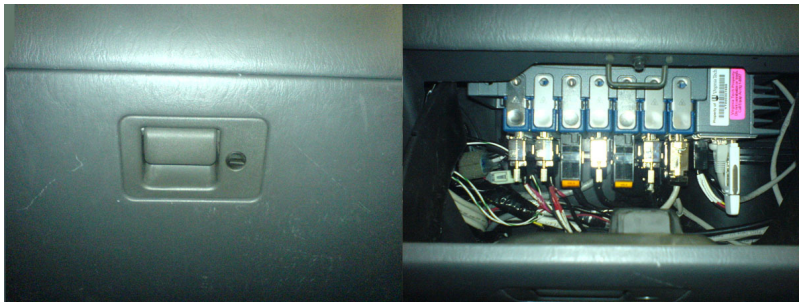

Figure 2.1: The RIO has been installed in the glove compartment space to preserve the aesthetics of the vehicle interior.

The HP servers are referred to by their nicknames Linus and Bill which respectively run Linux and Windows operating systems. Each server has dual processor sockets with at least one quad-core processor. Bill has an additional quad-core processor because Bill has the expensive processing task of reading in raw camera data. In addition, Bill is running a less efficient OS than Linus, so it was decided that Bill would get the extra chip. The servers each have 4 GB of RAM and RAID1 250 GB hard drives.

Linus is divided into four Virtual Machine (VM). The VM's, named Alpha, Bravo, Charlie, and Delta, act as independent machines running Linux OS, when in fact they are simply multiple instances of Linux running on a single platform. This architecture was chosen to give more control over resources and delegate them more efficiently. Each VM

has access to all of Linus's communications ports and other hardware. In addition, if one of the VM's needs to be shut down because of a crash or other error, this can be done without affecting the other machines.

Odin's computers are connected to a Local Area Network (LAN) via an HP Procurve 24 port switch. Linus handles all of the network management including assigning IP addresses to all of the other machines and devices.

2.5 Sensors

Odin is equipped with an array of sensors that allow it to perceive its surroundings as it moves through the DUC course. The sensor array was designed such that Odin will be able to detect obstacles on all sides, and be able to detect road coverage and lane markings in front. Sensors associated with the vehicle's position and other internal vehicle states will not be discussed in this section, and will instead be discussed in greater detail in Chapter 3.

The primary object detection sensor is the Ibeo XT Fusion system. The Fusion system consists of two Ibeo XT Light Detecting and Ranging (LIDAR) units and an Electronic Control Unit (ECU). The system has an advertised range of 200 meters and a horizontal field of view of 220 degrees. The Ibeo XT's are mounted to the front corners of the Escape as shown in Figure 2.2. The units themselves are protected by 1/4" wall 1.5" diameter steel tubing. In addition, the mounts have been designed to break away from the vehicle in the event of a head on collision to prevent the Ibeo units from being crushed.

On the roof of the vehicle, there is an array of four Sick LIDAR and two Imaging Source 1024 x 768 color Charge Coupled Device (CCD) cameras. This sensor array is shown in Figure 2.3. Two of the Sick LIDAR are angled such that their scan planes intersect the ground 15 and 18 meters in front of the vehicle. The purpose of the forward

Figure 2.2: The protection for the Ibeo XT's is designed to break away from the vehicle in the event of a collision.

looking Sicks is to scan the road in front of the vehicle to detect small objects and negative obstacles, and scan the sides of the road to find curbs and other road defining boundaries. The two remaining Sicks are pointed down and to the sides of the vehicle, intersecting the ground 5 meters from either side of the vehicle. These sensors monitor the blind spots on the sides of Odin.

Figure 2.3: The sensor mounts on the roof rack are designed such that the orientation of any one sensor can be easily adjusted.

The cameras have a combined 90 degree field of view in front of the vehicle. The primary function of the cameras is to detect road markings such as lane lines and stop lines.

The cameras are also used to find the edges of the road, as well as provide redundancy in classifying objects seen by the Ibeo sensor such as cars. The cameras are also used to visually recognize stop lines when approaching intersections.

When all of Odin's sensors are working together, they provide the field of view illustrated in Figure 2.4. This sensor coverage allows Odin to detect obstacles far enough in advance to be able to appropriately react to them. The range of coverage to the front of the vehicle is larger than the range of coverage to the sides or rear because objects approaching the vehicle from the front can be approaching up to twice as fast as objects approaching from any other direction.

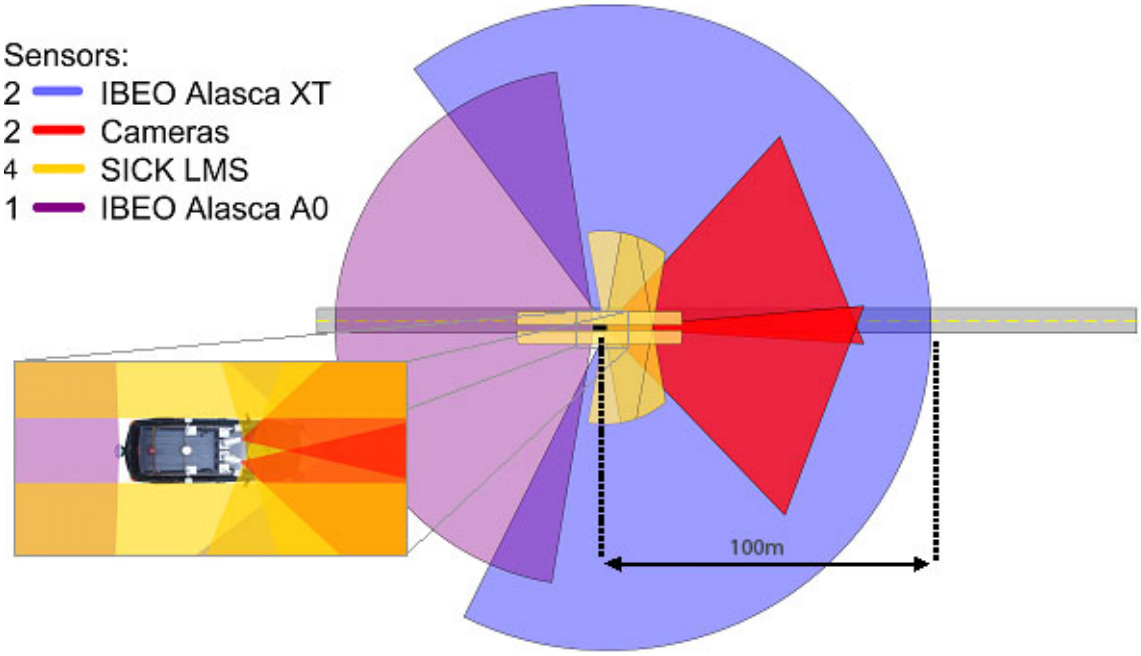

Figure 2.4: Odin's sensor layout provides 360 degrees of sensor coverage.

2.6 Software Architecture

To better distribute the burden of software development, team *V*ictor*T*ango divided Odin's software into discrete modules. These software modules fall into two categories: percep-

tion and planning.

2.6.1 Perception

The perception category encompasses all vehicle software involved in sensing the vehicle's environment as well as the vehicle's state within the environment. The individual software modules in the perception category are: Object Classification (OC), Road Detection, and Localization. Each software module is responsible for processing sensor inputs, packaging the data into Joint Architecture for Unmanned Systems (JAUS) messages, and reporting the messages to the planning modules. Figure 2.5 illustrates the flow of information from raw sensor data to JAUS messages.

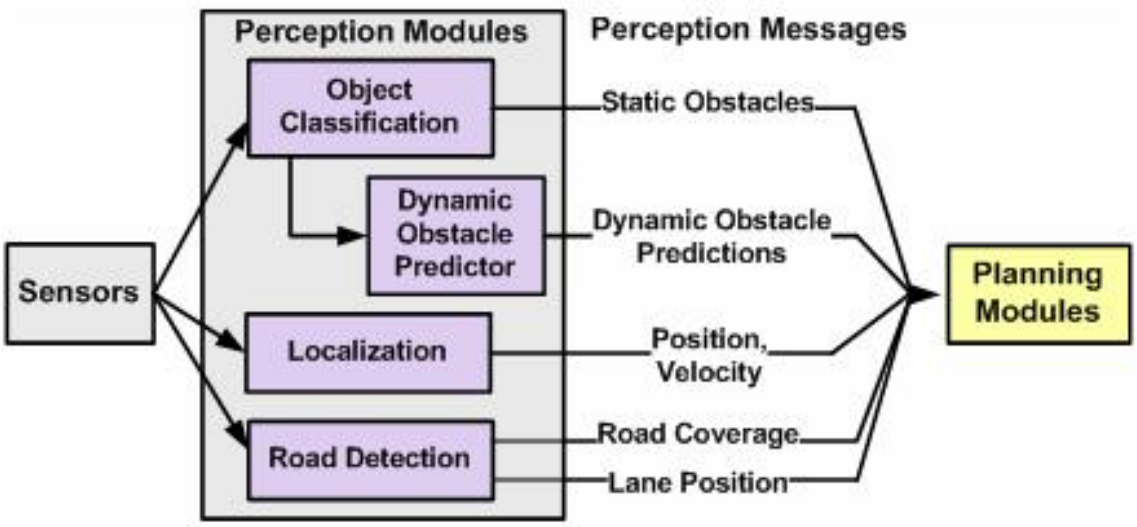

Figure 2.5: Raw sensor data is packaged into JAUS messages and reported to the planning modules.

The OC module is responsible for detecting and reporting any obstacles within Odin's field of view. These obstacles are classified as either static or dynamic obstacles. The DUC rules do not mention or imply the possibility of moving obstacles such as pedestrians or bicycles being present during the competition, therefore the OC module assumes that all objects having an appreciable absolute velocity (i.e. absolute velocity $> 3\ m/sec$)

are vehicles. Additional processing must be done to distinguish between large static obstacles and dynamic obstacles that are momentarily stopped, such as vehicles queued at an intersection.

OC relies primarily on the Ibeo XT scanners to detect and classify obstacles. OC receives both object data, and raw scan data from the Ibeos. The Ibeo object data provides OC with a preliminary classification, the range to the object, and the absolute velocity of the object. The Ibeo's classification of obstacles is based mostly on the size of the object, therefore OC must perform its own sanity checks to ensure that data given by the Ibeo is correct. These sanity checks are done by integrating camera data into the classification algorithm. OC uses the vision data to detect the presence of features unique to cars, such as tail lights and license plates.

Once an object has been determined to be either static or dynamic, the object data is packaged into the appropriate message and sent to the planning modules. For static obstacles, OC sends the approximate size and centroid of the obstacle. If an obstacle is classified as dynamic, the data is then sent to the Dynamic Obstacle Predictor (DOP). DOP is a sub-module of OC and is responsible for projecting the path of dynamic obstacles into the future. The DOP module allows the planning modules to make decisions based on where dynamic obstacles will be rather than where they are currently. This is something that human drivers do almost subconsciously and is completely necessary for making correct decisions when interacting with other moving vehicles.

The next module in the perception category is the Localization module. Localization is perhaps the most important of the perception modules because it is responsible for estimating and reporting the vehicle's position, velocity, and attitude to all other modules in the system including the other two perception modules. This module will not however be discussed further in this section due to the fact that it is the subject of this thesis, and will be discussed in great detail in the remaining chapters.

The final module in the perception category is the Road Detection module. The Road Detection module can be broken down further into two sub-modules: Drivable Area Coverage (DRAC) and Report Lane Position (RLP). DRAC is responsible for using camera data to find flat, uniform surfaces that are considered to be drivable. RLP is responsible for fusing lane data from line detection algorithms with the lane data provided by the RNDF to produce a message that lets other modules know which lane Odin is currently occupying.

2.6.2 Planning

The planning category contains all of the higher level intelligence that controls how Odin behaves in a given situation, and how the vehicle will maneuver through the course. The individual planning components are: Route Planner, Driving Behaviors, Motion Planning, and Vehicle Interface. The responsibilities of these modules are organized in a hierarchical manner with Driving Behaviors being at the top of the hierarchy and Vehicle interface being at the bottom. The flow of data is illustrated in Figure 2.6.

The first module in the planning category is the Route Planner. Although the Route Planner can be considered to be at the top of the planning hierarchy, it is actually the simplest of the planning modules. The Route Planner uses an A* graph search algorithm to determine the shortest route that will achieve all checkpoints specified by the MDF while traveling the shortest possible route through the RNDF. The Route planner sends a JAUS message to the other planning modules giving them a list of waypoints to follow.

