FILTER-BASED SLIP DETECTION FOR A COMPLETE-COVERAGE ROBOT

by

EDWARD JOSEPH KREINAR

Submitted in partial fulfillment of the requirements
For the degree of Master of Science

Thesis Advisor: Dr. Roger Quinn

Department of Electrical Engineering and Computer Science
CASE WESTERN RESERVE UNIVERSITY

August, 2013

CASE WESTERN RESERVE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis of

EDWARD JOSEPH KREINAR

candidate for the Master of Science degree*.

Roger Quinn

Committee Chair

M. Cenk Cavusoglu

Committee Member

Francis Merat

Committee Member

(date) April 26, 2012

* We also certify that written approval has been obtained for any proprietary material contained therein.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank those involved in the CWRU Cutter robot snow-mower team: Dr. Quinn, CWRU Cutter advisor, for his support of the project and for the opportunity to lead the team during the past two years; Bradley Hughes for his mentorship and outstanding contributions which helped begin CWRU Cutter's long and successful life at Case; Graduate students Lu Li and Charles Hart for their help with robot development and for participating in the 2013 Autonomous Snowplow Competition; all mower team members during the past three years at Case, especially David Chrzanowski, Nick Badger, Tom Shkurti, and Henry Snow.

I would also like to thank Dr. Frank Merat and Dr. Cenk Cavusoglu for generously serving on my thesis committee, and for providing the academic foundation on which this thesis is built.

The CWRU Cutter project is funded by MTD Products, Inc. I would like to thank the Robotics Team at MTD, especially Steve Smith and Micah Wolf, for their continued support of the CWRU Cutter project, and Jim Green for initiating the mower project in 2008. My time working on the CWRU Cutter robot has been my most valuable learning experience at Case.

Finally, thank you to my family for their constant support, encouragement, and motivation which allowed me to get to where I am today.

List of Abbreviations

AKF	Augmented Kalman Filter
ALC	Autonomous Lawnmower Competition
AMCL	Adaptive Monte Carlo Localization
ASC	Autonomous Snowplow Competition
BRB	Big Red Button
cRIO	Compact Real-Time Input/Output
CWRU	Case Western Reserve University
EKF	Extended Kalman Filter
FKF	Federated Kalman Filter
FPGA	Field Programmable Gate Array
GPS	Global Position System
IEKF	Iterated Extended Kalman Filter
IMU	Inertial Measurement Unit
ION	Institute of Navigation
KF	Kalman Filter
LIDAR	Light Detection and Ranging

MCL	Monte Carlo Localization
NI	National Instruments
PCB	Printed circuit board
PWM	Pulse width modulation
ROS	Robot Operating System
RTK	Real-Time Kinematic
UKF	Unscented Kalman Filter
VHDL	VHSIC Hardware Description Language

Filter-Based Slip Detection

for a Complete-Coverage Robot

Abstract

by

EDWARD JOSEPH KREINAR

Complete-coverage robots, such as a lawnmower or snowplow, require a centimeter-level localization solution in order to navigate reliably. Unmodeled wheel slip or other odometry errors may cause localization to diverge beyond the bounds of uncertainty. Specifically in the case of a robot snowplow, errors due to wheel slip may be significant. This thesis uses the CWRU Cutter autonomous robot as a test platform to address the dual issues of (1) robust localization and (2) odometry error handling. Both an Extended Kalman Filter and an Adaptive Monte Carlo Localization procedure are derived and implemented specifically on the CWRU Cutter robot. Finally, a new augmented Extended Kalman Filter with general-purpose wheel-velocity error states is derived. The augmented EKF is shown to fully estimate the robot state and the wheel velocity error due to wheel slip during logged data from the 2013 Institute of Navigation's Autonomous Snowplow Competition.

1 Introduction

Over the past decade, consumer robots have become more prevalent with the introduction of faster, smaller, and more affordable processors. However, in order to develop intelligent consumer mobile robotics, a precise, robust, and inexpensive localization system is critical. When put into the context of an outdoor mobile robot, localization must be accurate on a centimeter level in order to produce complete-coverage path planning, obstacle avoidance, and high level operation which rivals the effectiveness of a human performing the same task.

Sensors typically used for academic robotics research (differential GPS, Laser Range Finder, etc) are prohibitively expensive for consumer robotics ($10,000 to $20,000 per sensor). As a result, intelligent complete-coverage robots in the consumer space are difficult to find. The iRobot line of vacuum cleaners, for example, exhibits only limited planning capabilities: unguided "bouncing" around an indoor area [1]. Similar robot lawnmowers are currently in production by Husqvarna, Friendly Robotics, and LawnBott [2][3][4]; they sell for $1000-$3000 depending on the model. Unfortunately, these lawnmower robots all operate using random motion and require a buried wire around the yard perimeter. As of the time of writing, no consumer robot lawnmower performs localization or drives organized paths. No consumer autonomous robot snowplow currently exists at all.

At the $1000 to $3000 consumer target price point, the precision sensors enabling intelligent robotics research throughout academia clearly cannot directly translate to consumer robots. For this reason, a primary area of interest for the

CWRU Cutter autonomous lawnmower team, and for this thesis in particular, is establishing outdoor-capable localization and filtering algorithms. Ideally, filtering algorithms should be able to fuse data from multiple low-cost, low-precision sensors in order to generate a comparable location solution to the expensive sensor result.

Secondly, localization must be as robust as possible in order to remain within 5-10cm location tolerance at all times. As a specific example, during the 2013 Insitite of Navigation's Autonomous Snowplow Competition, snow-pushing robots commonly experienced significant wheel slip (non-systematic odometry error)– in fact, wheel slip was occasionally of such magnitude that unmodeled odometry error caused the CWRU Cutter localization solution to diverge beyond the bounds of uncertainty. Because of this, plowing snow illustrates a significant issue with mobile robot localization robustness: if a robot experiences unmodeled sensor error, localization is prone to malfunction. Encoder-based errors are particularly troublesome because wheel encoder sensors will be necessary on a consumer autonomous robot. This thesis addresses the wheel-slip problem by improving the robustness of the CWRU Cutter localization algorithm.

Given the long terms goals of localization and filter robustness for complete coverage robots, the specific goals for this thesis are threefold:

1. Revise CWRU Cutter software and electronics in order to support the desired algorithms and fast-turnaround development.

2. Derive and implement standard localization algorithms (Extended Kalman Filter and Adaptive Monte Carlo Localization) using the CWRU Cutter robot platform.

3. Modify localization to estimate an arbitrary wheel-slip/non-systematic odometry error. Demonstrate the improved localization algorithm handles actual wheel-slip errors without diverging, thereby improving filter robustness in the wheel-slip case.

All above goals are achieved and will be discussed in this thesis. The improved CWRU Cutter robot provides a software and hardware platform upon which localization and odometry error handling can be developed. Localization theory for both Extended Kalman Filter and Adaptive Monte Carlo Localization is derived and implemented on the CWRU Cutter robot using high-precision sensors. Finally, mobile robot wheel slip (non-systematic odometry error) is estimated using a new augmented Extended Kalman Filter (AKF) with general-purpose wheel-velocity error states, derived here. It is shown that a judicious choice of robot states in the Extended Kalman Filter causes the augmented EKF to become a conceptually simple modification. The augmented EKF is implemented on the CWRU Cutter robot platform, and data logs from the 2013 Institute of Navigation Autonomous Snowplow Competition demonstrate the proposed AKF effectively estimates and localizes in the presence of significant wheel slip.

It is important to note here that low-cost sensors are not yet tested. Rather, the research incorporates available precise (yet expensive) sensors in order to develop a fundamentally-sound theory and foundation for CWRU Cutter localization; if the localization theory is incorrect, neither precise nor imprecise sensors will function correctly.

This thesis is organized as follows: Chapter 2 briefly discusses mobile robot localization research with a specific focus on methods for handling encoder faults. Chapter 3 details the CWRU Cutter software, electronics, and architectural improvements during the course of this thesis research. Chapter 4 outlines the localization approaches derived and implemented on the CWRU Cutter robot (both Extended Kalman Filter and Adaptive Monte Carlo Localization). Chapter 5 demonstrates the need for robust wheel-slip handling and proposes the augmented Extended Kalman Filter with wheel-velocity error states. Chapter 6 presents localization results in the no-slip situation, in the presence of wheel slip, and finally under a systematic odometry error (an incorrect odometry parameter).

2 Background

2.1 CWRU Cutter Project Overview

The "CWRU Cutter" robot family is a series of intelligent, autonomous mobile robots designed and built at Case Western Reserve University for two primary purposes:

1. A prototype platform for various research pursuits
2. A competition entry for the Institute of Navigation's Autonomous Lawnmower Competition (ALC) and Autonomous Snowplow Competition (ASC)

Beginning in 2008, the CWRU Cutter project explored the feasibility of autonomously mowing grass using commercially available sensors for the purpose of succeeding at the Institute of Navigation's Autonomous Lawnmower Competition. The 2012 robot, known as "CWRU Cutter Cinco," is the fifth iteration of the robot which competed in the 2012 Autonomous Lawnmower Competition. To learn additional details about the previous robot, the reader should refer to the technical report submitted for the 2012 ALC [5].

The new robot platform, significantly modified during the process of this thesis development, is known as "CWRU Cutter Hex." CWRU Cutter Hex competed in the 2013 Institute of Navigation's Autonomous Snowplow Competition. During the snowplow competition, teams are required to push as much snow as possible out of a given rectangular area within a fixed time limit.

The CWRU Cutter research team is a small multidisciplinary group of 6-8 undergraduate and graduate students at Case Western Reserve University in the

Biologically Inspired Robotics Lab. Students represent backgrounds from the Electrical Engineering, Mechanical Engineering, Computer Science, and Computer Engineering fields.

Previous research on the CWRU Cutter lawnmower robot addresses sensor development for localization and obstacle-detection. Beno developed a gimbaled LIDAR mount, allowing for forward obstacle detection in 3 dimensions rather than the typical planar range scan [6]. Also dealing with obstacle detection, Schepelmann used image processing to identify grass and avoid areas which are not grass [7]. Image processing was shown to detect common lawn obstacles at around 80% success rate.

Custom localization sensors have also been developed for the CWRU Cutter robot: In [8], Bennett developed a proof-of-concept triangulation procedure using Received Signal Strength Indicators (RSSI) from a rotating radio-frequency antenna; unfortunately, localization could not achieve sub-meter accuracy necessary for a complete coverage robot. Hughes implemented an open-source library for Real Time Kinematic differential GPS processing, lowering the cost for an RTK GPS sensor by a factor of 30 [9].

While many custom and commercial sensors have been investigated with varying degrees of success throughout the CWRU Cutter project so far, CWRU Cutter research has not yet carefully investigated the issue of sensor filtering to improve robustness for a complete-coverage robot. In this sense, this thesis poses a different research question: instead of asking "how can CWRU Cutter use *more* sensors?" this thesis asks "how can CWRU Cutter better use its *current* sensors?" With a long term goal to lower overall robot cost, localization theory and filter robustness is a valid area of concern for the CWRU Cutter project.

2.2 Mobile Robot Localization

State estimation and localization are central concepts throughout this thesis. State estimation, a general term, refers to the process of observing and combining measurements of a system in order to generate an approximation of the true state. State estimation is necessary in situations where sensors do not directly measure all states or where multiple sensors are used to measure the same states. In addition, state estimation may also attempt to filter out measurement noise on the sensor readings.

In the field of mobile robotics, the specific process of determining the robot's position through state estimation is known as localization. This section discusses popular parametric state estimation approaches (with a focus on the Kalman Filter), and non-parametric state estimation approaches (with a focus on Monte Carlo Localization). In the simple case, localization assumes that a perfect map of the environment is known ahead of time. When the map is not known, the problem requires simultaneous localization and mapping (SLAM), which is not considered in this thesis. Both localization approaches discussed here, parametric and non-parametric filters, are implemented on the CWRU Cutter robot as baseline localization algorithms. Chapter 4 discusses the CWRU Cutter-specific implementation of localization algorithms.

State Estimation: Parametric Filters

State estimation originated from state observers in control theory [10]. The concept of a state observer is that knowledge of a particular system's dynamics allows the entire state to be determined from sensor measurements which may be sensitive to only a subset of the state. A well-known observer is the Kalman Filter [11], which generates a statistically optimal state estimate from noisy input measurements. The Kalman filter (FK) is a parametric, linear observer that represents all estimates using a Gaussian distribution. The result is a mean state, which is the optimal estimate, and a covariance matrix, which is the uncertainty on the state

estimate. There are a few critical constraints on Kalman filter use: The system must be linear, and the system must be uni-modal (i.e., only one state hypothesis) exhibiting zero-mean Gaussian noise.

Differential drive mobile robots such as the CWRU Cutter autonomous lawnmower are not linear systems. Because of the system dynamics, an Extended Kalman Filter must be used in order to correctly propagate covariance through the filter [12]. Essentially, the Extended Kalman Filter (EKF) linearizes the system around the estimated operating point during each filter iteration using Jacobians of the system and measurement functions. The EKF is a relatively fast algorithm (it scales quadratically with additional states) which works well for many nonlinear situations, but it retains two drawbacks: The system must always be uni-modal, and the result is no longer theoretically optimal like the linear Kalman filter. Nevertheless, because the EKF can handle a large subset of nonlinear systems, it remains a widely-used algorithm for robotics and control systems today. The EKF theory and implementation for CWRU Cutter is covered in detail in Section 4.1.

To address some of the EKFs deficiencies, other parametric filter adaptations have evolved:

- The Iterated Extended Kalman Filter [13] performs the same measurement update multiple times, linearizing around an increasingly accurate state estimate until a certain repeatability threshold is reached. The Iterated EKF has demonstrated better performance than the EKF, but requires more calculation time.

- The Unscented Kalman Filter (UKF) propagates "sigma points" through the filter [14], which eliminates some of the computational complexity involved with the EKF because Jacobian matrices are not necessary. The UKF, however, requires additional tuning to find a suitable sigma-point distribution.

- The Federated Kalman Filter (FKF) decomposes a centralized Kalman Filter

into multiple sub-filters [15]. Each sub-filter receives its own sensor measurement. The FKF benefits fault detection and isolation; if a sensor fault is detected, it can be removed from the overall solution easier than with the typical Kalman Filter.

State Estimation: Non-Parametric Filters

Non-parametric filters, in contrast to Kalman Filters, are not constrained by any predetermined probability distribution. Non-parametric/ Bayesian techniques (known broadly in mobile robotics as Markov localization) attempt to calculate the probability that the robot is at any possible location based on all sensor readings. As a result, Markov localization techniques offer distinct advantages in certain situations:

1. When the robot pose is either completely unknown or initialized incorrectly.
2. When sensor measurements create a nonlinear, multi-modal estimate of the state.

Because these situations are frequently encountered when developing a robust, noise-tolerant mobile robot, Markov localization has grown in popularity since the 1990s.

Monte Carlo Localization (MCL) , developed in [16] by D. Fox, et al., is a particle filter, sampling-based solution which was created to address deficiencies with more computationally-intensive Markov localization. MCL outperforms grid-based localization in terms of computation requirements, improves accuracy, and also retains the desirable qualities of all Markov localization techniques. The particle filter procedure will be covered in more detail in Section 4.2.

After MCL was developed, several significant variations and improvements on the original MCL particle filter were proposed.

1. The Mixture MCL algorithm [17] diverts a small subset of particles to sample the PDF of the absolute sensor (whereas the typical MCL algorithm samples

the PDF of relative sensors only, such as wheel encoders). It is shown that Mixture MCL helps localize when using extremely accurate sensors, or when the state estimate diverges unexpectedly.

2. KLD Sampling [18] varies the number of particles used for MCL based on a dynamically computed statistical bound. KLD sampling produces a computational improvement without sacrificing overall precision.

With these developments, MCL and other particle filter-based algorithms have an enduring foothold in mobile robot localization due to their noise-tolerance and multi-hypothesis tracking.

2.3 Odometry Error Handling

Wheel encoders provide a precise and inexpensive measurement of wheel rotation. When used on a mobile robot, encoders measure rotational distance which is directly proportional to each wheel's angular velocity and (assuming no odometry errors or wheel slip) forward velocity. These wheel velocity measurements constitute robot "odometry." Wheel odometry is critical for robot filtering algorithms in order to smooth absolute sensor data.

Odometry errors are referred to in literature as either "systematic" or "non-systematic" errors [19]. A systematic error typically results from incorrectly calibrated odometry parameters: the encoder ticks-per-meter conversion (wheel diameter) or the robot track width. A non-systematic error, on the other hand, represents error from the robot environment: wheel slip, external forces, etc. For the purposes of this thesis, the term "wheel-slip" will be used interchangeably with non-systematic odometry error.

Systematic odometry errors are well-discussed; many researchers appreciate the need for an odometry parameter "autocalibration." An early method to estimate systematic odometry errors is the UMBMark procedure from 1994 [19], which requires a robot to drive a predefined square path at least ten times. More re-

cently in 2006, the Bi-Directional Circular Path Test requires a robot to drive a circle in both the clockwise and counterclockwise directions, after which data post-processing reveals the systematic odometry errors [20]. Online algorithms are also used to estimate systematic odometry errors; Martinelli [21]and Caltabiano [22] implement an augmented Kalman Filter with dedicated states for each odometry calibration parameter (right/left wheel ticks-per-meter and track width). When fused with other sensors (e.g., GPS or Lidar), the augmented KF estimates odometry calibration during localization. In 2010, Antonelli et al. demonstrates simultaneous odometry and camera calibration using only onboard measurements and no other supplemental sensors [23].

Because systematic odometry errors are well-discussed in literature and can in fact be calibrated prior to running on the CWRU Cutter robot, systematic odometry errors are primarily not of interest in this thesis. It is, however, worth noting that the proposed augmented KF with wheel-velocity error states can also handle systematic odometry errors (but it cannot, at this point, estimate these errors).

Non-systematic odometry error has not been addressed as comprehensively as systematic odometry error. An early approach [24] attempts to use redundant sensor data in a unique robot featuring a separate encoder trailer wheel to better estimate non-systematic odometry errors. In 2003, Rudolf implements an approach to non-systematic error estimation by tracking position using an error-state Kalman Filter (as is often used for inertial navigation) [25]. The algorithmic complexity of Rudolf's error-state KF is significant. Adaptive particle filters are used by Zajac in 2011 [26] for robot self-diagnosis, which is shown to effectively filter robot faults and even detect a flat tire. In another approach, Yi, et al., kinematically models wheel slip error for a skid-steer mobile robot [27]. Wheel-slip velocity errors are then detected using an error-state EKF, which is derived and implemented on a robot test platform, but the kinematics include specific constraints for skid-steered robots which do not apply to the CWRU Cutter differential drive robot. In 2006,

a study on slip-compensated path driving for a Mars rover [28] uses an error-state EKF with an independent statistical threshold filter based on wheel odometry and visual odometry.

The closest-related approach to this thesis for non-systematic error estimation on a differential drive robot is a procedure discussed in Martinelli, et al. from 2007 [29]. Martinelli presents a framework for estimating both systematic and non-systematic errors for a differential drive robot, implementing two separate filters: (1) An augmented KF for systematic error estimation, and (2) an "Observable Filter" which stochastically monitors the output of the first augmented KF. Experimental results, conducted in an indoor environment with a smooth floor, showed better than 1% error filtering systematic odometry error, yet 90% error filtering non-systematic odometry error.

The non-systematic error estimation proposed in this thesis implements a similar approach as [21][22][29] using an augmented Kalman Filter. Critical differences compared to past approaches for non-systematic error estimation are: wheel-slip detection uses a centralized Kalman filter as opposed to multiple filters, whole-value position measurements as opposed to error-states, and an "analog" estimate of wheel-slip as opposed to a Boolean threshold value. As will be seen in Chapters 5 and 6, the proposed augmented EKF greatly eliminates algorithmic complexity over previous approaches and is able to localize the CWRU Cutter robot in the presence of significant wheel slip.

3 CWRU Cutter Robot

In previous years, the CWRU Cutter robot implemented a monolithic LabVIEW solution using National Instruments compact Real-Time Input/Output (cRIO) product line. The LabVIEW implementation, while allowing for fast initial development, became hardware limited when attempting to perform more computation-heavy development. Real-time image processing, object-oriented programming, and floating point computations were not feasible on the single board RIO used (sbRIO-9074).

Therefore, the CWRU Cutter Hex robot required significant development to redesign the electronics and software. The new robot software is primarily based on the open source Robot Operating System (ROS), which allows for more robust, modular, and fast-prototype research. In addition, new components and sensors on the robot were revised to use a lower cost wherever possible.

