Classification of Ego Platform Motion for Platform Independent Plug and Play Navigation

by

Jonathan Ryan

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 7, 2016

Keywords: Navigation, Sensor Fusion, Multiple Model Particle Filtering

Approved by

David Bevly, Chair, Professor of Mechanical Engineering
David Cicci, Professor of Aerospace Engineering
George Flowers, Professor of Mechanical Engineering and Dean of the Graduate School
Subhash Sinha, Professor of Mechanical Engineering

Abstract

This dissertation presents a method of using these kinematic constraints without requiring this apriori knowledge of the application platform. The method is termed "joint navigation and classification" (JNC), and involves determining the platform type online and simultaneously using the platform type information to properly apply the kinematic constraints and improve accuracy. The JNC problem is solved with a form of multiple model particle filter which treats the platform type as a mode state upon which the navigation state vector is conditioned. The particle filter is marginalized using Rao-Blackwellization to improve computational efficiency. Additionally, this dissertation presents a method within the JNC particle filter of autonomously determining whether any constraints should be applied at all. Motivating this is a study which shows that applying constraints when the IMU is of high quality can actually hurt the solution. The final JNC particle filter is robust to this phenomenon. The algorithm is validated using data collected on three different platforms (ground vehicle, pedestrian, and aircraft) and with IMUs of varying quality. It is demonstrated that the JNC particle filter can autonomously determine the correct platform type and use that knowledge to improve the navigation solution. It is further demonstrated that the JNC filter can autonomously detect situations where it is advantageous *not* to apply the constraints, thereby avoiding the pitfall described above. The JNC particle filter offers a best of both worlds solution which is both flexible and optimized to the platform, in addition to being robust to situations with high quality IMUs. Many navigation systems take advantage of knowledge of the host platform type, such as a ground vehicle, to apply kinematic constraints to the system to improve the navigation solution. It has been well documented that such constraints can help reduce inertial drift, whether in concert with other aiding sensors or as the only aids to an otherwise unaided inertial system. However, using these

constraints implies an apriori knowledge of the host platform type during the design phase. This is commonly done and is acceptable for stovepiped solutions for which flexibility is not a concern. However, this does not allow for flexibility in either the design phase or in the field.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-0809382. First, I would like to thank my advisor Dr. David Bevly for his guidance and leadership throughout my time in graduate school. I love the work I get to do every day and am very excited about the career path which I am on, all of which was made possible by my graduate school experience. I have Dr. Bevly to thank as much as anyone for this. I also wish to acknowledge my committee: Dr. Cicci, Dean Flowers, and Dr. Sinha for their oversight on this dissertation and for the very positive learning experiences I enjoyed in their respective courses.

I would now like to acknowledge my parents, who have been encouraging and supportive throughout my entire life, and no less in graduate school. There is no doubt that I would not be where I am today without them. I thank them for believing in me and encouraging me along this journey. Likewise I want to thank my grandparents for their constant love and support, for always being there, and for always encouraging me to do my best, something which they have each modeled with great integrity.

I would like to thank the AFRL ASPN team at the Sensors Directorate, Wright-Patterson AFB, and the Consortium of Ohio Universities on Navigation and Timekeeping (COUNT) who performed the data collectionfor the data used in Chapter 5. Additionally I would like to acknowledge the Huntsville Leidos team including Kevin Betts, Mike Turbe, Jon Wetherbee, Troy Mitchell, Doug Stranghoener, and Patrick O'Leary, whose work I have built upon and whom I have learned from since joining the Leidos team myself.

Finally I wish to thank my wife Ginger Ryan with deepest gratitude for her unwavering love and support during this time. She has always completely supported me and believed in me throughout my studies. This work would not be what it is without her, for I would not

be who I am without her. For all of these people and this opportunity I give thanks to the Lord Jesus Christ.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Problem Definition: Joint Navigation and Classification

Consider a sensor fusion algorithm which incorporates measurements from various sensors to aid an inertial measurement unit (IMU) and provide an optimal navigation solution in real time. This navigation system consists of the processing unit, which hosts the sensor fusion algorithm, and the sensor suite itself. The system is physically carried on some host platform, be it a ground vehicle, aircraft, water vehicle, or even a person. The purpose of the navigation system is to provide real time estimates of the host platform position in the world. Given this, the goal of this work is to be able to autonomously identify the platform type and use this knowledge to improve the navigation solution, without assuming any apriori knowledge of what type of platform the navigation system is affixed to. This goal was largely inspired by an ongoing navigation effort in the Defense Advanced Research Projects Agency (DARPA), which seeks to develop an extremely flexible and robust navigation system. The following is a quote from the BAA for this effort, known as the All Source Positioning and Navigation (ASPN) program [1]: *The All Source Positioning and Navigation (ASPN) program seeks to enable low cost, robust, and seamless navigation solutions for military users on any operational platform and in any environment, with or without GPS. In particular, ASPN will develop the architectures, abstraction methods, and navigation filtering algorithms needed for rapid integration and reconfiguration of any combination of sensors. This will enable rapid adaptation to evolving missions as well as reduction of the system integration costs and time-to-market for navigation solutions in general.*

Autonomous platform type determination is not a stated goal of the ASPN program. However, this capability would help to achieve the following stated goals:

- Navigation accuracy
- Rapid system integration and reconfiguration
- Reduction of time-to-market and development costs

First, the benefits of knowing system platform type to the overall navigational accuracy are well documented as discussed in Section 1.2. Stated briefly, inertial measurements are the backbone of modern navigation systems, and they have errors which drift over time. Knowing the platform type allows for applying platform specific constraints on the navigation system which can help to bound this error drift. Second, autonomous platform identification can improve rapid system integration and reconfiguration. Current state of the art assumes that if kinematic constraints are going to be used in the navigation solution, then this platform type (which dictates the appropriate constraints) is known at the outset of the design. It is almost always the case that the designer has a target application in mind when developing the algorithm. Yet this means that specific algorithms must be configured for specific applications, and they are not flexible for use on other platforms. It would be ideal, and very much in the spirit of the ASPN program, if the navigation system could be removed in the field from one platform and attached to another and still have the benefits of constrained aiding. This flexibility would also speed general system integration in the field. Finally, autonomous platform identification capability enable a single, flexible navigation system to be designed and used for a myriad of applications. This reduces both development cost and development time. There will always be applications which warrant the extra time and cost of a stove-piped navigation system (i.e. a $100 million dollar fighter jet), but a flexible system would be very beneficial to the far more numerous applications which require navigation systems. Furthermore, this adaptable technology could be a strong aid to the growing commercial robotics and commercial navigation markets.

1.2 Kinematically Constrained Navigation

It is well known that unaided integration of inertial sensor signals will produce position, velocity, and attitude (PVA) solutions which degrade over time due to the errors in the inertial sensors. This problem is generally mitigated with aiding from some other external sensor such as GPS, magnetometers, or wheel odometers, etc. [23]. However, it is also common in the case of ground vehicle applications to make assumptions regarding the vehicle dynamics in order to help bound the navigation error growth. In particular, many authors take advantage of the fact that ground vehicles are normally constrained to the road. This leads to the idea that the lateral and vertical velocities along the surface of the road are zero. The authors of [11] apply these constraints in a Kalman filter structure as "virtual measurements" of the lateral and vertical velocities. This is done both for a standalone INS solution and a loosely coupled GPS/INS/wheel speed sensor solution, and the improvements obtained via the constraints are studied. Likewise, the authors of [20] apply the constraints as Kalman filter virtual measurements to a blended GPS/INS solution to improve dead reckoning navigation during GPS outages. Finally, the authors of [17] apply these same constraints to a tightly coupled GPS/INS/wheel speed sensor integration. Klein et al. proposes a novel method of applying vehicle motion constraints in [27]. Instead of only introducing the constraints using the measurement model, the authors also incorporate the constraints into the system model. For example, if a zero lateral velocity constraint is imposed, the authors replace the lateral accelerometer reading with zero in addition to applying a measurement of zero lateral velocity.

All of the above demonstrate the effectiveness of such constraints to improve inertial navigation performance during GPS outages or in a more general dead reckoning scenario. However, these works do not include any in depth analysis of what happens to the navigation solution when the constraints are violated. Travis et al. [39] investigated constraint violation errors in a multi-sensor blended solution. In that work, the lateral velocity constraint was applied at all times and the resulting position errors in the blended solution during dynamic

maneuvers were studied. The authors also investigated the effects of constraint violations on the yaw rate gyro bias estimation. They found that constraint violations introduce error into the blended solution estimate of the yaw gyro bias, which harshly degrades dead reckoning performance during GPS outages.

Several authors have applied forms of kinematic constraints to aid aircraft navigation. Euston et al. [16] propose to obtain steady state roll and pitch estimates using the accelerometers, and using these steady values to correct gyro drift. Thus the complementary low frequency accuracy of the accelerometers and the high frequency accuracy of the gyros can be fused together (hence the term "complementary filter"). The accelerometer based roll and pitch computation assumes that the aircraft is in steady level flight, so any time the aircraft turns this assumption will be violated. The authors mitigate deviations from that assumptions by using wind speed measurements to estimate the aircraft velocity in order to compute the centripetal acceleration. After removing the centripetal acceleration, the remaining signal in the accelerometers is assumed to be gravity only. Note that this also assumes zero sideslip in the aircraft motion, and also a knowledge of the angle of attack. Yoo et al. propose a complementary filter very much in the same spirit as Euston et al., except without requiring airspeed measurements and extra knowledge of the angle of attack [41]. Like Euston et al., Yoo et al., also observe that using the accelerometers when the aircraft is not in level flight will not work due to the additional non-gravity acceleration effects present in the signal. Therefore they propose, instead of trying to estimate these terms, to merely adjust the gains in the complementary filter to account for errors in the accelerometer based roll and pitch during dynamic maneuvers. They use the magnitude of the measured 3D acceleration signal to discern the dynamic maneuvers.

Kinematic navigation constraints are often applied in pedestrian navigation applications in the form of step counting to obtain user speed and/or distance along the forward axis. For example, Zhao [42] uses an empirical model of acceleration when a user is walking and running to design a step counter using a 3-axis accelerometer. This step counter produces

estimates of the user speed and distance which can be used in a larger pedestrian dead reckoning (PDR) algorithm. Terminology here is adopted from [22], where PDR refers to a fusion of step counting for speed/distance and gyros for heading. Groves et al. present an analysis of PDR vs. strap-down pedestrian navigation (SPN), in addition to a thorough review of pedestrian inertial navigation literature [22]. SPN consists of applying the standard strap-down navigation equations to a 6-DOF IMU to obtain the full navigation solution for the pedestrian, while also using the IMU to detect zero velocity conditions for additional aiding. Note that even with SPN, the ZVU detection relies on an apriori model of walking motion. Broadly speaking, PDR and SPN constitute the majority of the state of the art for pedestrian inertial navigation. Both PDR and SPN can be implemented in a shoe mounted system or a traditionally (torso) mounted system. Shoe mounted systems have the advantage of easy ZVU detection, but are more cumbersome to actually use. Torso mounted systems are easier to package and wear.

1.3 Ego Motion Classification

The basic problem addressed by this dissertation is using kinematic platform constraints to improve navigation systems when the platform type is not known apriori. Section 1.2 presented various examples of systems which use such constraints to improve navigation, given that the platform type is known. In this section, a summary of research related to the problem of determining the platform type is presented. To the best of the author's knowledge, there is no published work which seeks to solve the ego platform identification problem. That is, knowledge of the platform type which the navigation system is fixed to is taken for granted. This is understandable. While there are definite advantages to being able to adaptively identify the ego platform (discussed in Section 1.1), it is almost always the case that navigation systems are designed for specific applications. Sometimes there is flexibility built into a given navigation system (e.g. "high dynamic" and "low dynamic" settings for GPS receivers), but these settings are configured by the user ahead of time for

the given application [25]. However, there is a great deal of research published in the area of joint tracking and classification (JTC) for radar systems (and other modalities).

First consider the problem of tracking an aircraft using radar measurements, assuming the the target class is known apriori or is irrelevant. In contrast to the navigation problem there is no input information about the vehicle of interest, so there is no way to predict the next movement of the target. For this reason it is very common in tracking problems to frame the system as a multiple model problem. Each of the models is a hypothesis of the maneuver state of the target at the current time, and the mode can change based on some assumed mode transition probability [29]. Now consider the problem of joint tracking and classification. The goal of JTC is to simultaneously track and identify a target, where in particular the correlation between the type (aka class, mode) of the target and its dynamic behavior is considered [8]. For example, consider the problem of determining an aircraft type (e.g. fighter jet vs. cargo plane) using nothing but the *kinematic* information from radar measurements. JTC algorithms, in general, present various forms of Bayesian frameworks to recursively update the likelihood of candidate target class types given this kinematic information. They optimally fuse the kinematic radar information with apriori knowledge of the flight capabilities (e.g. acceleration and velocity envelopes) of each potential hypothesis. Over time, the JTC algorithm refines the likelihood of each hypothesis until the target is definitively classified. Furthermore, the assumed constraints of the target class implicitly improve the tracking performance. This is very similar to the goal of this work. In essence, the only difference is that it is the *ego platform* as opposed to some external target which must be identified. The measurements of the ego platform motion help to identify what the platform type is, and the knowledge of the platform type helps to constrain the error growth of the navigation solution. Finally, keep in mind the general structure of the JTC problem. First, there are multiple possible modes at the tracking level which correspond to potential maneuver states. These modes can change over time. Then, at the higher classification level, there are multiple possible modes for the type of vehicle being tracked. These modes do not

change over time, as the vehicle type doesn't change over time. So then, there are multiple static hypotheses which each consist of multiple dynamic hypotheses. In the navigation problem the vehicle state can be predicted, thereby eliminating the need for multiple models pertaining to the current vehicle maneuver state.

In [8], the authors propose to solve the JTC problem by using a generalized pseudo Bayes (GPB) filter to do the tracking. Each of the filters within the GPB are conditioned upon a hypothesis class of the tracked target. Classification is done by fusing direct measurements of the target class from electronic support measure (ESM) sensors with kinematic measurements from radar in a Bayesian framework. Likelihood functions of the target class given ESM measurements are derived from ESM sensor models. Class likelihood information obtained from radar kinematic measurements is done in the manner adopted from [10] (where a GPB is used). Finally, the authors feed class information back into the GPB to improve the tracking performance. They also compare this with a similar approach, where an interacting multiple model (IMM) filter is used instead of the GPB filter for the tracking part, and no classifier information is fed back into the tracker. The authors found that feedback of the classifier into the tracker improved overall classification performance.

Ristic et al. presents an Interacting Multiple Model (IMM) approach to JTC which assumes linear and Gaussian restrictions [33]. This is done by only using acceleration constraints in each of the class conditioned filters. The class conditioned filters are implemented as IMM filters themselves, due to the nature of tracking with the potential of unknown target maneuvers, and the overall aggregate filter which blends the class conditioned filters is itself an IMM filter. Therefore the overall algorithm can be considered as a meta-IMM filter. An approach similar to the IMM, but more general in its application potential is presented by McGinnity and Irwin [29]. Note that *tracking only* is considered in [29], not JTC. In this case, the modes upon which the dynamic model is conditioned are the aircraft maneuver hypotheses. In their work, the multiple model filter maintains the same number of mode conditioned filters as there are hypotheses, which is similar to the IMM. Additionally their

algorithm applies a density mixing technique at each time step, as in the IMM. However, unlike the IMM their algorithm uses particle filters for each of the mode conditioned filters, so it can handle nonlinear and non Gaussian mode conditioned models. The authors find that it performs better than the IMM in a simulated tracking problem, at the cost of increased computational complexity.

Angelova and Mihaylova [3] present a particle based approach to JTC, and compare this with a mixture Kalman filter (MKF) algorithm. The particle approach augments the target kinematic state with a mode state signifying the target class hypothesis, resulting in what is often referred to as a "hybrid state" model. The kinematic portions of the hybrid state are conditioned on the mode state, meaning that the target dynamics are a function of the target aircraft type. The likelihood of the entire state vector, given a history of radar measurements, is used to update the overall posterior. This means that by treating the mode state as part of the overall state vector, the class probability is estimated inherently via the radar measurements. No direct class measurements (e.g. ESM) are used. The MKF treats the mode state as a particle but marginalizes out the kinematic states. These are assumed to be linear and Gaussian and can therefore be estimated using Extended Kalman Filters (EKF), conditioned on the value of the corresponding mode particle. This approach treats the true posterior as a weighted sum of Gaussian distributions (hence *mixture* Kalman filter). The particle approach is very similar to the method in [21], which is discussed in detail in Section 4. The MKF approach is very similar to the method presented in [30], which is also discussed in detail in Section 3.6. The MKF is extended to cases where the mode conditioned models are not linear and Gaussian in [5] for the tracking only problem in the presence of significant glint. However, the overall distribution is still treated as a mixture of Gaussians as in the MKF. The difference is that the authors propose a method of optimally mixing the mode conditioned estimates in a way that accounts for the nonlinear/non-Gaussian characteristics of the mode conditioned models.

This summary of work in JTC field is presented because of the close similarities between the tracking/classification problem and the goal of this dissertation. However there are some important differences which must be clarified. First, the platform type (which is the "mode state") for the problem considered here is static. Whereas in the tracking problem, the mode state corresponds to the set of possible maneuvers of the tracked vehicle and thus changes over time. This difference means that the multiple model filters designed around optimal pruning of the mode history, such as the IMM, GPB, and MKF, are unnecessary. It is more important for the Joint Navigation and Classification (JNC) problem to optimally represent the state distributions and to ensure robustness in classification than it is to maintain an accurate *history* of the mode state. These algorithms all make assumptions about the posterior distribution in order to maintain a good mode state history. The IMM assumes that the posterior distribution is Gaussian, while the GPB and MKF assume that the posterior is at least a mixture of Gaussians. There are many applications for which this may be acceptable, even in certain cases of JNC, but the assumptions are made in order to obtain a result which is unnecessary for JNC. Therefore, these types of algorithms are not used. In Section 3.6, more background is given on particle filter specific multiple model methods which are more germane to JNC. This is done so that they can be considered in more detail after particle filtering foundations are discussed.

1.4 Contributions and Outline

In light of the preceding problem motivation and related works, the following contributions are presented in this dissertation:

- Development of an algorithm to aid ground vehicle navigation performance using kinematic constraints (results presented by the author in [34]).
- Analysis of navigation performance of the algorithm in simulation and with real data.

- Experimental data showing that naive application of kinematic constraints will actually degrade the navigation solution if the IMU is of high enough quality.
- Development of an algorithm which simultaneously classifies the vehicle platform online and improves the navigation output using kinematic constraints.
- Analysis of navigation and classification performance of the algorithm with real data from three different platforms.
- Development of an algorithm which can autonomously determine if kinematic constraints will be beneficial or harmful to the given inertial system online.
- Analysis of navigation and classification performance of the algorithm with real data from three different platforms, using various quality inertial measurement units.

The first contribution of this work is an algorithm which applies kinematic constraints in a robust manner to aid an inertial navigation system for a ground vehicle. This algorithm uses the assumption that ground vehicles move along the forward axis of the body, thus the motion perpendicular to the forward axis in the horizontal plane is zero, as is the motion perpendicular to the road surface. This algorithm uses the yaw rate gyroscope to detect vehicle turning motion. If the vehicle is not turning, then applying the constraint is appropriate and helps bound the inertial system error growth. If the vehicle is turning, then the constraint is not applied because there are small violations of the zero lateral velocity assumption when turning. These violations have the effect of accumulating error in the filter over time, and should therefore be avoided. The performance of the algorithm is analyzed using simulated data and data collected from a sensors mounted on an all terrain vehicle. This work also includes an analysis of the relationship between the IMU quality and the errors introduced by improper assumptions. It is shown that there is a level of IMU quality which is high enough that the inertial drift is less than the error accumulated from violating the lateral motion constraints. This is an important finding, because it implies

that kinematic constraints should not be applied indiscriminately to all navigation systems using a known platform. Therefore some knowledge of the inertial measurement unit quality is required in order to make this distinction.