The next module in the planning software hierarchy is Driving Behaviors. Driving Behaviors is responsible for the highest levels of intelligence as well as short term navigation. Driving behaviors receives perception data and the planned route provided by the Route Planner, and determines how the vehicle should behave to handle the perceived situation. Driving Behaviors outputs a list of target points and sends this list to the Motion

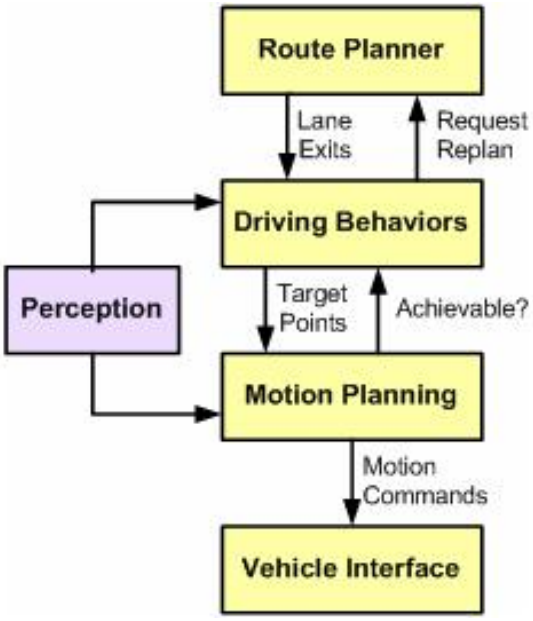

Figure 2.6: The planning modules work together to correctly interpret perception data and execute the correct vehicle behavior and motion for any given situation.

Planning module via JAUS message. These target points are essentially smaller pieces of the route provided by the Route Planner.

The next module in the planning category is the Motion Planning module. Motion Planning is responsible for generating the exact path that the vehicle will follow to achieve the target points it receives from Driving Behaviors. Motion Planning does this by determining a mathematical curve that connects all of the target points while considering the constraints of the vehicles motion. Motion Planning is also responsible for obstacle avoidance and zone navigation. Motion planning outputs motion profiles which consist of a series of steering rates and accelerations. These commands are sent to the Vehicle Interface via JAUS message.

The final module in the planning category, and the lowest level of vehicle control, is the Vehicle Interface. The Vehicle Interface is responsible for converting commanded vehicle movements into actuator signals. The Vehicle Interface controls all vehicle level tasks such as speed control, steering, braking, and shifting. The Vehicle Interface also

controls vehicle functions such as turning on turn signals and beeping the horn.

Chapter 3

Methods Of Localization

Effective autonomous vehicle navigation requires the vehicle's ability to answer three questions: "where am I?", "where am I going", and "how do I get there?" The answer to the first question is the subject of vast amounts of research in the unmanned systems world and is known as the localization problem. The preceding elements of autonomous navigation are in fact completely dependent on solving the localization problem. A map is useless if one cannot determine one's current position within the map, and solving a path to a goal is impossible if one does not know one's initial position relative to the goal. An autonomous vehicle is therefore irretrievably lost if it does not have the ability to localize itself.

The degree to which an autonomous vehicle can localize itself also affects the usefulness and flexibility of the autonomous platform. For example, a vehicle may be able to determine its location based on visually recognizing features within an environment, but if the vehicle is placed in a different environment, with different features, or if its vision

sensors fail, its localization capabilities fail, and the vehicle is lost. Robust localization is a key element in increasing the independence and usefulness of robotic vehicles in varying environments. A flexible autonomous vehicle should be able to combine diverse localization techniques so that it will be able maintain its navigational capabilities when one of its localization methods fail.

This chapter will present a brief background on the general problem of localization. First this chapter will define the localization problem for different types of maps. This chapter will then describe techniques for relative and absolute localization. Finally this chapter will discuss popular methods of combining relative and absolute position solutions.

3.1 Defining Localization

The term localization has been defined as determining one's location within an environment. However, the realization of this can vary depending on what type of map is used to represent the environment. There are two main categories of map representations: metric and topological.

Metric maps are named so because they are based on quantitative measurements of the environment they represent. These types of maps generally consist of occupancy grids or 2-D coordinate frames that are superimposed on the area of interest [2]. Localization for this type of map is the task of determining the position of the vehicle with respect to the coordinate system. Figure 3.1 shows a simplified example of an occupancy grid. Figure 3.1 shows a vehicle with an initial position of (0,0). The vehicle is navigating to a goal with coordinates (7,7). An important characteristic of this map representation is that the position of the vehicle, as well as other objects in the environment, is depicted as occupying an entire cell in the grid, even though the size of the object may not actually

fill the cell completely. In this way, the quality of the environmental representation is limited by the resolution of the map. A more useful and globally referenced example of a metric map representation is the Universal Transverse Mercador (UTM) coordinate system. UTM coordinates provide much higher resolution, allowing users to position objects with millimeter precision.

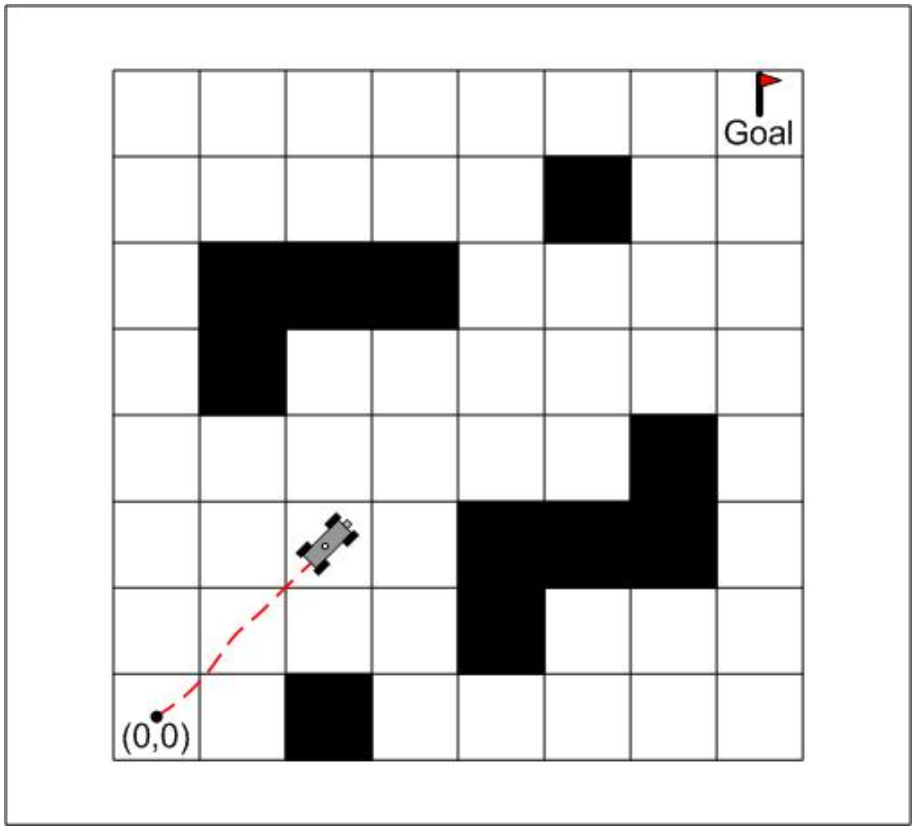

Figure 3.1: The position of objects within the map are the coordinates of the grid cell they occupy.

Topological mapping techniques use landmarks to describe an environment. Localization for a topological map requires landmark detection using vision, range finding, or a combination of both. This type of localization is very effective in structured environments, such as indoor environments. Indoor environments usually contain uniform man made features that are ideal for topological mapping and localization. Topological localization using natural landmarks in an outdoor environment is more difficult because

there is a great deal of variation between landmarks and desirable features such as straight edges or flat surfaces are not common in nature. Positioning objects in a topological map is done by determining the object's relative distance to landmarks within the map.

3.2 Absolute Localization

Absolute localization refers to localization techniques that provide a direct measurement of global position. Absolute localization is an essential part of any autonomous vehicle application because, while it is still important for the vehicle to know its position relative to its immediate surrounding, its usefulness is greatly limited if it does not know its position in the world.

The following section will discuss several methods for achieving absolute localization. This section will also discuss the difficulties associated with these techniques, and the fundamental strengths and weaknesses of absolute localization in general.

3.2.1 Absolute Landmark Detection

One of the oldest techniques for absolute localization, used by humans and robotic vehicles alike, is visual landmark detection. Conceptually, this technique is extremely simple: have a priori knowledge of the global position of a landmark, detect the landmark, and then determine range to the landmark to calculate position. In practice, however, this process is much more difficult.

The first challenge in localization through landmark detection is detecting the landmark. This is not a trivial process. Several methods may be used to perform landmark detection. One common method is LIDAR. LIDAR provides accurate range measurements, but it is difficult to use for landmark detection because it can only convey geometric features of the landmark. This makes it difficult to determine with any certainty if the object

the vehicle is ranging off of is the landmark of interest, or just an object with a similar size and shape.

Another means of detecting landmarks is computer vision. This can be broken down further into monoscopic and stereoscopic vision. For both monoscopic and stereoscopic sensing, the first task is to recognize the landmark through pattern matching or the use of a visual template. These techniques can be rather processor intensive, but are generally straight forward. The next task is to determine the range from the vehicle to the landmark. For stereoscopic cameras this task is trivial assuming the processing required for stereo vision is already in place. For monoscopic cameras, range information is much harder to come by. For this case range can be determined using Structure From Motion (SFM) techniques, however, it can be difficult to achieve range information that will be usably accurate using this method. In addition, the vehicle must be moving for range to be observable using SFM.

The next problem that needs to be solved when localizing via absolute landmarks is dealing with multiple solution ambiguities [1]. Once a landmark is detected and range is determined, the vehicle position can be any point along a circle with radius equal to the range to the landmark. Different methods can be used to reduce the number of position solutions to just two possible position, but a unique solution cannot be calculated without having more information about the landmark. This information can include orientation of the landmark or distinct features of the landmark that would indicate from which direction the object is being viewed. Of course, if the absolute heading of the vehicle is already known, a unique position solution can be calculated directly.

Absolute localization using landmark detection can be used as a primary means of vehicle localization, but in most cases it cannot be used independently. The first disadvantage is that landmarks can be sparse, requiring a vehicle to navigate solely by the use of relative localization techniques until a new landmark can be acquired. Perhaps the

greatest drawback of absolute landmark detection is that it can only be done when there is a priori knowledge of landmarks in an area. This greatly limits the applicability of vehicles dependent on landmark detection for localization, because they cannot be deployed in an environment unless landmarks in that area have already been surveyed.

3.2.2 Global Positioning Systems

In recent years, Global Positioning System (GPS) has become the preferred solution for absolute localization. GPS units such as the U-blox can be purchased for $200 and provide position accuracy as low as 2 meters Circular Error Probable (CEP), or a more expensive unit ($9,000) such as the Novatel Propak LBplus can be used to achieve position accuracy as low as 10 cm CEP. GPS units are typically easy to interface and can provide reasonable update rates, usually ranging from 1 to 50 Hz.

Like most absolute localization solutions, GPS has its faults. The first and most obvious limitation of GPS technology is that it requires that the GPS antenna maintain a clear view of the sky, and be able to observe at least four satellites at all times. Four satellites are needed to carry out the 3D trilateration techniques used to calculate GPS position [6]. GPS is therefore not an option for indoor robotics, and is of limited use to vehicles traveling in environments with occlusions such as tree canopies or tall buildings.

For vehicles traveling between tall buildings there are also other problems. GPS signals can be deflected by solid objects causing the signal to have an increased time of flight. Since GPS position calculations are based on the time of flight of the satellite signal, this deflection corresponds directly to position errors. This phenomenon is known as multipath and is most often experienced on city streets lined with tall buildings commonly referred to as urban canyons.

3.3 Relative Localization

Relative localization, also known as dead reckoning, is the process of determining position relative to a starting point. This starting point may be a location with known global coordinates, or it may be considered the origin of an arbitrary position frame. In either case, relative localization is characterized by the exclusive knowledge of initial position, with no means of directly measuring position afterward. Relative localization is accomplished by integrating proprioceptive measurements over time to achieve a change in position. The two most widely practiced methods for implementing this are odometry and inertial navigation.

3.3.1 Odometry

Odometry is the most widely used method for relative localization in mobile robotics [3]. This is because it can be accomplished using inexpensive sensors and simple calculations and can produce highly accurate position estimates at high update rates. Odometry works by integrating wheel speeds over time to produce incremental vehicle motions. Summing the vehicle movements produces a position estimate. Of course, the direction of the incremental vehicle motion must also be known. In simple differential drive AGVs, a relative heading can be estimated with wheel speed measurements alone, but for larger four-wheeled Ackerman steered vehicles, heading or yaw rate measurements are necessary.

The drawback of this method is that each measurement of vehicle movement has error, and as these movements are added together the total error in the position estimate will continue to grow without bound. There are two types of error associated with odometric measurements: systematic errors, and non-systematic errors. Systematic errors include errors associated with imperfections in the kinematic vehicle model, and errors in

the measurement of wheel base and wheel diameter. Systematic errors can typically be compensated for through calibration.

Non-systematic errors are errors that arise due to interaction between the vehicle and its environment. This includes errors such as lateral and longitudinal slip between the wheels and the ground, and loss of contact between the wheel and the ground due to uneven or rocky terrain. These errors are much more difficult to overcome. In general, the only way to compensate for non-systematic errors is to update the relative position solution periodically with an absolute position measurement, essentially reseting the relative position solution.