3.1 New CWRU Cutter Hex Platform

The overall system diagram (Figure 3.1) shows the significant systems included in the CWRU Cutter Hex platform. CWRU Cutter Hex uses a tiered hardware/software approach to ensure robustness and reliability in all robot stages:

- Low-level electronics ("CWRU Cutter Senses," Section 3.1.1) provide basic input and output functionality, including a critical hardware-based safety system.

- Mid-level software ("CWRU Cutter Spine," Section 3.1.2) handles closed-

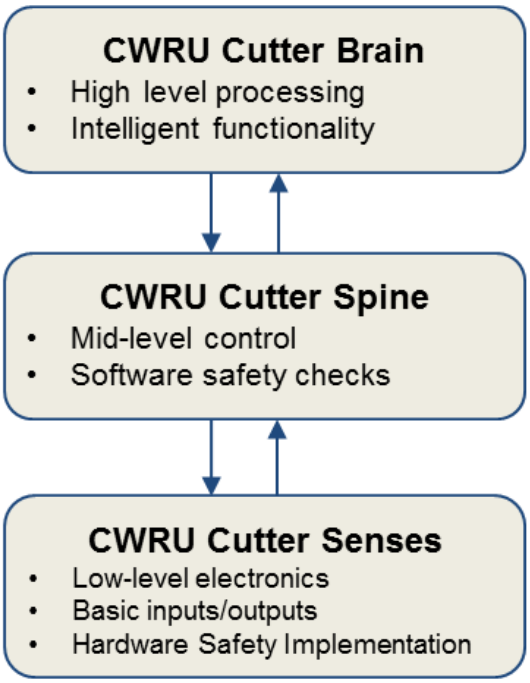

FIGURE 3.1: CWRU Cutter Hex System Overview

loop wheel control at 100 Hz and provides a software-based input to the safety system electronics.

- High-level software ("CWRU Cutter Brain," Section 3.1.3) allows for algorithm development centered around complex robot behavior (i.e., localization, path planning).

All electronics and software systems shown were redesigned during the course of this research. Other portions of the CWRU Cutter development, such as mechanical design, are similar to past CWRU Cutter robots and will not be discussed. Please refer to [5] for mechanical design and component selection if necessary.

3.1.1 Low Level Electronics: CWRU Cutter Senses

Low-level electronics are nicknamed "CWRU Cutter Senses" because of their focus on reliable input/output at the lowest level. Low-level electronics are designed to be "on" and operational at all times when the robot power is on. CWRU Cutter Senses consist of the following two critical subsystems:

1. Compact RIO Breakout Board: Includes various input and output functionality, plus safety-system hardware

2. Emergency Stop Safety Chain: Includes dedicated safety system hardware on the cRIO Breakout Board plus relays to enable/disable hazardous robot functionality.

Compact RIO Breakout Board

The cRIO Breakout printed circuit board (PCB) is a purpose-built breakout board which interfaces with the mid-level compact RIO through a 37-pin connector and the NI 9403 (cRIO bidirectional digital I/O module). The breakout board includes:

- Emergency Stop Safety Chain
- 3x Relay Drivers
- Battery Voltage and Current Monitoring
- 6x Analog Inputs
- 8x Low-current (2-5mA sink or source) Digital Inputs/Outputs
- 8x Buffered (up to 30mA source only) Digital Outputs
- PWM Inputs from Futaba RC receiver
- Dedicated Switch Inputs from front-panel switches

To summarize, the Breakout Board includes a large number of inputs and outputs, along with custom analog circuity for the Emergency Stop Safety Chain. The result is a reliable, flexible low-level system that integrates well with the higher level. The CWRU Cutter Senses Breakout Board layout is shown in Figure 3.2. The Breakout Board was designed such that it is not specific to the cRIO– that is, in the future, the mid-level cRIO may be replaced by a lower cost system which communicates with the same Breakout Board.

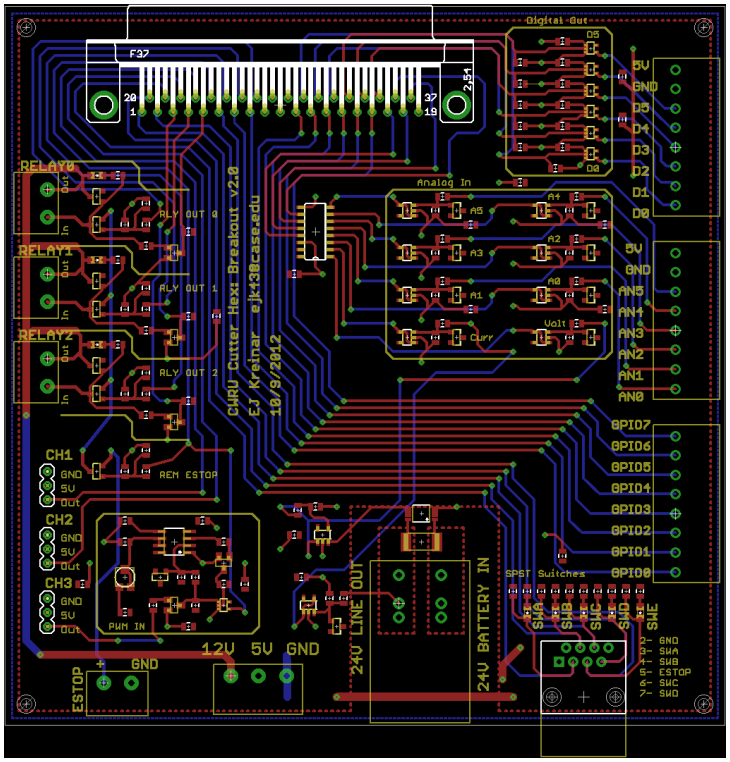

FIGURE 3.2: Compact RIO Breakout Board Layout

Emergency Stop Safety Chain

A critical component on the Breakout PCB is the analog Emergency Stop Safety Chain. The Safety Chain provides a triple-redundant safety check consisting of the following three modules:

1. Physical Emergency Stop: Big Red Button (BRB) mounted on the back of the robot (Figure 3.3)

2. Remote Emergency Stop: The remote emergency stop is built into the Futaba Remote Control (also used to manually drive the robot).

3. Software Enable: The mid-level software, CWRU Cutter Spine, outputs a software enable signal that must be true for the safety chain to be enabled. This Software Enable is the result of two situations: A watchdog "heartbeat" verifies software is functioning as expected, and mid-level software enters a "drive" state.

FIGURE 3.3: CWRU Cutter Hex Rear Panel

All three conditions (physical e-stop, remote e-stop, and software enable) must be true for the wheel-relay to turn on, allowing power to go to the wheels. This "AND" operation is performed using a hardware logic chain implemented on the Breakout PCB according to the schematic in Figure 3.4. The safety chain is implemented three times to create three separate relay drivers (allowing for additional relay outputs which depend on the same emergency stop hardware while using different software enables).

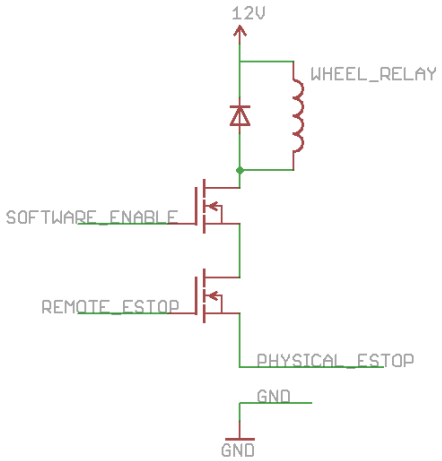

FIGURE 3.4: CWRU Cutter Hex Safety Chain

Also of note is the mixed signal PWM-to-Digital conversion for the Futaba Remote Control Emergency Stop. The Futaba RC E-Stop outputs a PWM (pulse width modulation) signal varying from approximately 10% to 20% duty cycle– 10%

indicates "E-Stop Enabled" while 20% indicates "E-Stop Disabled." Integrating the remote emergency stop into the Futaba RC allows for reliable long distance operation. The Custom Breakout board implements a rectangular pulse timer (a 555 IC with a time constant tuned to approximately 15% duty cycle) triggered at the rising edge of the input PWM. The input Futaba RC PWM is inverted and passed to a D-Flip Flop, which is triggered at the falling edge of the rectangular pulse timer. Therefore, if the PWM duty cycle is high (above 15% duty cycle), then the D-Flip Flop output is a constant active low, and vice versa. The RC E-Stop timing circuit schematic is shown in Figure 3.5.

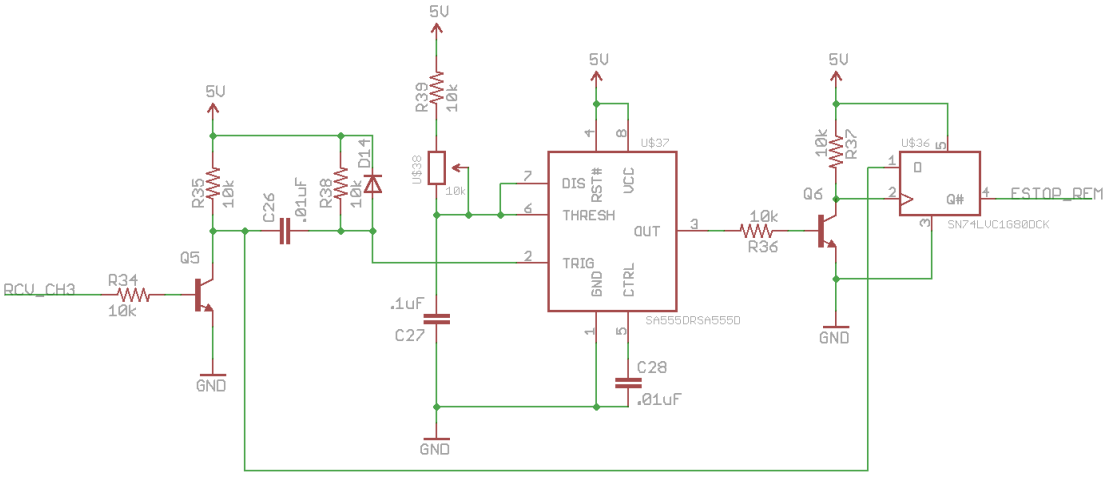

FIGURE 3.5: Futaba Remote Control E-Stop Timing Schematic

3.1.2 Compact RIO Interface: CWRU Cutter Spine

Mid-level processing, nicknamed "CWRU Cutter Spine," is designed with the intent of being an "always-on" robot system (similar to the low-level electronics). That is, whenever the robot is turned on, the Spine is active. CWRU Cutter Spine's primary responsibilities include closed-loop wheel velocity control and software safety checks. In fact, the robot can always be manually remote controlled using just the CWRU Cutter Spine and Senses, even if the high-level processing (CWRU Cutter Brain) is not operational. When the CWRU Cutter Brain is operational, the Spine acts as a simple "pass-through" mechanism whereby high level velocity and angular velocity commands are input to the Spine and executed. This

separation of responsibility allows for explicit encapsulation and modularity that was not present in past versions of CWRU Cutter.

CWRU Cutter Spine provides 100 Hz wheel control, safety state machine handling, and communication between the low-level electronics and the higher-level software. In CWRU Cutter Hex, mid-level processing is implemented using a National Instruments compact RIO-9074, which includes both a real-time processor and a Field Programmable Gate Array (FPGA). An FPGA is a programmable digital logic integrated circuit– allowing for parallel algorithm implementation in hardware instead of software. Note that a primary goal for the CWRU Cutter Hex redesign is to move away from performing significant processing tasks with the NI RIO/ Labview toolchain; however, in this case, the cRIO functions mainly for quick-turnaround FPGA development. And, because the cRIO's role is limited to a subset of pre-defined tasks, the team intends to replace the cRIO with a more cost effective alternative in the future.

Spine FPGA

CWRU Cutter utilitizes an FPGA for time-critical tasks. FPGA behavior is mostly intact from previous versions of the CWRU Cutter robot. However, critical functionality of the FPGA has been reimplemented and tested in VHDL (an industry standard Hardware Description Language used to program the digital logic inside an FPGA) so as to facilitate the transition from the cRIO to a lower cost FPGA in the future. The CWRU Cutter FPGA is directly responsible for:

1. Wheel Encoder Counting (VHDL): The FPGA counts encoder ticks from two quadrature Greyhill 256-count wheel encoders. The motors (two independently driven Invacare wheelchair motors) have a 24:1 transmission, so the full ticks-per-wheel-revolution is around 24000. With this resolution, an FPGA encoder counting implementation ensures that processing is completely parallel and that no ticks are lost at any point.

2. PWM Input (VHDL): Two PWM Futaba Remote Control commands are

input to the CWRU Cutter Spine and interpreted as forward and angular velocity commands for manual remote control. The FPGA precisely detects the period of the input PWM.

3. PWM Output (VHDL): In the case that a PWM output is necessary to drive any motors (e.g., the lawnmower blade), the FPGA PWM peripheral can output a PWM wave to the CWRU Cutter Senses cRIO Breakout Board.

4. Closed-Loop Wheel Velocity Control (LabVIEW): Two wheel motors are controlled at 100 Hz in the FPGA using a manually-tuned PID controller. The Wheel Control peripheral takes a right-wheel and left-wheel desired velocity from the cRIO processor, then uses the Wheel Encoder input as feedback for a PID controller.

5. Serial Communication Peripherals (LabVIEW): CWRU Cutter Spine sends velocity commands to the motor controller over RS-232. CWRU Cutter Spine also requires bi-directional communication with the high-level processor, CWRU Cutter Brain. Interaction with the cRIO RS-232 module is specifically handled in the LabVIEW FPGA.

Spine Real-Time Processor

The CWRU Cutter Spine real-time processor is responsible for only two tasks:

1. Velocity and Angular Velocity Command Multiplexer: The CWRU Cutter Spine processor sends a desired velocity to the FPGA for closed-loop wheel velocity control. This velocity may originate from either the manual remote control commands or from the CWRU Cutter Brain serial communication (higher level intelligence).

2. Safety State Machine: CWRU Cutter Spine combines various software status signals into a single "Software Enable" which is output to the CWRU Cutter Senses hardware safety chain.

The CWRU Cutter Spine real-time processor also parses serial communication to and from the higher level processing (velocity commands, encoder counts, and hardware status updates).

3.1.3 High Level Software: CWRU Cutter Brain

The CWRU Cutter Brain high-level processing is responsible for all intelligent robot functionality. The Brain outputs the commanded robot velocity and angular velocity at 10 Hz. Brain inputs are essentially unlimited (limited only by the number of peripherals which can attach to the high-level hardware). Robot research, whether localization, path planning, or other development, is intended to operate at the "Brain" level for its ease-of-use and isolation from physical robot behavior.

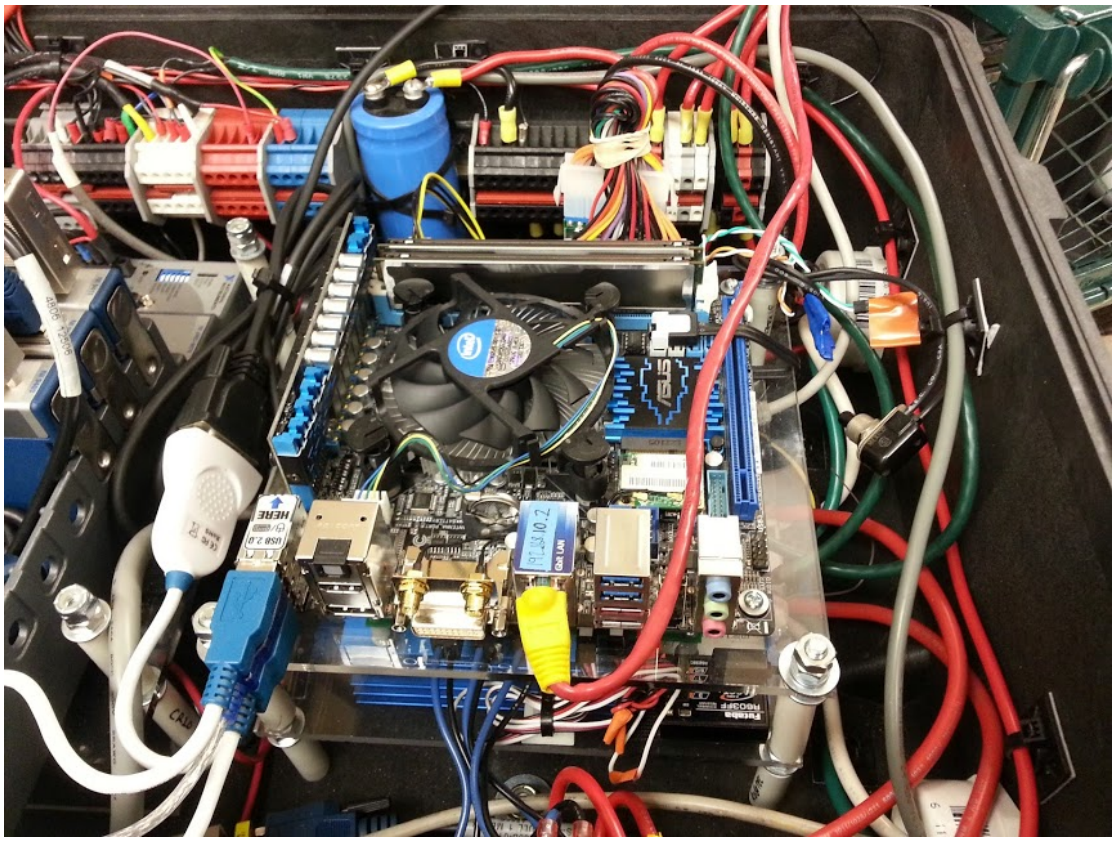

FIGURE 3.6: Onboard Processor (CWRU Cutter Brain)

Processor

CWRU Cutter Brain is implemented using an Intel i3 dual-core CPU. It uses an ASUS mini-itx motherboard with 8-gigabytes of DDR3 SDRAM and a 128-gigabyte solid-state disk. Figure 3.6 shows the Brain processor currently inside the robot.

The x86 computer running Ubuntu Linux enables the robot to use Willow Garage's Robot Operating System (ROS). ROS is an open source software framework that provides a soft real-time scheduling and message passing architecture for an unlimited number of parallel software nodes. ROS was chosen because the node-and-message-based structure enables organized and parallel development, which has been lacking in past versions of CWRU Cutter. Also, ROS community support is strong and many common robot features are included which have been historically very difficult to implement in a custom LabVIEW program (for example, logging, simulation, serial communication, and localization algorithms such as particle filtering).

High Level Sensors

High level sensors (sensors not essential to either CWRU Cutter Spine or low-level electronics) are incorporated directly into the CWRU Cutter Brain. This avoids the inconvenient situation of using the CWRU Cutter Spine as a pass-through sensor bottleneck, and instead allows the onboard PC to communicate directly with its required sensors. The high level sensors used on CWRU Cutter for the Autonomous Snowplow Competition are:

- Novatel Global Positioning System: The Novatel differential real-time-kinematic GPS provides a centimeter-level absolute position reference.
- 6-Axis Christa Inertial Measurement Unit (IMU): The Christa IMU includes a precise 3-axis gyroscope, of which the z-axis is used to detect the robot's rotational velocity.

- SICK Light Detection and Ranging (LIDAR): The SICK LIDAR provides range measurements in a 180 degree arc in front of the LIDAR unit, useful for obstacle detection or localization.

Custom serial drivers were created for the Novatel GPS, Christa IMU, and other low-cost sensors (e.g., analog input from a contact-based whisker) to interface all peripheral sensors with ROS. Other sensors may be integrated easily in the future.

Software Structure

CWRU Cutter software is organized into several systems (nodes), each of which have well-defined inputs and outputs (messages) governing their operation. Figure 3.7 illustrates the basic software structure developed and used during the 2013 Institute of Navigation's Autonomous Snowplow Competition.