Another contribution of this work is a novel approach to increase navigation system accuracy while maintaining flexibility regarding the deployed application. An algorithm is designed which is able to autonomously determine online the type of vehicle platform to which it is affixed, and then simultaneously use this knowledge to improve the navigation performance of the system. This means that this new navigation system can be strapped to a particular vehicle or person in the field and provide a navigation solution which is optimized to the platform type, without requiring any apriori knowledge of the host platform. The algorithm is a variant of a multiple model particle filter, which has the majority of the state vector marginalized via Rao-Blackwellization in order to optimize computational tractability. Each platform hypothesis has a unique reference trajectory subject to its specific model in addition to an independent set of particles which model the error in the reference trajectory. The probability of each mode hypothesis is updated at each time step via the particular dynamic propagation model and the likelihood functions of received measurements. Over time, the measurements rule out the unlikely platform modes and leave the true vehicle platform mode set alive. The constraints inherent to the identified model serve to improve the overall trajectory of the navigation solution. The performance of this algorithm is evaluated using data sets from ground vehicle, pedestrian, and aircraft platforms. The algorithm is able to properly classify the platform type online, with no apriori knowledge regarding the platform. The navigation performance of the algorithm is evaluated by comparing the results with ground truth and with results from an unconstrained system. It is shown that in all three cases the proper platform is identified and the corresponding constraints actually improve the navigation solution when compared with the unconstrained solution. The unconstrained solution has no other differences besides the constraints; all other things are equal for the comparison.

These results demonstrate that the algorithm meets the desired goals of this dissertation. However, these data sets use IMU's that are not of very high quality. Therefore it is possible, as previously stated, that imposing these constraints when a high quality IMU is used can degrade the solution. The algorithm must be able to handle this in a a way which doesn't compromise the flexibility benefit that the online classification provides. The final contribution of this dissertation solves precisely this problem. A variant of the algorithm is presented which also determines online whether any constraints should be applied at all, regardless of the platform type. This algorithm autonomously decides between each of the hypotheses representing a constrained platform type and a "null" hypothesis representing an unconstrained model. The null hypothesis is agnostic of the platform type. It implies that no matter what the platform type is, the IMU is of high enough quality that it is more accurate than any constraints which would be applied. This modified version of the algorithm is tested on data sets from pedestrian, ground vehicle, and aircraft platform types with tactical grade IMUs. It is shown that the algorithm properly decides on the unconstrained model and applies no constraints. Additionally, an example case with a ground vehicle and a tactical grade IMU is studied using the original filter which doesn't include the null hypothesis. It is shown that in this case the constraints to degrade the navigation performance, as predicted.

This work is outlined as follows. The remainder of Chapter 1 provides important background on inertial navigation and Kalman filtering. Inertial measurement units are the primary component in modern navigation systems, and as such their fundamentals are discussed in detail. Following this is a discussion of Kalman filtering. Kalman filters are used in Chapter 2, and elements of Kalman filtering permeate the Joint Navigation and Classification (JNC) algorithm which is ultimately developed in Chapter 4. Chapter 2 presents a study of inertial constraints applied to ground vehicle navigation. This is a case study which motivates the need for inertial constraints and proves their benefits using experimental data collected on an all terrain vehicle (ATV). Furthermore Chapter 2 motivates the need for distinguishing between high and low quality IMUs when using kinematic constraints. Chapter

3 introduces particle filtering fundamentals and provides a detailed discussion of more advanced multiple model techniques which are foundational to the online classification portion of the JNC filter. Important considerations for implementing multiple model particle filters in real applications are discussed. Chapter 4 builds on Chapter 3 and develops the Joint Navigation and Classification filter. The indirect filter framework which is used is described, in addition to the Rao-Blackwellization method which is adopted. The constrained platform models are presented, as are the measurement models for all sensors used. Chapter 5 analyzes the JNC filter performance using real data from three different platforms and several different IMUs. Finally, Chapter 6 presents the conclusions of this dissertation and discusses potential future work.

1.5 Inertial Navigation

Inertial sensors are the backbone of all so-called kinematic, or sensor fusion based, navigation systems. As the inertial sensors provide measurements of the state derivatives, they are used to propagate the system forward in time. Therefore attention is given here to the dynamic equations of inertial navigation. First the dynamic equations are discussed. This is followed by a discussion of the first order approximation of the dynamics, which becomes important for uncertainty propagation in many cases [9]. In this work the attitude state is maintained as a quaternion q, which represents the orientation of the body frame with respect to the North-East-Down (NED) frame. The attitude is propagated according

to Equations (1.1 - 1.3)

$$\dot{q} = \frac{1}{2} q \omega_{BN}^{B} \tag{1.1}$$

$$\omega_{BN}^{B} = \omega_{BI}^{B} - R_{N}^{B} \omega_{NI}^{N} \tag{1.2}$$

$$\omega_{NI}^{N} = \omega_{e} \begin{bmatrix} \cos(\lambda) \\ 0 \\ -\sin(\lambda) \end{bmatrix} + \begin{bmatrix} \frac{v_E}{R_\Phi + h} \\ -\frac{v_N}{R_\lambda + h} \\ -\frac{v_E \tan(\lambda)}{R_\Phi + h} \end{bmatrix} \tag{1.3}$$

where ω_{BN}^{B} signifies the orientation of the body frame with respect to the NED frame, expressed in the body frame. The term ω_{BI}^{B} is the rotation rate of the body frame with respect to the inertial frame. The term ω_{NI}^{N} is known as the transport rate. It accounts for the rotation of the local NED reference frame due to the motion of the vehicle across the globe. It is resolved into the body frame via R_{N}^{B}, which is the directio cosine matrix (DCM) of the vehicle attitude. The transport rate is a function of the earth rotation rate ω_e, the vehicle velocity $[v_N, v_E]$, and the vehicle geodetic position $[\lambda, h]$.

The velocity propagation equations are given in Equations (1.4-1.7).

$$\dot{v}_N = -\left[\frac{v_E}{(R_\Phi + h)\cos(\lambda)} + 2\omega_e\right] v_E \sin(\lambda) + \frac{v_N v_D}{R_\lambda + h} + a_N \tag{1.4}$$

$$\dot{v}_E = \left[\frac{v_E}{(R_\Phi + h)\cos(\lambda)} + 2\omega_e\right] v_N \sin(\lambda) + \frac{v_E v_D}{R_\Phi + h} + 2\omega_e v_D \cos(\lambda) + a_E \tag{1.5}$$

$$\dot{v}_D = -\frac{v_E^2}{R_\Phi + h} - \frac{v_N^2}{R_\lambda + h} - 2\omega_e v_E \cos(\lambda) + g + a_D \tag{1.6}$$

$$\begin{aligned} g = {} & 9.780327(1 + 5.3024 \times 10^{-3} \sin^2(\lambda) - 5.8 \times 10^{-6} \sin^2(2\lambda)) \cdots \\ & - (3.0877 \times 10^{-7} - 4.4 \times 10^{-9} \sin^2(\lambda)) h \cdots \\ & + 7.2 \times 10^{-14} h^2 \ {}^{m}/_{s^2} \end{aligned} \tag{1.7}$$

The linear acceleration of the vehicle in the NED frame is represented by the $a_{[N,E,D]}$ terms. The first terms in Equations (1.4) and (1.5) are Coriolis terms resulting from the motion of

the NED frame, as are the third terms in Equations (1.5) and (1.6). The second terms in Equations (1.4) and (1.5) are centripetal terms, as are the first two terms in Equation (1.6). The down velocity derivative also contains the gravitational component, which is computed via Equation (1.7).

The position states are maintained in geodetic terms $[\lambda, \Phi, h]$, and as such the rates must be transformed from rates of NED position to geodetic rates. This is done via scaling, which is a function of position, in Equations (1.8) and (1.9). The altitude rate is merely the negative of the down velocity (1.10).

$$\dot{\lambda} = \frac{v_N}{R_\lambda + h} \tag{1.8}$$

$$\dot{\Phi} = \frac{v_E}{(R_\Phi + h)\cos(\lambda)} \tag{1.9}$$

$$\dot{h} = -v_D \tag{1.10}$$

The inertial sensor models each contain error contributions from a scale factor $\kappa_{\{g,a\}}$, a constant "turn-on" bias $b_{\{gs,as\}}$, a time-varying bias $b_{\{g,a\}}$, and Gaussian white noise $\eta_{\{gv,av\}}$. The gyroscope and accelerometer models are shown, respectively, in Equations (1.11) and (1.12). The dynamic bias for each are modeled as first order Markov-processes having a specific time constant $\tau_{\{g,a\}}$ and driven by Gaussian white noise $\eta_{\{gu,au\}}$ (1.13 - 1.14).

$$\tilde{\omega}_{BI}^B = (I_{3\times3} + \kappa_g)\,\omega_{BI}^B + b_g + b_{gs} + \eta_{gv} \tag{1.11}$$

$$\tilde{a}^B = (I_{3\times3} + \kappa_a)\,a^B + b_a + b_{as} + \eta_{av} \tag{1.12}$$

$$\dot{b}_g = -\frac{1}{\tau_g} b_g + \eta_{gu} \tag{1.13}$$

$$\dot{b}_a = -\frac{1}{\tau_a} b_a + \eta_{au} \tag{1.14}$$

$$a^N = R_B^N a^B \tag{1.15}$$

The acceleration in the body frame is transformed into the NED frame given Equation (1.15).

It will be necessary when processing Kalman filtering equations to apply a first order linearization to the error dynamics (δ) of the above navigation equations. The error dynamics ($\delta\dot{x}$) are derived directly from their parent equations as described in [9]. The first order linearization of the error dynamics is given by (1.16 - 1.17).

$$\Phi_k = e^{F_k \Delta t} \tag{1.16}$$

$$\delta\dot{x} = \Phi\delta x + G\underline{\eta} \tag{1.17}$$

The matrix Φ is the discrete approximation of the state transition matrix. The dynamics are linearized in continuous time to form the matrix F, and then discretized according to (1.16) to obtain Φ. The matrix G is the linearization of the process noise input matrix. F and G are given below:

$$F_k = \begin{bmatrix} F_{11} & F_{12} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ F_{21} & F_{22} & F_{23} & F_{24} & F_{25} & 0_{3\times3} & 0_{3\times3} & F_{28} & 0_{3\times3} \\ F_{31} & F_{32} & F_{33} & 0_{3\times3} & 0_{3\times3} & F_{36} & F_{37} & 0_{3\times3} & F_{39} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{44} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{66} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{1.18}$$

$$G = I_{27\times27} \tag{1.19}$$

The primary terms of the linearization are shown in Equations (1.20-1.29), with the remaining terms are defined in Appendix A.

$$F_{12} = \begin{bmatrix} \frac{1}{R_\lambda + h} & 0 & 0 \\ 0 & \frac{\sec(\lambda)}{R_\Phi + h} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{1.20}$$

$$F_{23} = -R_B^N \left[\hat{a}^B \times\right] \tag{1.21}$$

$$F_{24} = -R_B^N (I_{3\times 3} - \hat{\kappa}_a) \tag{1.22}$$

$$F_{25} = -R_B^N (a^B - \hat{b}_a) \tag{1.23}$$

$$F_{28} = -R_B^N (I_{3\times 3} - \hat{\kappa}_a) \tag{1.24}$$

$$F_{36} = -(I_{3\times 3} - \hat{\kappa}_g) \tag{1.25}$$

$$F_{37} = -(\omega_{BI}^B - \hat{B}_g) \tag{1.26}$$

$$F_{39} = -(I_{3\times 3} - \hat{\kappa}_g) \tag{1.27}$$

$$F_{44} = -\frac{1}{\tau_a} I_{3\times 3} \tag{1.28}$$

$$F_{66} = -\frac{1}{\tau_g} I_{3\times 3} \tag{1.29}$$

1.6 Implementation of the Inertial Navigation Equations

The inertial navigation equations described in Section 1.5 are implemented at the rate of the IMU signals by applying numerical integration. Initial conditions for the navigation states are required. Equations (1.1 - 1.10) are a function of the navigation states and the measured inertial signals. In (1.1 - 1.10) the current state estimate is used in place of the true value, which is unknown. The inertial signals are compensated for the estimated errors according to the error models (1.11 - 1.15), where $\hat{()}$ represents an estimated quantity. The IMU error estimates are resolved in the frame of the IMU, and thus the correction is done in this frame. Equations (1.30) and (1.31) show how the measured inertial signals are

compensated with the estimated sensor errors.

$$a^S_{(i)} = \frac{(\tilde{a}^S_{(i)} - (\hat{b}_{as,(i)} + \hat{b}_{a,(i)}))}{(1 + \hat{\kappa}_{a,(i)})} \tag{1.30}$$

$$\omega^S_{SI,(i)} = \frac{(\tilde{\omega}^S_{SI,(i)} - (\hat{b}_{gs,(i)} + \hat{b}_{g,(i)}))}{(1 + \hat{\kappa}_{g,(i)})} \tag{1.31}$$

Next, the DCM of the IMU with respect to the body frame is used to resolve the compensated IMU signals into the body frame via Equations (1.32) and (1.33),

$$a^B = R^B_S a^S \tag{1.32}$$

$$\omega^B_{BI} = R^B_S \omega^S_{SI} \tag{1.33}$$

where $i \in \{x, y, z\}$. Having evaluated all of the state derivative equations, these are now numerically integrated to obtain the state estimate at the next time step. In this work, Heun's second order Runge-Kutta method is used. The numerical integration process is given in Equations (1.34-1.36).

$$\tilde{x}_{k+1} = x_k + \Delta t f(t_k, x_k) \tag{1.34}$$

$$x_{k+1} = x_k + \frac{\Delta t}{2}\left[f(t_k, x_k) + f(t_{k+1}, \tilde{x}_{k+1})\right] \tag{1.35}$$

$$f(t_k, x_k) \triangleq \dot{x}_k \tag{1.36}$$

1.7 Kalman Filtering

The Kalman Filter is a widely popular filtering algorithm which implements the weighted linear least squares algorithm in a recursive manner [37]. Given a linear system subject to additive Gaussian noise in the process dynamics and the measurement model, the least squares estimator is the best linear unbiased estimator in terms of the estimation error

variance [37]. Such a system is described in Equations (1.37) - (1.40).

$$y_k = Hx_k + \eta \tag{1.37}$$

$$x_{k+1} = Ax_k + Bu_k + G\omega \tag{1.38}$$

$$\omega \sim N(\underline{0}, Q)_{n_x} \tag{1.39}$$

$$\eta \sim N(\underline{0}, R)_{n_y} \tag{1.40}$$

The vector y_k represents the full measurement vector at time k, and H encodes the linear relationship between the measurement and the state vector x_k. The state propagation is linear in the state vector and the process inputs u_k, where the matrix B describes how the input enters the system. The system dynamics may also contain additive Gaussian noise ω, which enters via G. Both ω and η are assumed to be zero mean with normal distributions. The Kalman filter maintains an estimate of the state ($\hat{x}$) and the estimation uncertainty (P) which is based on the system dynamic model and the statistical model of the stochastic elements. Given the measurement model, the estimation uncertainty at time k, and the measurement uncertainty R, the optimal update gain L is computed using Equation (1.41).

$$L_k = P_k H^T (H P_k H^T + R)^{-1} \tag{1.41}$$

The gain L is often referred to as the "Kalman gain," and it is optimal in the sense that P is minimized. Once the Kalman gain is computed, the state is updated using the gain and the difference between the measurement and the predicted measurement (also known as the measurement "innovation"). Note that if the assumptions of the Kalman filter are met, then the Kalman gain is the optimal amount with which to scale the measurement innovation in order to update the state estimate. The state update is given by Equation (1.42).

$$\hat{x}_k = \hat{x}_k + L_k(y_k - H\hat{x}_k) \tag{1.42}$$

Having adjusted the state estimate using information from the latest measurement, the uncertainty in the filter should also be adjusted. This is done in Equation (1.43).

$$P_k = (I - L_k H)P_k \tag{1.43}$$

The filter propagates the estimate and the uncertainty forward in time using Equations (1.44) and (1.45).

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k \tag{1.44}$$

$$P_{k+1} = AP_kA^T + GQG^T \tag{1.45}$$

Equation (1.44) is merely the model of the deterministic components of the system, while the second term in (1.45) accounts for the effects of stochastic components on the estimation uncertainty. The Kalman filter makes several important assumptions, which are listed below:

1. The system dynamics and measurement models are linear.
2. The system model (A, B, G, H, Q, R) is perfectly known.
3. The stochastic elements (ω, η) are normally distributed with zero mean and their distribution parameters (mean and variance) are known.
4. The errors (ω, η) are uncorrelated in time with themselves or each other.

Assumption 1 is rarely true in an absolute sense, and some linearization is often required. In this case the system model (A, B, G, H, Q, R) is linearized about the current best state estimate, where the model is a function of the state. Furthermore, Equations (1.42) and (1.44) are replaced with the full nonlinear functions respective to each, as given in Equations

(1.46 - 1.47).

$$\hat{x}_{k+1} = f(\hat{x}_k, u_k) \tag{1.46}$$

$$\hat{x}_k = \hat{x}_k + L_k(y_k - h(x_k)) \tag{1.47}$$

Applying these linearizations results in the well known Extended Kalman Filter (EKF). An implementation of the EKF is discussed in Section 2.2, where additional details are given. Assumption 2 is often impractical also. In order to deal with potential errors in the system model it is common to increase the variances (Q, R) according to the assumed model uncertainty. This mitigation technique is still subject to assumption 3, however. Filter designers seek to ensure assumption 3 via apriori knowledge of the stochastic terms. In sensor fusion applications, which are considered in this work, this amounts to experimental testing and modeling of the sensor errors. Thus where is it determined that the process noise or measurement noise terms are not zero mean, an additional state which represents this stochastic parameter can be added to the state vector. This may be modeled as a time correlated value or a static value. Note that assumption 3 doesn't mean that R must be diagonal, only that the measurement vector is uncorrelated with itself at any other time instant, and that it is uncorrelated with the process noise. If assumption 4 is mostly valid for a given application, then simply increasing the R is usually sufficient to overcome the deviations from this requirement. However if this is not true, then augmenting the state vector with a time correlated measurement error state or applying other techniques will be required.

Chapter 2

Kinematic Constraints for Improving Navigation

2.1 Introduction

This chapter presents a method of applying kinematic constraints to an unaided INS system which is used on a ground vehicle. In contrast to [39], a similar study, no external aiding sensors are used. Recall that the goal of this dissertation is to show an algorithm which can classify the platform type online and use this information to improve the navigation solution. Therefore it is first important to consider the impact of kinematic constraints and to prove that they actually benefit a real navigation system. Only then does it make sense to consider online platform classification. Two particular ground vehicle constraints are considered: zero lateral velocity and zero vertical velocity. These are applied in a Kalman filter structure as virtual measurements of the lateral and vertical velocities in the vehicle coordinate frame. Additionally, the lateral and vertical accelerometer biases are estimated using these constraints. The research presented in this chapter studies the error growth in position and velocity estimates which can occur if the assumed constraints are violated. This chapter also proposes a method of detecting lateral slip conditions without any outside aiding, making it possible to take advantage of the lateral velocity constraint when it is appropriate without introducing additional errors when the constraint is violated. Section 2.2 describes the navigation algorithm and the method of applying the constraints. Section 2.3 considers the effects of constraint violations on the navigation error and presents simulation studies which analyze this problem. Section 2.4 presents navigation results from processing experimental data collected at Auburn University. The results of this chapter have been published previously [34].