The magnitude of these types of errors depends on the type of surface the vehicle is traveling on, and the speeds at which the vehicle is moving. For instance, if a vehicle is traveling on a smooth surface at low speeds, the wheels will experience less lateral slip, and are less likely to loose contact with the ground. For this case the magnitude of non-systematic errors will be small and the vehicle will be able to maintain an accurate position solution over longer distances. In general, the faster a vehicle travels, the more lateral wheel slip it will induce. Similarly, the more uneven or loose the terrain is that the vehicle is traveling over, the more longitudinal slip it will experience.

3.3.2 Inertial Navigation

Inertial navigation is a more sophisticated and more expensive solution for relative localization. Inertial navigation is done by integrating accelerations twice and angular rates once to determine incremental vehicle movements. The acceleration and angular rate measurements are typically provided by an Inertial Measurement Unit (IMU), which consists of three orthogonal accelerometers and three laser ring gyroscopes. This arrangement provides 3D acceleration measurements and angular rate measurements about each axis.

The primary advantage inertial navigation when compared to odometry is that it is not

susceptible to outside error sources such as wheel slip. However, because the angular rate measurements must be integrated to produce angular position, the angle measurements will drift with time [3] and the position error will therefore increase without bound as the vehicle moves. Inertial navigation is therefore only useful over relatively short distances.

Chapter 4

Kalman Filtering

A critical part of robust localization is the ability to simultaneously use diverse localization methods in a manner that takes advantage of the strengths of each method. Since 1960, the tool of choice for accomplishing this task has been the Kalman Filter.

The Kalman filter is an optimal recursive data processing algorithm [10] that was introduced by Rudolph E. Kalman in 1960 as a solution to some of the problems left unsolved by his predecessor Norbert Wiener and his Wiener filter. The Wiener filter was developed for the United States during World War II and was released to the public in 1949 [11]. The Wiener filter quickly earned the nickname, "The Yellow Peril", due to the paper's yellow cover, and the mathematical complexity found within. In his 1960 paper, R.E. Kalman addressed the limitations of the Wiener filter and presented a new method that proved to be less abstruse mathematically and better suited to machine computation [7]. This new method became known as the Kalman filter, and is still used 47 years later to solve estimation problems in a wide array of applications.

The following chapter will present an overview of the Kalman filter. This overview will assume that the reader has at least a familiar knowledge of linear systems theory and state-space representations, as well as an understanding of probability theory. First, this chapter will review the necessary linear systems and probability theory needed for the Kalman filter formulation. Then, this chapter will discuss the state-space models used in Kalman filters. Finally, this chapter will expound upon extensions of the Kalman filter for nonlinear systems. The material discussed in this chapter is a compilation of material gathered from [8], [4], and [11].

4.1 The Linear Kalman Filter

The linear Kalman filter formulation begins with a state space model of the system. Because the work presented in this thesis is done using the discrete form of the Kalman filter, the continuous Kalman filter will not be discussed. Moreover, the continuous form of the Kalman filter, though not to be discounted entirely, is of limited use in the digital world in which we live.

The system we wish to apply the Kalman filter to consists of two parts: the states of the system, and the measurements taken to monitor these states. The states are described by a discrete first order linear system given by

$$\bar{x}_{k+1} = \Phi_k \bar{x}_k + \Gamma_k \bar{w}_k \tag{4.1}$$

where $\bar{x}_k$ is the state vector at time t_k, Φ_k is the state transition matrix that relates $\bar{x}_k$ to $\bar{x}_{k+1}$, and Γ_k is the process noise distribution matrix which puts the process noise $\bar{w}_k$ into coordinates of $\bar{x}_k$. In the absence of a forcing function, the state transition matrix will propagate the state estimate through time based on the dynamics of the system.

The measurements are also modeled as a discrete first order linear equation of the

form

$$\bar{z}_k = H_k \bar{x}_k + \bar{v}_k \tag{4.2}$$

where $\bar{z}_k$ is the measurement at time t_k, and H_k is the observation matrix that relates $\bar{x}_k$ to $\bar{z}_k$. The measurements can directly or indirectly measure states of the system, and not all measurements must be available for a given time although one must take precautions to ensure that missing measurements will not cause the system to become unobservable.

At this point, some assumptions must be made about the noise terms in the each of the preceding equations. In order for the Kalman filter formulation to be valid, the noise terms $\bar{w}_k$ and $\bar{v}_k$ must be white noise sequences, and must have known covariance. The covariance matrices for $\bar{w}_k$ and $\bar{v}_k$ are given by

$$E\left[\bar{w}_k \bar{w}_i^T\right] = \begin{cases} \mathbf{Q}_k, & i = k \\ 0, & i \neq k \end{cases} \tag{4.3}$$

$$E\left[\bar{v}_k \bar{v}_i^T\right] = \begin{cases} \mathbf{R}_k, & i = k \\ 0, & i \neq k \end{cases} \tag{4.4}$$

$$E\left[\bar{w}_k \bar{v}_i^T\right] = 0, \quad for\ all\ k\ and\ i \tag{4.5}$$

These equations also imply that the process noise and the measurement noise are uncorrelated.

Now that the system model is considered known, the linear Kalman filter equations may be implemented. The linear Kalman filter equations are as follows:

$$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k\right]^{-1} \tag{4.6}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left[\bar{z}_k - H_k \hat{x}_k^-\right] \tag{4.7}$$

$$P_k = \left[I - K_k H_k\right] P_k^- \tag{4.8}$$

$$\hat{x}_{k+1}^{-} = \Phi_k \hat{x}_k \tag{4.9}$$

$$P_{k+1}^{-} = \Phi_k P_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T \tag{4.10}$$

The proof of these equations can be found in [7]. The hat symbol in $\hat{x}$ denotes an estimate of the state vector $\bar{x}$. The super minus on $\hat{x}_k^-$ and P_k^- denote that these are estimates of these terms prior to updating based on new measurements. These equations are not run all at once, and for some cycles, some of these equations will not be run at all. The flowchart in Figure 4.1 illustrates how these equations are used.

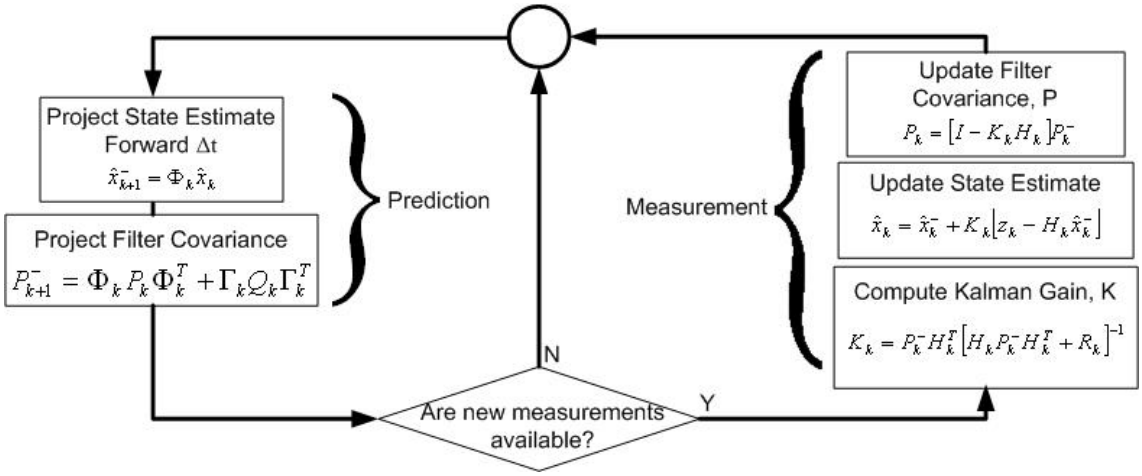

Figure 4.1: The system equations can continue to be projected forward in time without measurement updates, but the accuracy of the estimates will depend only on the accuracy of the state model.

As depicted in Figure 4.1, the state estimates are calculated by linearly projecting the state model forward by the time step, Δt. If the process model were perfect, these equations could be run indefinitely without measurement updates. Because the process model will never be perfect, measurements must be incorporated periodically to keep the solution from diverging. When measurements are available, the filter equations will update the prediction. The state estimate is never fully based on the prediction or the measurements. Instead, the Kalman gain, K, is used to optimally blend the prediction and the measurements based on the covariance of each.

4.2 The Extended Kalman Filter

The preceding section presented a linear formulation of the Kalman filter. However, for many systems, the process model and/or the measurement model are nonlinear. To apply a Kalman filter tow these system, linearization of the process and measurement models must be performed. The following section presents an Extended Kalman Filter (EKF) formulation for nonlinear systems.

One common method for applying the Kalman filter to nonlinear systems is to linearize the process around a precomputed nominal trajectory. The problem with this method is that over extended missions, the difference between the estimated trajectory and the nominal trajectory can grow to the point that linearization about the nominal trajectory is no longer a good estimate of the true trajectory of the vehicle. The EKF is a linearization solution that is better suited for long missions. This is because the EKF linearizes the state model around the estimated trajectory instead of a precomputed trajectory. Intuitively, it would seem that using the updated estimated trajectory for linearization would be better than using the nominal trajectory. This is true for most cases, but if the updated estimate is actually poorer than the nominal trajectory, the next estimate will also be poorer and the estimates will continue to deviate from the nominal trajectory until the filter is completely diverged. In other words, while the EKF may produce better estimates over extended missions, it is a riskier approach because bad measurements can quickly lead to divergence of the filter.

To begin the EKF formulation, consider a system that may be written in the form

$$\dot{\bar{x}} = f(\bar{x},t) + \bar{w}(t) \tag{4.11}$$

$$\bar{z} = h(\bar{x},t) + \bar{v}(t) \tag{4.12}$$

where f and h are nonlinear functions. For a given trajectory estimate, these nonlinear

functions can be approximated by their Jacobian matrices evaluated at the current trajectory estimate. This is written as

$$F_k = \frac{\partial h}{\partial \bar{x}}\left(\hat{x}_k^-\right) \tag{4.13}$$

$$H_k = \frac{\partial f}{\partial \bar{x}}\left(\hat{x}_k^-\right) \tag{4.14}$$

The linear approximations of the nonlinear functions can now be used in the Kalman equations as follows:

$$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k\right]^{-1} \tag{4.15}$$

$$\hat{x}_k = \hat{x}_k^- + K_k\left[\bar{z}_k - H_k \hat{x}_k^-\right] \tag{4.16}$$

$$P_k = \left[I - K_k H_k\right] P_k^- \tag{4.17}$$

$$\hat{x}_{k+1}^- = \Phi_k \hat{x}_k \tag{4.18}$$

$$P_{k+1}^- = F_k P_k F_k^T + \Gamma_k Q_k \Gamma_k^T \tag{4.19}$$

where H and F are now the measurement Jacobian matrix and the system Jacobian matrix respectively. These equations are run in the same manner as the linear Kalman filter equations with the only difference being that the measurement and system Jacobian matrices must be calculated at the beginning of each cycle.

Chapter 5

Filter Design

When considering a localization solution for an AGV, it is important to think about how the localization data will be used. For route planning and global navigation, it is important to know as accurately as possible the location of the vehicle so that navigation decisions can be made such as when to change lanes or when to exit a road segment. In contrast, it is not important for the perception modules to know the true global position of the vehicle as much as it is important for them to have a steady continuous position solution for keeping track of obstacles within sensor range. In designing the localization software for Odin, these needs were taken into account and the solution was to provide both types of localization solutions simultaneously. The localization software will provide the planning modules with an accurate absolute position while also providing the perception modules with continuous relative position. This chapter discusses how this was accomplished and presents the formulation of the Absolute Position Filter (APF) and the Relative Position Filter (RPF).

5.1 Absolute Position Filter

The APF is designed to provide the planning modules on Odin with a reliable global position solution. Odin is equipped with a high precision INS, wheel encoders, and steering sensors, as well as cameras that can be used to aid the APF by sensing absolute landmarks.

Odin's INS provides the APF with a filtered GPS and inertial measurement solution. This system could be used as a stand alone localization solution for the vehicle, however, as with any inertial solution, the position estimate will drift when the vehicle is traveling without at least four satellite observations. The APF will blend odometry measurements into the inertial solution in order to decrease the rate at which this drift occurs. The APF will also provide a means of incorporating other absolute position measurements asynchronously such as absolute landmarks.

Finally, the APF will afford the user more control over the behavior of the global position estimate. The user can tune the filter to adjust how the position estimate will react to GPS pops and how quickly the position estimate will converge to the new position solution solution after a GPS pop.

5.1.1 System Model

Development of the APF begins with modeling the vehicle dynamics. Some assumptions have been made to simplify this model. First, the velocities and angular rates are assumed to be constant from one time step to the next. This is known as the low dynamics assumption and is a reasonable assumption for vehicles traveling at low speeds over relatively even terrain. The greatest benefit of making this assumption is that it considerably reduces the number of states necessary to model the system by eliminating acceleration states from the process model.