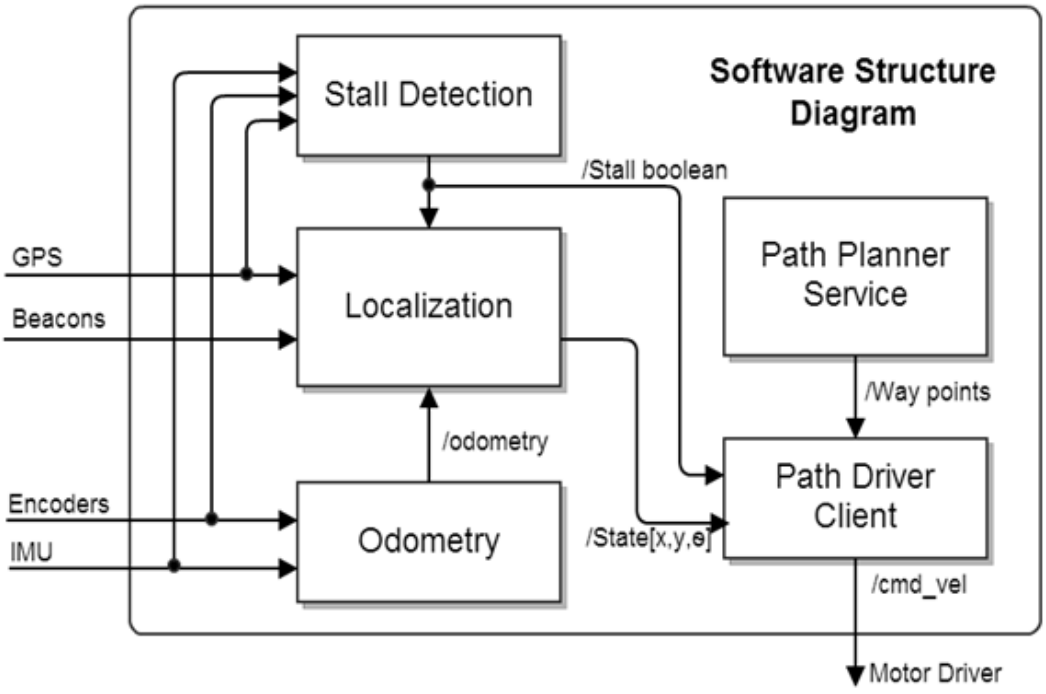

Figure 3.7: CWRU Cutter Hex Software Structure Diagram

The CWRU Cutter software systems are as follows:

- Odometry: Accepts inputs from wheel encoders (and, potentially, an IMU with a gyroscope) to generate an output "odometry" message representing the robot's dead-reckoned belief.

- Localization: Accepts inputs from odometry and other sensors (GPS and Beacons are illustrated here; other possibilities include LIDAR, IMU, etc.). The Localization system then filters sensory information and finds the robot's location in an absolute coordinate frame. See Section 4 for more detailed information on CWRU Cutter Localization.

- Path Planner: For the ASC, the path consisted of a series of static, predetermined waypoints. The Path Planner simply outputs these waypoints when requested. A more intelligent planning node may be implemented in future work, but was not necessary for the 2013 ASC or this thesis.

- Path Driver: The Path Driver inputs waypoints from the Path Planner and the robot state from Localization. The Path Driver commands robot velocity and angular velocity outputs at 10Hz to drive the robot to converge on the desired path.

- Stall Detection: The robot attempts to detect and react to stalls during the ASC using a Stall Detection system (detailed more closely in Section 5.3). This system has been replaced after the ASC with the introduction of localization from this thesis.

Because the CWRU Cutter software architecture is now defined in terms of node inputs and outputs, any system may be individually replaced. In this thesis, the Localization system is investigated and improved to handle wheel slip errors using an augmented Extended Kalman Filter; To run the robot with the improved localization, all other systems (path planning, path driving, odometry, etc.) may remain the same while Localization is substituted for the improved version.

Offline Processing and Simulation

The Robot Operating System software includes utilities for simulation, data logging, and playback.

The ROS simulation evironment, Stage, allows for testing and development of robot algorithms. When a node outputs a desired velocity over the "cmd_vel" topic, the Stage simulator updates a simulated robot's state. A robot visualization indicates the position change. On CWRU Cutter Hex, Path Planning and Path Driving systems were developed completely in the Stage simulator before being tested on the physical robot. In addition, localization algorithms can use the Stage simulator as a ground truth comparison to verify the localization system works correctly before testing on the robot.

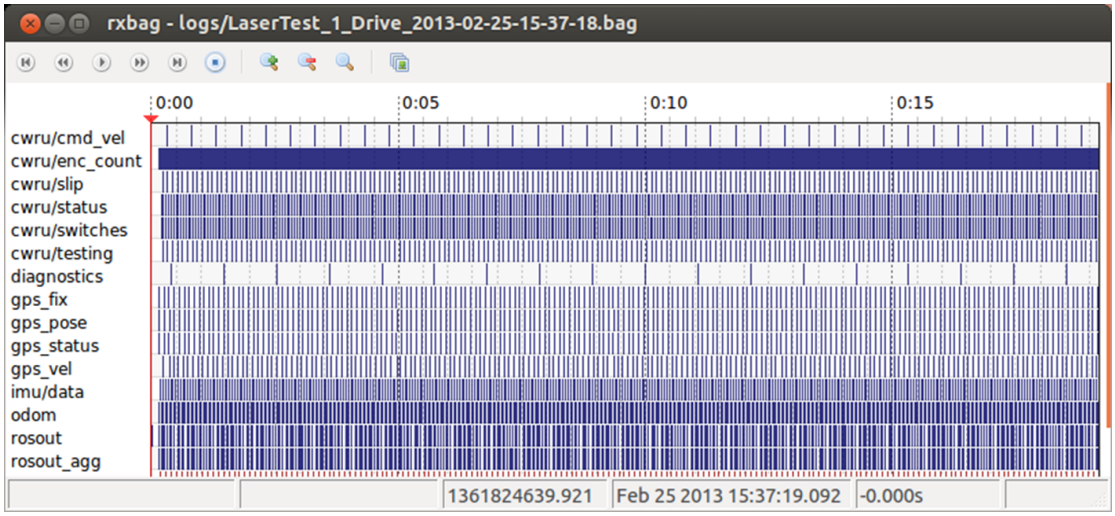

FIGURE 3.8: Data recording visualization

Data logging and playback is accomplished through the use of ROS "bags." A rosbag records all messages over a given time period. Then, when desired, the rosbag plays back all messages in real-time. In this way, the rosbag playback faithfully replicates both the data content and the time of arrival for each message. Figure 3.8 illustrates recorded data over a 20-second time period; each line represents an individual message. Because ROS uses a message-based structure, a localization node remains literally unchanged whether input messages originate from sensors or from a rosbag playback; this allows for sensor-realistic tuning of localization algorithms. Logging/playback was used extensively for development and testing throughout this thesis.

4 Localization

Multiple localization filtering systems were developed and implemented on CWRU Cutter Hex. This chapter, centered around localization, describes the baseline mobile robot localization techniques implemented, including both parametric filters and non-parametric filters. The following filters are covered:

1. Extended Kalman Filter: Various measurements, system equations, and filter architectures

2. Monte Carlo Localization: Measurements, benefits, and implementations are discussed

The filters were implemented in simulation and onboard the CWRU Cutter robot. Filter theory, motivation, and practical considerations are investigated for each filter, with a particular emphasis on the CWRU Cutter specific integration. Chapter 5, Filter-Based Slip Detection and Rejection, builds on the concepts discussed here in order to localize while experiencing severe wheel slip which would otherwise cause localization to diverge.

4.1 Extended Kalman Filter

The Kalman Filter, mentioned in the Background Section 2.2, is a recursive filter which generates an "optimal" estimate given a set of measurements. The optimal estimate finds the maximum likelihood probabilty given both measurements and known measurement noise. The Extended Kalman Filter (EKF) is the well-used,

non-linear extension of the Kalman Filter. The CWRU Cutter robot, and all other non-holonomic robots, follow a non-linear set of motion equations due to the system's dependence on the robot heading, θ. Therefore, the Extended Kalman Filter and its variants are used for parametric filtering.

The discrete Extended Kalman Filter aims to predict the filter state $\boldsymbol{x}_k$ at time k which is subjected to the following nonlinear stochastic differential equations [12]:

$$\boldsymbol{x}_k = \mathbf{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{w}_{k-1} \tag{4.1}$$

$$\mathbf{z}_k = \mathbf{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{4.2}$$

Where $\mathbf{f}$ and $\mathbf{h}$ represent the nonlinear system and measurement models, respectively. The additive system noise, $\boldsymbol{w}_{k-1}$, and measurement noise, $\boldsymbol{v}_k$, are both zero mean Gaussian noise with associated covariance matrices:

$$\boldsymbol{w}_k \sim N(0, \boldsymbol{Q}_k) \tag{4.3}$$

$$\boldsymbol{v}_k \sim N(0, \boldsymbol{R}_k) \tag{4.4}$$

The Extended Kalman Filter keeps track of a state estimate vector, $\hat{\boldsymbol{x}}_k$, and a covariance matrix, $\boldsymbol{P}_k$, at each iteration. The EKF operates using two steps:

1. System Update: The known system dynamics $\mathbf{f}(\boldsymbol{x}, \boldsymbol{u})$ propagate the state estimate from $\hat{\boldsymbol{x}}_{k-1}$ to $\hat{\boldsymbol{x}}_k^-$. The system update increases the overall uncertainty, expressed as $\mathbf{P}_k^-$, where $\mathbf{P}_k^- > \mathbf{P}_{k-1}$.

2. Measurement Update: A measurement is applied according to the measurement model $\mathbf{h}(\boldsymbol{x})$. The measurement update adjusts $\hat{\boldsymbol{x}}_k^-$ to $\hat{\boldsymbol{x}}_k$. Because the measurement update is based on sensor data, the overall uncertainty decreases; the covariance propagates to $\mathbf{P}_k$, where $\mathbf{P}_k < \mathbf{P}_k^-$.

Equations 4.1 and 4.2 govern the state estimate and measurement estimate of the Extended Kalman Filter. In order to propagate the covariance correctly through-

out the EKF, the nonlinear functions $\mathbf{f}$ and $\mathbf{h}$ must be linearized around the operating point, $\boldsymbol{x}_k$, at each time step. The Jacobians of $\mathbf{f}$ and $\mathbf{h}$ create the necessary matrices for the EKF update:

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}}\right|_{x_k = \hat{x}_k^-} \tag{4.5}$$

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{x_k = \hat{x}_k^-} \tag{4.6}$$

The typical System Update (Step 1) is performed according to Equations 4.7 and 4.8. The System Update follows the equations for state and covariance propagation through a known system.

$$\hat{\boldsymbol{x}}_k^- = \mathbf{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_k) \tag{4.7}$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{4.8}$$

Finally, the Measurement Update (Step 2) is applied to the EKF:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{4.9}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}\left(\hat{\boldsymbol{x}}_k^-\right)\right) \tag{4.10}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^- \tag{4.11}$$

Additional detail on the mathematical derivation of the Kalman gain and measurement function can be found in [30] or [12].

The Kalman gain is easily conceptualized using a one-dimensional case where $\mathbf{H} = 1$:

$$K_k = \frac{P_k^-}{P_k^- + R_k} \tag{4.12}$$

Clearly the Kalman gain weights the measurement innovation based on the KF covariance P_k^- and the measurement noise R_k. If the filter covariance is very large compared to the measurement noise, then $K_k \approx 1$ and the updated state will reflect the sensor measurement. On the other hand, if the covariance is small

while the measurement noise is large, then $K_k \approx 0$ and the measurement is not used.

If no measurement update is possible, then the update may be skipped by replacing Equations 4.10 and 4.11:

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- \tag{4.13}$$

$$\mathbf{P}_k = \mathbf{P}_k^- \tag{4.14}$$

Filter tuning is accomplished by varying the noise values $\mathbf{Q}_k$ and $\mathbf{R}_k$ from Equations 4.3 and 4.4. A typical KF use-case involves quantifying measurement noise for each sensor and iteratively modifying the noise values to achieve desired filter performance.

4.1.1 CWRU Cutter System Model

The CWRU Cutter system is a non-holonomic differential drive mobile robot. The robot origin is centered between the two differentially driven wheels. The state of interest for the Extended Kalman Filter used on CWRU Cutter Hex is:

$$\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v & \omega \end{bmatrix}^T \tag{4.15}$$

Where x, y refers to the origin coordinates, θ refers to the robot heading, and v, ω refers to the velocity and angular velocity of the origin, respectively. Many common approaches for mobile robot localization do not include v, ω states [30][31]. Including v, ω here is a conscious decision that eventually allows for a convienient and intuitive modification for augmented wheel velocity error states. Other incentives for including v, ω are discussed in Section 4.1.3.

The Extended Kalman Filter using the state vector in Equation 4.15 is hereafter referred to as the "5-State" EKF. For the 5-State EKF, the following system model

describes the transition for each individual state:

$$\mathbf{f}(\boldsymbol{x}) = \begin{bmatrix} f_x & f_y & f_\theta & f_v & f_\omega \end{bmatrix}^T \tag{4.16}$$

The first three equations of motion for a differential drive mobile robot can be approximated [32][33] as:

$$f_x = x_k = x_{k-1} + v_{k-1}\,\mathrm{d}t\cos(\theta_{k-1} + \frac{\omega_{k-1}\mathrm{d}t}{2}) \tag{4.17}$$

$$f_y = y_k = y_{k-1} + v_{k-1}\,\mathrm{d}t\sin(\theta_{k-1} + \frac{\omega_{k-1}\mathrm{d}t}{2}) \tag{4.18}$$

$$f_\theta = \theta_k = \theta_{k-1} + \omega\mathrm{d}t \tag{4.19}$$

These equations represent a first order velocity-model approximation which is improved by using the heading midpoint in equation 4.17 and 4.18: $\theta_{k-1} + \frac{\omega_{k-1}\,\mathrm{d}t}{2}$. This approximation produces a more accurate result than simply using the old heading, θ_{k-1}, and the complexity is less than using the full non-linear velocity-based motion equations. This model is sufficient for CWRU Cutter needs when running at a filter update rate of 10 Hz.

The velocity states, v, ω are modeled as a random walk (Equations 4.20 and 4.21). Because the CWRU Cutter state vector does not include acceleration states, velocity and angular velocity remain constant during the system update.

$$f_v = v_k = v_{k-1} \tag{4.20}$$

$$f_\omega = \omega_k = \omega_{k-1} \tag{4.21}$$

Now that the CWRU Cutter system model $\mathbf{f}(\boldsymbol{x})$ is known, the system Jacobian $\mathbf{F}_k$ is calculated with respect to the state vector $\boldsymbol{x}$:

$$\theta_{mid} = \theta_k + \frac{\omega_k\,\mathrm{d}t}{2} \tag{4.22}$$

$$\mathbf{F}_k(\boldsymbol{x}) = \begin{bmatrix} 1 & 0 & -v_k\,\mathrm{d}t\sin(\theta_{mid}) & \mathrm{d}t\cos(\theta_{mid}) & -\frac{v_k\,\mathrm{d}t^2}{2}\sin(\theta_{mid}) \\ 0 & 1 & v_k\,\mathrm{d}t\cos(\theta_{mid}) & \mathrm{d}t\sin(\theta_{mid}) & \frac{v_k\,\mathrm{d}t^2}{2}\cos(\theta_{mid}) \\ 0 & 0 & 1 & 0 & \mathrm{d}t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.23}$$

The 5-State system model $\mathbf{f}(\boldsymbol{x})$ and the system Jacobian matrix $\mathbf{F}_k$ are implemented in the CWRU Cutter Extended Kalman Filter. The discrete system noise matrix, $\mathbf{Q}_k$, is chosen experimentally such that the Extended Kalman Filter performs well. The selected $\mathbf{Q}_k$ values during robot implementation are discussed in Section 6.2, Implementation Results.

4.1.2 CWRU Cutter Measurements

Each sensor used on CWRU Cutter has a corresponding measurement model $\mathbf{h}(\boldsymbol{x})$ which describes the measurement's sensitivity to the robot state. Measurements may be applied in "batch" mode, where all measurements are combined into a single EKF measurement update during each timestep (Equations 4.9 through 4.11). Or, alternatively, measurement updates may be applied sequentially. Because CWRU Cutter uses multiple asynchronous sensors, option 2 (successive measurement updates) is implemented so that the EKF may be updated with new measurement information as soon as new data arrive.

GPS Measurement

The CWRU Cutter GPS is a differential Real-Time-Kinematic (RTK) Novatel GPS system. This GPS is used for precise localization, but its integration into the CWRU Cutter filtering algorithm has been somewhat "ad-hoc" in the past. While a long term goal of the CWRU Cutter project is to lower the overall robot cost through custom sensor development and lower-precision/lower-cost sensors, the RTK-GPS is used here to provide a baseline x, y measurement in order to verify the EKF theory and implementation. In the future, the same measurement

update theory discussed here may be applied to any arbitrary x, y measurement source.

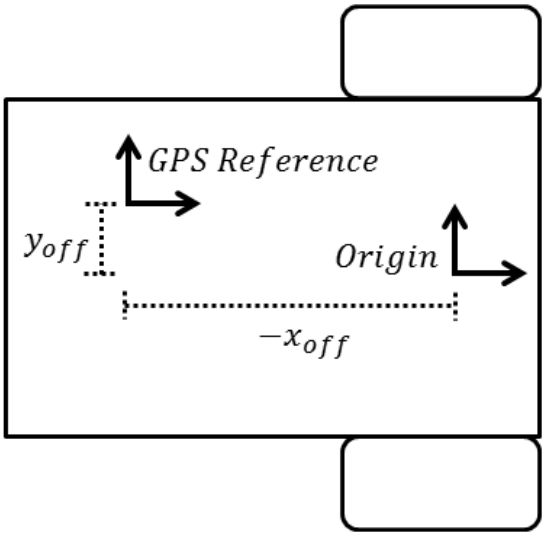

FIGURE 4.1: GPS Lever Arm Offset Schematic Diagram

The GPS measurement is defined here as any measurement which provides an x, y reading offset at some fixed coordinates from the robot origin $(x_{\text{off}}, y_{\text{off}})$, termed the "lever-arm offset." An arbitrary lever arm offset is shown in Figure 4.1. For CWRU Cutter's GPS (shown in Figure 4.2), the lever arm offset, measured in meters, is typically:

$$\begin{bmatrix} x_{\text{off}} \\ y_{\text{off}} \end{bmatrix} = \begin{bmatrix} -0.45 \\ 0.0 \end{bmatrix} \tag{4.24}$$

However, for this analysis, the actual lever arm offset parameters are not important. It should be noted that while the Novatel GPS system also outputs messages such as GPS velocity and heading, these messages are ignored in the CWRU Cutter Hex EKF localization (it is assumed that the GPS is solely an x, y measurement).