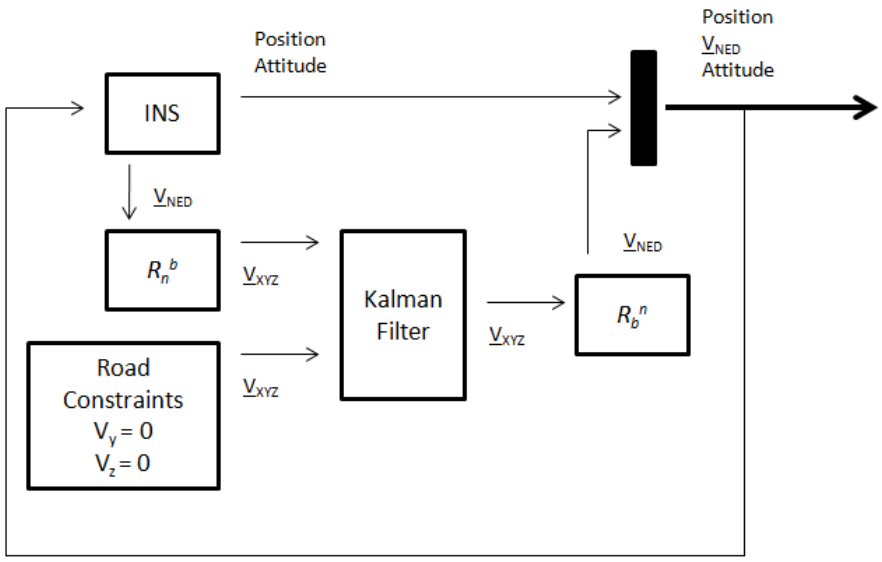


Figure 2.1: Diagram of Navigation Algorithm.

2.2 Navigation Algorithm

The core of the navigation algorithm is the standard INS mechanization routine, whereby the accelerometer and gyro signals are resolved into the navigation frame using the current attitude solution and integrated to obtain the new position, velocity, and attitude. Generally speaking, the IMU signals are also compensated for estimated bias effects, Coriolis effects from the Earth's rotation, local gravitational variations, etc. In this case, Coriolis effects and the gravity model are not included, because the target application is a ground vehicle. The operational speeds are low enough that the Coriolis effects can be neglected, and the altitude changes are small enough that gravity can be assumed constant. The equations used to calculate the INS position, velocity, and attitude (PVA) solution in the NED frame are described in [35].

The constraints on V_y and V_z (in the vehicle frame) are applied using a Kalman filter. This is shown in Figure 2.1, where the variable R_n^b represents the direction cosine matrix of the vehicle body frame with respect to the navigation frame. This DCM is a function of the current attitude solution. Note also that the variable R_b^n represents the inverse rotation. At each time step, the NED velocities from the navigation solution are rotated into the body

frame using the current attitude estimates in order to obtain current estimates of the body XYZ velocities as in Equation (2.1). The states of the Kalman filter are given by Equation (2.2).

$$\begin{bmatrix} \hat{V}_x \\ \hat{V}_y \\ \hat{V}_z \end{bmatrix} = R_n^b \begin{bmatrix} \hat{V}_n \\ \hat{V}_e \\ \hat{V}_d \end{bmatrix} \tag{2.1}$$

$$\hat{X} = \begin{bmatrix} \hat{V}_y & \hat{b}_y & \hat{V}_z & \hat{b}_z \end{bmatrix} \tag{2.2}$$

Again, note that $\dot{V}_y$ and $\dot{V}_z$ are not calculated and propagated explicitly. At each time step, V_y and V_z are computed from the navigation state via a rotation transformation. The bias estimates are assumed constant in between updates. At each time step, the Kalman gain is computed and the measurement update is applied, per Equations (1.41 - 1.43) presented previously. In this case, the measurements are virtual measurements, such that R is simply tuned to reflect the confidence in the validity of the constraints. I is a 2x2 identity matrix, and y_k is the measurement vector. The vector y_k is given by

$$Y = \begin{bmatrix} V_y \\ V_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.3}$$

If both constraints are applied, H is given by

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.4}$$

If the lateral velocity constraint is not applied and the vertical constraint is applied, H is given by

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.5}$$

Therefore switching between whether or not the lateral constraint is applied is simply a matter of changing H(1,1). The covariance is propagated in between measurements as a function of the state dynamics and the noise input matrix. The equations to do so are given by [37]

$$\dot{P}_k = AP_k + P_kA^T + B_wQB_w^T \tag{2.6}$$

$$P_k = P_{k-1} + \frac{1}{2}\Delta t(\dot{P}_k + \dot{P}_{k-1}) \tag{2.7}$$

Q is the input noise matrix, and is tuned according to the anticipated accelerometer values. B_w represents how the process noise enters the system. In this case, it is an identity matrix. Note that simple trapezoidal integration is used to propagate the covariance forward in time. The matrix A represents the state dynamics. It is given as follows.

$$A = \begin{bmatrix} 0 & a_{12} & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \tag{2.8}$$

In the first portion of this chapter, accelerometer biases are not estimated. Therefore the algorithm is set to ignore b_y and b_z. This is done by setting a_{12} and a_{34} to 0. In this case, it doesn't matter what a_{22} and a_{44} are set to. If it is desired to estimate b_y and b_z, as is done later in this chapter, then a_{12} and a_{34} are set equal to -1. The terms a_{22} and a_{44} are defined by the model of the accelerometer. In this chapter, the accelerometer signal is modeled per Equation (2.9),

$$\underline{a}_{meas} = \underline{a}_{true} + R_n^b\underline{g} + \delta\underline{f} + \underline{w} \tag{2.9}$$

where "a" corresponds to the acceleration, $\underline{g}$ is the three dimensional gravity vector in the NED frame, w is zero mean wide band noise, and $\delta\underline{f}$ is the sensor bias. The bias is modeled

as having a static component and a drifting component in Equation (2.10),

$$\delta f = b_{static} + b_{drift} \tag{2.10}$$

$$\dot{b}_{drift} = -\frac{1}{\tau} b_{drift} + \mu \tag{2.11}$$

where the drifting term is modeled as a first order Markov process, given by Equation (2.11). Here, τ is the assumed time constant for the accelerometer bias and μ is the wide band noise driving the process [19]. The term δf is the scalar form of the vector term $\delta \underline{f}$ in Equation (2.9). The characteristics of w and μ are defined by Equations (2.12) and (2.13).

$$w \sim N(o, \sigma_w^2) \tag{2.12}$$

$$\mu \sim N(0, \sigma_b^2) \tag{2.13}$$

The standard deviations σ_w and σ_b, and the time constant τ, are provided in Table 2.1.

In this study the roll and pitch states are excluded from the filter for two reasons. First, tactical grade gyroscopes have very stable biases, and these can be easily identified using static data. When this is done the primary error growth in the attitude solution will be due to integrating the wide-band noise, and the growth is very slow when using a tactical grade IMU. Second, it is hypothesized that it will be extremely difficult, if not impossible, to separate the gravitational effects of the attitude from the accelerometer biases using only the virtual measurements of the constraints. This stems from the well studied observability problem of INS alignment [32].

Three variations of this algorithm are studied. In the first case, the lateral velocity constraint is assumed to be valid at all times and is therefore applied at all time steps. This is referred to hereafter as the "hard constraint" case. In the second case, the lateral velocity constraint is never applied ("no constraint"). In the third case, the lateral velocity constraint is applied only if the yaw rate signal remains within a predefined threshold ζ_r for a predefined length of time T_r. The constraint is lifted immediately if the yaw rate magnitude exceeds

ζ_r. This is hereafter referred to as the "soft constraint" case. The yaw rate gyro is used as a condition for the lateral constraint because it is a reasonable means of determining whether or not the vehicle is driving straight under most operating conditions. Furthermore, if the vehicle is driving straight, then under most operating conditions there will be no lateral velocity or sideslip and the lateral constraint will be valid. This method of straight driving detection was successfully used to aid a loosely coupled GPS/INS sideslip estimator in [8]. The threshold is selected as being three times the standard deviation of the assumed wide band noise characteristics of the yaw rate gyroscope. In all three cases the vertical velocity constraint is applied at every time step.

2.3 Positioning Errors Due to Sideslip

Several experiments are conducted in this study using Carsim, a high fidelity commercial vehicle simulation software package. All simulations include periods of driving straight (resulting in a valid lateral velocity constraint), as well as periods of turning during which sideslip is present and the constraint condition is violated. The inertial outputs of the simulation are corrupted with wide-band noise and a drifting bias, all according to specifications matching a MEMS grade IMU. Additionally, the same experiments are repeated with the outputs corrupted with errors matching a tactical grade unit. The specifications of the errors for each grade are listed in Table 2.1. Note that the values for σ_w were obtained for

Table 2.1: IMU Categorical Specifications

Accelerometers	Tactical	MEMS
$\sigma_m(m/s^2)$	0.011189	0.005778
$\tau_b(s)$	60	100
$\sigma_b(m/s^2)$	30E-6	1.80E-03
Gyroscopes		
$\sigma_w(deg/s)$	0.20443	0.11608
$\tau_b(s)$	100	300
$\sigma_b(deg/hr)$	0.5	240

the MEMS case by examining two hours of static data recorded on a Crossbow IMU-440

at 20Hz. For the tactical case, this value was obtained by examining 1.5 hours of static raw IMU data recorded from a Novatel SPAN system at 100Hz. The SPAN system uses a Honeywell HG1700 (AG58) IMU. The time constant and driving noise variance values were taken from the ranges given in [10] for the corresponding sensor grade.

The accuracies of the positioning solution when using the three different lateral constraint approaches are compared, and the results from the MEMS IMU case are compared with the results from the tactical grade IMU case. This results in a total of six different scenarios, which are labeled scenarios 1-6 in Table 2.2. Figure 2.2 shows the trajectory of

Table 2.2: Ground Vehicle Test Scenarios

Scenario	Lateral Velocity Constraint	Vertical Velocity Constraint	IMU Grade
1	off	on	tactical
2	soft	on	tactical
3	hard	on	tactical
4	off	on	MEMS
5	soft	on	MEMS
6	hard	on	MEMS

the simulated experiment. In this simulation the vehicle drives straight on a flat road for a period of time, followed sequentially by a left, right, and another left turn. The run lasts for 100 seconds, and is ended by a sustained period of straight driving. The dynamics in this experiment were aggressive. The vehicle speed was 65 mph, and lateral accelerations peaked at 0.7 g's on one turn and 0.8 g's on another. These corresponded to 3 and 4 degrees of sideslip. Figure 2.3 also includes velocity estimates from scenarios 1-3. It can be seen by inspection that the soft constraint case provides the most accurate final position. This can be examined further by observing the estimates of the vehicle (or body) frame velocities for each case. Figure 2.3 shows the true vehicle frame velocities along with each corresponding estimate from scenarios 1-3. The X and Y velocity estimates for scenario 1 ("no constraint") exhibit a slow drift consistent with integrating wide-band noise, as expected. The Z velocity estimate, by contrast, does not exhibit this drift because of the vertical velocity constraint. Upon examining the results for scenario 2, it is observed that the estimates of the X and

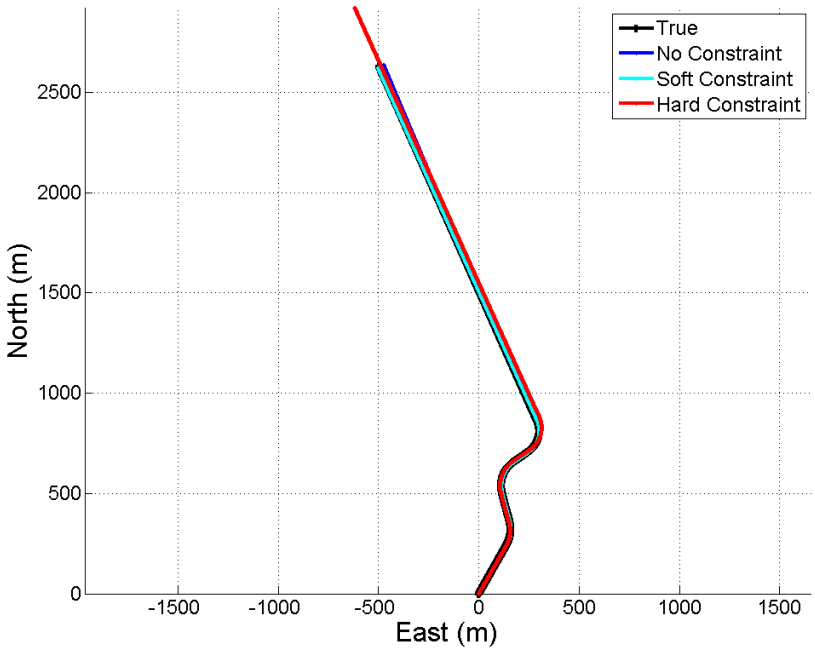


Figure 2.2: Trajectory of Simulated Experiment with Example Estimates.

Z velocities behave precisely as in the case of scenario 1, because the constraint conditions on these states are the same as in scenario 1. However, the Y velocity estimate is more accurate over time, because the no-sideslip constraint is applied when the vehicle is driving straight. This is in stark contrast to scenario 3, where the constraint is applied at every time step. The error in the lateral velocity estimate during turning maneuvers is clearly seen, as is the expected lack of drift during straight driving periods. Not only does the positioning in this case suffer from errors in the lateral velocity estimate, but the forward velocity estimate is degraded as well. This is because of the coupling between the velocities during the transformations between the navigation and vehicle coordinate frames. The error in the lateral velocity state will corrupt both the North and East velocity estimates, which are in turn used to obtain the X and Y velocities at the next time step. These results from Figure 2.3 are only from one simulated experiment, and are a function of the behavior of the underlying stochastic processes during that particular experiment. Therefore in order to

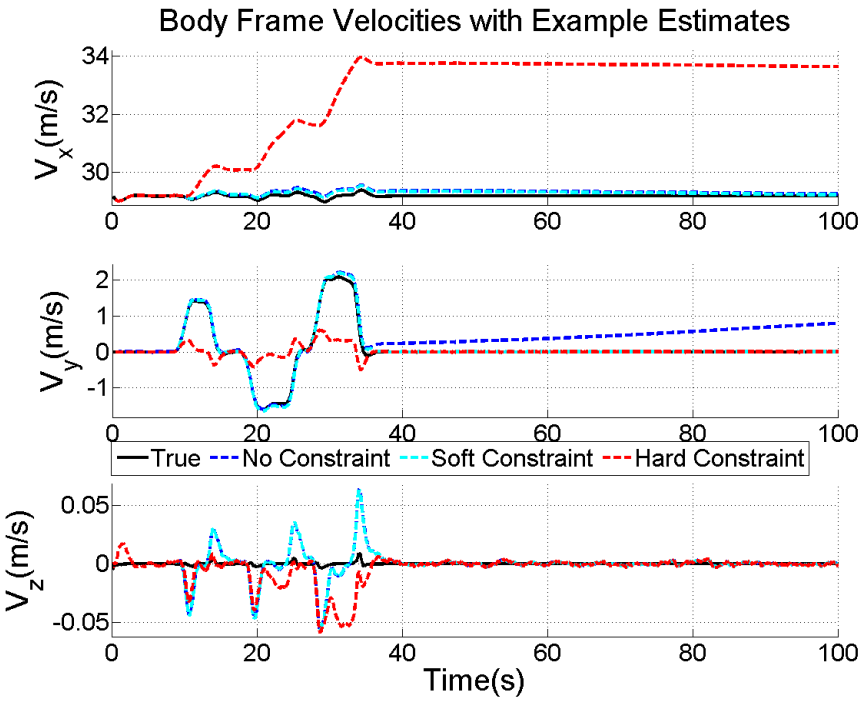
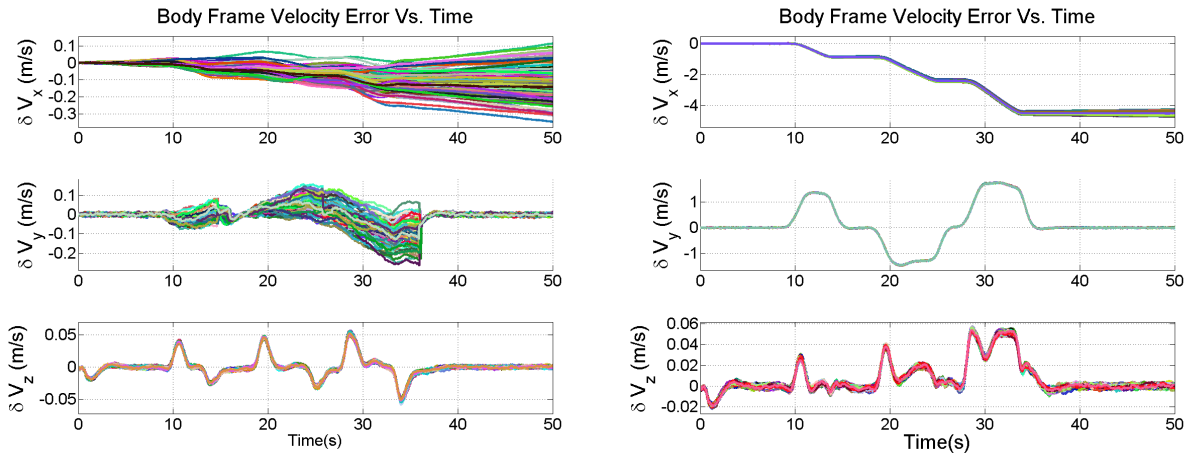


Figure 2.3: Velocity Estimates in the Vehicle Body Frame.

more accurately assess the navigation performance of the algorithms, scenarios 1-6 were simulated 100 times each in a Monte Carlo fashion. Figure 2.4a shows the body frame velocity errors for each of the 100 simulations of scenario 2 (soft lateral constraint). It is seen that the X velocity estimate exhibits the expected drift patterns, while the Y velocity estimate only drifts during the turns. Note also that the error returns to near zero when turning ceases, as the constraint is applied to correct the estimate. This is seen at t≈16s and t≈37s. This is an improvement over scenario 1, where the V_y estimate drifts throughout the entire run. Figure 2.4b shows the vehicle frame velocity estimation errors for scenario 3, and the results are dramatically different from scenarios 1 and 2. First, notice the large errors in the lateral velocity estimate. Indeed the errors are roughly equivalent to the true lateral velocity, because the filter constrains this estimate to zero. In cases such as this, where larger lateral velocities are present, these large errors will rapidly accumulate in the position solution. Note also that the error in the forward velocity estimate, which was considered in the discussion of Figure 2.3, is consistent. These errors will also contribute to the rapid



(a) Vehicle Frame Velocity Estimation Errors for 100 Monte Carlo Simulations, Scenario 2.

(b) Vehicle Frame Velocity Estimation Errors for 100 Monte Carlo Simulations, Scenario 3.

degradation of the positioning solution. Figure 2.5 shows the horizontal positioning error and the vertical position error over time, for each of the 100 experiments of scenario 1. Note that the horizontal error is calculated as the magnitude of the X and Y position errors, therefore it is always positive. The vertical error is not a magnitude; it is the true vertical error and may be negative. In addition, a histogram showing the statistics of the final position errors is displayed adjacent to the time plot. Recall that scenario 1 refers to the case where no lateral constraints are applied, and the vertical constraint is always applied. Figure 2.5 shows, then, that the horizontal position errors behave as expected, with the horizontal position estimates exhibiting random drift behaviors which arise from the wide-band noise present on the simulated IMU signals. It is also observed that the vertical error grows much slower by comparison, which is a product of the improvement from the vertical velocity constraint.