The other necessary assumption made concerning the system model arises from limi-

tations imposed by the localization sensor suite available on Odin. A linear system model could be developed that would sufficiently model the dynamics of the vehicle, but with the given sensor suite, that system would not be observable. The nonlinear system projects all vehicle velocities on the the vehicle x-axis and all vehicle rotation rates to rotations about the vehicle z-axis. This essentially assumes that velocities along the lateral and vertical axes of the vehicle are negligible, as are the rotation rates about the longitudinal and lateral axes of the vehicle. Since Odin is designed to travel on paved roads in an urban environment, and at reasonably low speeds, these assumptions can be made without detrimentally degrading the validity of the vehicle model.

The resulting nonlinear vehicle model is as follows:

$$\bar{x}_{k+1} = \Phi\, \bar{x}_k$$

$$\begin{bmatrix} N_{k+1} \\ E_{k+1} \\ D_{k+1} \\ V_{k+1} \\ \phi_{k+1} \\ \theta_{k+1} \\ \psi_{k+1} \\ \dot{\beta}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & cos\theta cos\psi dt & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & cos\theta sin\psi dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -sin\theta dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & tan\theta cos\phi dt \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & cos\phi dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & cos\phi sec\theta dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} N_k \\ E_k \\ D_k \\ V_k \\ \phi_k \\ \theta_k \\ \psi_k \\ \dot{\beta}_k \end{bmatrix} \quad (5.1)$$

where N, E, and D are the Northing, Easting, and Down position of the vehicle, ϕ, θ, and ψ represent the roll, pitch, and yaw of the vehicle respectively, and V and β represent the projection of all vehicle velocities onto the vehicle x-axis, and the projection of all angular rates about the vehicle z-axis respectively. By projecting the vehicle velocities onto the vehicle x-axis and the angular rates onto the vehicle z-axis, the number of states in the system model is reduced, thus decreasing the computational requirements of the

system model and solving the observability problem.

5.1.2 Measurement Model

The next part of the APF development is to determine the measurement model. Some of the relationships between the measurements and the process model are nonlinear, so a nonlinear form of the Kalman filter must be formulated. An EKF has been chosen for this system and the measurement model. As discussed in the previous chapter, the measurement matrix in the EKF is actually the Jacobian matrix of the nonlinear function that relates the measurements to the process model. This section will populate the measurement Jacobian matrix with the functions determined by the sensors used in the filter.

Odin is equipped with a Novatel Propak LBplus GPS/INS. From this system, the filter will receive 3D global position, 3D velocity, and three-axis rotation angles. The Propak system is the primary sensor for the localization system. Other sensors will serve only to improve the solution provided by the Propak, but cannot be used for extended periods of time without it. The measurement Jacobian for these measurements is

$$H_{INS} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos\phi sin\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & sin\phi sin\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.2}$$

Odin is also equipped with four wheel encoders that provide wheel speeds that can be

read directly from the vehicle's CAN bus. The data taken from the CAN bus is already preprocessed and is output as a direct velocity measurement, so no further processing is necessary to get usable data. The measurement Jacobian matrix for the wheel speed measurements is

$$H_{WS} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.3}$$

To measure the vehicle steering angle, two string potentiometers have been installed on the vehicle's steering shaft. This data is also processed at the vehicle interface level, so no further processing is necessary to achieve steering angle. The steering angle measurements are incorporated into the measurement model using a classical bicycle model. The bicycle model will convert steering angle measurements and velocity estimates into yaw rate measurements. The derivation of the bicycle model equations is done in Appendix A and the resulting equations are,

$$\frac{\partial \delta}{\partial V} = \frac{-L\dot{\beta}}{1+\left(L\dot{\beta}\right)^2} \tag{5.4}$$

$$\frac{\partial \delta}{\partial \dot{\beta}} = \left(\frac{LV}{1+\left(L\dot{\beta}\right)^2}\right) \tag{5.5}$$

where δ is the steering angle, L is the length of the vehicle, V is the estimated forward velocity of the vehicle, and $\dot{\beta}$ is the estimated yaw rate. The measurement Jacobian for the steering angle measurement is

$$H_{SA} = \begin{bmatrix} 0 & 0 & 0 & \frac{\partial \delta}{\partial V} & 0 & 0 & 0 & \frac{\partial \delta}{\partial \dot{\beta}} \end{bmatrix} \tag{5.6}$$

The final localization measurement on Odin is absolute landmark position. The landmarks that will be detected are stop lines. The vehicle will have *a priori* knowledge of the location of all stop lines in the DUC course, and will be required to stop within 1 meter of

the painted lines. The range to the stop line will be determined and used to translate the stop line position to vehicle position using the equations

$$N = N_{SL} - d_{SL} cos\psi \tag{5.7}$$

$$E = E_{SL} - d_{SL} sin\psi \tag{5.8}$$

where N and E are the northing and easting of the vehicle respectively, N_{SL} and E_{SL} are the northing and easting of the stop line respectively, d_{SL} is the distance to the stop line, and ψ is the heading of the vehicle. These equations provide a direct measurement of the vehicle northing and easting and their measurement Jacobian is therefore

$$H_{SL} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.9}$$

Combining the preceding measurement Jacobians into the total measurement Jacobian

for the measurement model yields

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos\phi sin\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & sin\phi sin\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial \delta}{\partial V} & 0 & 0 & 0 & \frac{\partial \delta}{\partial \beta} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.10}$$

5.1.3 Noise Model

The final step necessary for the formulation of the APF is to quantify the error associated with both the process model and the measurement model. At this point is important to note that accurately quantifying the process and measurement noise is not essential to the optimal operation of the EKF. What is important is that the relative weight of the individual process state errors and the individual measurement errors are correct. This will allow the filter to trust in more accurate measurements or state estimates than less accurate ones, thus optimizing the available measurements and the process model predictions.

The primary source of error in the process model is Taylor series truncation error. A constant velocity model has been used to approximate the system, therefore the errors

in the model will be velocity errors caused by the fact that the velocities are not truly constant. The system covariance matrix, Q_k, is assumed to be diagonal because there is little or no knowledge of the correlation of errors between states. These off diagonal terms are therefore assumed to be zero. The error induced by this assumption will be handled by the filter covariance matrix, P_k, which will populate the off diagonal covariance terms naturally as the filter runs.

The error for each state will now be approximated as the Taylor remainder in the dead reckoning equations. The first three states in the process model are linear position states. The error associated with these states is the double integral of the neglected accelerations. This integral can be approximated as

$$\sigma_x = \sigma_y = \sigma_z = \frac{a_{max}(\Delta t)^2}{2} \tag{5.11}$$

where Δt is the time step of the filter and a_{max} is an estimate of the maximum acceleration that the vehicle is likely to experience. A weighting vector will also be used to account for the fact that most of the acceleration the vehicle experiences will be along the longitudinal axis of the vehicle. The weighting vector for the vehicle accelerations is

$$\bar{a}_{max} = \begin{bmatrix} 1.0a_{max} & 0.1a_{max} & 0.3a_{max} \end{bmatrix}^T \tag{5.12}$$

For the angular position states in the process model, it is the angular velocity terms that have been neglected. Therefore, the error for these states will be a function of angular velocity and the Kalman filter time step. These errors are written as

$$\sigma_\phi = \sigma_\theta = \sigma_\psi = \Omega_{max}\Delta t \tag{5.13}$$

where Ω_{max} is the maximum angular velocity the vehicle is likely to experience and Δt is

the time step of the EKF. This is a very rough estimate of the angular position state error, but as previously mentioned, it is not important that these error estimates be accurate as much as it is important to have a mechanism by which the errors can be weighted. Just as with the linear position states, the magnitude of the state error will vary along each axis of the vehicle. A weighting vector is used for the angular position errors and is written as

$$\bar{\Omega}_{max} = \begin{bmatrix} 0.3\Omega_{max} & 0.3\Omega_{max} & 1.0\Omega_{max} \end{bmatrix}^T \tag{5.14}$$

This vector weights angular velocities about the vehicle z-axis more heavily because it is about this axis that the vehicle is most likely to see higher angular velocities.

The error associated with the linear velocity states is a direct result of neglecting accelerations in the process model. The linear velocity error can therefore be written as

$$\sigma_V = a_{max}\Delta t \tag{5.15}$$

where a_{max} is the same acceleration term from the linear position state error, and Δt is still the time step of the EKF.

Now that the process noise has been sufficiently estimated, the measurement noise must be determined. This is far more critical to the operation of the filter. The measurement noise model is what the filter will use to determine which measurements should be trusted and which measurements should be ignored. For most of the localization sensors the noise model will simply be set to the error given in the sensor specifications.

The noise model for the Novatel Propak LBplus is actually quite simple to determine due to the fact that the covariance for the Novatel INS solution is an output of the sensor. This output will be inserted directly into the measurement covariance matrix and will be updated as new covariance messages are received from the Propak. This takes care of all position, velocity, and attitude measurements received from the Propak.

The noise model for the wheel encoder is a function of the error induced by the scale factor used to convert the encoder pulses to a velocity, and the velocity of the vehicle. This function is written as

$$\sigma_{WS} = SFE_{WS} V_{WS} \tag{5.16}$$

where SFE_{WS} is the estimated scale factor error and V_{WS} is the wheel speed measurement.

The steering measurement is not considered a high fidelity measurement, so precisely estimating the steering measurement error is unnecessary. The steering measurement error will therefore be set to a constant to be determined when tuning the filter.

The uncertainty associated with the stop line measurements will be determined by the Road Detection module. The Road Detection module will assign a confidence value to the stop line position estimate and send it along with the stop line data.

The state uncertainties were estimated in vehicle frame coordinates because it is more intuitive to do so. To use these error estimates in the APF however, they need to be transformed into global frame coordinates. This is accomplished with the Γ matrix. The Γ matrix for this system is written as

$$\Gamma = \begin{bmatrix} \left[R_b^n\right] & 0 & [0] & 0 \\ 0 & 1 & 0 & 0 \\ [0] & 0 & [\Omega] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17}$$

where R_b^n is the rotation matrix that transforms linear and angular position state uncertainties from the body frame to the nav frame, and Ω is transforms angular velocity uncertainties from the body frame to the nav frame. The kinematics for these coordinate frame transformations are presented in [9], with the modification that the navigation frame in this thesis is a North-East-Down coordinate system and the navigation frame in [9] is a

North-East-Up coordinate system.

5.2 Relative Position Filter

The second half of Odin's localization software is the RPF. The RPF will provide Odin's perception modules with a position in a local position frame that is guaranteed to be free of discontinuities such as GPS position pops. This is perhaps the most important function of the localization solution because GPS pop is very likely to occur in an urban setting and without a relative frame to place perceived objects, GPS pops will result in phantom obstacles.

5.2.1 System Model

This system model chosen for the RPF consist of only three states: velocity, heading, and yaw rate. This simple model was chosen to avoid nonlinearities which would require an EKF to estimate the system, and to avoid observability problems caused by eliminating all localization measurements influenced by the GPS. The state space equation for this system model is written as

$$\bar{x}_{k+1} = \begin{bmatrix} V_{k+1} \\ \psi_{k+1} \\ \dot{\psi}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_k \\ \psi_k \\ \dot{\psi}_k \end{bmatrix} \tag{5.18}$$

5.2.2 Measurement Model

A major criteria of the RPF is that the position that it estimates be based entirely on relative localization measurements. This is essential for eliminating the possibility of position pops, which is the main purpose of the RPF. There are two relative localization

techniques to choose from on Odin. These techniques are dead reckoning with odometry and inertial navigation. Due to the amount of preprocessing required to use the raw inertial measurements provided by the IMU and the added complexity to the system model, only odometry was used in the RPF.