The measurement function $\mathbf{h}_{gps}(\boldsymbol{x})$ is defined according to Equation 4.25. Keep in mind that the measurement function $\mathbf{h}_{gps}(\boldsymbol{x})$ expresses the measurement x_{gps}, y_{gps} in terms of the robot state, $\boldsymbol{x}_k$ (i.e., $\mathbf{h}_{gps}(\boldsymbol{x})$ represents the measurement "forward

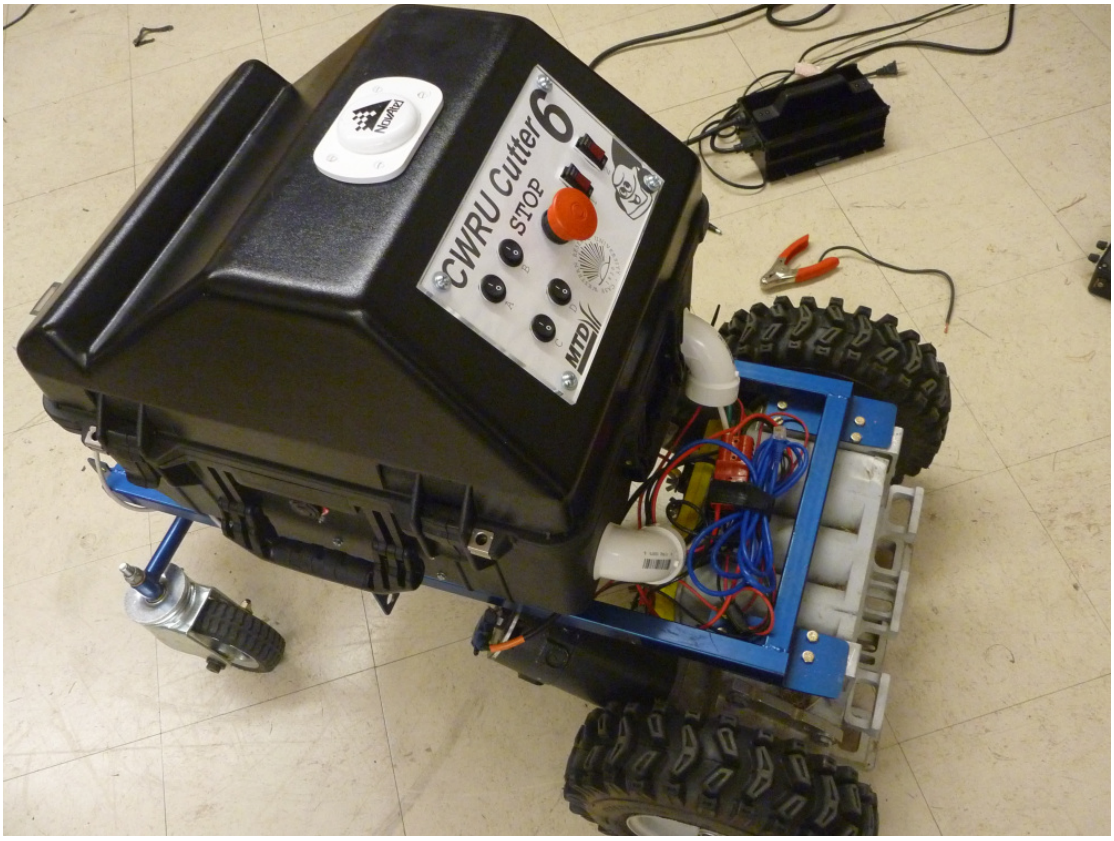

FIGURE 4.2: CWRU Cutter Robot with offset GPS antenna

kinematics").

$$\mathbf{h}_{gps}(\boldsymbol{x}_k) = \begin{bmatrix} x_{gps} \\ y_{gps} \end{bmatrix} = \begin{bmatrix} x_k + x_{\text{off}} \cos\theta_k - y_{\text{off}} \sin\theta_k \\ y_k + x_{\text{off}} \sin\theta_k + y_{\text{off}} \cos\theta_k \end{bmatrix} \tag{4.25}$$

The measurement function Jacobian creates the corresponding matrix $\mathbf{H}_{gps}$ (Equation 4.26) used for the EKF measurement update at each time step:

$$\mathbf{H}_{gps}(\boldsymbol{x}_k) = \begin{bmatrix} 1 & 0 & -x_{\text{off}} \sin\theta_k - y_{\text{off}} \cos\theta_k & 0 & 0 \\ 0 & 1 & x_{\text{off}} \cos\theta_k - y_{\text{off}} \sin\theta_k & 0 & 0 \end{bmatrix} \tag{4.26}$$

It is important to realize that at this point, the entire state vector $\boldsymbol{x}$ can be estimated using only the system model $\mathbf{f}(\boldsymbol{x})$ and the measurement model $\mathbf{h}_{gps}(\boldsymbol{x})$. This assertion arrives from the non-linear observability test for an Extended Kalman Filter [34], which evaluates the rank of the Observability matrix using the lin-

earized Jacobian matrices:

$$\text{rank}(\boldsymbol{O}) = \text{rank} \begin{bmatrix} \mathbf{H} \\ \mathbf{HF} \\ \mathbf{HF}^2 \\ \vdots \\ \mathbf{HF}^n \end{bmatrix} = n \tag{4.27}$$

Typically, a linear system is considered observable when the Observability matrix $\boldsymbol{O}$ is full-rank. Though it will not be derived here, $\boldsymbol{O}$ in Eq. 4.27 is full-rank when using the linearized Jacobians for the robot system, $\mathbf{F}$, and GPS measurement, $\mathbf{H}_{gps}$ (Equations 4.23 and 4.26). In this case, because the CWRU Cutter system is non-linear as opposed to linear, a full-rank observability matrix indicates that the Extended Kalman Filter converges on the true state as long as the estimate is close to the true value. This serves as a verification test to make sure that the derived system and measurement equations will operate as expected.

Gyroscope Measurement

The gyroscope measurement originates from a Christa 6-degree-of-freedom Inertial Measurement Unit (IMU) onboard the CWRU Cutter robot. This IMU provides a precise measurement of the robot angular rate which may include a constant bias term β_{imu}. This constant bias can be handled in two ways:

1. Ignore the bias by artificially inflating the IMU measurement noise, $\mathbf{R}_k$
2. Estimate the bias by including an augmented IMU bias state in the EKF

For the purposes of this implementation, CWRU Cutter includes an augmented IMU bias state in the Extended Kalman Filter (option 2).

The measurement function $\mathbf{h}_{imu}(\boldsymbol{x})$ is a linear function of the robot state (Eq 4.28). Including the augmented IMU bias state, the estimated robot state vector

becomes: $\left[\hat{\boldsymbol{x}} \;\middle|\; \hat{\beta}_{imu}\right]$.

$$\mathbf{h}_{imu}(\boldsymbol{x}_k) = \left[\omega_{imu}\right] = \left[\omega_k + \beta_{imu}\right] \tag{4.28}$$

And the measurement Jacobian is simply:

$$\mathbf{H}_{imu}(\boldsymbol{x}_k) = \left[\begin{array}{ccccc|c} 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] \tag{4.29}$$

From this point forward, keep in mind that an additional state, β_{imu}, is present whenever an IMU is used, but the baseline EKF will still be referred to as a "5-State" EKF for consistency.

Encoder Measurement

CWRU Cutter includes a quadrature wheel encoder on each motor. Quadrature encoders essentially return an integer number of "ticks" indicating how far the encoder has turned during each timestep. Using a $\frac{\text{ticks}}{\text{meter}}$ (tpm) conversion, the velocities of each wheel can be approximated over a timestep dt:

$$v_R^{enc} = (\text{ticks}_R)/\text{tpm} * \text{d}t \tag{4.30}$$

$$v_L^{enc} = (\text{ticks}_L)/\text{tpm} * \text{d}t \tag{4.31}$$

Furthermore, the velocity of the robot origin v, w can be expressed in terms of the velocities of the left and right wheel (where the left and right wheel are separated by the track width, b).

$$v = \frac{v_R^{enc} + v_L^{enc}}{2} \tag{4.32}$$

$$\omega = \frac{v_R^{enc} - v_L^{enc}}{b} \tag{4.33}$$

Equations 4.32 and 4.33 are standard kinematic equations for a differential drive mobile robot. Solving for v_R^{enc}, v_L^{enc} leads to:

$$v_R^{enc} = v + \frac{b}{2}\omega \tag{4.34}$$

$$v_L^{enc} = v - \frac{b}{2}\omega \tag{4.35}$$

To incorporate the encoder measurement into the CWRU Cutter Extended Kalman Filter, the measurement function $\mathbf{h}_{enc}(\boldsymbol{x})$ must express the measurement $\begin{bmatrix} v_R^{enc} & v_L^{enc} \end{bmatrix}^T$ in terms of the state $\boldsymbol{x}$. The EKF measurement function $\mathbf{h}_{enc}(\boldsymbol{x})$ is based on Equations 4.34 and 4.35:

$$\mathbf{h}_{enc}(\boldsymbol{x}) = \begin{bmatrix} v_R^{enc} \\ v_L^{enc} \end{bmatrix} = \begin{bmatrix} v_k + \frac{b}{2}\omega_k \\ v_k - \frac{b}{2}\omega_k \end{bmatrix} \tag{4.36}$$

Finally, the Jacobian matrix for the encoder measurement is:

$$\mathbf{H}_{enc}(\boldsymbol{x}) = \begin{bmatrix} 0 & 0 & 0 & 1 & \frac{b}{2} \\ 0 & 0 & 0 & 1 & -\frac{b}{2} \end{bmatrix} \tag{4.37}$$

Notice that in the case of a linear measurement (i.e., encoders, IMU), the Extended Kalman Filter measurement update simplifies to the linear Kalman Filter measurement equation: $\mathbf{z} = \mathbf{H}\boldsymbol{x}$. But, since we are already dealing with an Extended Kalman Filter, it is easy enough to think in terms of the EKF measurement update from Equation 4.2.

4.1.3 Other Parametric Filters

Additional parametric filters have been developed and tested for the CWRU Cutter robot. These filters will be discussed at a high level with a particular focus on their implementation specifically for the CWRU Cutter robot.

3-State Extended Kalman Filter

A common alternative to the 5-State mobile robot system model is a 3-State EKF model which filters only $\boldsymbol{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$. The 3-State EKF uses the forward and angular velocities as control inputs $\boldsymbol{u} = \begin{bmatrix} v & \omega \end{bmatrix}^T$. The control inputs are typically determined at each time step using wheel encoder outputs or other odometric sensors. While the 3-State system model is commonly used in research and implementation for differential drive mobile robots, the 5-State EKF was chosen for CWRU Cutter over the alternative 3-State EKF for two primary reasons:

1. Measurements are able to improve the estimated v, ω beyond what is typically used as control inputs. For example, gyroscope measurements, wheel encoder measurements, and subsequent filtered x, y measurements are merged inside the EKF to improve the velocity estimate.

2. The 5-State EKF reduces computational complexity calculating the system update. While this may seem counterintuitive, moving v, ω from the input vector $\boldsymbol{u}$ to the state vector $\boldsymbol{x}$ enables the system update (Equations 4.7 and 4.8) to function without considering any control inputs.

The reader is referred to [30] and [31] for futher information on the 3-State mobile robot Extended Kalman Filter if desired.

Iterated Extended Kalman Filter

The Iterated Extended Kalman Filter attempts to provide a more accurate EKF in the presence of a non-linear measurement update. Conceptually, the typical EKF measurement update involves linearizing a non-linear measurement function around an operating point to create the matrix $\mathbf{H}_k\big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}^-}$. If the operating point $\hat{\boldsymbol{x}}$ is close to the true state $\boldsymbol{x}$, then the measurement Jacobian $\mathbf{H}_k$ is more accurate. The Iterated EKF repeats the non-linear measurement update, "re-linearizing" around the better estimate of the operating point, in order to generate a better

estimate of $\hat{\boldsymbol{x}}^+$. A technical discussion including IEKF theory can be found in [12].

For CWRU Cutter, the Iterated Extended Kalman filter is applied specifically to the non-linear GPS measurement with a lever arm offset. In most cases, the Iterated Extended Kalman Filter does not significantly improve behavior (see the Results Section 6.1). However, simulation does show that the Iterated Extended Kalman Filter performs more iterations, theoretically improving the filter estimate, during times when the GPS measurement is particularly nonlinear (during turns, etc).

Unscented Kalman Filter

The Unscented Kalman Filter uses a completely different approach than the Extended Kalman Filter. While the EKF approximates non-linear noise as a Gaussian by analytically calculating Jacobians of the system model and measurement models, the Unscented Kalman Filter sends so-called "sigma-points" (samples scattered throughout the n-dimensional state-space) through the non-linear system and measurement models directly. After passing sigma points through the system and measurement models, the resulting distribution is used to model the covariance $\mathbf{P}$.

While an analytical Jacobian is not used within the Unscented Kalman Filter, the UKF is unfortunately sensitive to a small number of parameters which must be experimentally tuned. For thorough details on the Unscented Kalman Filter, the reader is referred to [14].

In simulation, the Unscented Kalman Filter demonstrated improved performance in some areas but degraded performance in other areas (see the Results Section 6.1). Based on CWRU Cutter testing with the Unscented Kalman Filter, the UKF is a plausible filter option– but the EKF detailed here performs well and there is not sufficient data to support using the UKF in lieu of the Extended Kalman Filter.

4.2 Monte Carlo Localization

Monte Carlo Localization (MCL) is also implemented on the CWRU Cutter robot. Adaptive MCL (AMCL), which also incorporates elements of KLD sampling and mixture MCL, is used on CWRU Cutter for the primary three use-cases:

1. GPS with a Lever-arm Offset: AMCL offers a straightforward way to deal with the non-linear GPS measurement model.

2. Landmark-Based LIDAR Localization: The onboard LIDAR can be used to localize using landmarks with pre-defined locations.

AMCL is particularly suited for these problems because Monte Carlo Localization, as discussed in the Background section, is a type of non-parametric Filter, meaning that the non-linear system and measurement models of the CWRU Cutter robot are actually quite easy to implement because no Jacobians are necessary. AMCL also tracks multiple hypotheses, allowing for straightforward localization using map-based (or landmark-based) LIDAR measurements.

AMCL was in fact used for localization during the 2013 ION Autonomous Snowplow Competition using GPS measurements. AMCL will be discussed in a high level, conceptual sense, with a specific focus on the CWRU Cutter implementation. For additional detail on MCL theory, see [30][16][18].

AMCL represents the filtered state and covariance using "particles" (discrete estimates of the state vector $\begin{bmatrix} x & y & \theta \end{bmatrix}^T$), rather than the parametric values $\hat{\boldsymbol{x}}$ and $\boldsymbol{P}$ as in the EKF. Particle motion is sampled from the odometry inputs $\boldsymbol{u} = \begin{bmatrix} v & \omega \end{bmatrix}^T$. Particles are represented here as a set of state estimates $\hat{\boldsymbol{x}}_k^n$ where $n = 1, 2, \ldots, n$. The AMCL algorithm involves three distinct steps:

1. System Update: The known system dynamics $\mathbf{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ propagate each particle (state estimate) from $\hat{\boldsymbol{x}}_{k-1}^n$ to $\hat{\boldsymbol{x}}_k^n$. In addition, random noise is stochastically added to the inputs $\boldsymbol{u}_k$ when calculating the updated state estimates $\hat{\boldsymbol{x}}_k^n$; this random sampling represents the "system noise" and increases the overall distribution of the particles.

2. Measurement Update: The AMCL measurement update is responsible for "weighting" each particle based on its likelihood of agreeing with each sensor measurement. That is, particle weight is assigned as the probability that each particle is true based on an external measurement reading using a given measurement model: $\mathrm{P}(\mathbf{h}(\hat{\boldsymbol{x}}_k^n)\big|\mathbf{z}_k)$.

3. Resampling: Finally, particles are resampled using the normalized weights from Step 2 as the probability of resampling each particle. Resampling decreases the overall particle distribution (effectively lowering the covariance).

The result of AMCL resampling is that lower weighted particles (i.e., those particles that do not agree with the measurement readings), will "die off" over time. Particles which agree with the robot measurements will survive. Some considerations of the AMCL algorithm include:

- AMCL can diverge due to an unfortunate sequence of random sampling. This problem is usually avoided by using a large number of particles

- Extremely accurate sensors can in fact cause the filter to diverge because most particles will be weighted very low, causing an unrealistic resampling step in which too many incorrect particles survive. Very accurate sensors may require inflating the expected noise values.

AMCL: GPS Localization

Using AMCL to filter GPS measurements seems like overkill; however, before the Extended Kalman Filter GPS measurement was derived for this research in Section 4.1.2, the "GPS with a lever arm offset" problem plagued the CWRU Cutter team for multiple years. AMCL effectively handles a GPS with a lever arm offset, but the localization result is not nearly as accurate as the Extended Kalman Filter.

The system update is performed according to the typical AMCL system update for an odometry-based motion-model [30].

The GPS x, y measurement is applied to each particle using the measurement model $\mathbf{h}_{gps}(\hat{\boldsymbol{x}}_k^n)$ from Equation 4.25. This equation specifies the expected measurement values $\begin{bmatrix} x_{gps} & y_{gps} \end{bmatrix}^T$ for each particle, $\hat{\boldsymbol{x}}_k^n$, where $n = 0, 1, 2, \ldots, n$. AMCL weights each particle according to the GPS measurement probability, and resamples. Resulting position error for EKF and AMCL localization using GPS only is shown in Figure 4.3. Both filters use an inflated GPS measurement noise:

$$\begin{bmatrix} \sigma^2_{GPS_x} & 0 \\ 0 & \sigma^2_{GPS_y} \end{bmatrix} = \begin{bmatrix} 0.1^2 & 0 \\ 0 & 0.1^2 \end{bmatrix}$$

. As will be discussed further in Chapter 6, the noise level is inflated so as to simulate a less-precise sensor.

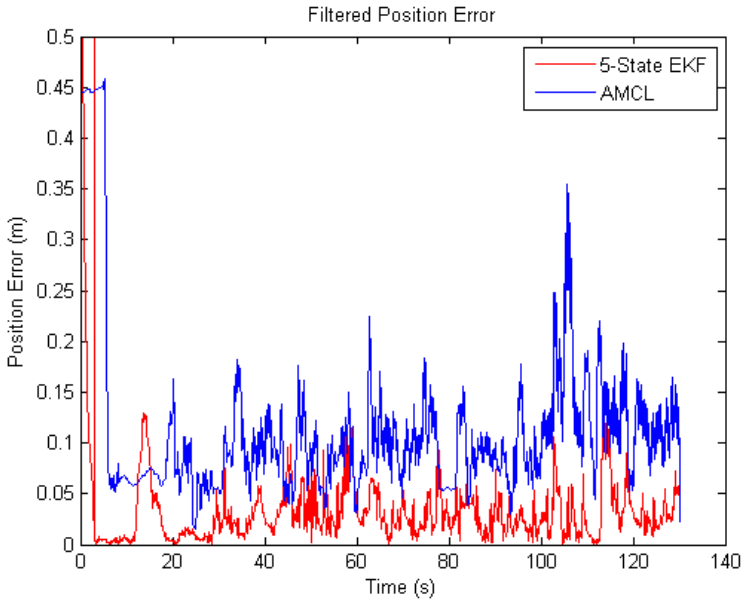

FIGURE 4.3: AMCL Localization vs EKF Localization Position Error

Both AMCL and the EKF converge. In this case, the mean AMCL error is 0.1m while the EKF error is only 0.03m. The AMCL error is, notably, approximately three times as large as the EKF error using the inflated noise values (Figure 4.3). This is expected behavior. Different noise parameters scale results similarly; if the noise level is modified to a smaller, more realistic representation of the differential GPS (i.e., $0.01m$ standard deviation instead of $0.1m$), then both filters

perform correspondingly better. Unfortunately, using a standard deviation of $0.01m$ occasionally causes AMCL to diverge because the sensor is too precise for the sampling-based approach without special handling. Nevertheless, even though the EKF performs better when filtering GPS measurements, the lever-arm-offset GPS measurement was correctly implemented first using AMCL, and AMCL GPS filtering ran successfully on CWRU Cutter during the 2013 ION Autonomous Snowplow Competition.

AMCL: Landmark Based Localization

The Institute of Navigation's Autonomous Snowplow Competition is located in an "urban-canyon" which makes GPS signals extremely unreliable. Fortunately, the CWRU Cutter team was able to successfully use GPS during the 2013 ASC, but GPS should not be used to compete in future ASC competitions due to persisting acquisition issues associated with the competition environment.

Instead, the ASC allows landmarks to be placed around the competition field. An AMCL Landmark-Based localization system with pre-defined landmark locations has been tested on CWRU Cutter Hex. AMCL localization offers several benefits over an analytical, parametric solution in this situation:

- No feature extraction is necessary from the lidar scans
- Robot localization is possible with a small number of landmarks visible (1-2 landmarks allows for localization, while the robot can track position for a short time with 0 landmarks visible)
- Extra "noise" such as people, obstacles, and even falling snow are ignored

For testing, four cones were arranged in a rectangle (Figure 4.4). Localization was performed in post-processing, using a Likelihood-Field LIDAR measurement model [30] where the four cone locations are surveyed and known with complete certainty. Figure 4.5 shows the localization position error.

FIGURE 4.4: Landmark Localization

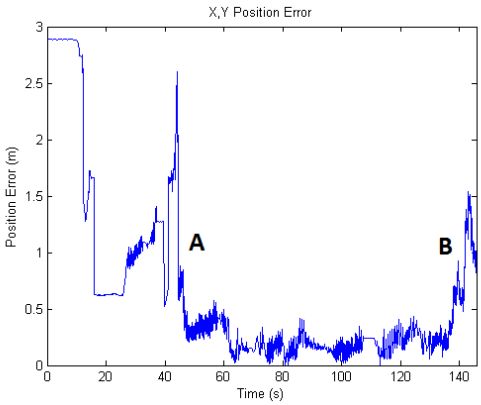

FIGURE 4.5: Landmark Localization Error

Once the AMCL localization is correctly initialized (Point A), then the position error remains around approximately 0.25 meters until Point B. At Point B, the robot increases speed to around 1m/s (unrealistic for a robot snowplow) and moves out of view of landmarks; the solution diverges. Note that in this test, there were only 0-2 landmarks visible at any time.

So, during a use-case demonstration of AMCL localization, the CWRU Cutter robot can in fact localize using a sparse set of landmarks with pre-defined locations. This procedure holds potential for future CWRU Cutter entries into the Autonomous Snowplow Competition. Future work may require more than 4 landmarks in order to localize within the $0.1m$ necessary for a complete-coverage robot snowplow.

5 Filter-Based Slip Detection and Rejection

The CWRU Cutter localization techniques introduced in Chapter 4 represent standard localization algorithms for differential drive mobile robots. These filters perform well overall. However, neither filter successfully handles wheel-slip faults. This chapter addresses wheel-slip by introducing an Extended Kalman Filter with an augmented state vector and measurement matrices for non-systematic odometry errors.