Table 2.3 gives the mean and standard deviations of the 100 final position errors for all 6 scenarios. The mean horizontal error for scenario 1 of ≈17m can be seen by inspection in Figure 2.5. The remaining entries of Table 2.3 shed light on the navigation performance as a function of both sensor quality and constraint violations. First, it is clear that the position performance is a function of the magnitude of the wide-band noise on IMU signals, which is

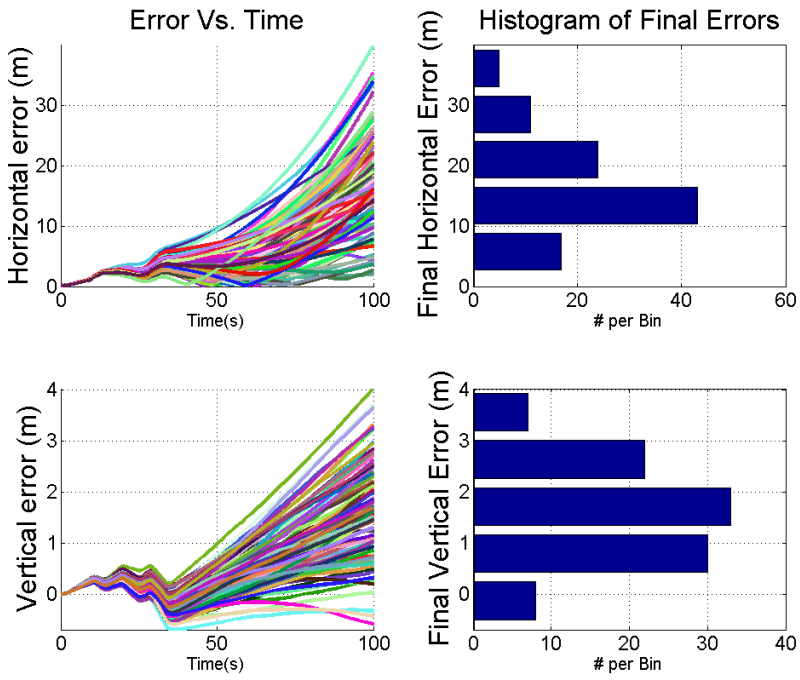


Figure 2.5: Position Estimation Errors for 100 Monte Carlo Simulations, Scenario 1.

well known. What is of interest here is whether or not the errors resulting from constraint violations surpass the inherent sensor errors. Scenarios 1-3 show that the navigation performance of a tactical grade IMU is superior without lateral velocity constraints vs. when the constraints are present and are violated. This result, of course, is a function of the magnitude of the lateral velocities which the vehicle experiences, and of the time duration of these velocities. However, it is concluded from scenarios 1 and 3 (tactical grade IMU) that when moderate to large lateral velocities are present, even over short periods of time, it is better

Table 2.3: Simulation Results: Navigation Error

	Horizontal Error (m)		Vertical Error (m)	
Scenario Number	mean	std	mean	std
1	16.72	7.95	1.64	0.96
2	12.29	6.56	1.64	0.96
3	318.17	11.25	4.03	1.09
4	682.41	350.50	-38.38	60.11
5	448.46	322.31	-19.82	57.92
6	502.91	340.70	-17.12	63.44

to use no lateral velocity constraint than to always assume zero sideslip. Fortunately, this is not the only option, as the results from scenario 2 demonstrate. Recall that in scenario 2 the algorithm utilizes a conditional or "soft" velocity constraint where the condition is based on the vehicle yaw rate. It would be expected that if the yaw rate condition accurately represents whether or not sideslip is present, then this method would be superior to both cases 1 and 3. This is because there is no drift during straight driving, as opposed to case 1, but neither is there additional error introduced by constraint violations, as opposed to case 3. Table 2.3 shows that indeed scenario 2 shows superior performance to both scenarios 1 and 3.

The results from scenarios 4-6 show the performance of the algorithm for the case of the simulated MEMS grade IMU. Scenario 4 corresponds to no lateral constraint, scenario 5 to the soft lateral constraint, and scenario 6 to the hard constraint. Table 2.3 shows that when using the MEMS grade IMU in simulation, the performance is better when using the hard constraint versus no constraint. This is in contrast to the tactical grade simulation results, where using the hard constraint radically increased the errors. In the tactical grade case, the errors introduced from constraint violations are much larger than the error growth from the sensor errors. In the MEMS grade case, the error growth due to sensor errors is much larger than the errors caused by constraint violations. These results are only applicable to this specific simulation, however it does provide the important insight that whether or not it is better to always apply the constraints vs. never applying them is a function of both the vehicle dynamics and the sensor quality. Finally, the results from both the MEMS grade simulation case and the tactical grade case show that the conditionally applied lateral velocity constraint provides the best navigation performance of the three constraint approaches.

2.4 Experimental Testing and Results

Several experiments were conducted in order to validate the simulation results using a Polaris electric drive ATV instrumented with a Novatel SPAN navigation system. The

SPAN system is a commercial GPS/INS integration which features a Novatel single antenna GPS receiver and a tactical grade Honeywell HG1700-AG58 IMU. The vehicle was driven along a gravel road in a field on the campus of Auburn University. Speeds were maintained near 20 mph throughout the run. The experiment included periods of straight driving along with aggressive turning maneuvers. Lateral acceleration peaks of ≈ 0.5g to ≈ 0.6g occurred during these maneuvers, and sideslip peaks of 7 degrees were generated in some cases and up to 12 degrees in others. The raw IMU signals from the HG1700 were logged at 100Hz, while the PVA blended solution from the SPAN was recorded at 2Hz. The PVA solution is used in this chapter as a reference solution for comparison, due to its very high accuracy, while the raw IMU signals are used as inputs for the navigation algorithm. The raw IMU signals are not compensated with any error estimates from the SPAN blended solution.

Figure 2.6 shows the horizontal and vertical position errors for three different variants of the navigation algorithm. The differences between the tests have to do with how the constraints are applied. It can be seen in Figure 2.6 that the best horizontal positioning performance is achieved using the soft lateral velocity constraint, a result which is in accord with the findings from the previous simulation results. Note also the differences between the horizontal positioning errors of the no constraint and the hard constraint cases. Over the duration of the run, the hard constraint results in approximately 150m more error than pure integration of the X and Y velocities. The reason for this was mentioned in the discussion of Figure 2.3. That is, the error in the Y velocity resulting from the improper constraint will propagate into both the North and East velocity estimates. These effects are also noticed in the clear dependence of the horizontal error of the hard constraint case on the dynamics, seen in Figure 2.6 during the time interval $110s < t < 160s$.

Figure 2.7 shows the X and Y velocity estimates when using each of the three lateral velocity constraint approaches. The most visible feature here is the ever increasing error in the longitudinal velocity estimate for the hard constraint case. This is the primary cause of the rapid horizontal error growth seen in Figure 2.6. The filter introduces error into the

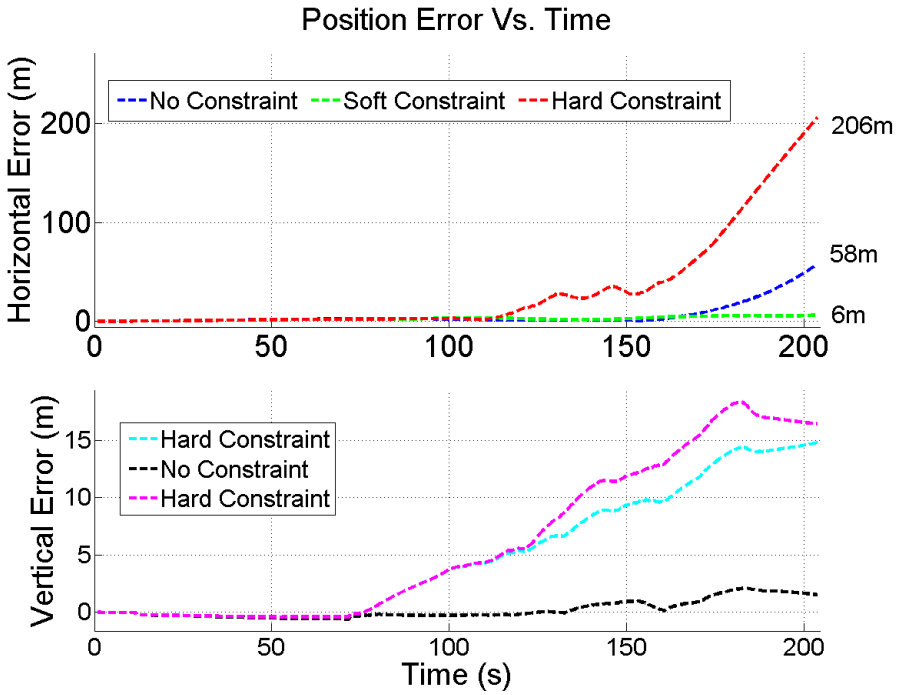


Figure 2.6: Position Estimation Error, Experimental Data.

lateral velocity state by assuming that there is no lateral velocity when there actually is. This error propagates into the north and east velocity states, which are then transformed back into the body frame at the next time step in order to obtain the next estimate of the vehicle frame velocities. The lateral velocity reaches magnitudes of around 1m/s multiple times during the run, and the hard constraint algorithm cannot capture these dynamics. Note also the difference in the lateral velocity estimates of the no constraint and soft constraint approach. These remain similar throughout the beginning of the run, until the estimate from the pure integration begins to drift off at $t > 160s$.

The underlying cause of this error is drift in the roll estimate due to bias drift, although that is not the focus of this study. What is germane to this discussion is that the soft constraint is able to reject these errors while the pure integration cannot. In short, the soft constraint implementation avoids both the errors of constraint violations and unaided signal drift.

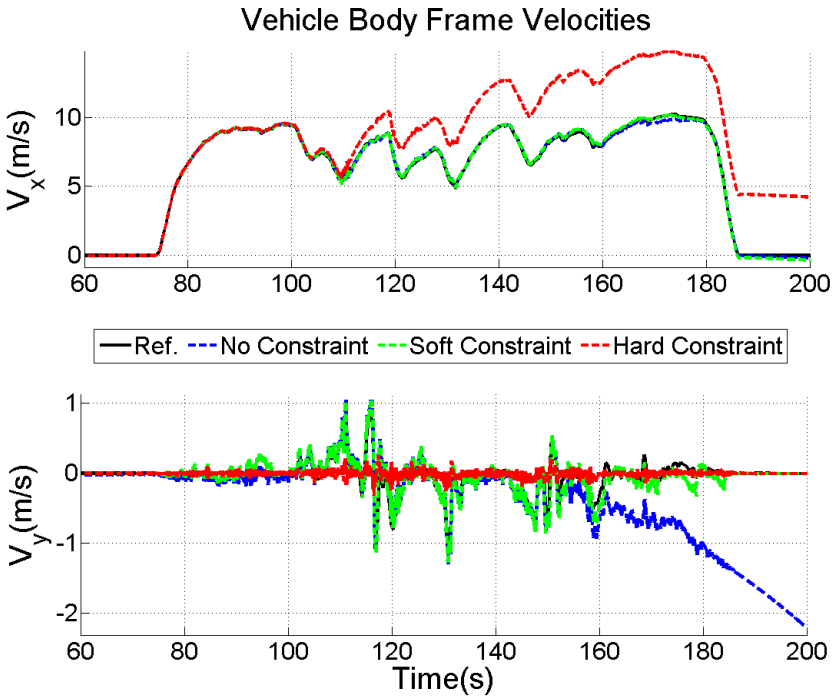


Figure 2.7: Body Frame Velocity Estimates, Experimental Data.

A more intuitive glimpse into the behavior of the 3 algorithms is shown in Figure 2.8, which represents the true and estimated trajectories of the ATV. It is again clear that the soft constraint provides the best estimate throughout the run, as has already been shown. Furthermore, additional insight into the nature of the constraint violations can be seen. Notice that the error in the hard constraint case is fairly stable during the straight segments, and that it increases primarily around corners. This is of course because the constraint violations occur during turning maneuvers. Finally, note the simple drift that is seen in the first trajectory, which is characteristic of un-aided integration of inertial signals. The drift continues even after the vehicle has stopped (seen in the lateral drift in the grid 0-50 East by -50-0 North). The soft and hard constraints prevent this behavior.

Figure 2.9 shows the Z velocity reference in the frame of the vehicle and the estimates produced when using the hard vertical constraint and when neglecting the constraint. It is clear from the figure that the vertical velocity in the vehicle frame is very rarely actually zero, as the constraint assumes. The first reason for this is that the terrain over which the ATV

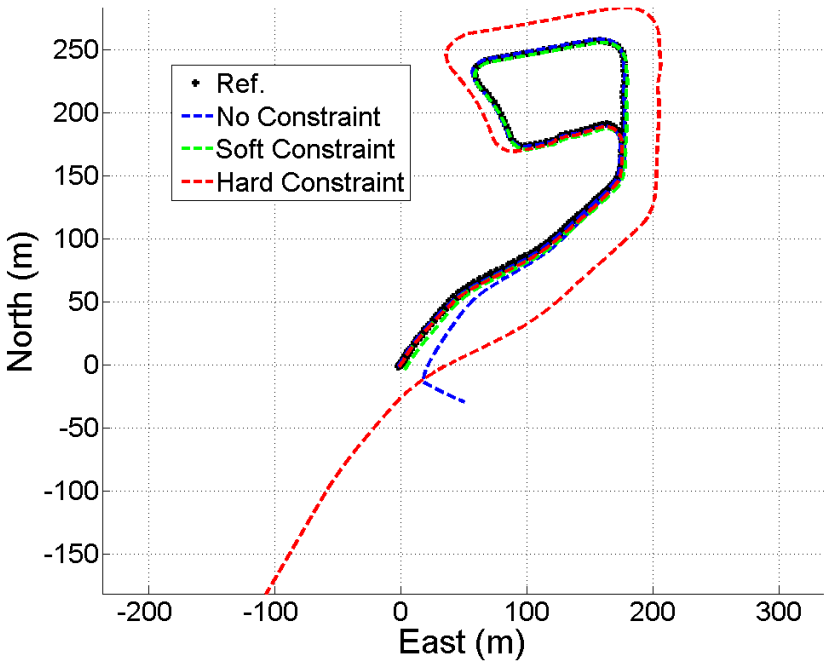


Figure 2.8: Trajectory of Real World Test, with Estimates.

was driven is a gravel road through an un-level field. There are many places where bumps cause bounce effects, violating the constraint. However, the more likely primary cause is the alignment between the IMU coordinate frame and the road coordinate frame. If there is any relative pitch angle θ_{rel} between the surface of the road and the x axis of the IMU, then a component of the longitudinal velocity of the vehicle proportional to $sin(\theta_{rel})$ will be present in the vertical axis of the IMU according to Equation (2.14).

$$V_z^{imu} = V_z^r cos(\theta_{rel}) + V_x^r sin(\theta_{rel}) \tag{2.14}$$

The superscript imu represents the coordinate frame of the IMU, while the superscript r indicates the coordinate frame of the road surface. These errors in the vertical velocity estimate will propagate into the vertical position estimate over time and degrade the solution. This is seen in Figure 2.6, where the vertical position estimation performance is best when the constraint is not applied.

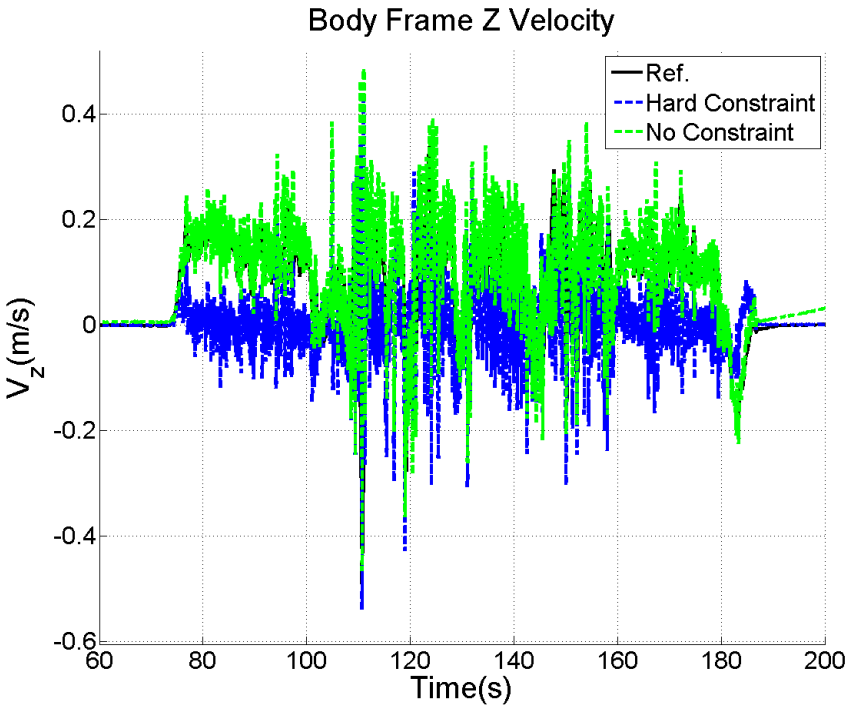


Figure 2.9: Body Frame Vertical Velocity Estimates, Experimental Data.

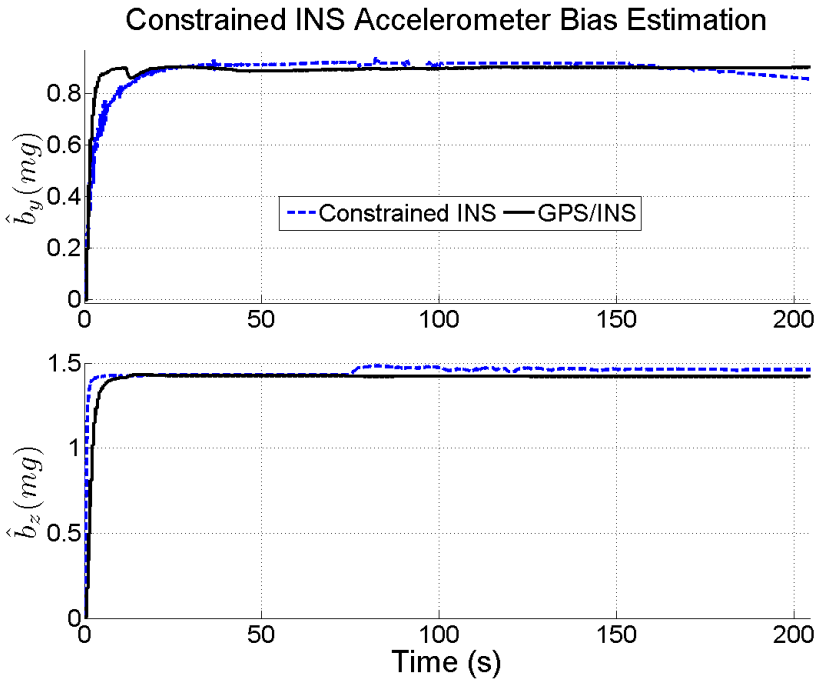


Figure 2.10: Bias Estimation Results, Experimental Data.

Figure 2.10 shows the bias estimation performance of algorithm (using the soft lateral and hard vertical velocity constraints), compared with the bias estimates obtained from a blended (loosely coupled) GPS/INS integration, which is used as a reference. The blended solution was obtained by integrating the raw IMU output of the SPAN with the position, velocity, and attitude solutions. This was done in order to obtain reference values of the uncompensated biases on the raw IMU signal. The GPS/INS solution is not used as a part of the navigation solution presented in this chapter; it is only used to provide a baseline reference estimate of accelerometer bias values. All of the bias states in a loosely coupled integration are fully observable when position, velocity, and attitude measurements are used, so it is concluded that the accelerometer bias estimates obtained from the GPS/INS integration are accurate enough to serve as the baseline reference. The vertical accelerometer bias estimate produced by the navigation algorithm converges to within 0.1mg of the reference estimate. This accuracy is made possible by the vertical velocity constraint, and is achieved despite the fact that the vertical constraint is technically not satisfied (as shown by Figure 2.9). It can be seen, however, that the error in the vertical velocity constraint has a small adverse affect on the bias estimate, beginning at time $t = 70s$. Likewise, the lateral accelerometer bias estimate converges to within 0.05mg of the reference. In this case, a 1mg bias was added to the accelerometer signal in post processing, because it was determined that the true signal had only a miniscule bias during the time of the run and it was desired to show the bias estimation performance. The value of 1mg was chosen because it is the maximum possible value for the sensor bias as specified by Novatel.