The first measurement used in the RPF measurement model is the wheel speed measurement provided by the Vehicle Interface. This is a direct measurement of the velocity state in the RPFs system model, so the observation matrix for this measurement is written as

$$H_{WS} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{5.19}$$

The next measurement used in the RPF measurement model is the heading measurement provided by the Novatel INS solution. Technically, this measurement is influenced by the GPS, which violates the criteria of the RPF, however, this is the only heading measurement available on Odin, so it will suffice. In addition, although the heading measurement is updated with GPS measurements in the Novatel INS filter, this measurement does not typically exhibit any strange behavior or discontinuities, even when GPS pops occur. The observation matrix for this measurement is

$$H_{\psi} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{5.20}$$

The next measurement used in the RPF measurement model is the steering angle measurement. This is converted to a yaw rate and velocity measurement using the same classic bicycle model used in the APF measurement model. Again, the equations are

$$\frac{\partial \delta}{\partial V} = \frac{-L\dot{\psi}}{1+(L\dot{\psi})^2} \tag{5.21}$$

$$\frac{\partial \delta}{\partial \dot{\psi}} = \left(\frac{LV}{1+(L\dot{\psi})^2} \right) \tag{5.22}$$

where δ is the steering angle, V is the velocity estimate of the filter, L is the distance between the front and rear axle, and $\dot{\psi}$ is the yaw rate. For this measurement model, $\dot{\psi}$ is used instead of $\dot{\beta}$ because this model uses only the rotation about the vehicle z-axis to estimate yaw rate rather than incorporating projections of rotations about other axes. The observation matrix for this matrix is written as

$$H_{SA} = \begin{bmatrix} \frac{\partial \delta}{\partial V} & 0 & \frac{\partial \delta}{\partial \dot{\psi}} \end{bmatrix} \tag{5.23}$$

The last measurement to be added to the RPF is the yaw rate measurement provided by the gyro in the Propak's IMU. So far the only measurement of yaw rate comes from the steering angle measurement via simplified bicycle model. As stated in the previous section, this measurement is a very low fidelity measurement. Therefore, the raw yaw rate measurement provided by the IMU gyro will be used to help stabilize the yaw rate estimate in the RPF. The observation matrix for this measurement is written as

$$H_{gyro} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{5.24}$$

Having fully populated the observation matrix for the RPF, the full measurement model can be written as

$$\bar{z}_k = \begin{bmatrix} V_{WS} \\ \psi_{INS} \\ \delta \\ \dot{\psi}_{gyro} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{\partial \delta}{\partial V} & 0 & \frac{\partial \delta}{\partial \dot{\psi}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_k \\ \psi_k \\ \dot{\psi}_k \end{bmatrix} \tag{5.25}$$

5.2.3 Noise Model

The noise model for the RPF is very similar to the noise model for the APF. Just as with the APF, little is known about the cross correlation of the state errors or the measurement errors, so the system covariance matrix, Q, and the measurement covariance matrix, R, for the RPF are assumed to be diagonal matrices.

As mentioned in the discussion of the APF noise model, it is not as important that errors be accurately quantified as much as it is that they have the correct relative weights. When tuning the APF, it became clear that the error estimates did not make much sense in terms of physical parameters, but the filter could still be tuned correctly by making sure that state errors and measurement errors had the correct relative magnitudes. For this reason, the system error estimates for the RPF have been simplified. The system errors will be adjusted with error constants that are not based on real world parameters. The error constants will simply provide a means of adjusting the integrity of the predictions provided by the system model. The system error estimates will still be functions of the filter time step. This makes sense intuitively because one would expect errors to increase with time due to the fact that the system model is a linear approximation of the actual vehicle dynamics. The system errors are then written as

$$\sigma_V = VEC\Delta t \tag{5.26}$$

$$\sigma_\psi = HEC\Delta t \tag{5.27}$$

$$\sigma_{\dot{\psi}} = YEC\Delta t \tag{5.28}$$

where VEC is the velocity error constant, HEC is the heading error constant, and YEC is the yaw rate error constant.

The measurement noise estimates for the RPF will be the same as those for the APF. This is because the measurements used in the RPF are also used in the APF with the

exception of the yaw rate measurement. The yaw rate measurement error is set to a constant of 0.01 rad/sec or approximately 0.57 deg/sec.

Chapter 6

Filter Implementation and Testing

Perhaps the most daunting task in the development of any Kalman filter is the actual implementation. This is a tedious process involving hours of tuning and testing in order to arrive at a truly optimal filter (or as close to optimal as the filter can be). The following chapter details the design of Odin's localization software and the procedures for tuning and testing the absolute and relative position filters.

6.1 Software Overview

Odin's localization software is responsible for providing accurate position, velocity, and angular orientation data to all other planning and perception modules. Because the entire software suite is so dependent on localization data, it is imperative that the localization module be robust and able to recover quickly from any errors it may encounter.

Odin's localization software was developed using National Instrument's LabVIEW

software. LabVIEW was chosen because it decreases software development time by providing an easy to use graphical programming environment that is more intuitive for engineers with little or no background in software engineering or computer science. LabVIEW also allows for quick and easy development of a Graphical User Interface (GUI) and a means of making on-the-fly code changes, without the need to recompile code.

The first task of the localization module is to interface all localization sensors. The primary sensor is the Novatel Propak LBplus. The interface for the Propak was developed for the 2005 DARPA Grand Challenge, and, due to its modular design, could be placed directly into Odin's localization code with little or no changes. The interface for all other localization sensors is taken care of by either the Vehicle Interface software module, which provides the wheel speeds and steering angle, or the Road Detection module, which provides stop line position data. Within the localization software, receiving these measurements is simply a matter of parsing the corresponding JAUS message.

To save time, and because testing time with Odin was limited, data logging was added to the localization software. All measurements used in the APF and RPF are logged along with a boolean array that indicates which measurements were used for a particular cycle of the filter. This allows filter tuning to take place in simulation using real data.

6.2 Absolute Position Filter Tuning

The goal when tuning the APF is to weigh state and measurement errors such that the filter will rely on the measurements when the conditions are ideal and measurement data is at its best, and rely on the process model when good measurement data is not available. In other words, the filter should be tuned so that the state estimates are as good as or better than the best measurements available, but never worse.

The best case scenario for the APF is for it to be receiving a good solution from the

Novatel system while observing ten or eleven satellites well distributed throughout the sky. For this situation the filter should output a position estimate that is as close to the unfiltered output of the Novatel system as possible. In other words, it is important to ensure that the APF will not make the position estimate worse than a position estimate that is known to be accurate. To tune the filter for this, sensor data was collected while the vehicle was driven manually around a known path. The path driven was in a large empty parking lot with a good view of the sky, and ten satellites were being observed by the Novatel system with good Geometric Dilution of Precision (GDOP). The collected sensor data was then run through the filter and the resulting position estimate was overlaid onto a plot of the sensor data collected from the Novatel system.

The APF must also be tuned to appropriately handle GPS pops. What is appropriate is completely up to the user. The user must decide whether it is desirable for the filter solution to immediately jump to a new solution when it experiences a pop, or for the filter solution to gradually merge with the new position solution. Figure 6.1 shows how the APF handles GPS pops when the GPS measurement is heavily weighted. In this case the estimated position immediately jumps to the new GPS position.

For Odin, it was decided that it would be better for the motion planning module for the position to gradually merge with the new position solution, rather than jump instantaneously. This is achieved by increasing the magnitude of the GPS position error, which effectively tells the filter to trust the GPS position measurements less. The result is shown in Figure 6.2. This plot shows that the position estimate gradually merges to the new position rather than popping over instantly. This behavior is much less confusing to path planning modules because the apparent motion in the position estimate is smooth and does not violate the kinematic constraints of the vehicle.

As scene in the previous figures, a critical decision must be made about the behavior of the APF. Tuning the APF such that it is heavily reliant on the GPS measurements results

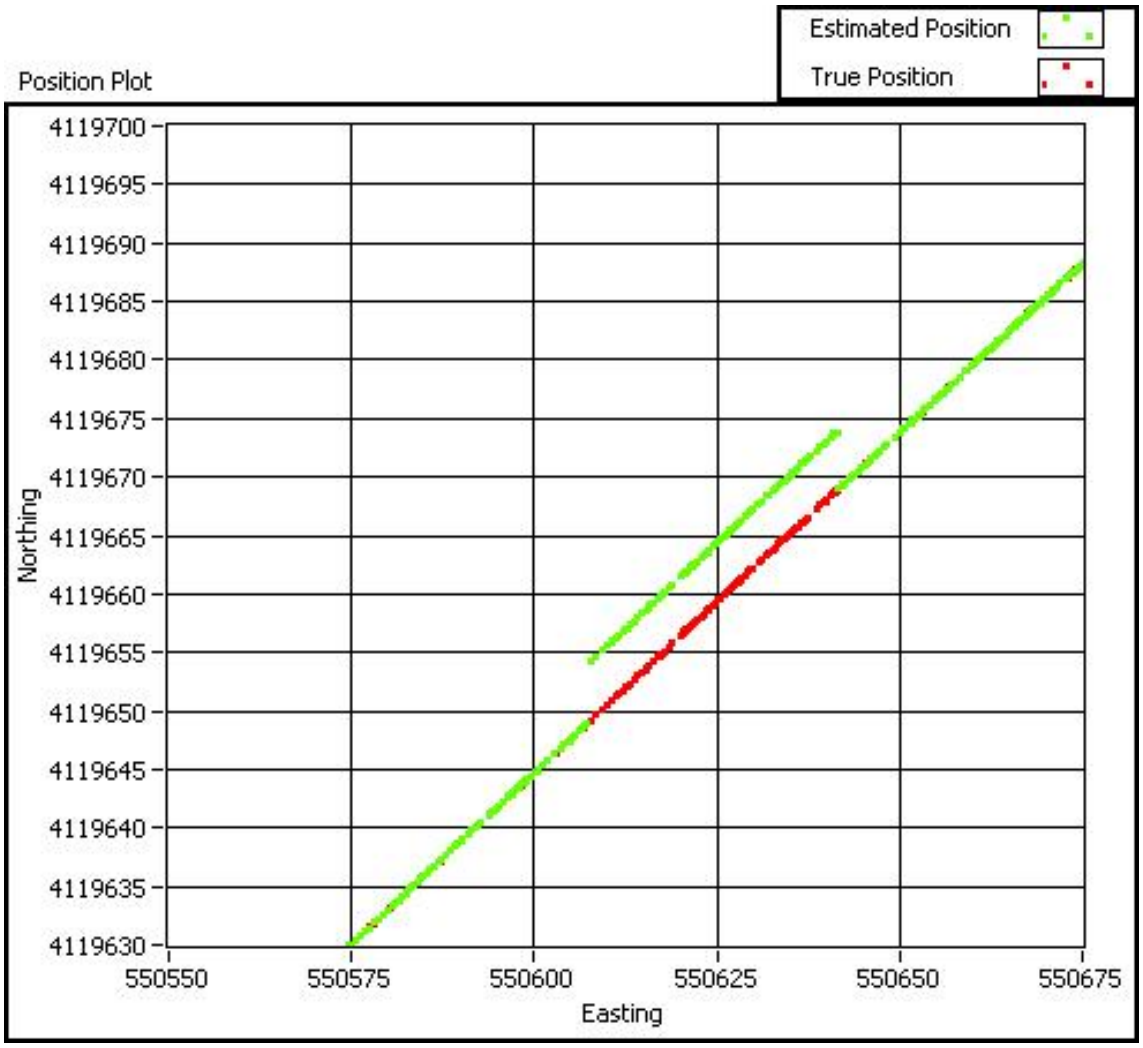

Figure 6.1: For this case, 5 meters of position bias were added instantaneously to illustrate the effect of GPS pop.

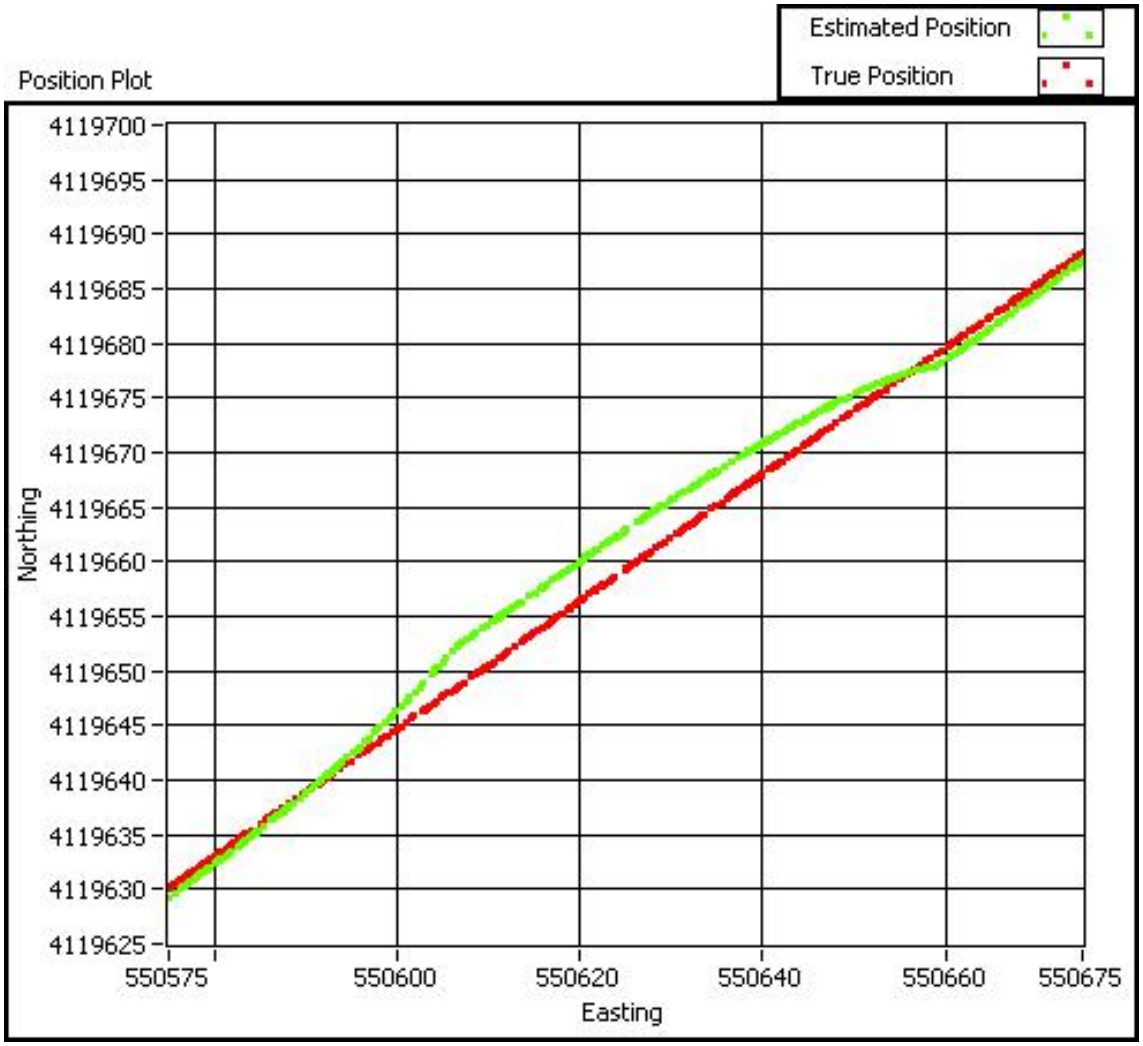

Figure 6.2: The APF smooths out the GPS pop, but the overall position estimate is less accurate.

in more accurate position measurements, but leads to instantaneous position jumps when GPS is bad. These jumps can be smoothed by decreasing the filter's faith in the GPS measurement, but doing so decreases the accuracy of the position estimates, even when GPS measurements are good. In the end, the decision was made for accuracy rather than a smooth position estimate. This decision was based on the fact that the Novatel system typically provides an accurate solution.