Section 5.1 first discusses and demonstrates the problem of unmodeled wheel slip. Section 5.2 describes the augmented Extended Kalman Filter designed to estimate wheel-slip errors for a differential drive mobile robot. Finally, Section 5.3 outlines plausible alternate approaches to wheel-slip detection/rejection which were not found to perform as well as the augmented Extended Kalman Filter. The following chapter, "Wheel-Slip Results," demonstrates simulation and implementation results using the theory outlined in this chapter and in Chapter 4.

5.1 The Wheel-Slip Problem

A wheel-slip is defined here as any encoder measurement where the measured wheel velocity $v_{R/L}^{enc}$ does not equal the true wheel velocity $v_{R/L}^{true}$. This definition includes, but is not limited to, non-systematic odometry errors as discussed in the background Section 2.3. The encoder measurement is now expressed as the sum

of the true velocity v^{true} and some unknown error velocity v^{err}:

$$v_R^{enc} = v_R^{true} + v_R^{err} \tag{5.1}$$

$$v_L^{enc} = v_L^{true} + v_L^{err} \tag{5.2}$$

Unfortunately, because the typical encoder measurement model (Equation 4.36) assumes the encoders are a true measure of the robot velocity (i.e., $v_{R/L}^{err}$ is always equal to 0), any wheel-slip fault causes localization to falsely believe an incorrect measurement. If the velocity error due to wheel slip is significant, localization may diverge. Minor wheel slip or wheel slip fused with other sensors such as an IMU may also cause an increased position error even though the robot filter might not catastrophically diverge. For the purpose of outdoor complete coverage robots such as a lawnmower or snowplow, any unmodeled non-systematic odometry error is not acceptable because the robot must localize with centimeter-level accuracy for reliable and repeatable navigation.

Throughout this thesis, CWRU Cutter slip behavior is primarily examined using logged data from the 2013 ION Autonomous Snowplow Competition in order to compare filter performance in a worst-case scenario. Specifically, the following examples apply the discussed localization methods (EKF and AMCL) to CWRU Cutter's second Autonomous Snowplow Competition run. It is shown here that unmodeled wheel slip degrades both standard mobile robot localization methods: Extended Kalman Filter and Adaptive Monte Carlo Localization. While a worst-case snowplow scenario is used to demonstrate the corresponding filter errors, keep in mind that any amount of wheel slip will cause a degraded state estimate.

Wheel-Slip Example: Extended Kalman Filter

The Extended Kalman Filter is considered here to have "diverged" when the state estimate $\hat{\boldsymbol{x}}$ is farther from the true state by more than 3σ (where σ is determined using the square root of the diagonal elements of the covariance matrix $\mathbf{P}$).

Wheel-slip clearly affects the position and heading estimate in the following filter example. This EKF example uses the 5-State system model, GPS measurements, IMU angular velocity measurement, and encoders. The measurement noise $\mathbf{R}_{gps}$ and $\mathbf{R}_{imu}$ are inflated by approximately 1 order of magnitude to simulate the EKF's belief level using less-precise sensors.

Figure 5.1 demonstrates EKF behavior during a 30-second section of the 2013 Autonomous Snowplow Competition. At this point, CWRU Cutter pushes snow, causing a severe non-systematic odometry error. The EKF optimally merges all sensor information including faulted data. The result, as seen in Figure 5.1 is a smooth state estimate where the odometry incorrectly propagates the EKF state and heading, while the GPS continues to "pull" the state estimate towards the absolute GPS measurement.

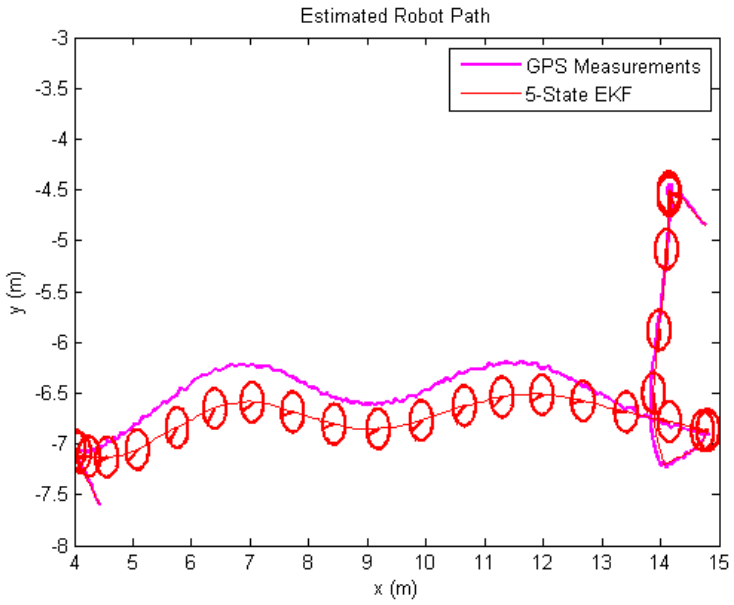

FIGURE 5.1: EKF Localization during severe wheel slip

Wheel-Slip Example: Adaptive Monte Carlo Localization

Baseline AMCL localization also diverges during the 2013 ASC encoder fault shown here. Interestingly, AMCL behaves differently from the EKF. AMCL, as a non-parametric filter, experiences several "jumps." As particles propagate incor-

rectly using the faulted odometry data, eventually systematic random noise allows localization to suddenly reconverge on the absolute GPS measurement. This procedure occurs several times throughout the course of the encoder fault.

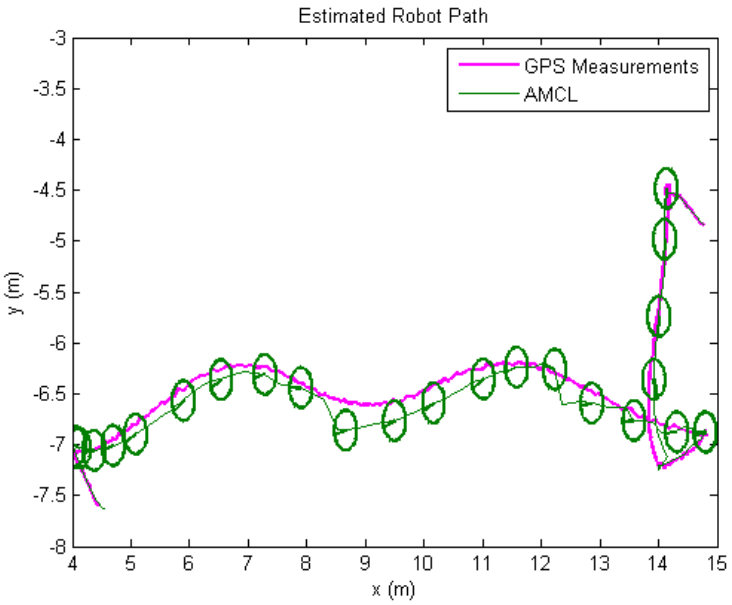

FIGURE 5.2: AMCL during severe wheel slip

5.2 Augmented Extended Kalman Filter

To solve the problem of non-systematic odometry errors, wheel-slip is detected and estimated using an augmented Extended Kalman Filter with wheel velocity error states v_R^{err} and v_L^{err}. It will be shown that the augmented Kalman Filter (AKF) requires a relatively small, conceptually straightforward adjustment to the baseline 5-State EKF. Wheel slip for the AMCL procedure is not solved within this thesis.

5.2.1 Motivation for Wheel-Slip Methodology

When a Kalman Filter (linear or non-linear) diverges beyond the parametric 3σ bounds of uncertainty, the error can usually be classified as one of two types of errors:

1. Noise-Based Error: The noise is incorrect for either the system noise ($\mathbf{Q}$) or the measurement noise ($\mathbf{R}$). Most noise-based errors can be eliminated (or at least reduced) by iteratively tuning noise parameters using actual robot data.

2. Model-Based Error: The system or measurement equations incorrectly model reality. Model-based errors are particularly troublesome for localization; If reality does not conform to the system model and measurement models, localization algorithms diverge or operate incorrectly.

According to Equations 5.1 and 5.2, wheel slip clearly constitutes a model-based error because reality is not expressed using the baseline system/measurement equations. The typical procedure to handle model-based errors is to either artificially inflate the system/measurement noise or to estimate the error using an additional state. A common example is already described in Section 4.1.2: the IMU measurement exhibits a constant angular velocity bias, so an additional bias state is added to the Extended Kalman Filter in order to estimate this bias whenever using an IMU.

Similarly, non-systematic odometry error will be addressed in this thesis using additional, augmented states inside the EKF. It was decided to use an augmented-EKF approach for non-systematic odometry error rather than a noise-based approach because:

1. Artificially inflating the encoder noise $\mathbf{R}_k^{enc}$ de-weights the encoder velocity measurements, causing sub-optimal filter performance during all times when wheels are not slipping.

2. Wheel slip can be easily modeled using augmented states, making non-systematic odometry error estimation conducive to a model-based solution.

5.2.2 Wheel-Slip Estimation

The velocity error due to wheel-slip is estimated by augmenting the EKF state vector with wheel velocity error states v_R^{err} and v_L^{err}. The augmented Extended Kalman Filter state vector becomes:

$$\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v & \omega & v_R^{err} & v_L^{err} \end{bmatrix}^T \tag{5.3}$$

Due to the addition of two extra states, this particular system model is referred to here as the "7-State EKF." Notice the first five states are exactly the same as for the 5-State EKF; this augmented EKF simply adds two additional states representing velocity error for each wheel.

The augmented 7-State EKF brings important implications to the system model and sensor measurement models derived for the 5-State EKF:

- System Model: No change necessary.
- GPS Measurement: No change necessary.
- IMU Measurement: No change necessary.
- Encoder Measurement: Each encoder measurement is sensitive to its corresponding velocity error as illustrated in Equations 5.1 and 5.2.

Notably, the only significant change from the 5-State EKF to the 7-State EKF is that each encoder measurement is sensitive to its wheel-velocity error state. The new encoder measurement, shown in Equation 5.4, now reflects both the true wheel velocity and the wheel velocity error:

$$\mathbf{h}_{enc}^{\text{7-State}}(\boldsymbol{x}_k) = \begin{bmatrix} v_R^{enc} \\ v_L^{enc} \end{bmatrix} = \begin{bmatrix} v_k + \frac{b}{2}\omega_k + v_{R,k}^{err} \\ v_k - \frac{b}{2}\omega_k + v_{L,k}^{err} \end{bmatrix} \tag{5.4}$$

The corresponding linearized Jacobian matrix for the encoder measurement model

becomes:

$$\mathbf{H}_{enc}^{\text{7-State}}(\boldsymbol{x}) = \begin{bmatrix} 0 & 0 & 0 & 1 & \frac{b}{2} & 1 & 0 \\ 0 & 0 & 0 & 1 & -\frac{b}{2} & 0 & 1 \end{bmatrix} \tag{5.5}$$

Other EKF equations remain essentially unchanged. The revised EKF equations are shown for the augmented 7-State EKF. The system model is:

$$\mathbf{f}^{\text{7-State}}(\boldsymbol{x}) = \begin{bmatrix} \mathbf{f}^{\text{5-State}}(\boldsymbol{x}) & f_{v_R^{err}} & f_{v_L^{err}} \end{bmatrix}^T \tag{5.6}$$

$$f_{v_R^{err}} = v_{R,k}^{err} = v_{R,k-1}^{err} \tag{5.7}$$

$$f_{v_L^{err}} = v_{L,k}^{err} = v_{L,k-1}^{err} \tag{5.8}$$

And the corresponding system model Jacobian is:

$$\mathbf{F}_k^{\text{7-State}}(\boldsymbol{x}) = \begin{bmatrix} \mathbf{F}_k^{\text{5-State}} & \mathbf{0}_{5\times 2} \\ \mathbf{0}_{2\times 5} & \mathbf{I}_{2\times 2} \end{bmatrix}_{7\times 7} \tag{5.9}$$

The 7-State GPS measurement model and Jacobian matrix are based entirely on the 5-State GPS measurement:

$$\mathbf{h}_{gps}^{\text{7-State}}(\boldsymbol{x}_k) = \begin{bmatrix} x_{gps} \\ y_{gps} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_{gps}^{\text{5-State}}(\boldsymbol{x}_k) \end{bmatrix} \tag{5.10}$$

$$\mathbf{H}_{gps}^{\text{7-State}}(\boldsymbol{x}_k) = \begin{bmatrix} \mathbf{H}_{gps}^{\text{5-State}} & \mathbf{0}_{2\times 2} \end{bmatrix} \tag{5.11}$$

Similarly, the 7-State IMU measurement model and Jacobian matrix are:

$$\mathbf{h}_{imu}^{\text{7-State}}(\boldsymbol{x}_k) = \begin{bmatrix} \omega_{imu} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_{imu}^{\text{5-State}}(\boldsymbol{x}_k) \end{bmatrix} \tag{5.12}$$

$$\mathbf{H}_{imu}^{\text{7-State}}(\boldsymbol{x}_k) = \begin{bmatrix} \mathbf{H}_{imu}^{\text{5-State}} & \mathbf{0}_{1\times 2} \end{bmatrix} \tag{5.13}$$

In summary, the proposed augmented Extended Kalman Filter allows independent wheel velocity error estimation. The AKF requires only two significant changes from a typical 5-State mobile robot EKF:

1. Augmented State Vector (Equation 5.3)

2. Modified Encoder Measurement Update (Equation 5.4)

These two changes are sufficient to estimate, detect, and reject a wheel velocity error during real-time robot operation.

5.2.3 Discussion

The proposed augmented Extended Kalman Filter is theoretically capable of estimating any wheel velocity error. For this thesis, the augmented EKF is specifically directed towards non-systematic odometry error estimation; however, another realistic wheel velocity error could be caused through incorrectly-calibrated odometry parameters (such as the encoder ticks-per-meter conversion or the robot track width). The proposed 7-State EKF is shown in Chapter 6 to correctly filter both wheel-slip errors and incorrect odometry parameters (systematic odometry errors) indescriminately.

Conceptually, the augmented EKF "works" because redundant sensor measurements allow the filter to simultaneously estimate both the true velocities v, ω and the wheel encoder error v_R^{err}, v_L^{err}. Redundant measurements in the CWRU Cutter system are visualized in Table 5.1. A darkened square in Table 5.1 indicates a measurement is sufficient to observe the associated state, while a non-darkened square indicates that other measurements must be applied for the state to be observable.

Table 5.1: State Observability for each EKF Measurement

	x	y	θ	v	ω	v_R^{err}	v_L^{err}
GPS (x_{gps}, y_{gps})	■	■	■	■	■		
Encoders (v_R^{enc}, v_L^{enc})				□	□	■	■
IMU (ω_{imu})					■		

Table 5.1 is primarily intended as a conceptual visualization, so it will not be rigorously proved here. However, Table 5.1 reveals several important considerations regarding the augmented EKF. A similar Observability test structured using

Equation 4.27 (in Section 4.1.2) would verify the following filter behavior:

1. The full 7-State EKF can be observed using only an encoder measurement update and an absolute x, y measurement (in this case a GPS, but the filter may be generalized to implement any absolute measurement instead).

2. Without a filtered x, y measurement, no redundant sensor currently exists on the CWRU Cutter robot to estimate the forward velocity v. This is a critical motivation for implementing wheel-velocity error detection using an Extended Kalman Filter which includes v and w states. Without the GPS measurement, wheel-velocity error would not be observable.

3. The IMU measurement is not necessary for the 7-State system to be observable, but it will help estimate the velocity states: $v, w, v_R^{err}, v_L^{err}$.

5.3 Alternate Approaches

Several alternate approaches for wheel-slip detection and handling were explored throughout the work for this thesis.

Ad-Hoc Slip Detection During 2013 ION ASC

The localization procedure during the 2013 ION ASC implemented an ad-hoc sensor fusion procedure which incorporated two elements for wheel slip handling:

1. The encoder and IMU angular velocities were fused according to the equation: $\omega_{tot} = \alpha\omega_{enc} + (1 - \alpha)\omega_{imu}$. For the ASC, α was chosen to be 0.1.

2. Forward velocity error was detected by comparing a moving average of the GPS velocity with the encoder forward velocity. When the moving averages fit certain logic comparisons, the robot is said to have "stalled."

These improvements were sufficient during the ION ASC, but exhibit critical drawbacks: First, the ω velocity fusion is a statically tuned weighting operating without

regard to the encoder or IMU measurement noise. While the onboard Christa IMU was precise enough to be used in place of the encoder w, this will not be the case in all situations; encoders are unfortunately ignored during times when they provide true information. Second, any forward velocity error (that is, any wheel slip occuring to both left and right wheels at the same time) is only detected during a stall condition; forward velocity errors are not removed from the overall localization solution.

The ad-hoc system for slip detection used on the CWRU Cutter robot during the 2013 ASC competition, while functional, was not a robust solution. Likely, the approach would fail with any sensors other than precise GPS and IMU sensors, and the ad-hoc approach would certainly not be suitable for a commercial robotic snowplow.

Federated Kalman Filter

The Federated Kalman Filter (FKF) architecture is occasionally used for Fault Detection and Isolation in inertial navigation [15]. The FKF, described briefly in Section 2.2, uses redundant sensors which each operate inside separate "local filters." Local filters are fused in a "master filter," which produces the global state estimate at each timestep and may or may not feed its estimate back to the local filters. Theoretically, the no-feedback FKF can completely isolate a sensor fault by instructing the master filter to simply ignore the faulted local filter.

Because of FKF fault tolerance, the FKF was briefly investigated for handling non-systematic odometry errors; the idea was that redundant sensors could provide velocity measurements which allow a local filter to detect when encoders exhibit faulted data. The FKF states were: $\boldsymbol{x} = \begin{bmatrix} v & \omega & \dot{v} & \dot{\omega} \end{bmatrix}$. The FKF did not include any absolute states (x, y, θ) so as to allow the filtered odometry result to be used inside either AMCL or the baseline 5-State EKF. In practice, critical drawbacks were found:

1. No way to observe v: An IMU gyroscope provides an ω measurement, but

CWRU Cutter currently does not implement any method to directly measure forward velocity. The IMU accelerometer, which provides a $\dot{v}$ measurement, was not precise enough to detect a forward velocity encoder error.

2. Fault detection was unreliable: FKF fault detection relies on logical comparisons of local-filter properties (state and covariance). The fault detection procedure often could not detect a small non-systematic odometry error or behaved differently in simulations depending on random noise.

3. Too many sensors are needed: The FKF works best with multiple sensor measurements (at least three). Unfortunately, CWRU Cutter hardware only includes a small number of sensors. Three separate measurements would be necessary for both v and ω.

Overall, the augmented Kalman Filter simply works better. The proposed AKF including position, heading, and velocity error states can estimate odometry errors with more accuracy and reliability than the simulated FKF.

6 Wheel-Slip Results

To evaluate the augmented Extended Kalman Filter with individual wheel velocity error states, tests were conducted both in simulation and on the CWRU Cutter Hex robot platform. Simulation tests aimed to verify the Kalman filter theory in a controlled environment. Implementation results incorporated selected Kalman filter architectures into the CWRU Cutter software and compared filter performance relying on data from GPS, encoders, and IMU.

6.1 Simulation Results

6.1.1 Overview and Simulation Setup

Simulation tests conducted in MATLAB verified the target Kalman Filter architecture to be implemented onboard the CWRU Cutter robot. Two critical results were found:

1. The Extended Kalman Filter with 7 states, $\begin{bmatrix} x & y & \theta & v & \omega & v_R^{err} & v_L^{err} \end{bmatrix}^T$, as described in Section 5.2, outperforms both a baseline EKF without error states and an EKF with two additional acceleration states, $\dot{v}$ and $\dot{\omega}$ (acceleration states $\dot{v}, \dot{\omega}$ were included in previous versions of the CWRU Cutter robot).

2. For this mobile robot system, the Extended Kalman Filter performs comparably to the Iterated Extended Kalman filter and the Unscented Kalman Filter.

To perform all tests consistently, the simulated robot follows a predefined trajectory consisting of a series of commanded velocity and angular velocity values. The trajectory results in a hidden true state, x_{true}, not known to the Kalman Filter estimator. For these tests, the x, y position of x_{true} forms the arbitrary track shown in Figure 6.1. The minimal possible sensors were simulated for the system to be fully observable in all filter configurations. The 7-state filter requires both a GPS measurement (an x, y reading) and an encoder measurement (a v_R^{enc}, v_L^{enc} reading). An IMU angular rate sensor (available on the CWRU Cutter robot) was not used in simulation because an angular rate measurement is not necessary for the system to be observable. GPS is simulated with a lever arm offset of $0.5m$. Any additional sensors should improve the estimate accuracy; however, in order to test the EKF it is important to show the filter is operational with the minimal sensor measurements.