2.5 A Note On Filter Parameter Tuning

The algorithms in this chapter contain several parameters which must be tuned. First, the matrix Q, which represents the errors in the system model, must be tuned. The system model in this chapter is based on the kinematic sensor relationships; therefore these expected errors can be obtained from sensor analysis or the given sensor specifications. The values

in Table 2.1 were used to tune Q. Usually, this is also the case for the matrix R, which models the errors in the measurements. However in this chapter there are no aiding sensors, and R models the expected error in the constraints. For this reason there are many possible ways to tune R. In this chapter, a value of $(0.1m/s)^2$ was chosen for R_{22}, which corresponds to the vertical constraint. This was chosen simply because typical GPS vertical velocity measurements have similar error magnitudes. Tuning R_{11} is more open ended. The approach taken in this chapter is to put extreme confidence in the constraint, because such confidence is appropriate when using the soft constraint. Therefore R_{11} was chosen as $(0.005m/s)^2$, which puts extreme weight on the measurement. Of course, this allows no room for constraint violations, which is why the performance of the hard lateral constraint case is so poor.

Another approach would be to use the hard lateral constraint and decrease its weight, allowing for constraint violations, but this is a tradeoff. First, the less weight that is given to R_{11}, the less the constraint will bound the drift during straight driving. If the soft constraint can be accurately applied, then this tradeoff is avoided. Second, tuning R_{11} in this manner makes R_{11} a function of the dynamics. For applications with very predictable dynamic conditions, this approach is likely appropriate. However, if the dynamics are not well known, then applying the high weight to R_{11} and using the soft constraint is recommended.

Of course, the yaw rate condition logic *also* has parameters which must be tuned, namely the yaw rate threshold ζ_r and the time duration T_r. In the simulated experiments, ζ_r was chosen as three sigma of the yaw gyro noise. However it was found in the real world experiments that environment noise made this impractical and a threshold of $\zeta_r = 2deg/s$ was chosen. T_r was selected as 0.25s. The author finds that tuning the yaw rate condition logic is preferable to tuning R_{11}, based on experience and intuition. Intuitively, it should be easier to quantify the noise on the yaw rate gyro, including the environment noise, than quantifying the measure of the constraint violations. However, it still holds that tuning is inevitably required no matter which route is taken.

Table 2.4: Tuning Parameters: Experimental Data

Parameter	Value
R_{11}	2.5e-5 $(^{m}/_{s})^2$
R_{22}	0.01 $(^{m}/_{s})^2$
T_r	0.25 s
ζ_r	2 $^{deg}/_{s}$

A table of all of the measurement related parameters is given in Table 2.4. The process noise parameters were given in Table 2.1.

2.6 Conclusions

In this chapter the effects of lateral and vertical velocity constraints on an otherwise unaided inertial navigation solution have been studied. In general, it is widely assumed that deviations from the assumptions of these constraints are negligible when compared to the error growth caused by the sensor errors. This study has shown that this is not always the case, and that this is largely a function of the sensor quality and the dynamics of the vehicle during the application. Studies were first conducted by simulation, showing that for higher grade sensors the position errors caused by large constraint violations are much larger than those caused by the inertial sensor errors. These results were validated using data collected with a tactical grade IMU mounted to an ATV. However, the simulated experiments also showed that for a MEMS grade IMU the constraint violation errors are smaller than the error growth from the inertial errors. Therefore in this case it would be better to apply the constraint despite the constraint violations than to leave the solution unaided.

This study also demonstrated a third approach to applying the constraint, which is based on certain conditions put on the yaw rate signal. If the yaw rate signal is within a predefined threshold for a predefined period of time, then the vehicle is considered to be driving straight and it is assumed that the lateral velocity constraint is true. This approach avoids the problem of the hard (or constant) constraint and does not introduce extra error during dynamic events. It also avoids the problems posed by allowing the lateral velocity estimate

to drift unchecked, and provides a best of both worlds solution to the lateral constraint application.

Errors in the vertical velocity constraint were also demonstrated to degrade the vertical position estimate. It was shown that if there is any pitch angle difference between the frame of the IMU and the surface of the road, then components of the forward velocity proportional to the sine of the relative pitch angle will be present along the Z axis of the IMU. Therefore the assumption of zero vertical velocity will be invalid. Such misalignment angles can be the result of pitch in the suspension, mounting misalignment, or both.

Finally, this chapter demonstrated the potential for using the velocity constraints to estimate biases in the lateral and vertical accelerometers. The bias states for these two accelerometers are modeled as a 1st order Markov process and incorporated into the Kalman filter. Real world experiments showed that the bias estimates for these two accelerometers converged to the expected values when using the soft lateral velocity constraint and the hard vertical velocity constraint.

Chapter 3

Particle Filters for Platform Classification

3.1 Introduction

This chapter introduces the fundamentals of particle filtering in addition to several important concepts which are used in the Joint Navigation and Classification (JNC) filter. First, the basics of particle filtering are presented in Sections 3.2 through 3.5. Section 3.2 presents some theoretical foundations of particle filtering including Monte Carlo estimation and importance sampling. Next, Section 3.3 presents the most basic implementation of the particle filter algorithm which is commonly known as the Sequential Importance Sampling (SIR) particle filter. Section 3.4 then discusses the problem of sample degeneracy and how it is often mitigated via the technique of resampling. Resampling is an important solution to the degeneracy problem, but it can lead to sample impoverishment. Section 3.5 discusses the problem of sample impoverishment and methods of alleviating this. The JNC filter uses a form of multiple model particle filtering, therefore details regarding multiple model particle filters must be introduced in addition to the particle filter basics. Section 3.6 presents in detail various approaches to the multiple model particle filter for applications which are nearest to the JNC problem. Finally Section 3.7 discusses which approach is the most suitable for the JNC filter.

3.2 Fundamentals of Particle Filtering

Particle filters belong to the class of recursive state estimation algorithms for dynamic systems. Consider a spectrum of this class of algorithms, where the algorithms are arranged in order of their ability to process nonlinear dynamic or measurement equations and non

Gaussian assumptions. On one side would be the well known Kalman filter, which is constrained to linear systems and assumptions of zero mean Gaussian distributed errors on the input and measurement signals. Next would be the extended Kalman filter, which works for nonlinear systems by applying a linearization around the best estimate of the state. The EKF, however, is still subject to Gaussian error assumptions. The Unscented Kalman filter (UKF) and interactive multiple model Kalman filter follow. The UKF attempts to minimize distortion of the uncertainty occurring during nonlinear propagation by using a set of so called sigma points to represent the uncertainty. This improvement allows the UKF to handle systems which have stronger nonlinearity in the dynamics than the EKF can approximate, but it still assumes that the true posterior state distribution is Gaussian. The IMM is designed for nonlinear systems which can be represented by a weighted sum of Gaussian distributions. Finally, the particle filter appears on the far side of the spectrum, opposite the standard Kalman filter. The particle filter operates by approximating the full probability density function of the state using a large number of discrete points (aka particles) with associated weights. In this way the particle filter is not constrained to a Gaussian distribution, nor does it suffer from applying nonlinear functions to the probability density function (PDF). This latter advantage is related to the approach of the UKF. The UKF avoids the nonlinear challenges by propagating sigma points instead of the PDF. The particle filter effectively does the same thing, only with a much greater number of points.

The two primary building blocks of particle filtering are Monte Carlo estimation of expected values and importance sampling. First, consider an arbitrary function $g(x)$ from which the expectation is desired. The Monte Carlo estimator of the expectation of $g(x)$ is denoted as $\hat{g}(x)$ and is given by

$$\hat{g}(x) = \frac{1}{N_i} \sum_{i=1}^{N_i} g(x^i) \tag{3.1}$$

where

$$x^i \sim f_x(x) \tag{3.2}$$

is a random sample drawn from the probability density function $f_x(x)$. This is a powerful tool which has many applications, but the discussion here is limited to particle filtering.

Importance sampling amounts to a specific and very useful application of Monte Carlo estimation. In particular, importance sampling is a method of using Monte Carlo estimation to solve an otherwise difficult definite integral. For example, consider a function $g(x)$ and its integral I_g over the interval $[a, b]$.

$$I_g = \int_a^b g(x)dx \tag{3.3}$$

The first step of importance sampling is to rewrite the integral as an expectation. Consider a probability density function of x, given by $f_x(x)$, and samples drawn from the density as x^i. Then Equation (3.3) can be written as

$$I_g = \int_a^b \frac{g(x)}{f_x(x)} f_x(x)dx = E[\frac{g(x)}{f_x(x)}] \tag{3.4}$$

which can be solved via Monte Carlo estimation. The Monte Carlo estimate of I_g, $\hat{I}_g$ is now given by

$$\hat{I}_g = \frac{1}{N_i} \sum_{i=1}^{N_i} \frac{g(x^i)}{f_x(x^i)} \tag{3.5}$$

In this way an approximation of the integral in Equation (3.3) is obtained via Monte Carlo simulation [2]. Furthermore, this estimate is unbiased as $N_i \to \infty$ [14].

The aim of particle filtering, or any filter in general, is to determine the probability distribution of a state vector x_k at time t_k, conditional on all of the observations Z_k up until t_k

$$p(x_k|Z_{1:k}) \tag{3.6}$$

The Monte Carlo estimate of a general probability distribution $f_x(x)$ is given by

$$\hat{f}(x) = \frac{1}{N_i} \sum_{i=1}^{N_i} \delta(x - x^i) \tag{3.7}$$

where

$$x^i \sim f_x(x) \tag{3.8}$$

and $\delta(x)$ is the Dirac delta function centered at x. This general estimation is foundational to the steps of particle filtering. Importance sampling is required when the desired distribution $f_x(x)$ is difficult to sample from, but may at least be evaluated up to proportionality. In this case, since it is impossible to sample from $f_x(x)$, samples from an importance distribution $q_x(x)$ are drawn and the estimate of the distribution is given by

$$\hat{f}(x) \approx \sum_{i=1}^{N_i} W^i \delta(x - x^i) \tag{3.9}$$

$$x^i \sim q_x(x) \tag{3.10}$$

$$W^i = \frac{w^i(x^i)}{\sum_{i=1}^{N_i} w^i(x^i)} \tag{3.11}$$

$$w^i \propto \frac{f_x(x^i)}{q_x(x^i)} \tag{3.12}$$

In order to utilize Equation (3.9) to estimate Equation (3.6) an appropriate density $q_x(x^i)$ is required. Assume that the prior $q(x_{k-1}|Z_{1:k-1})$ is available, which is a standard requirement for Bayesian filtering. If the system is a Markov process then

$$q(x_k|Z_{1:k-1}) = \int_{x_{k-1} \in X} p(x_k|x_{k-1}) q(x_{k-1}|Z_{1:k-1}) dx_{k-1} \tag{3.13}$$

This can be obtained by sampling from the distribution $p(x_k|x_{k-1})$, which is a function of the state transition model. Having done this, Bayes' equation can now be used to update

the distribution when a measurement becomes available.

$$q(x_k|Z_{1:k}) = \frac{p(z_k|x_k)q(x_k|Z_{1:k-1})}{\int p(z_k|x_k)q(x_k|Z_{1:k-1})} \tag{3.14}$$

The posterior distribution in Equation (3.6) can now be estimated using Equation (3.14) according to Equations (3.9-3.12).

3.3 The SIR Particle Filter

The standard, or baseline particle filter is a straightforward application of the above principles. It is generally referred to as the Sequential-Importance-Sampling (SIS) algorithm, for reasons discussed in this section. The steps of the SIS particle filter are well documented in the tutorial by Aruampalam et al. [4], which is regarded as the preeminent work on basic particle filtering. It consists of three principle steps for each iteration, which can mostly be discussed in the language of recursive filtering. Assuming that the initial state estimate and its uncertainty are known, the initial distribution can be defined. This distribution can be completely arbitrary; it is not subject to any Gaussian assumptions nor is it assumed to be of any form. The distribution is represented using the pair $\{x, w\}$, which are the particles and the associated weights, respectively. Note that each particle represents one estimated time history of the entire state vector. Given the initial distribution approximated by the particles and weights, the first step of one iteration of the filter is to sample from the importance distribution. The importance distribution is given generically (for the case of filtering) by

$$q_x(x_k|x_{k-1}, z_k) \tag{3.15}$$

where q is arbitrary and is selected by the designer. It is often common in the case of nonlinear filtering applications of the particle filter to select the importance density to be conditional on x_{k-1} alone, as it might be difficult to derive or sample from the true density $p(x_k|x_{k-1}, z_k)$. In certain cases this optimal importance density can be derived under certain

restrictions [4], but these will not be applied in this work. However, the density $p(x_k|x_{k-1})$ is known from the state equations, and it is commonly chosen as the importance density.

$$q_x(x_k|x_{k-1}, z_k) = p(x_k|x_{k-1}) \tag{3.16}$$

Sampling from this density merely amounts to applying the state equations to each of the particles and simulating each of the stochastic terms, much as would be done to the state vector of a Kalman filter. However in this case, there are virtually no limits on the dynamic equations. They may be nonlinear and may contain non-Gaussian error distributions. Another difference is that any stochastic terms are sampled from their assumed distributions and actually included in the state equations. The nonlinear, stochastic equations are effectively simulated in order to propagate them forward in time. For this reason particle filtering is referred to as a type of Monte Carlo method.

Once the particles x_k have been sampled from the importance distribution, the weights of the particles must be updated. In the general filtering case, the recursive update equation for the particle weights is given as

$$w_k \propto w_{k-1} \frac{p(z_k|x_k)p(x_k|x_{k-1})}{q_x(x_k|x_{k-1}, z_k)} \tag{3.17}$$

where $p(z_k|x_k)$ is the likelihood function of the measurement given the estimate. This can be calculated given the measurement equations, and is not subject to linear or Gaussian constraints. It is readily observed that if the importance density is chosen to be the state transition model, $p(x_k|x_{k-1})$, as described above, then the weight update equation reduces to

$$w_k \propto w_{k-1} p(z_k|x_k) \tag{3.18}$$

The weight update step is akin to what is commonly referred to as the "measurement update" step of a Kalman filter or other recursive filtering algorithms. However, the particle filter

allows for measurement likelihood functions which are nonlinear and non-Gaussian, just as it does for the state transition model as described above. This is a major advantage when measurements are used which relate to the state in a nonlinear fashion. For example, an RFID emitter simply outputs a signal that can only be received within a certain range. If the receiver picks up the signal, then the sensor must be within the range of the emitter. This information is most accurately represented as a uniform distribution shaped as a sphere, with the center at the emitter location and with a radius of the signal range. Such a distribution could not be used by a Kalman filter, and the information would have to be represented as a Gaussian distribution. The particle filter doesn't force the designer to fit square pegs into round holes.

Once the weights have been determined, the posterior distribution of the state x can be approximated using the particles and their respective weights.

$$p(X_k|Z_k) \approx \sum_{i=1}^{N_p} w_k^i \delta(x_k - x_k^i) \tag{3.19}$$

The state estimate at time t_k can be obtained by calculating the conditional mean from the approximated distribution. This is done via the sum

$$\hat{x}_k = \sum_{i=1}^{N_p} \tilde{w}_k^i x_k^i \tag{3.20}$$

where $\tilde{w}_k^i$ are the normalized weights. Figure 3.1 shows a graphical illustration of the steps in the SIR particle filter.

3.4 Sample Degeneracy and Resampling

Although the SIS filter offers many advantages over other recursive filters, it does suffer from a major drawback which is referred to as sample degeneracy. The two steps of the SIS filter basically amount to propagating the particles forward in time and updating their

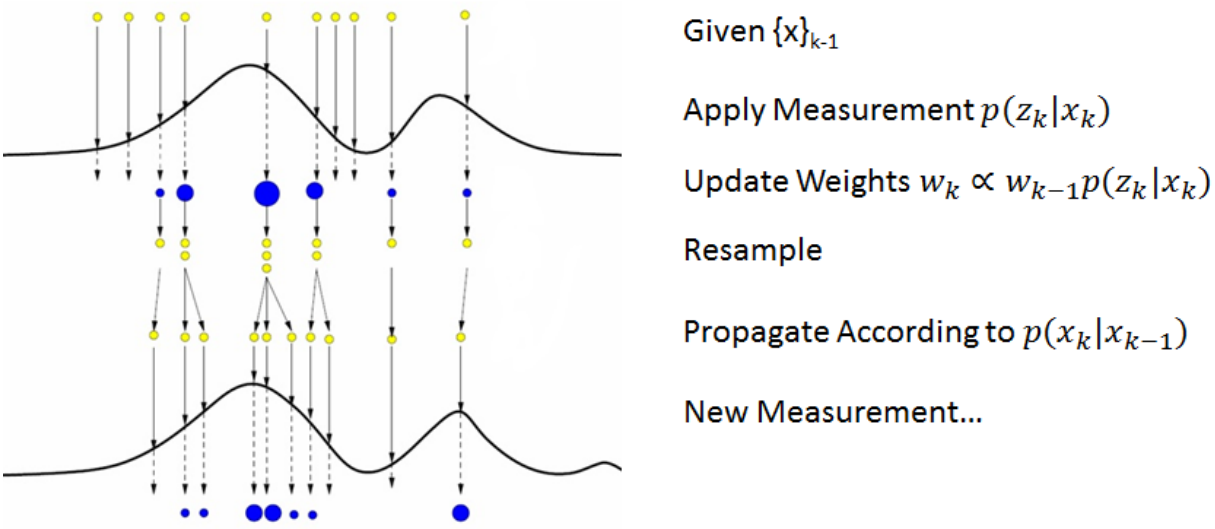


Figure 3.1: SIR Particle Filter Diagram.

weights using the measurement information. In effect, the SIS filter performs a survival of the fittest routine. Yet, after several iterations it is possible for all of the total weight in the filter to become concentrated on one particle, leaving the remaining particles with no weight. Recall the recursive update for the particle weights in Equation (3.18). This equation merely updates the previous weights, which means that if the previous weight for a given particle is very small, then the updated weight will also be very small. No weight can be larger than one (once they are normalized), so the updates can do little to increase the weights of deweighted particles. Therefore the posterior PDF approximation is effectively represented by only one particle after a time, and this can be inaccurate. The authors of [13] have proven that this degradation is inevitable, therefore a meaningful solution is required [4].

The standard solution to the degeneracy problem is to resample the particles based on their weights, generating a new set of particles $\{\underline{x}_k\}$ from the weighted set by sampling new particle $x_k^j = x_k^i$ with $P(x_k^j = x_k^i) = w_k^i$. Particles with higher weights are likely to be resampled multiple times, while particles with low weight are likely to be sampled few times, if any. The idea behind resampling is to reintroduce sample diversity in the vicinity of particles with high weight. Sample degeneracy is not a problem because the particle with all of the weight is the wrong particle; in fact the particle with the highest weight is the most

likely particle from that set. The problem is that once the weight has collapsed to one particle there no longer remains any modality for further diversity in the PDF approximation. This means that only one of many stochastic possibilities is tested at each time step. Resampling maintains the "center of mass" of the weights in the proper location in the state space, but it clusters many particles in this location which allows for more diversity and better coverage of the state space. There are several possible implementations of particle resampling which have been shown to yield comparable results. These are multinomial resampling, residual resampling, stratified resampling, and systematic resampling [12]. Systematic is the most commonly used due to its simplicity [4], [12], and it is likewise used in this work. The algorithm for systematic resampling is given by Algorithm 1 [4]. This algorithm is a simple

Algorithm 1 Systematic Resampling

Given weights w^i and particles x_k^i
Initialize CDF: $c_1 = 0$
for $i = 2 : N_p$ **do**
  Construct CDF: $c_i = c_{i-1} + w^i$
end for
Let CDF index $i = 1$
Draw starting point $u_1 \sim U\left[0, N_p^{-1}\right]$
for $j = 1 : N_p$ **do**
  $u_j = u_1 + N_p^{-1}(j-1)$
  while $u_j > c_i$ **do**
    $i = i + 1$
  end while
  Assign new sample $x_k^j = x_k^i$
  Assign uniform weight $w_k^j = N_p^{-1}$
  Assign parent $i^j = i$
end for

way of drawing N_p particles from an arbitrary distribution, where the probability of drawing a particle is directly proportional to its prior weight. Once the particles are drawn, all are reassigned an equal weight of $1/N_p$.