6.3 Relative Position Filter Tuning

In tuning the RPF, the goal is to minimize the accumulation of error between the estimated position and the actual position. It is impossible to eliminate all of the error and the error will continue to increase as the vehicle moves. This is a fundamental limitation of relative localization. All that is necessary is to maintain a reasonable position estimate over the range of the vehicles perception. This is because the RPF is only used to provide a frame for positioning perceived obstacles that is not susceptible to GPS pop. Therefore the overall error in this position is inconsequential as long as the position errors to not grow substantially within the range that the vehicle can observe a particular object.

The RPF was tuned using the same method used to tune the APF. Data was collected on the vehicle and then replayed in a simulator. Figure 6.3 shows sample data used to tune the RPF. As shown in the figure, the error between the estimated position and the true position increases as the vehicle travels. This plot also seems to indicate that the error has decreased to nearly zero by the time the vehicle returns to the starting point. This is only a coincidence. Position errors grow primarily in the direction that the vehicle is moving. Therefore, if the vehicle travels 100 meters due North, and then returns 100 meters due South, the position errors are likely to be roughly equal for each trip, thus cancelling each other. Since the vehicle traveled in a loop, the error accumulated while traveling East and

North was cancelled by the error accumulated while traveling West and South. The error is actually growing the entire time, but in different directions.

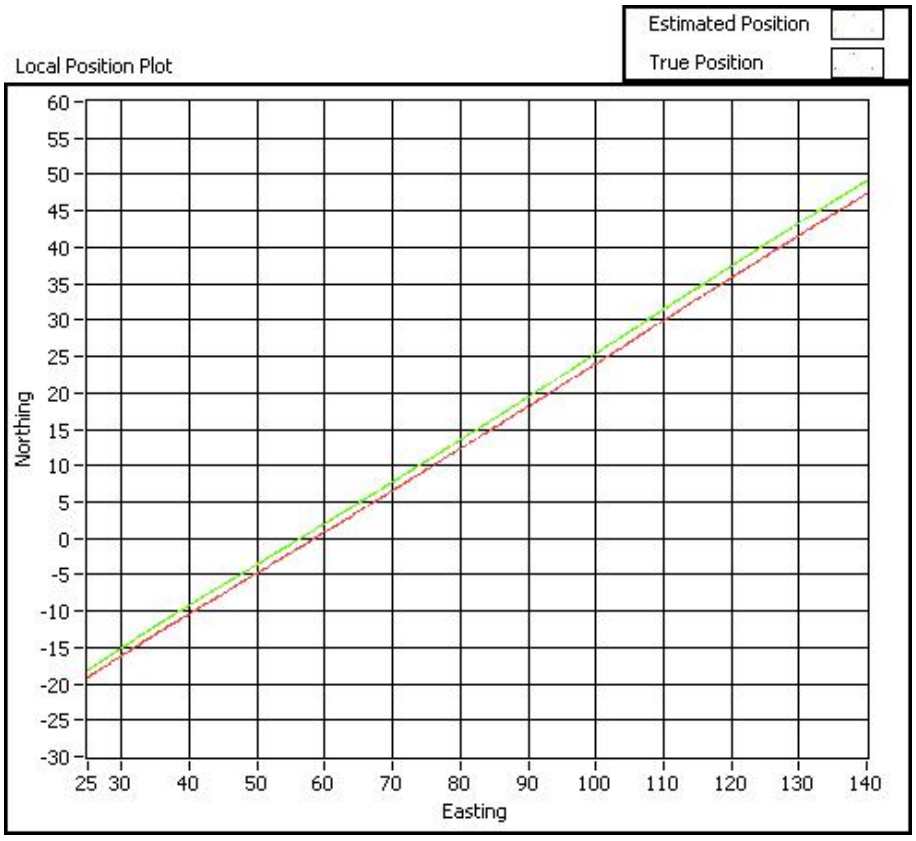

Figure 6.3: This data was collected while Odin was driving on a smooth paved surface. The error will increase on uneven or loose terrain.

After tuning the RPF, this error was minimized to 2.6 meters per 100 meters traveled, or 2.6% of the total distance traveled when traveling a straight path at 15 to 20 mph. This error will increase to 3.4% when the vehicle is traveling on a curved path. This is a direct result of the increase in lateral slip as the vehicle turns.

Chapter 7

Conclusion

The localization software presented in this thesis underwent hours upon hours of testing in the months leading up to the DUC. However, the true test of the software's abilities came when the team and Odin arrived in Victorville, CA for the NQE. This chapter will discuss the localization software's performance during the DUC, the downfalls of the localization software, and changes that can be made in the future to improve its performance and robustness.

7.1 The Urban Challenge

The DUC NQE and final event took place in Victorville, CA from October 26, 2007 to November 3, 2007. The site of the challenge was the former George's Air Force Base, which is now used as a MOUT site and for various other military training exercises. Contrary to its name, the Urban challenge site was not very urban. There were however

plenty of trees and buildings that partially blocked the view of the sky on certain parts of the course. This section will discuss Odin's performance while in Victorville in terms of localization, rather than overall failure or success in completing missions. Therefore any errors not related to or caused by localization failures will not be discussed.

7.1.1 NQE

The NQE tested each vehicle's ability to navigate safely through an urban environment. The test courses ranged from parking lots with moving traffic with which vehicles were required to merge, to suburban neighborhoods with four-way stops. Throughout the NQE, Odin's localization performed flawlessly. Odin experienced no GPS outages, position pops, or offsets. The team could not have asked for a better performance from the localization software, until the last day of the NQE.

The second test for qualification, test area B, led vehicles through what used to be base housing. In this area vehicles were tested on their ability to maneuver past a barrage of disabled vehicles and navigate through zones. Area B contained the most occlusions of any of the test courses, and was the only area in which Odin experienced a localization error. The error occurred as Odin was exiting the traffic circle shown in Figure 7.1. Odin had already navigated through Area B with no localization problems, but on his second attempt, as Odin exited the traffic circle, the position solution provided by the Novatel system popped almost 10 meters to the south west. Although position pops are a regular occurrence when navigating through areas subject to multipath and occlusions, a position pop of this magnitude is rarely seen. As shown in Figure 7.1, there are nearby buildings as well as trees that line the road. These images clearly show that satellite visibility in this area is potentially poor, but this does not completely explain why the Novatel system would experience such a severe jump in position.

Unfortunately, the raw data needed to fully diagnose the cause of the position jump

Figure 7.1: The top picture gives a south facing view and captures Odin in the vicinity of the position pop. The bottom picture provides a north facing view.

was not being logged. This error was especially mysterious given the fact that Odin had already navigated through this area successfully with no localization errors. Several theories concerning the cause of the problem have been suggested. Many believe that DARPA was actively jamming GPS signals at this point in the course to test the vehicle's robustness to GPS interference. There is no evidence to substantiate this claim, and DARPA has never admitted to any such activity. The other popular opinion is that the position jump was caused by multipath and/or satellite occlusions from nearby buildings and trees. While this is a possibility, this is contrary to months of testing with the Novatel system, during which Odin successfully navigated through far more cluttered environments with no localization problems.

While the cause of this problem may never be known, changes can be made to Odin's localization and perception software to handle these types of problems better. Had Odin been actively perceiving the road as it traveled through the NQE course, the RPF position could have been used to guide Odin through the course until a better GPS position measurement could be acquired. All of Odin's perception is done using the RPF position solution which is not susceptible to GPS position pops. Therefore, instead of thinking that the lane had suddenly shifted north, Odin would have maintained the correct lane position and would not have gotten stuck at the end of the segment.

7.1.2 UFE

In spite of the localization problems experienced in Area B during the NQE, Odin was admitted to the UFE and was seeded second among the eleven vehicles that had qualified. Odin was the first to leave the start chute on the morning of the competition and smoothly navigated through the DUC course.

Odin's UFE run was not completely flawless. Aside from other minor errors not related to localization, Odin experienced another position pop when exiting the George

Blvd. traffic circle onto Sabre Rd., the same place that the error occurred during the NQE. The position pop was not as severe as the previous one, and resulted in Odin leaving the road briefly and then returning. Once again, there is little evidence to explain the cause of this position jump. Odin successfully drove this stretch of road several times before the localization error, and had no problems navigating through this error during the remainder of the race. No software changes were made between the NQE and the UFE.

Odin completed all three missions and crossed the finish line with an official time of 276 minutes. Odin placed third behind Boss, of Tartan racing, and Junior, from Stanford.

7.2 Future Work

The software presented in this thesis was successful in the DARPA Urban Challenge. However, several changes can be made to increase the softwares usefulness and robustness. In addition, changes can be made to Odin that would make better use of the features provided by this localization solution.

The APF implemented in this localization solution provided accurate and reliable position estimates, as well as a means for incorporating additional measurements in the future. The APF took advantage of several localization techniques to produce this position estimate, but it is still overly dependent on only one sensor. Without the measurements provided by the Novatel SPAN system, the APF would quickly diverge. This is mainly due to the fact that the system becomes unobservable without the absolute position measurements provided by the Novatel system. In addition, all of the attitude measurements used by the APF were also produced by the SPAN system. Therefore, if this one sensor were to fail, the filter would not be able to maintain position estimates and the localization solution would fail. Future localization sensor suites should at the very least incorporate a low cost GPS to guide the vehicle if the primary sensor fails.

The RPF implemented in this localization solution accomplished its goal of providing a smooth position estimate that is not subject to GPS position pops. Odin's perception modules did not fully take advantage of this. To be successful in an urban setting, autonomous vehicles must be actively sensing the world around them. Furthermore, all perception must be done in a relative frame like the one provided by the RPF. This is the only way to avoid the perception failures caused by GPS position pops. The solution is not to eliminate position pops. In most cases the position is popping to a more accurate measurement, which is desirable.

References

[1] A. Bais, R. Sablatnig, J. Gu. *Single Landmark Based Self-Localization of Mobile Robots*, in Proceedings of 3rd Canadian Conference on Computer and Robot Vision, IEEE Computer Society, 2006.

[2] G. Bekey. *Autonomous Robots: From Biological Inspiration to Implementation and Control*. Cambridge, MA: MIT Press, 2005.

[3] J. Borenstein, L. Feng. *Measurement and Correction of Systematic Odometry Errors in Mobile Robots*, in IEEE Transactions on Robotics and Automation, Vol. 12, No. 6: 869-880, 1996.

[4] R. Brown, P. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*, Third Edition. New York, NY: John Wiley & Sons, Inc.,1997.

[5] Defense Advanced Research Projects Agency. *DARPA Grand Challenge Congressional Mandate*, 2004. Available Online: http://www.darpa.mil/

[6] B. Hofmann-Wellenhof, K. Legat, M. Wieser. *Navigation: Principles of Positioning and Guidance*. Graz, Austria: Springer-Verlag Wien, 2003.

[7] R.E. Kalman. *A New Approach to Linear Filtering and Prediction Problems*, in Transactions of the ASME-Journal of Basic Engineering, 82 (Series D): 34-45, 1960.

[8] A.J. Kelly. *A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles*. Pittsburgh, PA: CMU Robotics Institute Technical Report CMU-RI-TR-94-19, 1994.