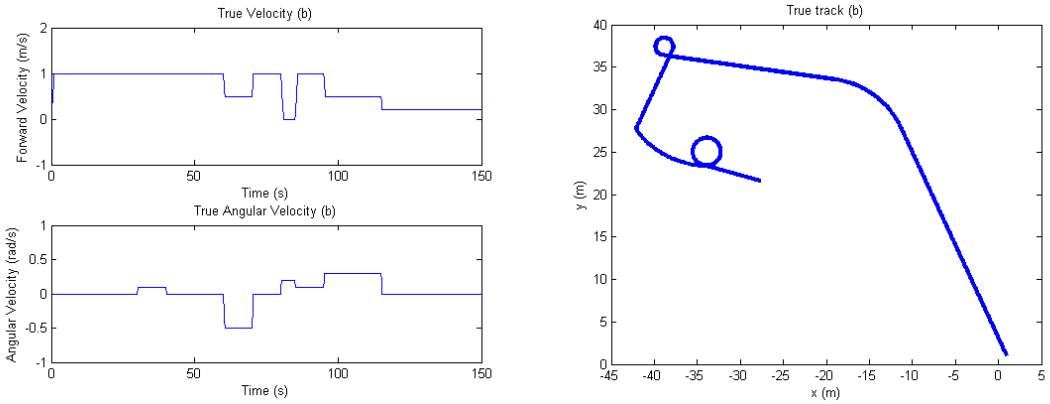

FIGURE 6.1: Simulated true robot trajectory and track

As discussed, the GPS sensor currently implemented on the CWRU Cutter robot is a high-precision RTK Differential GPS. Unless otherwise noted, it is assumed for simulation that the GPS x and y standard deviations are 5.0 cm each. Similarly, the encoders are assumed to be a very accurate measurement of the distance the wheel turns between sensor readings. System noise and measurement noise are set to nominal values representing the measurement accuracy of CWRU Cutter's sensors and a realistic system noise value for each filter.

Each simulated measurement is derived from the equation: $y = h_k(x_{true}) + v_k$, where $v_k \sim N(0, R_k)$, which represents zero-mean gaussian noise with covariance R_k. When comparing filter architectures, the random noise is reseeded with a known quantity so each filter receives exactly the same measurements at each timestep.

For simulation tests, a non-systematic odometry error (wheel-slip) is simulated as a constant wheel velocity step fault. That is, during the time $a <= t <= b$, the encoder measurement (before adding noise) is expressed as:

$$\begin{bmatrix} v_R^{enc} \\ v_L^{enc} \end{bmatrix} = \begin{bmatrix} v_R^{true} \\ v_L^{true} \end{bmatrix} + \begin{bmatrix} v_R^{err} \\ v_L^{err} \end{bmatrix} \tag{6.1}$$

Table 6.1 shows the parameters a, b, v_R^{err}, and v_L^{err} for the faults simulated in the following tests.

TABLE 6.1: Simulation fault parameters

	$v_R^{err}(m/s)$	$v_L^{err}(m/s)$	a	b
No Fault	0.0	0.0	–	–
Fault 1	0.1	0.0	70	80
Fault 2	0.3	0.3	20	30

To evaluate filter accuracy, a "position error" metric is used which is defined here as the norm of the distance between the estimated position and the true position (see Equation 6.2).

$$d_{err} = \sqrt{(x_{true} - x_{est})^2 + (y_{true} - y_{est})^2} \tag{6.2}$$

6.1.2 Test 1: 5-State, 7-State, and 9-State EKFs

In Test 1, the proposed augmented EKF with wheel-velocity error states is shown to detect and reject wheel faults. Three Extended Kalman Filter architectures are compared in Test 1:

1. 5-State Extended Kalman Filter without wheel velocity error states: $\boldsymbol{x} =$

$\begin{bmatrix} x & y & \theta & v & \omega \end{bmatrix}^T$. This filter is used as a control to compare performance.

2. 7-State Extended Kalman Filter: $\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v & \omega & v_R^{err} & v_L^{err} \end{bmatrix}^T$.

3. 9-State Extended Kalman Filter implementing additional acceleration states: $\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v & \omega & v_R^{err} & v_L^{err} & \dot{v} & \dot{\omega} \end{bmatrix}^T$. Because acceleration states were used in past versions of the CWRU Cutter robot, this test attempts to determine the practical effect of including additional acceleration states.

Figure 6.2 shows the filtered velocity and angular velocity error during a single test for Fault 1 of all three EKFs. Both the 7-State and 9-State EKFs with wheel velocity error states are able to track the true velocity, while the 5-State EKF diverges when the encoder measurement exhibits a fault. This is expected behavior and verifies the operation of the wheel velocity estimator. Table 6.2 shows the estimated position errors between the three filter configurations under the Fault 1, Fault 2, and No Fault cases, averaged over 50 simulation runs of 100 seconds each. Values of interest are bolded.

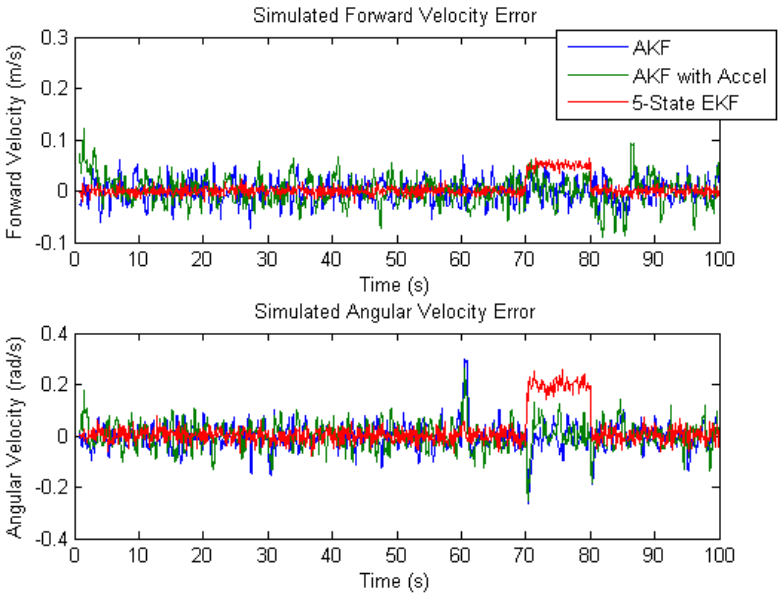

FIGURE 6.2: Velocity and Angular Velocity Error for Fault 1 ($x_{est} - x_{true}$)

According to Table 6.2, the 5-State EKF (without wheel velocity error states) shows an approximately 50% improvement for v, ω estimates over the 7-State

Table 6.2: EKF Simulation Error

		Dist (cm)	θ (deg)	v (cm/s)	ω (deg/s)
5-States	No Fault	2.2	1.5	**0.7**	**1.4**
	Fault 1	6.8	**5.2**	1.7	3.8
	Fault 2	**17.5**	1.5	**9.5**	1.4
7-States	No Fault	1.8	1.5	**1.9**	**2.3**
	Fault 1	1.8	**1.6**	2.1	2.5
	Fault 2	**2.4**	1.5	**4.3**	2.3
9-States	No Fault	1.8	1.6	2.0	2.5
	Fault 1	1.8	1.6	2.2	2.8
	Fault 2	2.4	1.6	4.4	2.5

EKF (with wheel velocity error states) during the No Fault case. This can be conceptualized through Figure 6.2: the filters with wheel velocity error states v_R^{err} and v_L^{err} exhibit a higher noise floor in the steady state (no wheel-slip) case.

The 7-State and 9-State EKFs perform significantly better than the 5-State EKF control filter when there is a wheel fault; in this simulation, the 7-State EKF improves the heading error for Fault 1 (a single wheel slip) by a factor of 3 and the position error for Fault 2 (both wheels slip) by a factor of 6. The acceleration error states do not appear to improve filter performance over the proposed 7-State EKF.

Simulation Test 1 demonstrates that the 7-State EKF handles an encoder fault correctly while an EKF without wheel velocity error states diverges during the time of the fault. Also, the first simulation result proves the augmented EKF theory is sound, the nonlinear system can be tracked, and the minimmum number of measurements (GPS, encoders) will observe the full system.

6.1.3 Test 2: EKF, Iterated EKF, and Unscented KF

Test 2 compares filter performance between the EKF and other common nonlinear estimation filters: the Unscented Kalman Filter and the Iterated Extended Kalman filter, both discussed in Section 4.1.3. In all three cases, the estimators use 7 states: $\begin{bmatrix} x & y & \theta & v & w & v_R^{err} & v_L^{err} \end{bmatrix}^T$. Table 6.3 shows the position errors of the three filters under test, for three different scenarios: No Fault, Fault 1, and

Fault 2.

Table 6.3: EKF, UKF, IEKF Simulation RMS Error

		Dist (cm)	θ (deg)	v (cm/s)	ω (deg/s)
EKF	No Fault	**1.3**	1.2	1.9	2.1
	Fault 1	**1.3**	**1.3**	2.1	2.2
	Fault 2	1.5	1.2	**3.9**	2.1
Iter. EKF	No Fault	**1.2**	1.1	1.9	2.1
	Fault 1	**1.2**	**1.1**	2.1	2.2
	Fault 2	1.5	1.1	**3.8**	2.1
Unsc. KF	No Fault	2.1	0.81	1.7	2.1
	Fault 1	2.1	0.85	1.7	2.2
	Fault 2	2.2	0.81	3.5	2.1

The filters under test perform with comparable results; the position error and heading error are within approximately 5-10% between EKF and IEKF, while the UKF performs better for some states and worse for others. The Unscented KF and the Iterated EKF both offer occasional benefits; in particular, the IEKF improves over the EKF when the measurement is significantly nonlinear, and the Unscented Kalman Filter better estimates the heading of the robot (but the position estimate is slightly less accurate).

Test 2 shows no clear "winning" estimator. The Unscented Kalman Filter may perform better with a thorough tuning, while the Iterated EKF would be the recommended filter choice for especially non-linear measurements (landmark measurements, a large GPS lever arm offset, etc), but neither the IEKF nor the UKF appear necessary for this system. So, for its simplicity, the 7-State Extended Kalman Filter was chosen to be implemented on the CWRU Cutter robot.

6.2 Implementation Results

The proposed augmented Extended Kalman Filter is implemented on the CWRU Cutter robot platform and tested using real-time data playback. The AKF with wheel velocity error states is shown to accurately reject wheel slip situations (non-systematic odometry errors) and outperform a baseline EKF lacking wheel velocity

error states. The AKF estimates the full state using GPS, encoders, and IMU measurements. In addition, the AKF is shown to robustly handle incorrect odometry calibration parameters (systematic odometry errors).

6.2.1 CWRU Cutter Setup

The augmented Extended Kalman Filter is implented in CWRU Cutter software according to Section 5.2 using the Robot Operating System (ROS) and the Bayesian Filtering Library. Notably, mainstream ROS users tend towards indoor mobile robots, so an outdoor-oriented EKF which fuses GPS measurements with other sensors is not widely accepted. Because of this, a ROS-compatible Extended Kalman Filter was created specifically for this thesis. The EKF runs nominally at 10Hz, which is the same frequency tested in the MATLAB simulations and used in past versions of CWRU Cutter.

The tests in this section compare results between 2 separate Extended Kalman Filter architectures:

1. 5-State EKF (no error states): $\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v & w \end{bmatrix}^T$
2. 7-State EKF (wheel velocity error states): $\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v & w & v_R^{err} & v_L^{err} \end{bmatrix}^T$

The system noise is initialized as a diagonal matrix with values as indicated in Equations 6.3 and 6.4. The system noise matrix, $\mathbf{Q}$, is held constant for all tests. Off-diagonal elements are all set to 0.

$$\begin{aligned} \mathbf{Q}_{diag}^{5-State} &= \begin{bmatrix} \sigma_x^2 & \sigma_y^2 & \sigma_\theta^2 & \sigma_v^2 & \sigma_\omega^2 \end{bmatrix} \\ &= \begin{bmatrix} 0.01^2 & 0.01^2 & 0.02^2 & 0.40^2 & 0.40^2 \end{bmatrix} \end{aligned} \tag{6.3}$$

$$\begin{aligned} \mathbf{Q}_{diag}^{7-State} &= \begin{bmatrix} \sigma_x^2 & \sigma_y^2 & \sigma_\theta^2 & \sigma_v^2 & \sigma_\omega^2 & \sigma_{v_R^{err}}^2 & \sigma_{v_L^{err}}^2 \end{bmatrix} \\ &= \begin{bmatrix} Q_{diag}^{5-State} & 0.30^2 & 0.30^2 \end{bmatrix} \end{aligned} \tag{6.4}$$

The measurement noise, $\mathbf{R}_k$, for the three measurements is given below. The measurements are all in accordance with the measurements discussed in Section 5.2.

1. GPS Measurement: The GPS measurement noise is represented as a diagonal matrix with fixed values for the noise in the x and y components (Equation 6.5). Note that the x and y standard deviation for the RTK GPS has been measured to be approximately 1-2cm, so the EKF measurement noise is inflated by an order of magnitude for the implementation tests.

2. IMU Measurement: The IMU is used for a single angular rate measurement (Equation 6.6). Similar to GPS, the IMU standard deviation has been measured at approximately an order of magnitude more precise than used in the EKF measurement noise.

3. Encoder Measurement: The encoder measurement noise is expressed as a function of the wheel motion, which was tuned experimentally. The result is that the encoders will inject some small amount of noise by default, plus an additional amount of noise dependent on the magnitude of the wheel velocity. For all tests, $\alpha = \epsilon = 0.001$.

$$\mathbf{R}_k^{gps} = \begin{bmatrix} 0.1^2 & 0 \\ 0 & 0.1^2 \end{bmatrix} \tag{6.5}$$

$$\mathbf{R}_k^{imu} = \begin{bmatrix} 0.2^2 \end{bmatrix} \tag{6.6}$$

$$\mathbf{R}_k^{enc} = \begin{bmatrix} \alpha \left| v_R^{enc} \right| + \epsilon & 0 \\ 0 & \alpha \left| v_L^{enc} \right| + \epsilon \end{bmatrix} \tag{6.7}$$

To perform tests, data were played back in real-time using ROS's logging system (Section 3.1.3). Messages are received in the playback program in the same order of arrival at the time of logging; the data log is a faithful representation of true sensor outputs.

$$\begin{bmatrix} x_{est} \\ y_{est} \end{bmatrix} = \mathbf{h}_k^{GPS}\left(\hat{\boldsymbol{x}}_k^+\right) \tag{6.8}$$

$$d_{err} = \sqrt{(x_{gps} - x_{est})^2 + (y_{gps} - y_{est})^2} \tag{6.9}$$

In order to compare filter performance, tests compare position error between the state estimate and the measured GPS coordinate at each time step (Equation 6.9). While it may seem contradictory to use differential GPS both in the filter and as an evaluation metric, it is in fact a valid comparison for several reasons:

1. The differential GPS is considered extremely accurate and precise (1-2 cm), so it is reliable as a "ground truth."

2. The filter measurement noise is inflated from this 1-2 cm level in all tests so as to instruct the EKF to act as though the GPS is less reliable. While differential GPS is a convenient measurement to use on the CWRU Cutter robot, an absolute measurement for many other robots will not exhibit the same level of precision.

3. The filter must estimate a state which includes heading, velocity, angular velocity, and potentially a lever arm offset in addition to a simple x, y. A precise GPS measurement does not guarantee filter stability.

4. When encoders experience a wheel slip fault, a typical EKF state estimate diverges beyond the bound of uncertainty.

Finally, additional notes specific to the CWRU Cutter software EKF implementation encountered during development and testing are described here:

1. The EKF is always initialized with an $\hat{\boldsymbol{x}}_0 = \mathbf{0}_{n\times 1}$ vector for the state estimate and a large value for all diagonal covariances, $\boldsymbol{P}_0$. Upon the first few system/measurement updates, the filter converges on the true state.

2. The EKF is timing sensitive. If the system update is applied asyncronously from the GPS measurement update, timing differences can cause unpredictable behavior. This could be addressed in the future by using an asyncronous system update and a forced system update upon receipt of absolute sensors measurement (GPS, landmark, etc). The system update would require a varying filter timestep, dt, depending on the last time that the system was updated.

3. Using an IMU requires an additional bias state in the EKF, as discussed in Section 4.1.2, "CWRU Cutter Measurements." The system noise on this bias state is very small (around $\sigma_{bias} = 0.0001$), which was chosen so the filter will converge on a slow-moving IMU bias. This is particularly necessary for the AKF with wheel velocity error states; if the IMU bias is not estimated, then the bias, however small, will manifest itself as an odometry error.

6.2.2 Non-Slip Baseline

Two non-slip cases, detailed in Table 6.4, were tested as control experiments. Both filters track the true state in the non-slip experiments as expected.

Table 6.4: Non-Slip Tests Summary

Test Name	Description	GPS Lever Arm	IMU Used?
Competition Practice	Practice run on dry pavement before the 2013 ION ASC competition	0.0	Yes
Spring Testing	A remote control path on grass without obvious wheel slips	-0.45	Yes

The "Competition Practice" test track is shown in Figure 6.3. At no point does the filtered state significantly diverge from the absolute measurement. The corresponding position error is shown in Figure 6.4. The estimated velocity errors v^{err}, w^{err} from the 7-State Kalman filter are shown in Figure 6.5. The velocity error estimates, while noisy, are centered around zero– indicating that the 7-State EKF correctly tracks the wheel velocity error in the no-slip case. Note that while

the AKF estimates wheel velocity states in terms of the wheel error (v_R^{err} and v_L^{err}), velocity errors are easier to visualize here in terms of v^{err} and w^{err}.

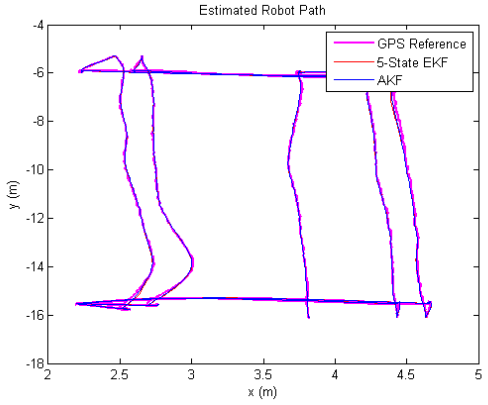

FIGURE 6.3: "Comp. Prac." Test Track

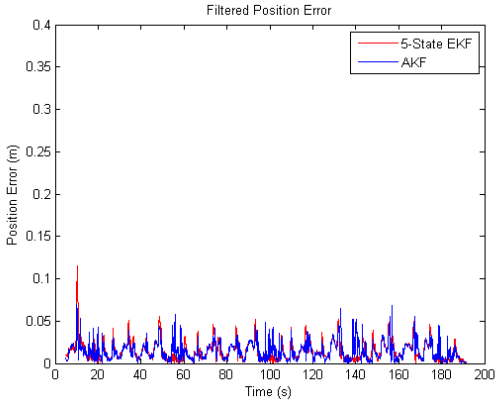

FIGURE 6.4: "Comp. Prac." RMS Error

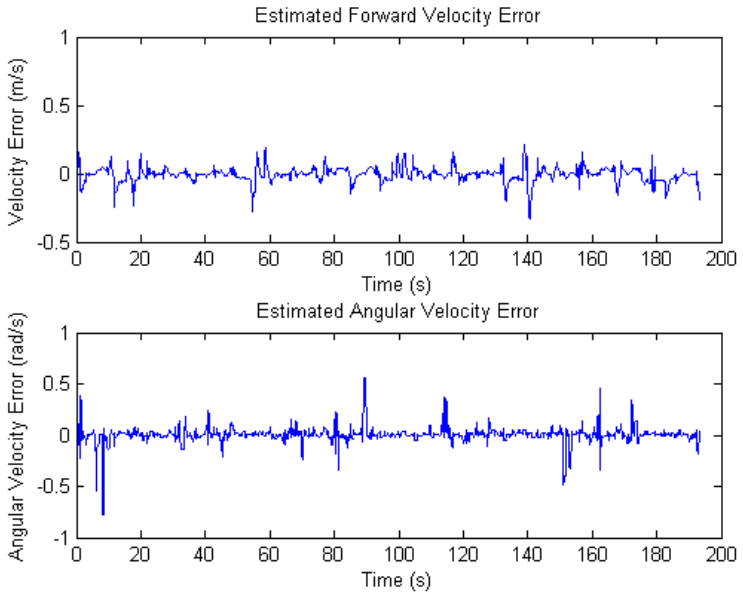

FIGURE 6.5: "Competition Practice" Estimated Velocity Errors

Next, the "Spring Testing" test track is shown in Figure 6.6. This test is different from other tests because the GPS lever arm offset is $-0.45m$ instead of $0.0m$. Figure 6.6 plots the x, y location of the GPS reading and the x, y location of the state estimates. At each individual time step, the location of the state estimate should be offset from the GPS by the lever arm offset (this explains why pivots create a circle of radius $0.45m$ around the robot origin, as can be observed in Figure 6.6). Position error is shown in Figure 6.7.