3.5 Sample Impoverishment and Mitigating Techniques

While resampling is an effective means of addressing the sample degeneracy problem, it leads to problems of its own which must also be addressed. The primary problem introduced by resampling is referred to as sample *impoverishment*. This is analogous to sample degeneracy, in that the problem is still a lack of diversity in the state space. Sample degeneracy exists when only one particle remains with any significant weight. Sample impoverishment exists when many particles are concentrated tightly in a region of the state space. In both cases, the result is that the "net weight" of all of the particles is very tightly clustered. This occurs when resampling is performed too frequently relative to the process noise, because the particles do not have time to spread out after a resample event. Resampling too often, when combined with very small process noise, is often the cause of sample impoverishment. A common and intuitive approach to improving sample impoverishment is to only resample the particles when they become degenerate. Arulampalam et al. [4] offer a method to quantify the degree of sample degeneracy using a measure of the "effective sample size" N_{eff}. This is calculated using the particle weights in the equation below.

$$N_{eff}(k) = \frac{1}{\sum_{i=1}^{N_p}(w_k^i)^2} \tag{3.21}$$

N_{eff} tends towards its maximum value as all of the weights approach $1/N_p$. In the extreme case of sample degeneracy, when all of the particles but one have zero weight, N_{eff} approaches 1. Thus a lower bound can be placed on N_{eff} such that resampling occurs when N_{eff} falls below this value. In this dissertation the threshold is set as $^2/_3 N_p$. Note that Equation (3.21) implies that increasing the number of particles will improve the effective sample size. Recall that Section 3.2 states that the Monte Carlo approximation of the expectation integral only converges as N_p approaches infinity. Here, Equation (3.21) offers a practical reason for the requirement of many particles. Essentially there are not enough particles to cover the state space with enough breadth and resolution.

It is often the case that applying Equation (3.21) alone isn't enough to overcome sample impoverishment (without a prohibitively large number of particles). Therefore, additional modifications to the basic resampling process are often made. Regularization is a means of introducing additional sample diversity in the sampling process by jittering the particle set with random samples drawn from a suboptimal density [4], [31]. The baseline resampling process approximates the solution by sampling from a discrete density. Regularization seeks to approximate a continuous density by drawing from a distribution which minimizes the difference between the true posterior and the estimated posterior, according to the mean integrated square error (MISE). This optimal density can be sampled by scaling the standard normal distribution according to Equations (3.22 - 3.26),

$$x_k^j = x_k^j + h_G D_k \sigma_u \tag{3.22}$$

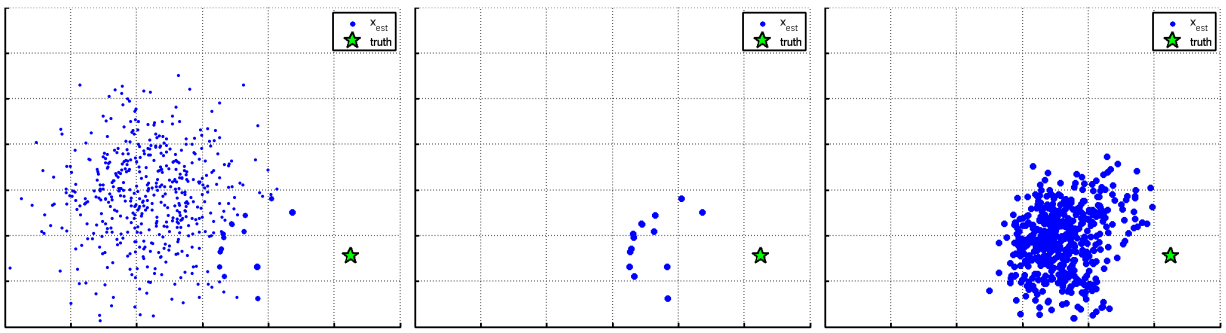
$$\sigma_u \sim N(0,1) \tag{3.23}$$

$$h_G = A N_p^{1/(n_x+4)} \tag{3.24}$$

$$A = 8 c_{n_x}^{-1} (n_x + 4)(2\sqrt{\pi}^{n_x})^{1/(n_x+4)} \tag{3.25}$$

$$D_k D_k^T = S_k \tag{3.26}$$

where x_k^j is a particle drawn from the systematic resampling algorithm, S_k is the empirically determined covariance of the particle states, n_x is the number of states in the nonlinear partition, and c_{n_x} is the volume of a unit hypersphere of an n_x dimensional space. These assume that all of the particles have the same weight, and that the true posterior is Gaussian in nature. The former is true given the systematic resampling, while the latter is an acceptable approximation in practice. Hence regularization is a suboptimal method [31]. It does not provide a sample from the optimal posterior distribution (in the sense of MISE), because the Gaussian assumption on the posterior is likely false. However, it does provide much needed diversity among the posterior particles, which is the primary reason for doing the regularizaiton. Figure 3.2 illustrates the effects of regularization. Note that it is also found



(a) (b) (c)

Figure 3.2: (A) Particles prior to resampling. (B) Particles post resampling. (C) Particles post regularization.

in practice that the regularization does not force the posterior to excessively conform to a Gaussian distribution. Consider Figure 3.2b, where the posterior after resampling has a distinct non-Gaussian structure but very little particle diversity. Performing the regularization results in Figure 3.2c, which has far more of a Gaussian structure than Figure 3.2b but still maintains the essential shape. Particle diversity is achieved and the underlying structure is maintained. This is also explained by Equation (3.22). The Gaussian noise is added to each particle individually, as opposed to collectively forcing them all into a Gaussian distribution. For example, regularization applied to a highly multimodal posterior would result in the same dominant modes but with some added noise around each mode.

A problem related to sample impoverishment occurs when a measurement is used which has a very strongly peaked likelihood function. As an example, consider a case where a set of particles representing geodetic position have drifted over time with no strong external aiding. If a GPS measurement were to become available, which has a very small uncertainty, then the resulting 2D likelihood is represented by a 2D Gaussian function which is very narrow. Consequently, very few if any of the particles will lie within an active region of the likelihood function. Being Gaussian, the likelihood function extends to infinity in both directions, but the slope of the curve flattens dramatically at the tails. This means two things. First, the scaling of the particles relative to one another will also be very slight. Second, the actual state update will be very slight. The update occurs as the highest weighted particles get

resampled. The particle set cannot "jump" to a region where there were previously no particles. They will gradually tend towards this peaked likelihood over time via weighting and resampling. Contrast this with a Kalman filter, where the state will jump towards the measurement (given an appropriate uncertainty for both). Doucet et al. [13] propose a method of incorporating the latest measurement into the proposal distribution. This is commonly referred to as a "local linearization particle filter" (LLPF) or "extended Kalman particle filter" (EPF). In Equation (3.16) the proposal density depended only on the prior state. The LLPF conditions the importance density on z_k by applying an extended Kalman filter update to each particle prior to resampling. The importance density is given by

$$q_x(x_k|x_{k-1}, z_k) = N(\hat{x}_{k|k-1}, P_k) \tag{3.27}$$

where $\hat{x}_{k|k-1}$ is the particle set having been updated by a Kalman filter. The local linearization particle filter is given in Algorithm 2. The matrix S_k is the innovation uncertainty

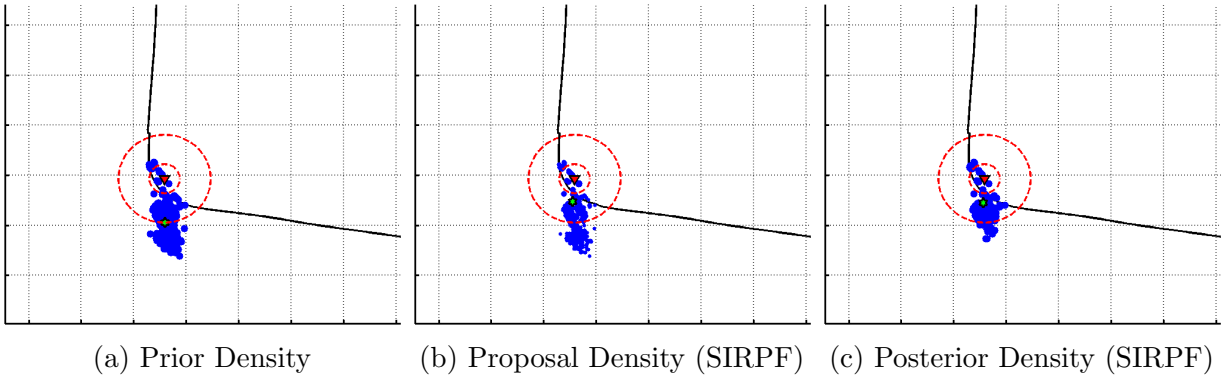
Algorithm 2 Local Linearization Particle Filter

Given weights w_k, particles $x_{k|k-1}$, measurement z_k, and measurement uncertainty R_k
Compute covariance P_k of particles empirically
for $i = 1 : N_p$ **do**
  Evaluate $w_k^i = (w_{k-1}^i)N(z_k; x_{k|k-1}^i, S_k)$
  Compute $x_{k|k}^i = EKF(x_{k|k-1}^i, P_{k|k}, z_k, R_k)$
end for
Resample set $\{x_{k|k}, w_k\}$ if necessary

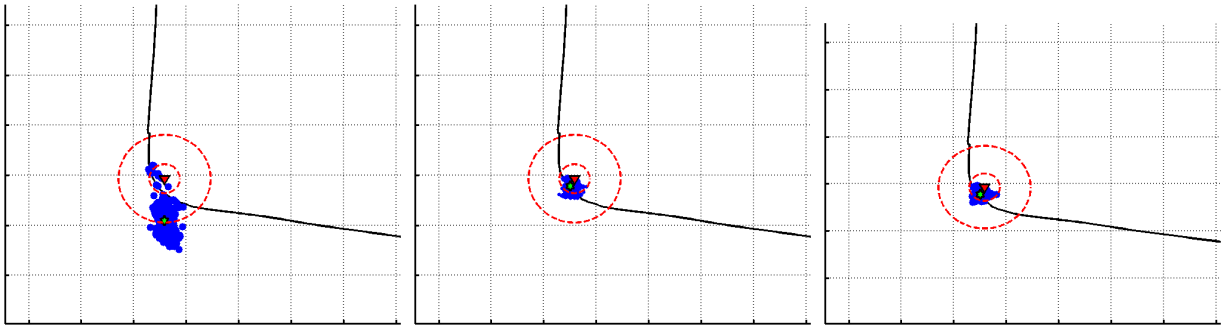
which is calculated as part of the standard Kalman filter update process. Figure 3.3 shows the effects of the local linearization particle filter.

3.6 Hybrid State Particle Filters

Boers and Driessen [7] present a general method of applying particle filtering techniques to systems which have both continuous and discrete states in the state vector. The discrete state(s) differs from the continuous states in that it can only assume values from a discrete



(a) Prior Density (b) Proposal Density (SIRPF) (c) Posterior Density (SIRPF)



(d) Prior Density (e) Proposal Density (LLPF) (f) Posterior Density (LLPF)

Figure 3.3: Comparison of SIRPF Resampling with LLPF Resampling. Red dashed lines represent Gaussian measurement 1σ and 2σ uncertainties.

set. Their method is general in that it can represent various type of information, such as a parameter, a mode of operation, a dynamic model, a measurement model, or other general "class" information. For the applications considered in this dissertation, the discrete state generally represents a mode of operation. Different modes can then be subject to different parameters in the dynamic or measurement models, or radically different system models altogether. The particle filter functions much like a standard particle filter, only now the discrete state must also be propagated and weighted according to the measurement likelihood functions. The algorithm is summarized in Algorithm 3. It is clear that this algorithm is a straightforward extension of the classical particle filter for non-hybrid systems, which is one advantage of the algorithm.

However, the primary drawback is that modes (or discrete states in general) which have low probabilities are likely to disappear from the particle set altogether in the resampling process. This is perhaps acceptable for the case where the discrete state represents a static

Algorithm 3 Hybrid State Particle Filter (Boers and Driessen) [7]

$X = \begin{bmatrix} x_c & x_d \end{bmatrix}^T$
x_c = continuous state
x_d = discrete state
Set initial distributions of x_c and x_d.
for each iteration **do**
  for all particles p **do**
    Propagate x_d according to Markov transition probabilities.
    Propagate x_c conditional on x_d.
    Calculate the weights according to measurement likelihood functions.
    Apply desired resampling algorithm.
  end for
end for

class which merely needs to be determined once. However this behavior completely disrupts the desired estimator functionality for the case where the mode can change. Even in the static mode case, this algorithm still suffers from a lack of robustness. This is because the resampling process can still shift all of the weight toward one particular mode via the randomness in the estimator, even if there is no true correlation between the measurement information and the set of potential modes. This effect is simply a case of sample impoverishment impacting the discrete state.

Gordon et al. [21] present a particle filtering solution for the problem of joint tracking and classification (JTC) of radar targets. JTC is one of the most common applications of the hybrid state particle filter. In this application, the dynamic state of the target (position, velocity, etc.) is the continuous state and the target "class" (friend or foe, aircraft type, etc.) is the discrete state. In this case, the class is the aircraft type. The authors propose the idea that information about the dynamics of the target ought to inform the estimate of the target class, and vice versa. For example, certain types of aircraft cannot exceed particular speeds, while others can. Therefore observing the speed of an unknown aircraft type could potentially distinguish its class. Likewise, knowing the type of aircraft can help to constrain estimates of the aircraft dynamics. This is achieved by implementing a type of hybrid state particle filter, where the dynamics of the aircraft depend on which aircraft type

it is. The authors address the problem of sample impoverishment in the discrete state by maintaining separate sets of equal number of particles for *every* possible mode. In this way it is ensured that none of the modes can disappear during resampling, because each of the different sets is resampled individually. The probability of each possible mode is then simply the sum of the weights for each mode, or the total weight for that mode. The authors note that maintaining an equal number of particles for all possible modes, even unlikely ones, increases the computational load as opposed to allowing the number of particles to vary (greater than zero of course) as in [24]. This is the price to ensure robustness against the sample impoverishment problem. The algorithm is summarized below in Algorithm 4.

Algorithm 4 Hybrid State Particle Filter (Gordon et al.) [21]

$X = \begin{bmatrix} x_c & m \end{bmatrix}^T$
x_c = continuous state
m = discrete state (mode)
Set initial distributions of x_c and m.
for each iteration **do**
  for all modes $m \in M$ **do**
    for all particles p **do**
      Propagate x_c conditional on m.
      Calculate the weights according to measurement likelihood functions.
    end for
    Sum the particle weights to obtain probability of the mode.
    Apply desired resampling algorithm to particles in this mode.
  end for
end for

The authors of [33] extend the basic ideas of [21] to the case when the mode conditioned system models are linear and Gaussian. In this case, an IMM filter can be used to estimate the mode state given the outputs of the mode conditioned filters, which can be implemented as Kalman filters or IMM filters themselves. The different mode conditioned models are based on different constraints on the {velocity, acceleration} plane, and the authors propose using IMM filters for the mode conditioned filters in order to implement the constraints. Basically, each mode conditioned filter is an IMM filter where each different mode conditioned filter has a different set of possible {velocity, acceleration} pairs to choose from. One class has

only one possible {velocity,acceleration}, while another has several possible pairs, while the third class accepts all possible pairs. At each time step the likelihood functions for each class are calculated and used as inputs into the overall IMM filter, which determined the most likely mode. This is not very different fundamentally from the work in [21], where the different mode conditioned particle sets can be thought of as generalizations of the mode conditioned IMM filters in [33]. The particle filter approach is preferred for this dissertation due to the flexibility of applying nonlinear constraints and using nonlinear measurements.

While Gordon et al. [21] present a particle filtering solution for hybrid state estimation which is robust to sample impoverishment of the mode (discrete) state, their method is only sufficient for the case when the mode is static. That is, the true mode state cannot change. Although the goal of the particle filter, or any true filtering algorithm, is to produce estimates of the system state at the current time, these states are predicated on what happened previously. It is actually the time history of the state that the filtering algorithm is fundamentally estimating, therefore hybrid systems which allow for mode switching cannot be estimated using filters based on static mode states. In order to solve this problem, Driessen and Boers [15] propose a hybrid state particle filter where the mode state can change according to a Markov transition model. The authors apply Monte Carlo methods to the continuous portion of the state while they derive the exact Bayesian equations for the evolution of the mode state probabilities. The transition model of the continuous state is conditional on the mode state, while the mode transitions are independent of the continuous state (in accordance with the Markov properties). In essence, the mode conditioned state is solved via particle filtering while the mode state is solved by directly sampling from its distribution. Additionally, this method allows the designer to control the number of particles per mode independently at each time step, thereby avoiding the sample degeneracy problem. The algorithm is described in Algorithms 5 and 6.

The filter proposed by Driessen and Boers solves the problem of sample degeneracy for particle filter based estimation of hybrid state systems when the mode state is dynamic.

Algorithm 5 Dynamic Hybrid State Particle Filter (Driessen and Boers) [15]

$X = \begin{bmatrix} s & m \end{bmatrix}^T$
$s =$ continuous state
$m =$ discrete state (mode)
Set initial distributions of s and m.
for each iteration **do**
  for all modes $m \in M$ **do**
    Do mixing according to Algorithm 6.
    Predict s according to state transition model and m.
    Determine particle weights from measurement likelihoods.
    Resample the particles in mode set m.
    Calculate mode probability according to Equations (3.29) and (3.30).
  end for
end for

Algorithm 6 Mode State Mixing Algorithm for [15]

for mode m_k **do**
  Calculate $P(m_{k+1}|Z_k)$ according to Equations (3.28 - 3.30).
  Calculate $P(m_k|m_{k+1}, Z_k)$ according to Equation (3.31).
  Resample $N_{m_{k+1}}$ modes from $P(m_k|m_{k+1}, Z_k)$.
  Set the weight for all s in mode m_k to $\frac{1}{N_{m_{k+1}}}$ and resample.
end for

$$p(m_{k+1}|Z_k) = \sum_{m_k=1}^{M} = p(m_{k+1}|m_k)p(m_k|Z_k) \tag{3.28}$$

$$p(m_k|Z_k) = \frac{\lambda_k^{m_k} p(m_k|Z_{k-1})}{\sum_{m=1}^{M} \lambda_k^{m_k} p(m_k|Z_{k-1})} \tag{3.29}$$

$$\lambda_k^{m_k} = \sum_{i=1}^{N_{m_k}} w_k^{i,m_k} \tag{3.30}$$

$$p(m_k|m_{k+1}, Z_k) = \frac{p(m_k|Z_k)}{p(m_{k+1}|Z_k)} \tag{3.31}$$

However, their method only applies when the mode state satisfies the Markov property. It does not apply when the mode transitions are a function of the continuous state and therefore do not satisfy the Markov property. One example given by Blom and Bloem [6] is that of an aircraft taxiing at the airport. Let the mode state be the set of possible maneuvers and let the continuous state include the position. The probability of the aircraft

turning depends on where it is, because it is only likely to turn when at a runway crossing. Therefore the probability of a mode switch is a function of the continuous state, ie. position. Motivated by such cases, Blom and Bloem develop an algorithm very similar to Driessen and Boers [15] with the distinction that their algorithm is able to handle mode states which are not constrained to Markov transitions. Functionally, the algorithms do the same thing apart from this difference. The particle prediction, correction, and mode probability calculations are the same as in [15], only the mixing calculations are different. The mixing algorithm of Blom and Bloem is shown below in Algorithm 7.