[9] A.J. Kelly. *Essential Kinematics for Autonomous Vehicles*. Pittsburgh, PA: CMU Robotics Institute Technical Report CMU-RI-TR-94-14, 1994.

[10] P.S. Maybeck. *Stochastic models, estimation, and control*, Volume 1. New York, NY: Academic Press, 1979.

[11] D. Simon. *Optimal State Estimation: Kalman, H infinity, and Nonlinear Approaches*. New York, NY: John Wiley & Sons, Inc.,2006

[12] S. Thrun, et al. *Stanley: The Robot that Won the DARPA Grand Challenge*. Journal of Field Robotics, 23.9, 662-692, 2006.

Appendix A

Kalman Filter Example

This appendix presents a simple example of a Kalman filter implementation. This is example is intended to provide readers with little or no previous experience with Kalman filtering a chance gain an understanding of how the Kalman filter is applied, without the added complexity of the 3D navigation problem presented in this thesis.

A.1 System Model

We will begin by defining our system. Consider the vehicle depicted in Figure A.1. The vehicle's position is described by its x and y coordinates, and its motion is described by its forward velocity, V, and its heading, θ. The vehicle heading is measured counterclockwise from the x-axis.

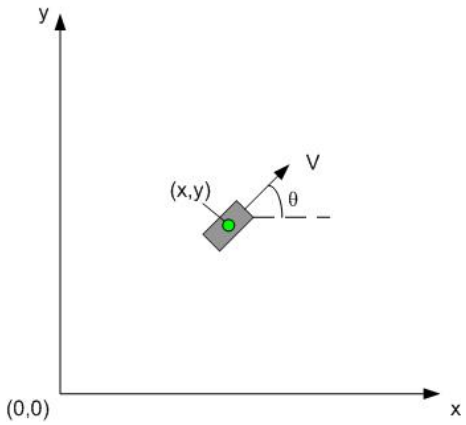

Figure A.1: The vehicle motion is confined to the 2D plane and is described by the forward velocity and heading of the vehicle.

The continuous state-space representation of this vehicle model is given by

$$\dot{\bar{x}} = \Phi\bar{x}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{V} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & cos\theta & 0 \\ 0 & 0 & sin\theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ V \\ \theta \end{bmatrix} \tag{A.1}$$

Notice in this equation that the acceleration of the vehicle, and the yaw rate of the vehicle have been neglected. This amounts to the assumption of a first order Taylor series approximation for the linear position of the vehicle, and a zero order Taylor series approximation for the angular orientation of the vehicle. In addition, there are no deterministic inputs to the system such as steering, braking, or throttle. This means that the V and θ terms will not update naturally. The only way to move these states forward in time is through measurement updates. These will be discussed in the next section.

This example is for a discrete Kalman filter, so the discrete form of this system model

is written as

$$\bar{x}_{k+1} = \Phi_k \bar{x}_k$$

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ V_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & cos\theta_k\, dt & 0 \\ 0 & 1 & sin\theta_k\, dt & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ V_k \\ \theta_k \end{bmatrix} \tag{A.2}$$

A.2 Measurement Model

The measurements that will be used for this Kalman filter are:

GPS

Encoders

Compass

which will provide the measurements:

2D position (x,y)

Forward Velocity

Heading

Having decided on the sensor suite, the measurement model can now be developed. The measurement model is written as

$$\bar{z}_k = H_k \bar{x}_k$$

$$\begin{bmatrix} x_{GPS} \\ y_{GPS} \\ V_{enc} \\ \theta_{comp} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ V_k \\ \theta_k \end{bmatrix} \tag{A.3}$$

Notice that the observation matrix, H, is simply a 4x4 identity matrix. This is a result of

choosing a sensor configuration that provides a direct measurement of all vehicle states. In addition, we now have a way to update the velocity and heading measurements. This is important because as previously mentioned, the system model will not propagate the velocity and heading forward in time since the accelerations and angular rates have been neglected.

A.3 Noise Models

Now that the system model and the measurement model have been developed, the noise associated with each model must be quantified. One major assumption of the Kalman filter is that the noise associated with the system model and the measurement model is an additive white noise sequence with a known covariance structure. This is of course impossible to achieve since true white noise sequences do not exist in nature. In addition, the covariance structure of the sequence, while it may be known well, is rarely known completely. In spite of these deviations from the Kalman filter requirements, the Kalman filter can still achieve an optimal solution.

First we will model the system model noise. For this example, the vehicle being tracked is a simulated vehicle, so the natural process noise that would occur on a real vehicle is not present in this system. Furthermore, even for a real vehicle with real process noise, the errors induced by the assumptions made in developing the vehicle model far outweigh the truly random noise in the system. Therefore, the covariance is assumed to be a function of the terms neglected in the vehicle model development. Finally, the position states, x and y, are orthogonal terms and therefore the covariance between x and y is zero. The velocity and heading states, V and θ, are not orthogonal to each other, or to the position states, so the noise between these states and other states is most likely correlated. However, how these state errors are correlated is not well known. For this

reason, the off diagonal terms in the system covariance matrix, Q, will be assumed to be zero. Any error induced by this assumption will be handled by the filter covariance matrix, P. The off diagonal covariance terms in P will evolve naturally as the filter updates.

Having applied the for mentioned assumptions, the diagonal elements of the system covariance matrix, Q, must now be approximated for each state. For most systems, it is more intuitive to think about the errors associated with the vehicle model in the vehicle frame. The covariance matrix would then be rotated into the navigation frame before it is used in the Kalman filter equations. Given the relative simplicity of the system in this example, the necessary rotations will be added directly to the diagonal terms in Q.

The noise associated with the linear position states, x and y, is due primarily to the first order Taylor series approximation of the system. As previously mentioned, the true random process noise would contribute on a real vehicle, but would have much less of an effect on the system model than the approximation errors. The variance of the linear position states will therefore be approximated by the equations

$$\sigma_x^2 = \left(\frac{a_{max} (\Delta t)^2}{2} cos\theta_k \right)^2 \tag{A.4}$$

$$\sigma_y^2 = \left(\frac{a_{max} (\Delta t)^2}{2} sin\theta_k \right)^2 \tag{A.5}$$

where a_{max} is the estimated maximum forward acceleration of the vehicle, Δt is the time step of the Kalman filter, and θ_k is the current estimate of the vehicle heading. The lateral and vertical accelerations of the vehicle have been completely neglected in this approximation.

The noise associated with the velocity and heading states will be approximated as functions of the neglected vehicle accelerations and vehicle angular rate respectively.

These terms are given by the equations

$$\sigma_V^2 = (a_{max}\Delta t)^2 \tag{A.6}$$

$$\sigma_\theta^2 = \left(\dot{\theta}_{max}\Delta t\right)^2 \tag{A.7}$$

where a_{max} is the same maximum forward acceleration from the linear position noise approximation, Δt is the Kalman time step, and $\dot{\theta}_{max}$ is the estimated maximum yaw rate of the vehicle.

Having estimated the noise associated with each state, the result is a system covariance matrix that can be written as

$$Q = \begin{bmatrix} \left(\frac{a_{max}(\Delta t)^2}{2} cos\theta_k\right)^2 & 0 & 0 & 0 \\ 0 & \left(\frac{a_{max}(\Delta t)^2}{2} sin\theta_k\right)^2 & 0 & 0 \\ 0 & 0 & (a_{max}\Delta t)^2 & 0 \\ 0 & 0 & 0 & \left(\dot{\theta}_{max}\Delta t\right)^2 \end{bmatrix} \tag{A.8}$$

Although these terms due not represent the true variance of the system states in a statistical sense, they do provide a good estimate of how the motion of the vehicle will deviate from the trajectory predicted by the system model, and that is the purpose of the Q matrix.

One qualitative interpretation of the Q matrix is that it is an estimate of how fast the state estimates will diverge if the process model equations are run open loop without measurement updates [8]. This is why it is not as important to analytically or numerically determine the true random process noise found in the system. For the system in this example, the primary contributors to the error in the system model are the acceleration and angular rate terms neglected in the modeling process. For some systems, it may be very important to determine the true random process noise, especially if that is the predominant error source in the system.

The final model to be determined before the Kalman filter can be applied is the measurement noise model. The noise associated with the individual measurements used in this example will be approximated using the specifications of the sensors. Since the sensors used are simulated sensors, the approximation of their noise is completely arbitrary. For this reason, the sensor noise for each sensor will be set to a constant and that constant will be adjusted to illustrate the effects of sensor noise on the state estimates. Just as with the system noise model, the measurement noise terms are assumed to be uncorrelated. This is usually a good estimate, especially for measurements provided by completely different sensors. The measurement covariance matrix, R, is therefore given by

$$R = \begin{bmatrix} \sigma^2_{GPS} & 0 & 0 & 0 \\ 0 & \sigma^2_{GPS} & 0 & 0 \\ 0 & 0 & \sigma^2_{Enc} & 0 \\ 0 & 0 & 0 & \sigma^2_{Comp} \end{bmatrix} \tag{A.9}$$

A.4 The Kalman Filter

Now that vehicle and the sensors have been completely modeled, these models can be used in the Kalman filter equations. The first step in the Kalman filter process is to initialize both the system states, and the filter covariance matrix. For simplicity, we will assume that the vehicle is starting at the origin of the navigation frame and will have zero velocity and heading. The initial value of the filter covariance, P, depends on how well the user knows the initial state of the vehicle, $\bar{x}_0$. If $\bar{x}_0$ is perfectly known, the initial value of P is zero. If $\bar{x}_0$ is not known at all, then P should be initialized to a very high value. Once measurements are incorporated and more knowledge is gained about the trajectory of the vehicle, the value of P will be updated and will decrease to a more reasonable value. Although the initial conditions of the vehicle for this example are perfectly known, P will

be initialized to a very small value, but not zero. This is because initializing P to zero will result in a trivial solution for all of the measurement update equations for the first cycle. This will become clear as the Kalman equations are presented.

We will now take the states, $\bar{x}$, and the covariance, P, through one step of the filter. The first equation run in the Kalman filter is the calculation of the Kalman gain, K. The Kalman gain equation is given by

$$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k \right]^{-1} \tag{A.10}$$

where P_k^- is our current estimate of the filter covariance before the measurements are incorporated as denoted by the $^-$. Notice that if the covariance matrix P_k^- were zero, the gain matrix, K, would also be zero and none of the measurements would be used in the following state update equations. This is the trivial solution previously mentioned and is the reason that P_0 was not initialized to zero. This equation gives the optimal gain for blending the measurements with the state estimates. This particular K is optimal because it minimizes the mean-square errors in the estimates of the state vector, $\bar{x}$, given by the major diagonal terms of the matrix, P_k. The proof that K is in fact the optimal gain can be found in [4].

Having determined the weighting for each measurement via the Kalman gain matrix, K, the measurements can now be used to update the state estimate. This is done using the equation

$$\hat{x}_k = \hat{x}_k^- + K_k \left[\bar{z}_k - H_k \hat{x}_k^- \right] \tag{A.11}$$

where the $\hat{}$ denotes an estimate of the state vector, x. In this equation, the a priori state estimate is being updated with weighted estimates of the error as given by the measurements. The error, defined in this equation by the difference between the measurements and the predicted state estimates, is weighted using the Kalman gain, K, and then added

to the predicted state estimate. In this way the filter is using the measurements to correct the error in the state estimate predicted using the system model. If the measurements are not reliable (i.e. the elements of the measurement covariance matrix, R, are high), the Kalman gain will be small and the measurements will have little influence on the state estimate.

Now that the measurements have been used to update the state estimate, the covariance estimate must also be updated. The covariance update equation is

$$P_k = [I - K_k H_k] P_k^- \tag{A.12}$$

This equation provides a means of updating the filter covariance based on the Kalman gain matrix, K. However, this is not the only form of this equation that can be used for updating P. This is the simplest from of the covariance update equation, but it is only valid if we assume that K is the optimal gain. More generic forms are valid for optimal or suboptimal conditions and can be found in [4] and [11]. For this example we will assume that K is optimal and use the simple version of the P update equation given above. Note however that in most real life Kalman filter applications, K is rarely the optimal gain. This is due to the fact that it is often difficult to perfectly model any system, and the differences between a process model and the true process will lead to a suboptimal K and therefore a suboptimal estimate [4]. For this reason it is often safer to use more generic forms of the P-update equation to ensure that the estimate will be valid for any K.

The covariance update equation concludes the measurement phase of the Kalman filter. An state estimate, $\hat{x}_k$, and a covariance estimate, P_k, have been determined based on an 'optimal' combination of model predictions and measurements. The next phase in the Kalman filter process is to predict the states for the next cycle of the filter. To project the

system forward one time step to time t_{k+1}, we use the equation

$$\hat{x}_{k+1}^{-} = \Phi_k \hat{x}_k \tag{A.13}$$

where $\hat{x}_{k+1}^{-}$ is the a priori state estimate that will be used for the next cycle. This prediction is based entirely on the dynamics of the system as given by the state transition matrix, Φ_k.