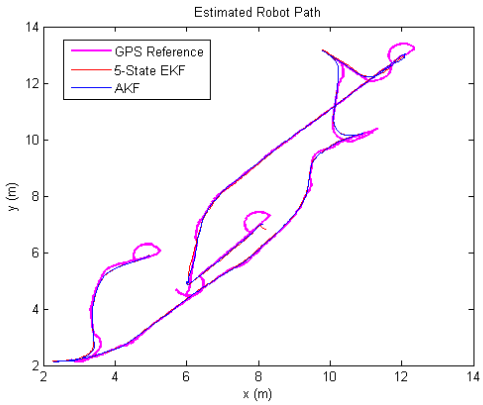

FIGURE 6.6: "Spring Testing" Track

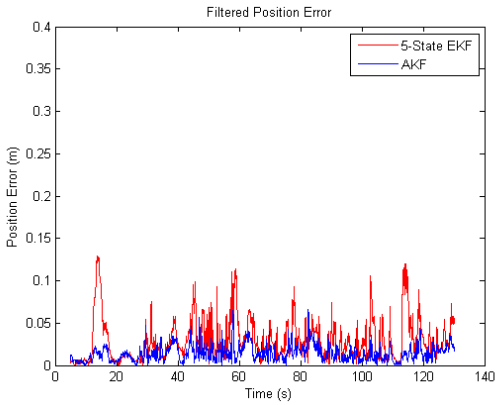

FIGURE 6.7: "Spring Testing" RMS error

In both tests, the mean position error is below $0.04m$ for both the 5-State EKF control and the proposed augmented Extended Kalman Filter. Interestingly, the 7-State AKF position error is smaller than the 5-State EKF error during the datalog when driving on grass (Figure 6.7); apparently, the robot experienced wheel-slip significant enough to degrade the localization solution but otherwise undetectable to a casual human observer. Overall, the baseline, no-slip results indicate here that the AKF with wheel velocity error states can perform with the same precision as a typical mobile robot EKF.

6.2.3 Driving With Wheel-Slip

Two wheel-slip cases were tested, detailed in Table 6.5. For both cases, the filtered (x, y) position is shown and certain sections of the test are highlighted. In both tests, the CWRU Cutter robot is plowing snow and experiences significant wheel slip.

TABLE 6.5: Wheel Slip Tests Overview

Test Name	Description	GPS Lever Arm	IMU Used?
Competition Run 1	CWRU Cutter Competition Run #1 for the 2013 ION ASC competition	0.0	Yes
Competition Run 2	CWRU Cutter Competition Run #2 for the 2013 ION ASC competition	0.0	Yes

"Competition Run 1" is filtered data from the first day of the Institude of Navigation's Autonomous Snowplow Competition. Unfortunately, due to a localization mistake during the actual competition, the robot could not successfully follow a path (which is why the robot veers off course). Nevertheless, post-processing localization can still filter data from this path. The x, y track is shown in Figure 6.8.

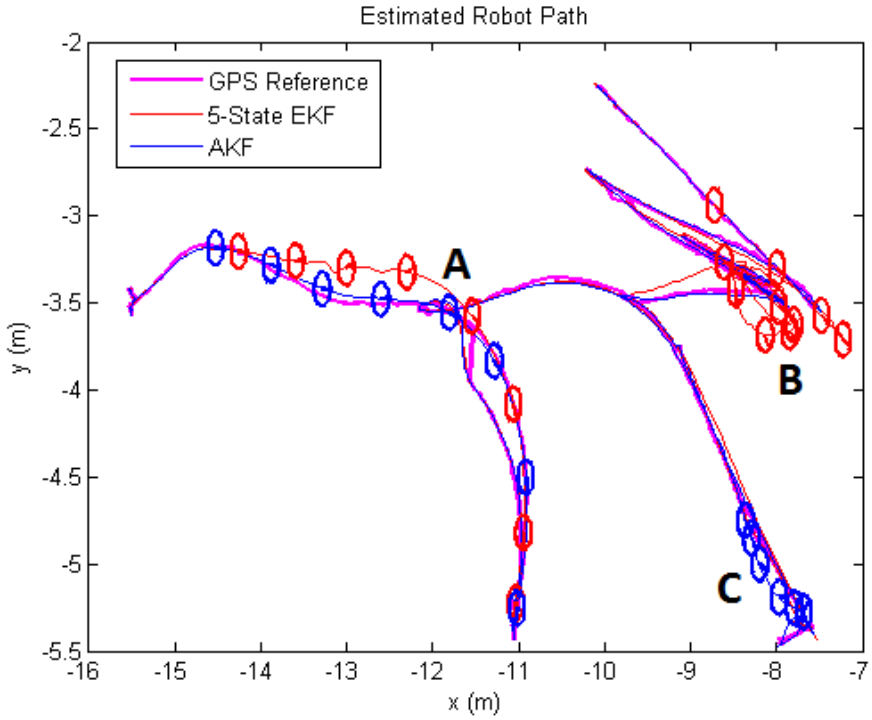

FIGURE 6.8: "Competition Run #1" Track

The three path segments of interested are marked in Figure 6.8:

A) CWRU Cutter Veers Right: The robot encounters snow, the wheels slip under the additional force, and the robot veers right.

B) CWRU Cutter Stalls: The robot stalls while pushing snow (i.e., the wheels turn but the robot does not move). The robot then backs up and resumes the path.

C) Erroneous GPS Reading: The robot receives an erroneous GPS reading which disrupts the filter estimation. Notice the 5-State EKF reconverges

on the true estimate faster than the 7-State EKF, which requires additional measurements for all 7 states to reconverge.

The 5-State EKF x, y estimate diverges from the true path in situations A and B. In comparison, the 7-State EKF with wheel velocity correction is able to maintain localization consistent with the absolute GPS measurements without diverging. The position error for both filters during Fault A is shown in Figure 6.9. Figure 6.10 shows estimated velocity error for the 7-State EKF during Fault A. Note that the encoder angular velocity error reaches nearly $1\frac{rad}{s}$, which indicates the right wheel experiences significant wheel slip during Fault A.

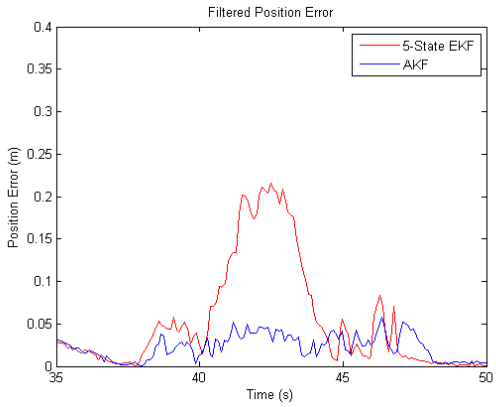

FIGURE 6.9: RMS Error (Test 1A)

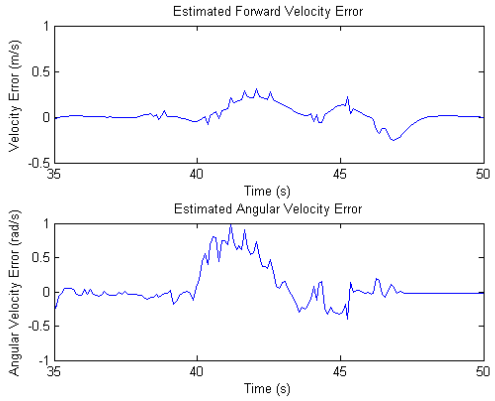

FIGURE 6.10: Velocity Error (Test 1A)

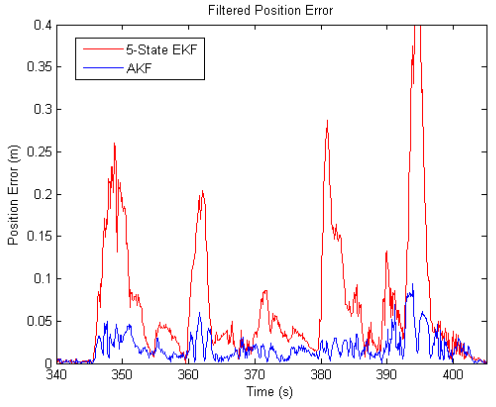

FIGURE 6.11: RMS Error (Test 1B)

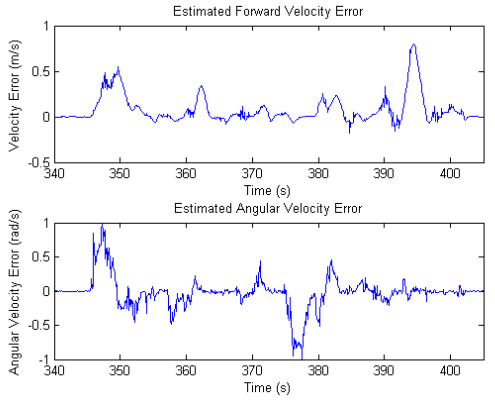

FIGURE 6.12: Velocity Error (Test 1B)

The position error for both filters during Fault B is shown in Figure 6.11, and the corresponding wheel velocity error is shown in Figure 6.12. Fault B is actually a series of several separate non-systematic odometry errors over approximately 60

seconds where the robot attempts to push a large pile of snow and consistently slips. Given the estimated state output from the 7-State Extended Kalman Filter, simple logical thresholds can be applied to determine if a stall occurs. For example, a stall occurs if: the estimated robot forward velocity is less than $0.1m/s$ and the estimated forward velocity error is greater than $0.3m/s$. This concept is applied to Fault B in Figure 6.13 to show a logical true if the robot stalls. Even during the repeated wheel stalls and large velocity errors, the 7-State EKF with wheel velocity error states remains under $0.1m$ position error.

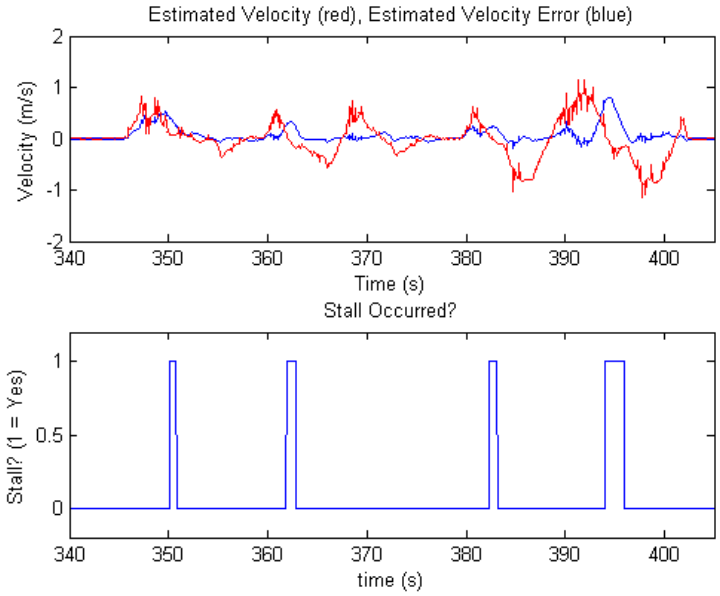

FIGURE 6.13: Stall Detection using the Augmented EKF

"Competition Run 2" uses filtered data from the second day of the Institude of Navigation's Autonomous Snowplow Competition. During this run, localization functioned well enough to drive paths. However, the competition state estimate was corrupted by wheel slip during the straightaways, causing paths to be severely curved as the majority of snow was removed from the course. The x, y track is shown in Figure 6.14. Two points of interest are also marked in Figure 6.14. Here, both Fault A and Fault B represent points when the 5-State EKF diverges from the GPS measurements due to wheel slip on the straightaways.

Faults A and B both exhibit positive angular velocity errors (the right wheel

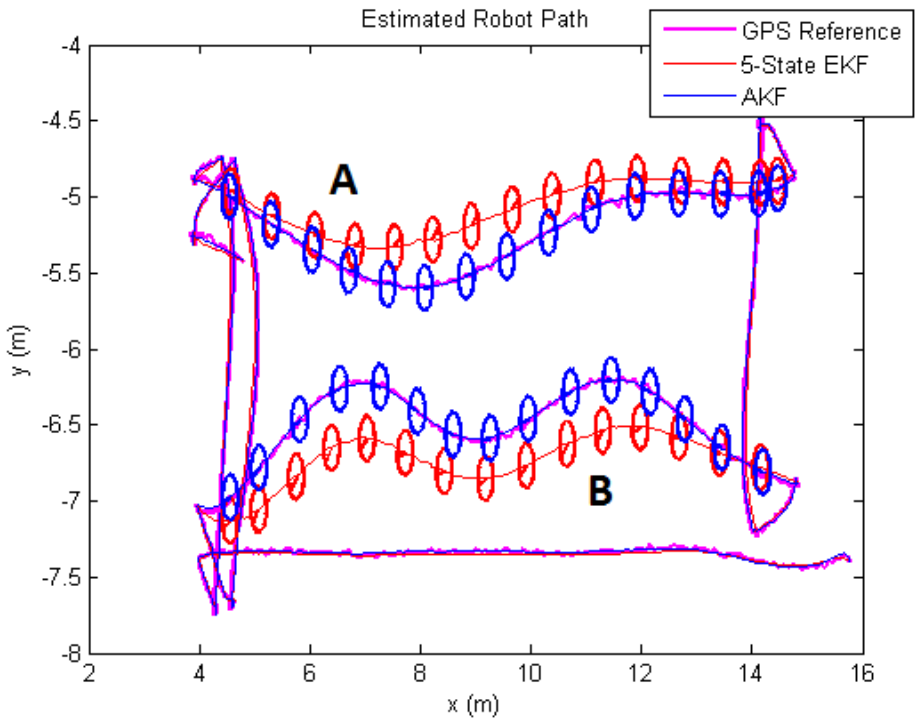

FIGURE 6.14: "Competition Run #2" Track

turns more than the left wheel) for an extended period of time as the robot pushes snow. The position error is shown for Fault A in Figure 6.15 and for Fault B in Figure 6.17. As seen in Figures 6.16 and 6.18, the estimated angular velocity error reaches and maintains approximately $0.5\frac{rad}{s}$ during both faults.

In addition, the estimated heading, θ, during Fault A and Fault B is significantly different between the 5-State and 7-State filters. To evaluate the heading error, subsequent GPS positions were used to find a noisy measure of the true robot

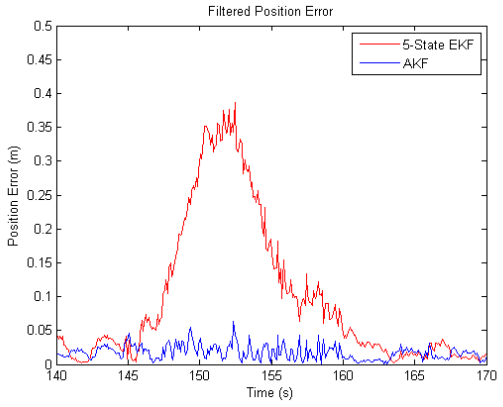

FIGURE 6.15: RMS Error (Test 2A)

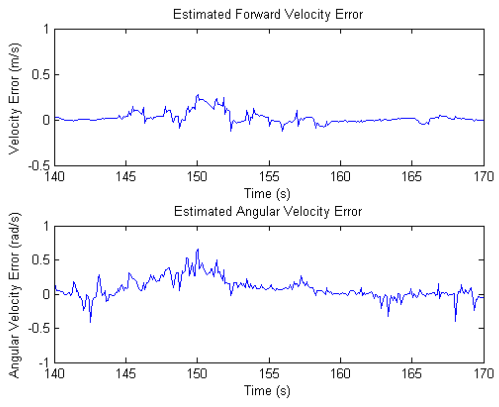

FIGURE 6.16: Velocity Error (Test 2A)

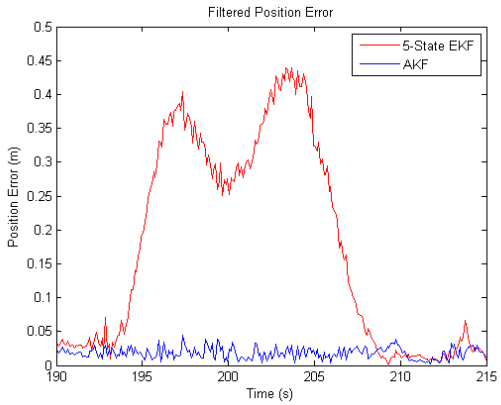

FIGURE 6.17: RMS Error (Test 2B)

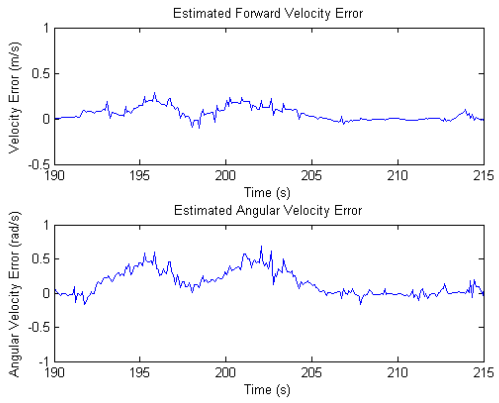

FIGURE 6.18: Velocity Error (Test 2B)

heading while the robot is in motion. This GPS heading was smoothed using a moving average with a window of 5 samples (0.5 seconds). Note that this GPS-based heading measurement is never used explicitly in any Extended Kalman Filter; it is simply a derived value used here for visualization. Figures 6.19 and 6.20 show a comparison of the estimated heading (including 3σ error bars) during Faults A and B, respectively. According to these figures, the 5-State EKF filter diverges in heading beyond the expected bounds of uncertainty, but the proposed AKF with wheel velocity error state continues to track the heading correctly during non-systematic odometry errors.

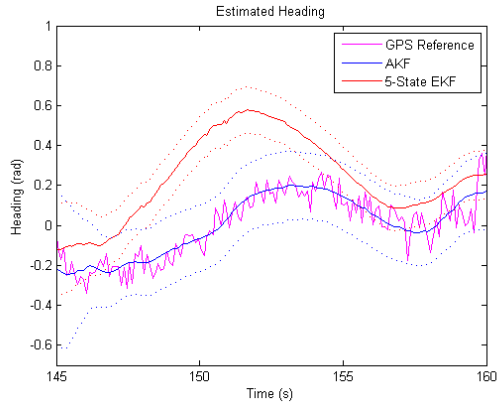

FIGURE 6.19: Heading Error (Test 2A)

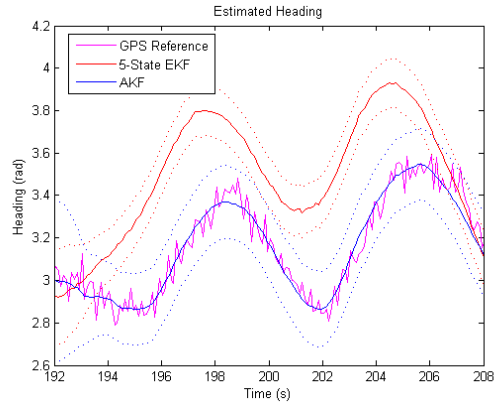

FIGURE 6.20: Heading Error (Test 2B)

The mean position error during all wheel slip faults is summarized in Table 6.6. The improvement factor is simply defined here as: $\frac{d^{5-State}}{d^{7-State}}$. An improvement factor greater than 1 indicates an improvement. Improvement factors are not normalized

based on magnitude of the fault or the duration of the fault, so comparing improvement factors between faults is not useful; however, the improvement factor is helpful to quantify how much the AKF improved the state estimate over the baseline 5-State EKF during a specific fault.