Algorithm 7 Mode State Mixing Algorithm for [6]

for mode m_k **do**
  Calculate the transition probabilities, which may be a function of the state.
  Calculate the mode probabilities $P(m_k|Z_{k-1})$ given the mode priors and the transition probabilities.
  Resample N_{mk+1} particles given the mode probabilities.
end for

The authors of [30] present a particle filter solution for estimating both the continuous and discrete states of a hybrid nonlinear system. In this context, the continuous states represent the system dynamics, while the discrete state(s) represent a particular mode of operation. Different modes could have different model structures, or simply different discrete values of one parameter. The algorithm is designed to be able to determine the current mode of operation and simultaneously provide an estimate of the dynamic system states. Traditionally, a standard or baseline multiple model particle filter implementation suffers from sample impoverishment when there are possible modes with low transition probabilities [21]. The authors overcome this, however, by employing a look ahead step in the resampling process. In between measurement updates, each particle is propagated according to *each* of the possible modes $m \in M$. When measurements arrive the posterior probability of each of these modes is determined based on the measurement likelihoods and the transition probabilities. The weight for each particle then becomes the sum of the posteriors across all M modes. The particles are now resampled as usual according to these weights. However,

the mode of each the new particles is sampled based on the previously mentioned posteriors. The algorithm is summarized in Algorithm 8, with a diagram of the method given in Figure 3.4.

Algorithm 8 Hybrid State Particle Filter (Narasimhan) [30]

for all particles x^i **do**
  for all modes $m^j \in M$ **do**
    Propagate a version of the particle according to mode m^j using Equation (3.32).
    Determine the weight of mode m^j using Equation (3.33).
  end for
  Calculate the weight of particle using Equation (3.34).
end for
Resample continuous particles according to weights.
for all particles x^{i*} **do**
  Draw discrete mode m^{j*} according to $w^{j,i*}$.
end for

$$x_k^i \sim p(x_k^i | x_{k-1}^i, m^j) \tag{3.32}$$

$$w_m^{j,i} = p(y_k | x_k^i, m_k^j) T_{mj} \tag{3.33}$$

$$w_p^i = \sum_{j=1}^{M} w_m^{j,i} \tag{3.34}$$

$$p(m^{j*}) = w_p^i \tag{3.35}$$

Tafazoli and Sun [38] also address the problem of sample impoverishment occurring in hybrid state particle filters. The standard particle filtering algorithm for hybrid system suffers from sample impoverishment when the transition probabilities to the true mode are small. This is because the mode is sampled using only the transition probability information, ignoring any new measurement information. Thus, low probability modes are unlikely to be sampled, even if measurements indicate that they are correct. Tafazoli and Sun propose a method of incorporating the measurement likelihoods into the mode sampling process to mitigate this problem. First, at every time step, N particles are sampled from *each* of the M possible modes (having a non-zero transition probability) using the importance function for each mode. Next, the likelihood functions of the measurements are applied to each of

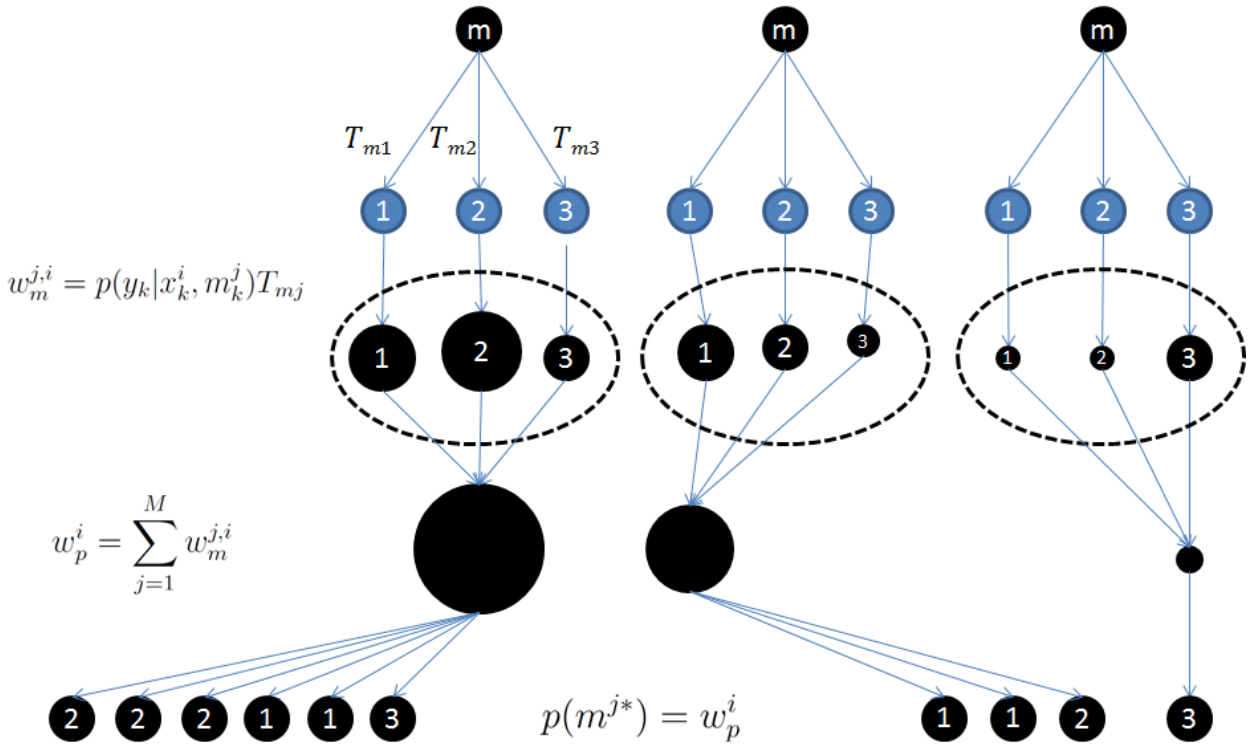


Figure 3.4: Narasimhan's Particle Filter Diagram.

the MN particles, and the average weight for each mode is computed. Finally, the winning mode is selected as the mode with the maximum of the average weight multiplied by the prior transition probability. The algorithm is summarized in Algorithm 9, with a diagram of the algorithm given in Figure 3.5.

3.7 Conclusion: Application of Hybrid State Particle Filter to JNC

This chapter has presented the fundamentals of particle filtering and some important methods which are required to mitigate problems common to many particle filter applications. First, the concepts of Monte Carlo approximation of integral and importance sampling were introduced. From this, the SIR particle filter was presented. Basic limitation of particle filter such as sample degeneracy and sample impoverishment were introduced, followed by important techniques for mitigating these limitations. These include resampling, regularization, and including the current measurement in the proposal density via a local linearization.

Algorithm 9 Hybrid State Particle Filter (Tafazoli and Sun) [38]

for all modes $m_i \in M$ **do**
  for all particles p_i **do**
    Propagate the particle according to mode m_j using (3.32).
  end for
  Determine the average weight of mode m_j using (3.36).
end for
Calculate the mode probabilities according to (3.37).
Select the mode with the maximum probability.
Resample from the particles in the winning mode according to their individual weights.

$$W_m^j = \sum_{i=1}^{N_p} \frac{w^i}{N_p} \tag{3.36}$$

$$p(m^j) = T_{ij} W_m^j \tag{3.37}$$

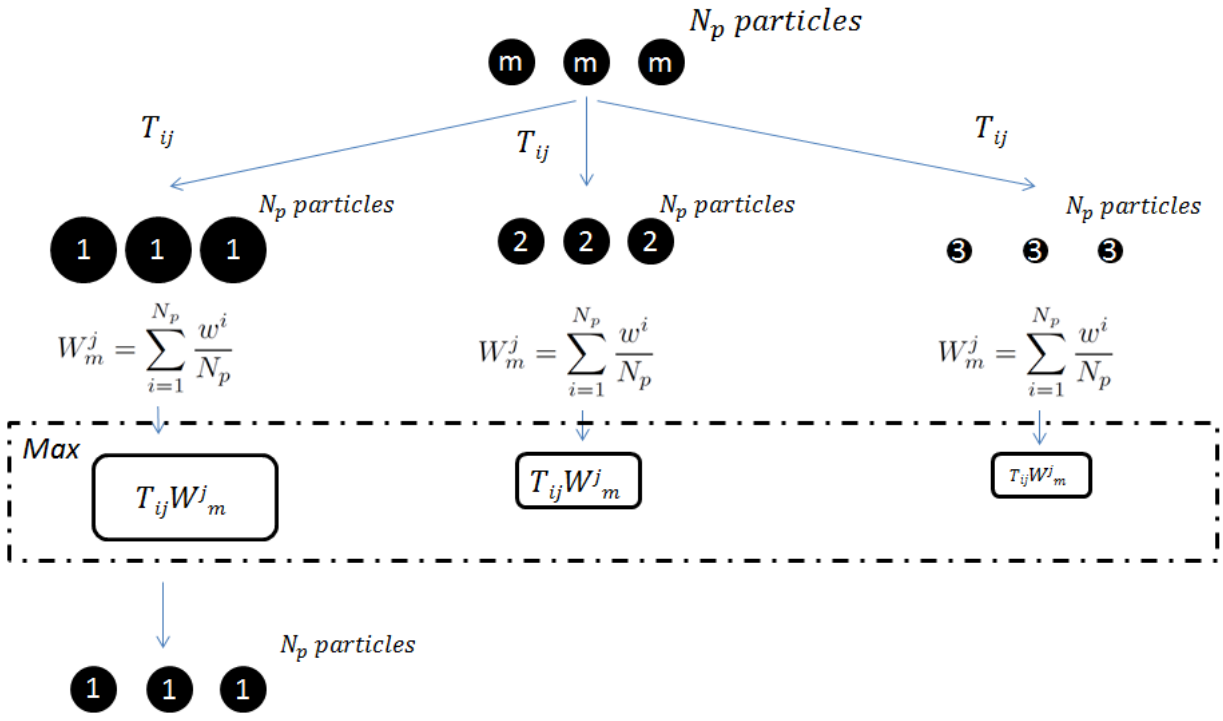


Figure 3.5: Tafazoli and Sun's Particle Filter Diagram.

Table 3.1: Taxonomy of Hybrid State Particle Filters

	Mode State	Mode Transition	Comments
Boers / Driessen [7]	Dynamic	Markov	Basic MMPF
Gordon et al. [21]	Static	N/A	Separate Modes
Driessen / Boers [15]	Dynamic	Markov	Mixing
Blom / Bloem [6]	Dynamic	General	Mixing
Narasimhan et al. [30]	Dynamic	General	Look Ahead
Tafazoli and Sun [38]	Dynamic	General	Look Ahead

Additionally this chapter has presented an overview of some extensions of the basic particle filter to hybrid state filtering, where the state vector consists of a continuous state vector and a discrete state vector. In this chapter, the survey is limited to hybrid state filters where the discrete state can be considered a *mode* upon which the continuous state dynamics are conditioned. Table 3.1 presents a taxonomy of these particle filters, neglecting the IMM filter presented by Ristic. The IMM filter is excluded because it does not allow for very nonlinear measurement types, and it can be considered merely a specific case of Gordon's hybrid state filter. Table 3.1 represents different philosophies of dealing with the mode impoverishment problem, where discrete states with low probability can "die off" in the resampling process before it is appropriate. This is especially problematic when the problem requires a longer time history of measurements to robustly sort out the mode, or when the mode can potentially change. It is reasonable to model the conditions of the JNC problem, as a first approximation, as a case where the mode cannot change over time. This rules out the need for dynamic mode states; indeed, the possibility of mode transitions can only hurt the filter performance. The Boers / Driessen filter includes a static mode in the state vector but does not implement any strategies for mitigating the mode impoverishment problem. However, the Gordon filter includes a static mode state and does not suffer from mode impoverishment. Therefore the Gordon filter is selected as a framework to be followed when considering the JNC problem.

Chapter 4

Joint Navigation and Classification Particle Filter

4.1 Introduction

The goal of this dissertation is to autonomously identify the vehicle platform type and apply the appropriate navigation constraints in order to improve the overall navigation. A type of multiple model particle filter is developed to achieve this goal. Chapter 3.2 presented the fundamental elements of particle filtering which are germane to the multiple model particle filter. This chapter presents the Joint Navigation and Classifcation Particle Filter (JNCPF) in detail. The JNCPF is an indirect, or error-state, filter as opposed to a direct-state filter. This is an important distinction which is outlined in Section 4.2. The JNCPF functions as a multiple model particle filter similar to the MMPF by [21] and discussed in Section 3.6, but a Rao-Blackwellization technique has been applied to make it more computationally feasible. This technique is presented in Section 4.3. The models which comprise the "multiple model" aspect of the JNCPF are detailed in Section 4.4. Next the measurements and sensor models which are used in this work are presented in Section 4.5. Finally, the JNCPF is summarized in Section 4.6 and a pseudo-code algorithm is given to illustrate the overall system.

4.2 Indirect Kalman Filtering

It is common in aided inertial navigation applications to use an indirect filtering formulation, due to the resulting modeling advantages [23]. This means that the state vector does not consist of the navigational states of interest. Rather the state vector is comprised of the errors in position, velocity, etc. Thus a reference state trajectory is maintained in addition

to the error state vector, and the true navigation state estimate is the combination of the two [23]. The error state vector is given by

$$\delta x = \begin{bmatrix} \delta r_N & \delta v_N & \delta \psi_N & b_a & \kappa_a & b_g & \kappa_g & b_{as} & b_{gs} \end{bmatrix}^T \tag{4.1}$$

where each of the terms is a 3×1 vector, resulting in a state vector that is 27 elements long. The position and velocity estimates are computed from the reference and the error states using Equations (4.2) and (4.3).

$$\hat{r}_{lla} = x_{1:3}^{ref} + \begin{bmatrix} \frac{1}{R_\lambda} & 0 & 0 \\ 0 & \frac{1}{R_\Phi \cos(\lambda)} & 0 \\ 0 & 0 & -1 \end{bmatrix} \delta \hat{x}_{1:3} \tag{4.2}$$

$$\hat{v}_N = x_{4:6}^{ref} + \delta \hat{x}_{4:6} \tag{4.3}$$

The attitude estimate must be computed using the reference and error state attitude quantities expressed as direction cosine matrices (DCM)'s. This is done in Equations (4.4) and (4.5) [28].

$$\hat{R}_N^B = \overline{R}_N^B \delta R_N^B \tag{4.4}$$

$$\delta R_N^B = I_{3\times 3} - \left[\delta \underline{\psi} \times\right] + \frac{1}{2}\left(\delta \underline{\psi} \delta \underline{\psi}^T - \delta \underline{\psi}^2 I_{3\times 3}\right) \tag{4.5}$$

The filter thus maintains the reference and error states separately and the best estimate is computed at each time step. The error estimates naturally grow over time, due to the error growth inherent in inertial sensors. The filter must be reset from time to time in order to mitigate this effect and retain the advantages of the indirect filtering approach. The reset frequency is chosen heuristically based on the IMU metadata. When a filter reset occurs, the reference trajectory is reset to the current best estimate of the total state. The error state vector is reset to zero, and the total state estimate remains unchanged from before the

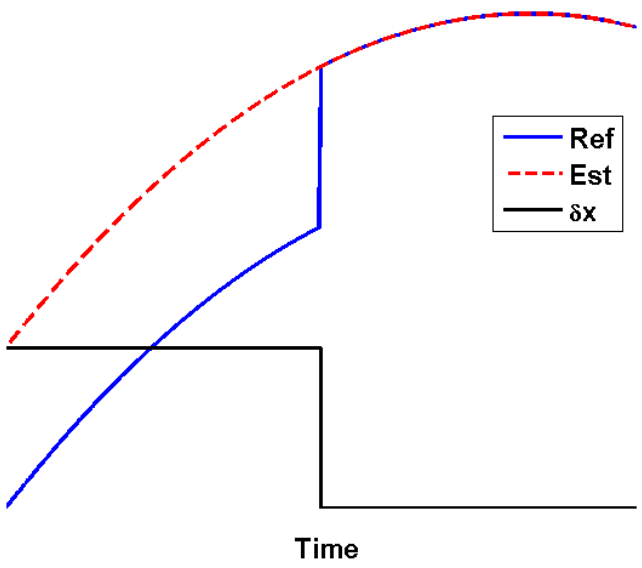


Figure 4.1: Generalized example of indirect filter structure.

reset to after. This is seen in Figure 4.1, where a generalized example of the relationship between the reference trajectory, the error state, and the state estimate is visualized. A reset is illustrated in the middle of the graph.

4.3 Rao Blackwellized Particle Filtering

One of the major drawbacks of particle filtering techniques is the heavy computational burden that can arise from running an adequate number of particles. Recall that the Monte Carlo approximation only holds as the number of particles tends to infinity. Realistically, of course, a finite number of particles is sufficient. Nevertheless the appropriate number can be very high, thus multiplying many of the filtering operations and hindering real time performance. The number of particles needed scales non-linearly with the number of states. This is a problem for aided INS navigation filters, which typically have a minimum of 15 states. The JNCPF uses 27 states, and applying particles to this many dimensions is utterly intractable [36]. Perhaps the most elegant solution to this scaling problem is Rao-Blackwellization. This is based on the idea that the state space can be partitioned into

linear and nonlinear subspaces, thereby limiting the number of states which actually use particles. Practically speaking this is very intuitive for navigation, because it is well known that the error dynamics are sufficiently approximated by a first order linearization. Hence the popularity of extended Kalman filters for aided INS applications. Most of the very non-linear relationships appear in the position measurement equations. For example, the ASPN data sets include measurements from RFID readers in several pedestrian cases. These are binary signals which simply indicate whether or not a sensor is within range of an RFID emitter at a known location. The measurement distribution is uniform over a region, rather than Gaussian.