Likewise, P must be predicted for the next cycle. The projection of P into the next time step is given by

$$P_{k+1}^{-} = \Phi_k P_k \Phi_k^T + Q_k \tag{A.14}$$

where P_{k+1}^{-} is the a priori estimate of the filter covariance for the next cycle of the filter. Once again, this estimate is based only on the system model and the system noise model. Also note that at this point all of the models developed in the previous sections have been used, and one full cycle of the Kalman filter is now complete.

A.5 Kalman Filter Tuning

Now that the pieces of the Kalman filter have been described, as well as the process for putting these pieces together, this section will explain how the tuning of the filter can determine its behavior. The process of tuning a Kalman filter involves adjusting system and measurement error estimates to achieve the desired behavior from the filter. For instance, the a_{max} term in the diagonal elements of the system covariance matrix, Q, can be adjusted to increase or decrease the covariance estimate for the position and velocity states in $\bar{x}$. Neither Q nor R may be exactly known when developing the system and measurement noise models, which is why the tuning process is necessary when implementing a Kalman filter. The update rate of the Kalman filter is also a factor that can be adjusted during the tuning process, and this is often determined by the rate at which measurements

are available.

Another important point to remember when tuning a Kalman filter is that the behavior of the filter is determined by the relative magnitude of the Q and R matrices. Whether or not Q and R have been accurately estimated, optimal filter performance can still be achieved as long as the ratio of Q to R is correct. To illustrate this, simulated data will be run through the Kalman filter for this system at varying values of Q and R.

To begin, the values of the R matrix will be set to constants. The measurements used in this example are simulated measurements, therefore we can decide exactly what the covariance of each measurement will be. Assume that the R matrix is given as

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{A.15}$$

The variance of the GPS measurements has been set to $1m^2$ and the variance of the velocity and heading measurement is set to 0.01. For the velocity and heading measurements the goal is to set the measurement error very low so that the filter will use these measurements almost exclusively instead of the predictions.

For the first test case, the maximum acceleration will be set to a very high value so that Q will be given as

$$Q = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 40000 & 0 \\ 0 & 0 & 0 & 400 \end{bmatrix} \tag{A.16}$$

The velocity and heading error terms are several orders of magnitude higher than the position error terms because they are directly proportional to Δt rather than Δt^2. For this

case, the process noise is estimated to be greater than the measurement noise. Figure A.2 shows the filter estimates plotted over the GPS measurements. As shown in the figure, the filter estimates are scattered, and tend to follow the measurements very closely. This is due to the fact that the measurements are being weighted very heavily compared to the process model predictions. The K matrix for one cycle of the filter equations is calculated as

$$K = \begin{bmatrix} 0.83 & 0 & 0 & 0 \\ 0 & 0.83 & 0 & 0 \\ 0 & 0 & 0.99 & 0 \\ 0 & 0 & 0 & 0.99 \end{bmatrix} \tag{A.17}$$

Notice that all of the diagonal terms in K are very close to one indicating that the measurements are very trustworthy and will have much more weight in the filter estimate than the process model predictions.

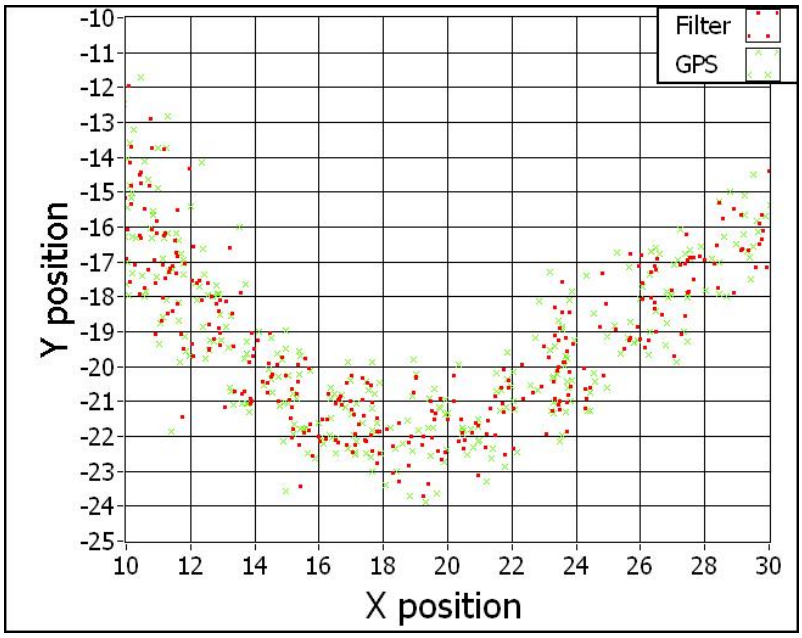

Figure A.2: Results when position terms in Q are much higher than GPS terms in R.

For the next case, the maximum acceleration terms in Q will be set to a more realistic

value of 1g. At a filter update rate of 50 Hz, the Q matrix is

$$Q = \begin{bmatrix} 4E^{-6} & 0 & 0 & 0 \\ 0 & 4E^{-6} & 0 & 0 \\ 0 & 0 & 40000 & 0 \\ 0 & 0 & 0 & 400 \end{bmatrix} \tag{A.18}$$

The velocity and heading terms have been left the same as the previous case to ensure that the measurements are still used over the predicted velocity and heading. Running the filter with this Q matrix gives the results shown in Figure A.3. Notice in this case that the solution is very smooth and is closer to what the actual motion of the vehicle would be. However, this position estimate diverges from the GPS measurements as the vehicle travels. This is due to the fact that a very small weight has been assigned to the GPS measurements and the filter is effectively running the open loop dead reckoning equations with only velocity and heading measurement updates. The K matrix after the filter has been running for some time is

$$K = \begin{bmatrix} 0.002 & 0 & 0 & 0 \\ 0 & 0.002 & 0 & 0 \\ 0 & 0 & 0.99 & 0 \\ 0 & 0 & 0 & 0.99 \end{bmatrix} \tag{A.19}$$

The elements of K associated with the GPS measurements are extremely small and as a result, the GPS measurements have little effect on the filter estimate.

In the previous case, the maximum acceleration term in the Q matrix was set to a perfectly reasonable value for a vehicle driving at a reasonable speed under normal conditions. This brings us to an important point. Regardless of what the true value of the system or measurement covariance is, if the filter does not produce useful results, these

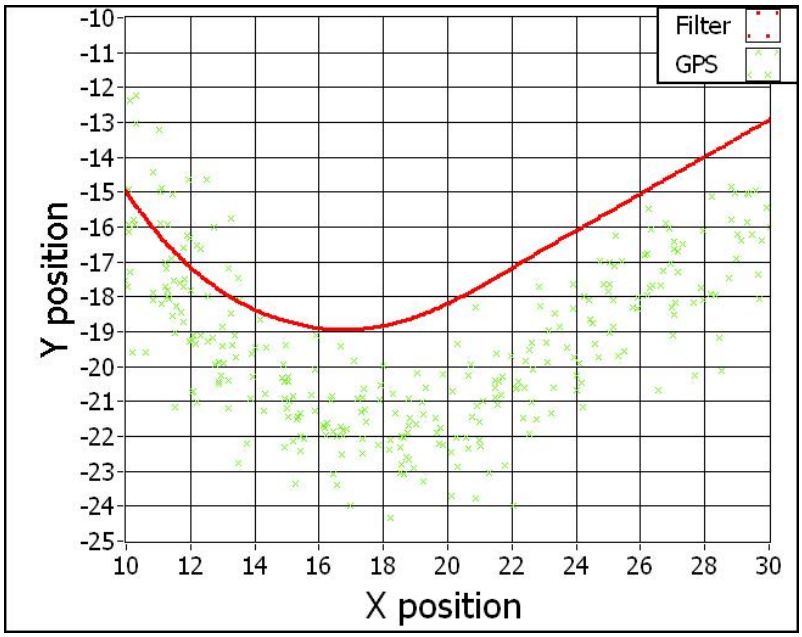

Figure A.3: Results when position terms in Q are much lower than GPS terms in R.

values must be changed until the desired behavior is achieved. Obviously a position filter that is completely diverged from the true position of the vehicle will not be useful for localizing a vehicle. For the next case, the maximum acceleration term in Q will be raised to $100\frac{m}{s^2}$. This is obviously not a reasonable estimate of the maximum acceleration that the vehicle is likely to experience under normal conditions, but as shown in Figure A.4 this produces a better estimate of the true position of the vehicle. In addition, the filtered position has smoothed the GPS measurements resulting in a continuous path that better represents the true path of the vehicle.

For the final case in this example, both the Q and R matrices will be multiplied by 100. Both matrices will have completely different values than they did in the previous case, but as shown in Figure A.5, the resulting filter estimate is identical. As mentioned previously, it is not as important that Q and R be accurately known as much as it is important that their relative magnitude be correct. For this case Q has increased to a value that makes even less physical sense that it did previously, and R has increased to a value that is known

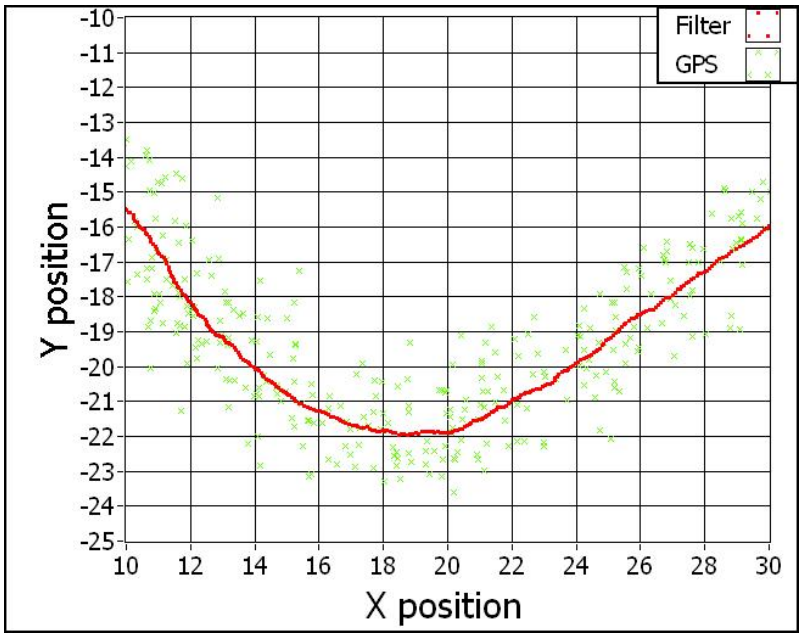

Figure A.4: Results with optimal tuning of Q and R matrices.

to be incorrect. However, the ratio of Q to R has remained constant, and the behavior of the filter remains unchanged.

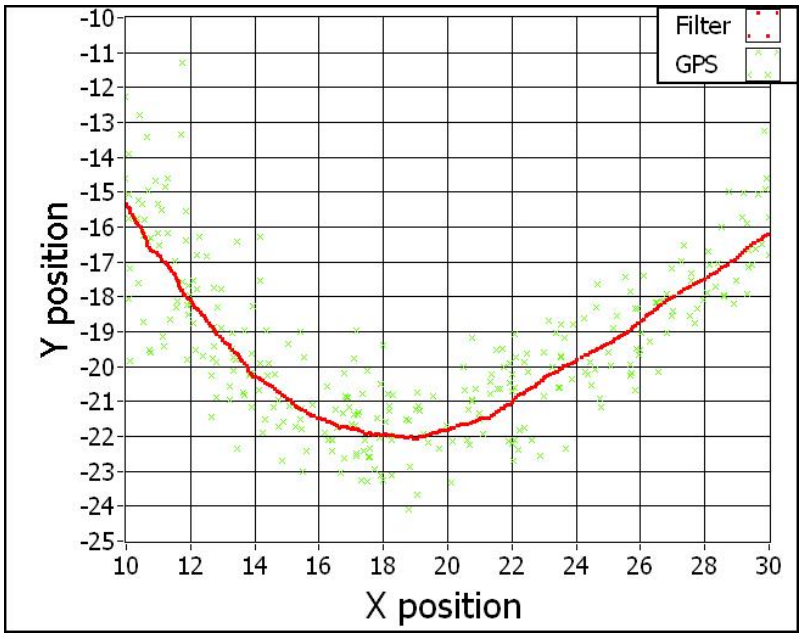

Figure A.5: Results with Q and R increased by one order of magnitude.

A.6 Conclusion

The implementation of a Kalman filter can be a tedious process, but the difficulties rarely involve the Kalman filter equations themselves. The difficulty lies in modeling the system and the measurements and determining at least a rough estimate of the noise associated with each one. Once this is done, hours upon hours must be spent calibrating sensors and tuning the noise models until the desired behavior is achieved from the filter. The Kalman filter equations themselves are very straight forward and do not need to be rederived or adjusted to be used on a particular problem. This is what makes the Kalman filter such a useful tool and is the reason it has been used on such a wide variety of applications.