Table 6.6: Wheel Slip Tests: Position Error

Test Name	Fault	5-State Position Error (cm)	7-State Position Error (cm)	Improvement Factor
Competition Run 1	Fault 1A	5.4	2.3	2.3
	Fault 1B	7.1	1.8	3.9
	Fault 1C	3.4	6.6	0.52
Competition Run 2	Fault 2A	17.3	2.1	8.2
	Fault 2B	17.4	1.8	9.6

Faults 1A, 1B, 2A, and 2B are all wheel-slip faults. According to Table 6.6, the augmented Extended Kalman Filter with wheel velocity error state performs significantly better in all slip-based errors. Fault 1C, as mentioned before, is an erroneous GPS measurement which causes the AKF to take longer to reconverge on the true state– this is a potential downside of the augmented Extended Kalman Filter, but has not been a significant factor in localization tests so far. To summarize, the augmented Kalman Filter with general-purpose wheel-velocity error states is shown here to significantly outperform a baseline EKF during non-systematic odometry errors.

6.3 Odometry Parameter Compensation

An additional, unexpected benefit of adding wheel velocity error states is that incorrect odometry parameters (ticks-per-meter for left/right wheels, and robot track width) or other systematic odometry errors can be compensated using the proposed augmented Extended Kalman Filter. However, because the filter is designed for wheel slip detection and rejection, any odometry parameter error manifests in the filter as a "pseudo" wheel slip. As a result, odometry parameters can not be *calibrated* accurately at this time, but an incorrect odometry parameter

Table 6.7: Odometry Parameters

Parameter	Tuned Valued	Tested Value
Right Wheel ($\frac{ticks}{m}$)	26500	26500
Left Wheel ($\frac{ticks}{m}$)	**27150**	**20000**
Track Width (m)	0.55	0.55

can still be compensated using the augmented 7-State EKF.

6.3.1 Faulted Odometry Setup

In this test, a systematic odometry error is simulated by modifying wheel odometry parameters as shown in Table 6.3.1. Essentially, the typical left wheel scaling, 27150 ticks-per-meter, is replaced with a false odometry parameter: 20000 ticks-per-meter (a 25% change). The following two filters are compared in this test:

1. 5-State EKF (no error states): $\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v & w \end{bmatrix}^T$
2. 7-State EKF (wheel velocity error states): $\boldsymbol{x} = \begin{bmatrix} x & y & \theta & v & w & v_R^{err} & v_L^{err} \end{bmatrix}^T$

For this test, only GPS and encoder measurements are used (no IMU used). GPS has a lever arm offset of $-0.45m$. The system noise is initialized as a diagonal matrix with values as indicated in Equations 6.10 and 6.11. The measurement noise for the GPS and encoders remain the same as in Section 6.2.1.

$$\begin{aligned} \mathbf{Q}_{diag}^{5-State} &= \begin{bmatrix} \sigma_x^2 & \sigma_y^2 & \sigma_\theta^2 & \sigma_v^2 & \sigma_\omega^2 \end{bmatrix} \\ &= \begin{bmatrix} 0.04^2 & 0.04^2 & 0.04^2 & 0.40^2 & 0.40^2 \end{bmatrix} \end{aligned} \tag{6.10}$$

$$\begin{aligned} \mathbf{Q}_{diag}^{7-State} &= \begin{bmatrix} \sigma_x^2 & \sigma_y^2 & \sigma_\theta^2 & \sigma_v^2 & \sigma_\omega^2 & \sigma_{v_R^{err}}^2 & \sigma_{v_L^{err}}^2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{Q}_{diag}^{5-State} & 0.05^2 & 0.05^2 \end{bmatrix} \end{aligned} \tag{6.11}$$

Data playback and analysis are performed as described in Section 6.2.1.

6.3.2 Faulted Odometry Results

Results indicate that the augmented Extended Kalman Filter can localize using only GPS measurements and *incorrectly* calibrated odometry. In comparison, the baseline 5-State EKF diverges under the systematic odometry error. The x, y track is shown in Figure 6.21 with 3 points of interest where the 5-State EKF (red) diverges. Estimated velocity for both filters is shown in Figure 6.22, with the corresponding Fault points marked.

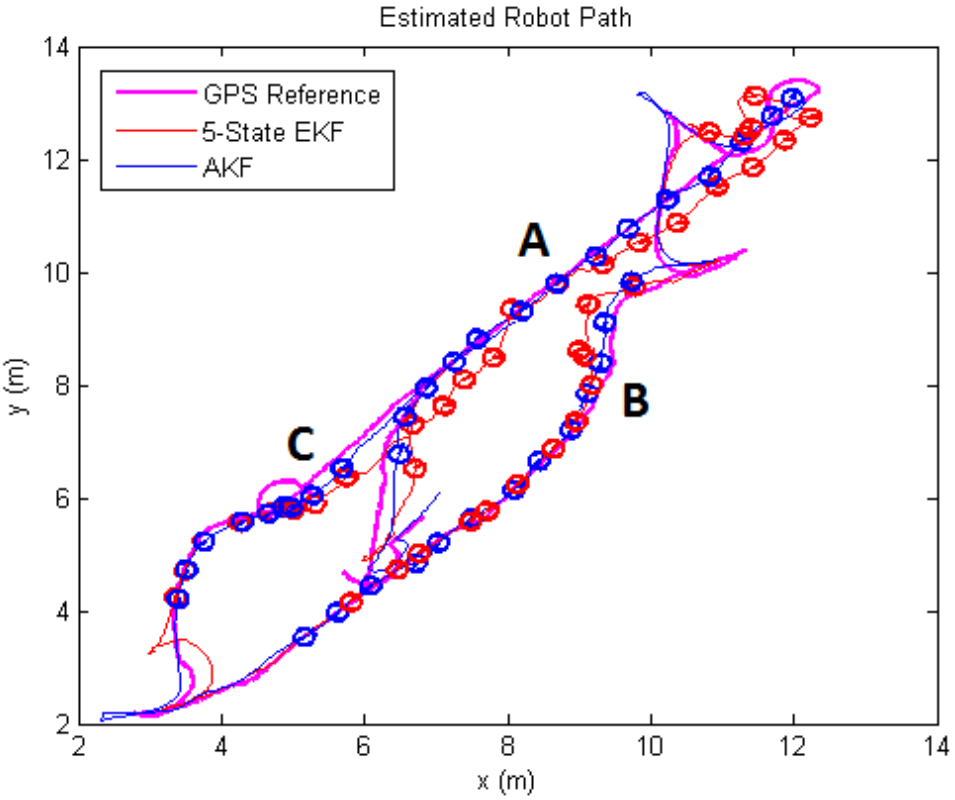

FIGURE 6.21: Odometry Correction: Track

Table 6.8 summarizes the filter behavior at all three points of interest. Relevant velocity values are bolded. In Faults B and C, the 5-State EKF is so degraded that the filter estimates the robot moves in reverse with its heading offset by 180 degrees; notice the estimated forward velocity for the 5-State EKF is approximately negative of the 7-State EKF during Faults B and C (Figure 6.22). Furthermore, Figure 6.23 shows the estimated heading compared with the GPS-extracted heading during Fault B.

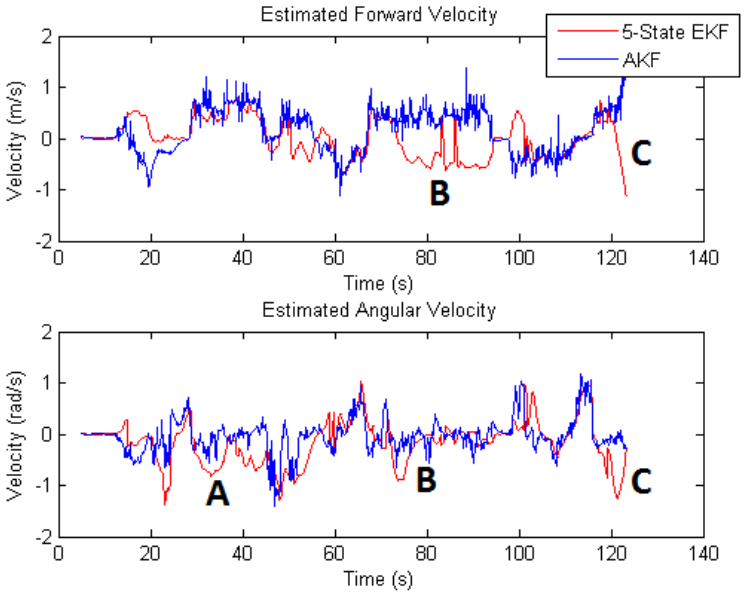

FIGURE 6.22: Odometry Correction: Velocity

TABLE 6.8: Odometry Correction: Average v, w at Points of Interest

		5-State EKF		7-State EKF	
	What Happened?	v $(\frac{m}{s})$	ω $(\frac{rad}{s})$	v $(\frac{m}{s})$	ω $(\frac{rad}{s})$
Fault A	5-State EKF heading diverges	0.52	**-0.48**	0.65	**0.004**
Fault B	5-State EKF moves backwards	**-0.41**	-0.11	**0.45**	-.04
Fault C	5-State EKF turns to move backwards	0.14	**-0.42**	0.52	**-0.02**

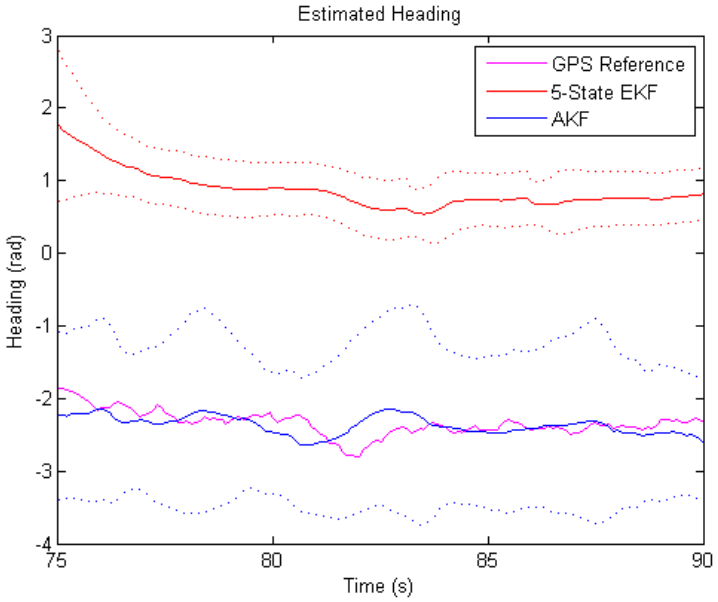

FIGURE 6.23: Odometry Correction: Estimated Heading during Fault B

To illustrate how the proposed AKF with wheel velocity error states compensates for an incorrect odometry parameter, Figure 6.24 compares the estimated wheel velocity (v_R^{true}, v_L^{true}) with the estimated wheel velocity error (v_R^{err}, v_L^{err}). Remember for this test, the systematic odometry error is an incorrect ticks-per-meter scaling for the left wheel. According to Figure 6.24, the estimated right wheel error is consistently centered around zero within 3σ bounds of uncertainty, but the estimated left wheel error varies proportionally with the true left wheel velocity. This is expected behavior and indicates that the AKF with general-purpose wheel velocity error states is correctly estimating the entire state using only GPS and encoder measurements.

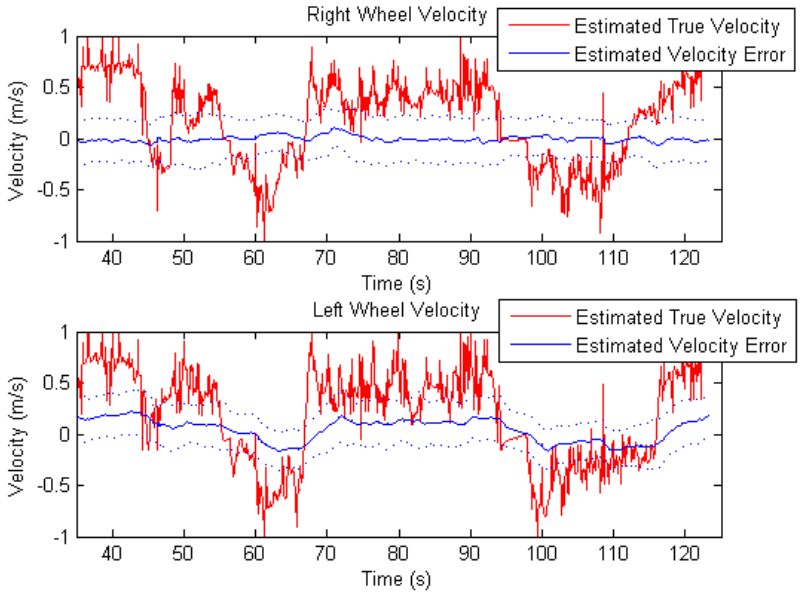

FIGURE 6.24: Odometry Correction: Estimated Wheel Velocity for 7-State EKF

In summary, results indicate that the proposed 7-State AKF is capable of filtering any general wheel-velocity error. While this thesis specifically focuses on its application to non-systematic odometry errors, it is also shown here that the AKF can remain localized in the presence of severe systematic odometry errors as well.

7 Conclusions

CWRU Cutter electronics and software were redesigned during the course of this test development. Multiple localization algorithms (e.g. Extended Kalman Filter, Adaptive Monte Carlo Localization) were derived and implemented successfully using high-precision sensors with inflated noise. The baseline Extended Kalman Filter was designed to include both forward velocity and angular velocity states, and an augmented EKF (AKF) with general-purpose wheel-velocity error states was proposed to robustly handle non-systematic odometry errors. The new AKF modification is conceptually and algorithmically simple because of the included velocity and angular velocity states in the baseline EKF.

The new AKF with wheel-velocity error states was tested both in simulation and using actual logged sensor data from the 2013 Institute of Navigation's Autonomous Snowplow Competition. The AKF was shown to perform consistently better than the baseline EKF during wheel slip conditions; depending on the magnitude of the non-systematic error, position localization during wheel slip in the 2013 ASC was anywhere from 2 times to 8 times closer to the ground truth when using the proposed AKF over the baseline EKF.

Future Research

The augmented Extended Kalman Filter with wheel-velocity error states performed well in simulation and using actual sensor data. Localization algorithms were also shown to accurately localize the CWRU Cutter robot using high-precision sensors. Thesis results lead to several pending avenues of future research for the

CWRU Cutter robot:

1. Low-Cost Sensor Evaluation: Perform localization tests using low cost sensors. What level of sensor precision is necessary for standard EKF and AMCL localization? What level of precision is necessary for augmented wheel-velocity error states?

2. Systematic Odometry Calibration: Wheel-velocity error states are able to compensate for systematic odometry errors, but the proposed AKF is not sufficient to calibrate odometry parameters on its own. How can a robot self-calibrate odometry parameters (ticks-per-meter, track width) while also estimating non-systematic wheel velocity errors? Two ideas include adding additional states for each odometry parameter, or possibly creating a control loop which observes the ratio between estimated wheel velocity and estimated wheel velocity error.

3. Path Planning: Even though the "true" robot velocity and wheel slip can be estimated, the AKF has not currently been used for path control. Once a wheel slip is detected, how does a robot use this information to re-plan a path? Similarly, should the robot Path Driver implement a combined control law including both "true" robot velocity and the wheel slip velocity? This will be especially useful for the CWRU Cutter autonomous snowplow functionality in future years.

Bibliography

[1] iRobot Corporation, "iRobot roomba vacuum." http://www.irobot.com/en/us/robots/home/roomba.aspx, Apr. 2013.

[2] Husqvarna, "Robotic lawnmowers for homeowners." http://www.husqvarna.com/us/products/robotic-mowers, Apr. 2013.

[3] Friendly Robotics, "Robomow." http://www.robomow.com/en-USA/, Apr. 2013.

[4] Kyodo America, "Lawnbott robotic mower." http://lawnbott.com, Apr. 2013.

[5] E. Kreinar, H. Snow, N. Badger, D. Siedlak, D. Chrzanowski, J. Hall, C. Hart, and R. Quinn, "CWRU Cutter Cinco: Case Western Reserve University's Autonomous Robotic Lawn Mower." http://robomow.ion.org/wp-content/uploads/2012/09/2012_Report_Case.pdf, Apr. 2013.

[6] J. A. Beno, "CWRU Cutter: Design and Control of an Automous Lawn Mowing Robot," Master's thesis, Case Western Reserve University, 2010.

[7] A. Schepelmann, H. Snow, B. Hughes, F. Merat, R. Quinn, and J. Green, "Vision-based obstacle detection and avoidance for the cwru cutter autonomous lawnmower," in *Technologies for Practical Robot Applications, 2009. TePRA 2009. IEEE International Conference on*, pp. 218–223, 2009.

[8] D. A. Bennett, "Angle Based Localization of an Autonomous Lawnmower via

Radio Frequency Beacons and a Directional Antenna," Master's thesis, Case Western Reserve University, 2010.

[9] B. E. Hughes, "A Navigation Subsystem for an Autonomous Robot Lawn Mower," Master's thesis, Case Western Reserve University, 2011.

[10] R. Kalman, "On the general theory of control systems," *Automatic Control, IRE Transactions on*, vol. 4, no. 3, pp. 110–110, 1959.

[11] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Transactions of the ASME. Series D, Journal of Basic Engineering*, vol. 83, pp. 95–107, 1961.

[12] A. Gelb, *Applied optimal estimation*. MIT Press, 1974.

[13] R. P. Wishner, J. A. Tabaczynski, and M. Athans, "A comparison of three non-linear filters," *Automatica*, vol. 5, no. 4, pp. 487–496, 1969.

[14] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 3068, pp. 182–193, July 1997.

[15] N. Carlson, "Federated filter for fault-tolerant integrated navigation systems," in *Position Location and Navigation Symposium Record. Navigation into the 21st Century. IEEE PLANS '88., IEEE*, pp. 110–119, 1988.

[16] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *Proc. of the National Conference on Artificial Intelligence*.

[17] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[18] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *International Journal of Robotics Research*, vol. 22, p. 2003, 2003.

[19] J. Borenstein and L. Feng, "Umbmark: A benchmark test for measuring odometry errors in mobile robots," 1995.

[20] T. Abbas, M. Arif, and W. Ahmed, "Measurement and correction of systematic odometry errors caused by kinematics imperfections in mobile robots," in *SICE-ICASE, 2006. International Joint Conference*, pp. 2073–2078, 2006.

[21] A. Martinelli, N. Tomatis, A. Tapus, and R. Siegwart, "Simultaneous localization and odometry calibration for mobile robot," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, pp. 1499–1504 vol.2, 2003.

[22] D. Caltabiano, G. Muscato, and F. Russo, "Localization and Self-calibration of a Robot for Volcano Exploration," in *International Conference on Robotics and Automation*, vol. 1, pp. 586–591, 2004.

[23] G. Antonelli, F. Caccavale, F. Grossi, and A. Marino, "Simultaneous calibration of odometry and camera for a differential drive mobile robot," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 5417–5422, 2010.

[24] J. Borenstein, "Internal correction of dead-reckoning errors with the smart encoder trailer," *International Conference on Intelligent Robots and Systems*, vol. 1, pp. 127–134, 1994.

[25] A. Rudolph, "Quantification and estimation of differential odometry errors in mobile robotics with redundant sensor information," *International Journal of Robotics Research*, vol. 22, no. 2, pp. 117–128, 2003.

[26] M. Zajac, "Mobile robot self-diagnosis with a bank of adaptive particle filters," in *Proceedings of the Second international conference on Adaptive and intelligent systems*, ICAIS'11, pp. 168–179, 2011.

[27] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, "Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation," *Robotics, IEEE Transactions on*, vol. 25, no. 5, pp. 1087–1097, 2009.

[28] D. M. Helmick, S. I. Roumeliotis, Y. Cheng, D. S. Clouse, M. Bajracharya, and L. H. Matthies, "Slip-compensated path following for planetary exploration rovers," *Advanced Robotics*, vol. 20, 2006.

[29] A. Martinelli, N. Tomatis, and R. Siegwart, "Simultaneous localization and odometry self calibration for mobile robot," *Auton. Robots*, vol. 22, no. 1, pp. 75–85, 2007.

[30] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Intelligent robotics and autonomous agents, The MIT Press, Aug. 2005.

[31] E. Kiriy and M. Buehler, "Three-state extended kalman filter for mobile robot localization," 2002.

[32] P. Bonnifait and G. Garcia, "A multisensor localization algorithm for mobile robots and its real-time experimental validation," in *Int. Conf. on Robotics and Automation, 1996*, vol. 2, pp. 1395–1400 vol.2, 1996.

[33] C. Wang, "Location estimation and uncertainty analysis for mobile robots," in *Int. Conf. on Robotics and Automation, 1988*, pp. 1231–1235, 1988.

[34] Y. Song and J. Grizzle, "The extended kalman filter as a local asymptotic observer for nonlinear discrete-time systems," in *American Control Conference, 1992*, pp. 3365–3369, 1992.