Schon et al. [36] present a Rao-Blackwellization methodology for general state space models which have linear and non-linear components. The result is called a Rao-Blackwellized Particle Filter (RBPF). The dynamic model of the system is given by Equations (4.6) - (4.9).

$$x_{k+1}^n = f_k^n(x_k^n) + A_k^n(x_k^n)x_k^l + G_k^n(x_k^n)\omega_k^n \tag{4.6}$$

$$x_{k+1}^l = f_k^l(x_k^n) + A_k^l(x_k^n)x_k^l + G_k^l(x_k^n)\omega_k^l \tag{4.7}$$

$$y_k^{ekf} = h_k(x_k^n) + C_k(x_k^n)x_k^l + \eta_k \tag{4.8}$$

$$y_k^{pf} = h_k(x_k^n) + \epsilon_k \tag{4.9}$$

The model is partitioned into so-called nonlinear (superscript n) and linear (superscript l) states. The dynamics of each of the partitions includes a function of the nonlinear partition ($f_k^*(x_k^n)$), a function of the linear partition $A_k^*(x_k^n)$, and noise w_k^*. The noise enters the system through the noise dynamics matrix $G_k^*(x_k^n)$. (The superscript * simply denotes both the n and l terms). The measurement model (Equation (4.8)) includes both nonlinear ($h_k(x_k^n)$) and linear ($C_k(x_k^n)x_k^l$) terms and Gaussian noise (η_k). In order to minimize notation confusion, the measurements are denoted with superscripts ekf and pf, instead of l and n. This is because there are some measurements (e.g. GPS position) which are *linear* functions of the nonlinear states, and these are treated according to Equation (4.8) (e.g. using the GPS

measurement model presented in Section 4.5). Yet there are other measurements (e.g. RFID) which are very nonlinear functions of the nonlinear states. These are treated according to Equation (4.9). In the JNCPF the horizontal position states are assigned to the nonlinear partition, while all others are assigned to the linear partition. This is because the primary nonlinear measurement requiring particles is the RFID sensor, which is a function of the position states. The vertical position state for pedestrian and ground vehicle cases is already constrained to the digital terrain elevation database (DTED) altitude, so particles are not needed for altitude. The noise models are given by Equations (4.10 - 4.13), and the initial distribution of the linear partition is given by Equation (4.14) and is assumed Gaussian.

$$Q_k = \begin{bmatrix} Q_k^l & Q_k^{ln} \\ (Q_k^{ln})^T & Q_k^n \end{bmatrix} \tag{4.10}$$

$$\omega_k = \begin{bmatrix} \omega_k^l \\ \omega_k^n \end{bmatrix} \sim N(0, Q_k) \tag{4.11}$$

$$\eta_k \sim N(0, R_k) \tag{4.12}$$

$$\epsilon_k \sim f(\epsilon_k^i) \tag{4.13}$$

$$x_0^l \sim N(\bar{x}_0, \bar{P}_0) \tag{4.14}$$

The noise terms ω_k and η_k are assumed to be linear, while ϵ_k can have an arbitrary distribution. The initial distribution of the nonlinear states, however, may have an arbitrary known distribution. The measurement update to the linear partition is described in

Equations (4.15) - (4.18).

$$\hat{x}^l_{k|k} = \hat{x}^l_{k|k-1} + K_k(y_k - h_k - C_k\hat{x}^l_{k|k-1}) \tag{4.15}$$

$$P_{k|k} = P_{k|k-1} - K_k M_k K_k^T \tag{4.16}$$

$$M_k = C_k P_{k|k-1} C_k^T + R_k \tag{4.17}$$

$$K_k = P_{k|k-1} C_k^T M_k^{-1} \tag{4.18}$$

These correspond with the standard Kalman filter update equations. Equations (4.15) - (4.16) give the updates for the linear state and covariance, while Equations (4.17) - (4.18) describe the requisite calculations for the innovation uncertainty matrix M_k and the Kalman gain K_k. The linear partition equations of the filter only differ from a standard Kalman filter in the state time update portion of the algorithm. This is shown in Equations (4.19) - (4.23).

$$\hat{x}^l_{k+1|k} = A^l_k \hat{x}^l_{k|k} + G^l_k (Q^{ln}_k)^T (G^n_k Q^n_k)^{-1} z_k + f^l_k + L_k(z_k - A^n_k \hat{x}^l_{k|k}) \tag{4.19}$$

$$P_{k+1|k} = A^l_k P_{k|k} (A^l_k)^T + G^l_k Q^l_k (G^l_k)^T - L_k N_k L_k^T \tag{4.20}$$

$$N_k = A^n_k P_{k|k} (A^n_k)^T + G^n_k Q^n_k (G^n_k)^T \tag{4.21}$$

$$L_k = A^l_k P_{k|k} (A^n_k)^T N_k^{-1} \tag{4.22}$$

$$z_k = x^n_{k+1} - f^n_k \tag{4.23}$$

The most important new term in Equation (4.19) is the last term ($L_k(z_k - A^n_k \hat{x}^l_{k|k})$). This is the means by which correlation from the nonlinear partition is fed back into the linear partition. The term z_k corresponds to the difference between the nonlinear state change due to nonlinear terms and linear terms. Recall that f^n_k and f^l_k are defined in Equations (4.6) and (4.7). The gain term L_k handles all of the correlation between the linear and nonlinear partition. It also appears as a part of the last term in in Equation (4.22). This term adjusts the linear partition uncertainty to account for the fact that information has been added to

the filter from the nonlinear partition. Without this term, the uncertainty would be too large. Note that it is often the case that the term Q_k^{ln} is zero, which implies that there is no correlation between the process noise of the linear and nonlinear partitions. This is true in this work also, and thus the term $G_k^l(Q_k^{ln})^T(G_k^nQ_k^n)^{-1}z_k$ is zero. The nonlinear partition is updated in a similar manner as the classical particle filter. Equation (4.24) gives the "pf" measurement update for the nonlinear partition.

$$p(y_k^{pf}|X_k^n, Y_{k-1}) = f_{y|x}(y_k^{pf}, \hat{x}_k^n, \Theta) \tag{4.24}$$

An example of such a measurement would be from an RFID sensor. This is a likelihood function with an arbitrary density which is evaluated at the sample point y_k, given the state estimate $\hat{x}_k^n$ and any density parameters Θ (which must be known). This likelihood corresponds to the first term in Equation (3.17), which is used to update the total weights. The nonlinear partition can also be updated using *linear* measurements of the nonlinear states. Equation (4.25) gives the update, which is also how the local linearization aspect of the particle filter described in Section 3.5 is implemented.

$$p(y_k^{ekf}|X_k^n, Y_{k-1}) = N(h_k + C_k\hat{x}_{k|k-1}^l, C_kP_{k|k-1}C_k^T + R_k) \tag{4.25}$$

This is important because particle filters can suffer when used in conjunction with unimodal measurements of the particle states when the measurement uncertainty is very low. GPS is a perfect example of this problem. The GPS accuracy will likely be very small relative to the filter uncertainty after periods with no strong aiding, and the local linearization equations allow the particle filter to properly include them. Note that it is possible, as is the case in the GPS position example, that this measurement is not a function of the linear partition. In that case C_k becomes zero. The time update for the nonlinear partition is given by Equation

(4.26).

$$p(x_{k+1}^n|X_k^n, Y_t) = N(f_k^n + A_k^n \hat{x}_{k|k}^l, A_k^n P_{k|k}(A_k^n)^T + G_k^n Q_k^n (G_k^n)^T) \tag{4.26}$$

Coupling with the linear state vector is accomplished via A_k^n, the linearized state transition matrix. The matrix A_k^n also is used to include the coupling of the particle state uncertainty with the linear uncertainty. This is seen in the term $A_k^n P_{k|k}(A_k^n)^T$. The process noise for the nonlinear state is Gaussian in this work.

This work takes advantage of an innovation on Schon et. al's Rao-Blackwellized particle filter which is discussed in [40]. Leidos has developed a version of the above RBPF in which the division between the linear and nonlinear partions of the state space can be adjusted on the fly according to the availability of highly nonlinear measurements. If all of the measurements which are available at a certain time are suitable for linearization (as in an EKF), then the filter can operate with all of the states in the linear partion. When a measurement becomes available which is more suitable for particles, the filter can adjust and move the appropriate states into the nonlinear partion. This can all be done on the fly.

In the case of the JNCPF presented in this work, this dynamic reconfiguration occurs in two circumstances. The JNCPF begins with the horizontal position states and the mode state in the nonlinear partion, and all other states in the linear partion. Once the mode has been identified, the position states are moved back into the linear partion unless a nonlinear measurement source is available. The mode state is then no longer required and is discarded. If the filter has converged on a mode hypothesis and all of the states are in the linear partition then it will transition the appropriate states back into the nonlinear partion should nonlinear measurements become available. In this work only the horizontal position states and the mode state are ever used in the nonlinear partion.

4.4 Dead Reckoning Models

The JNCPF makes use of different kinematically constrained models for each platform type in order to determine online what platform type the IMU is affixed to. Different platform reference models are used to propagate each individual partical state estimate forward, according to the corresponding kinematic model indicated by the mode state of the particular particle. As the particles propagate forward in time, the motion model which most closely matches the platform motion will produce particle estimates nearest to the measurements. The measurements in turn weight these particles higher than the others and the filter converges on the most likely mode. The foundational equations for all of the reference models are the dynamic equations for position, forward speed, and heading. These are given by Equations (4.27 - 4.29) [40].

$$\dot{r}_{lla} = \begin{bmatrix} \frac{1}{R_M+h} & 0 & 0 \\ 0 & \frac{1}{(R_p+h)cos(\lambda)} & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v_x cos(\psi) \\ v_x sin(\psi) \\ 0 \end{bmatrix} \tag{4.27}$$

$$\dot{v}_x = a_x^b - gsin(\theta) \tag{4.28}$$

$$\dot{\psi} = \frac{\omega_y^b sin(\phi) + \omega_z^b cos(\phi)}{cos(\theta)} \tag{4.29}$$

The vector r_{LLA} is the geodetic position. The scalars R_m and R_p are parameters of the WGS-84 reference ellipsoid. The terms h and λ correspond to geodetic altitude and latitude, respectively. Forward speed is represented by v_x, and heading by ψ. Roll and pitch are given by ϕ and θ. The inertial measurements are given by a^b and ω^b. All of the reference models propagate the position and heading according to Equations (4.27) and (4.29). The ground vehicle and aircraft models propagate the forward speed using Equation (4.28). The pedestrian reference model calculates the forward speed using a step detector [42], as opposed to integrating the estimate of forward acceleration. The primary difference between the ground vehicle and aircraft reference models comes from how the roll and pitch are calculated.

4.4.1 Ground Vehicle Model

While the dead reckoning constraints help reduce the navigation sensitivity to roll and pitch estimation errors, the roll and pitch estimates still play an important roll in constrained navigation. The pitch angle in particular has a strong effect on the forward acceleration computation (Equation (4.28)). The roll angle is far less significant in the ground vehicle dead reckoning case. The roll and pitch angles are computed using the accelerometers instead of the gyros. This is because it is assumed that over time both angles approach zero mean, and that both angles change at a relatively low frequency compared with the gyro signals. Integration of the gyro signals leads to unbounded drift in the attitude estimates, primarily due to the high frequency noise and the sensor biases. Computing the attitude from the estimated gravity vector in the accelerometer signals mitigates this problem, at the expense of potential lag during high dynamic situations. Equations (4.30) and (4.31) show how roll and pitch are calculated from the estimated gravity vector (a^b_{grav}),

$$\phi = \arctan 2(u^b_{down}(2), u^b_{down}(3)) \tag{4.30}$$

$$\theta = \arcsin(-u^b_{down}(1)) \tag{4.31}$$

while Equations (4.32) and (4.33) show how the gravity vector is computed [40].

$$u^b_{down} = \frac{-a^b_{grav}}{\|a^b_{grav}\|} \tag{4.32}$$

$$a^b_{grav} = a^b_{meas} - \begin{bmatrix} \dot{\tilde{v}}_x \\ v_x\dot{\tilde{\psi}}\cos(\phi) \\ -v_x\dot{\tilde{\psi}}\sin(\phi) \end{bmatrix} \tag{4.33}$$

The gravity vector is estimated by removing the centripetal acceleration, which is computed using low pass filtered approximations of the forward acceleration ($\dot{\tilde{v}}_x$) and the heading rate ($\dot{\tilde{\psi}}$). The reference speed and heading estimates are numerically differentiated and low pass

filtered to obtain these approximations. The state transition matrix given in Equation (1.18) is not valid for the dead reckoning ground vehicle model, because the state dynamic equations have changed. The new state transition matrix (in continuous form) is given by Equation (4.34) [40].

$$F_k^{GV} = \begin{bmatrix} F_{11}^{gv} & F_{12}^{gv} & F_{13}^{gv} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{24}^{gv} & F_{25}^{gv} & 0_{3\times3} & 0_{3\times3} & F_{28} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{4.34}$$

The position is purely a function of speed and heading in the ground vehicle model. Therefore the primary coupling with the position error states are with the horizontal velocities (Equation (4.35)) and heading (Equation (4.36)).

$$F_{12}^{gv} = \begin{bmatrix} \frac{V_N \cos(\psi)}{|V|} & \frac{V_E \cos(\psi)}{|V|} & 0 \\ \frac{V_N \sin(\psi)}{|V|} & \frac{V_E \sin(\psi)}{|V|} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.35}$$

$$F_{13}^{gv} = \begin{bmatrix} 0 & 0 & -V_x \sin(\psi) \\ 0 & 0 & V_x \cos(\psi) \\ 0 & 0 & 0 \end{bmatrix} \tag{4.36}$$

The terms in Equation (4.37) correspond to the position error state coupling with the other position errors and are very small. The denominators of these terms are on the order of the

radius of the earth expressed in meters.

$$F_{11}^{gv} = \begin{bmatrix} 0 & 0 & \frac{-V_x \cos(\psi)}{R_\lambda + h} \\ \frac{V_x \sin(\psi)\sin(\lambda)}{(R_\lambda + h)\cos(\lambda)} & 0 & \frac{-V_x \sin(\psi)}{R_\Phi + h} \\ 0 & 0 & 0 \end{bmatrix} \tag{4.37}$$

Note that the horizontal accelerometer biases (Equation (4.38)) and scale factors (Equation (4.39)) enter the system primarily as a function of heading also, adjusted by the pitch angle.

$$F_{24}^{gv} = \begin{bmatrix} \cos(\theta)\cos(\psi) & 0 & 0 \\ \cos(\theta)\sin(\psi) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.38}$$

$$F_{25}^{gv} = \begin{bmatrix} a_x^B \cos(\theta)\cos(\psi) & 0 & 0 \\ a_x^B \cos(\theta)\sin(\psi) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.39}$$

The ground vehicle is assumed to the constrained to the surface of the Earth, therefore the altitude error state does not substantially affect other states. The noise input matrix Equation (1.19) is unchanged.

4.4.2 Pedestrian Model

The pedestrian navigation solution is obtained in much the same way as the ground vehicle cases. Equation (4.27) is used to propagate position, Equation (4.29) is used to propagate heading, and roll and pitch are obtained via Equations (4.30 - 4.33). The primary difference is how the X velocity is calculated. In this work, a simple step detection algorithm is used to determine the speed of the pedestrian based on an assumed stride length [42]. The stride length assumes an average pedestrian height of 5'6", thus the author of this work has never been selected to validate the step detection algorithm. Equation (4.40) gives the state

transition matrix for the pedestrian dead reckoning model.

$$F_k^{PD} = \begin{bmatrix} F_{11} & F_{12} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ F_{21} & F_{22} & 0_{3\times3} & F_{24} & F_{25} & 0_{3\times3} & 0_{3\times3} & F_{28} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{36} & F_{37} & 0_{3\times3} & F_{39} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{44} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{66} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{4.40}$$

The most striking distinction of this model is that the position error is completely decoupled from the heading error. The dynamic model assumes that the pedestrian is walking straight, which of course makes the position dependent on the heading. However, this model is easily violated if the person looks left or right (turning the torso and not only the head) while walking, or steps sideways. Furthermore, the IMU is mounted on the torso of the pedestrian, as opposed to the classical foot mount option for pedestrian navigation. This is in keeping with the spirit of the ASPN program, which discourages such platform specific modifications. Developing a pedestrian navigation model which is suited to the more challenging torso mount is beyond the scope of this work, therefore the simplistic model is used. The philosophy of this decision is that the solution is dominated by the heading readings from a magnetic compass, and that position information is dominated by RFID readers (or GPS if available). Position measurements are not relied upon for observability of the heading, and the solution effectively follows the heading measured by the compass at the speed estimated by the step detector for propagation between position updates.

4.4.3 Aircraft Model

In contrast with the ground vehicle reference model, the aircraft reference model uses both the gyro and accelerometer signals in order to estimate aircraft attitude. In this case the gyros are incorporated because the higher frequency dynamics are more significant, for example in a bank to turn maneuver. Nevertheless it remains that pure integration of the gyros will lead to unbounded drift, which is unacceptable. Therefore an approach is adopted which combines the advantages and frequency characteristics of the accelerometers and gyros. Yoo et al. [41] present a so called complementary filter which leverages the unbiased, low frequency characteristics of the accelerometer derived attitude and the high frequency tracking ability of the gyros. A diagram of their complementary filter is given in Figure 4.2. The two attitude estimates are fused in a manner similar to a classical PI

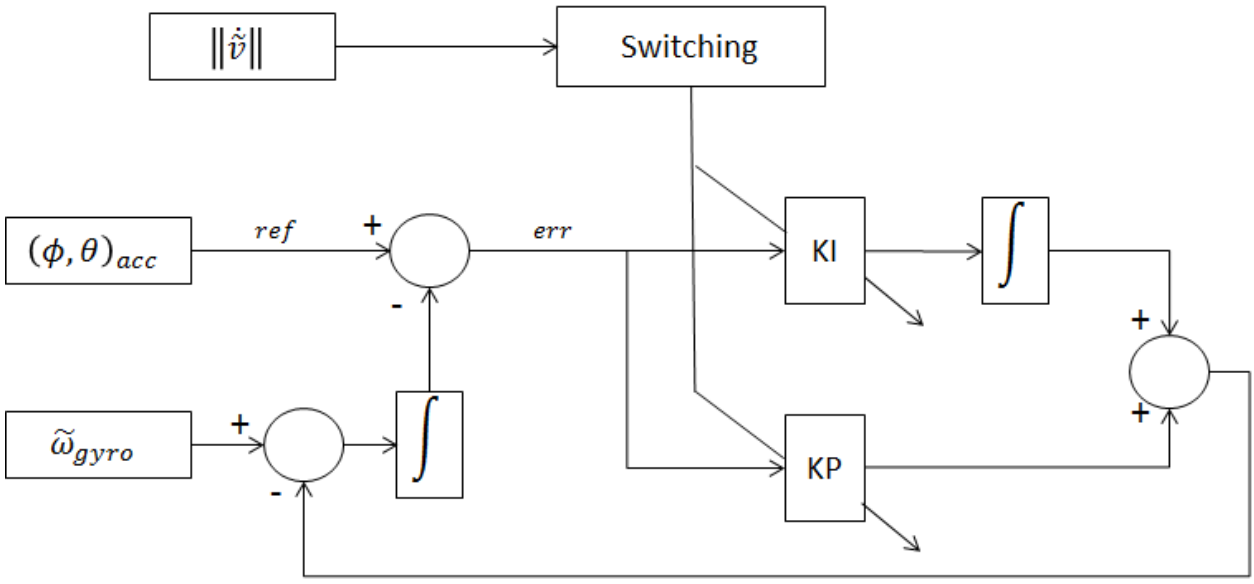


Figure 4.2: Complementary filter for aircraft attitude estimation [41].

controller. The accelerometer based attitude measurement is treated as the reference signal with which the integrated gyros are compared. The error between these is computed and passed through a PI control law, which provides a feedback adjustment to the Euler rate terms. The closed loop result is a low pass filter on the difference between the accelerometer

attitude and the gyro attitude, which represents a trade off between the high frequency estimation of the gyros and the steady state stability of the accelerometers. High gains drive the estimate to the reference signal, allowing little high frequency changes. Low gains allow the gyros to dominate the estimate, resulting in steady state drift. Yoo et al. point out that while it is possible to find satisfactory middle ground for particular circumstances, it is difficult to do so over the variety of realistic CONOPS which an aircraft will experience. Therefore they propose gain scheduling based on the dynamic conditions in three regimes of operation. The norm of the measured acceleration is compared with the magnitude of the gravitational force, and this difference provides the metric used for gain scheduling.

Yoo et al. compute the accelerometer based attitude estimate using Equations (4.30) and (4.31). The accelerometer signals are compensated for centripetal acceleration using the measured gyro signals and an external estimate of velocity. The authors take advantage of apriori knowledge of aerodynamic coefficients to estimate the velocity using an airspeed sensor. In this work the reference pitch angle is adequately approximated using Equation (4.31). The reference roll angle is solved for iteratively by solving Equation (4.41) using the secant method [26].

$$v_x\omega_y \sin(\phi) + v_x\omega_z \cos(\phi) - g\tan(\phi) = 0 \tag{4.41}$$

The state transition matrix for the aircraft dead reckoning model is given by Equations (4.42 - 4.44).

$$F_k^{AC} = \begin{bmatrix} F_{11} & F_{12} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ F_{21} & F_{22} & F_{23}^{ac} & F_{24} & F_{25} & 0_{3\times3} & 0_{3\times3} & F_{28} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & F_{33}^{ac} & 0_{3\times3} & 0_{3\times3} & F_{36} & F_{37} & 0_{3\times3} & F_{39} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{44} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{66} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{4.42}$$

$$F_{23}^{ac} = \begin{bmatrix} 0_{3\times1} & 0_{3\times1} & F_{23}^{(c3)} \end{bmatrix} \tag{4.43}$$

$$F_{33}^{ac} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ F_{33}^{(31)} & F_{33}^{(32)} & 0 \end{bmatrix} \tag{4.44}$$

The primary difference is that the velocity error is no longer dependent on pitch and roll, but rather primarily on heading (notwithstanding inertial sensor errors). Nor are the roll and pitch errors dependent on anything but the gyroscope errors, although the heading is marginally dependent on roll and pitch. All other terms remain the same as the unconstrained state transition matrix.

4.4.4 The Null Hypothesis

The JNCPF operates with mode states for each of the three platform types under consideration. However, a fourth mode possibility is considered for many tests presented in this work. This fourth mode is referred to as the "h_0" mode (for "null hypothesis"), and

corresponds to a completely unconstrained mode state. The particles having the h_0 mode are propagated according to the original inertial navigation equations presented in Section 1.5. The state transition matrix used for uncertainty propagation is also the same described in Section 1.5. Some initial tests do not include the h_0 mode, and only have modes for the three platform types. However, as is shown in Section 5.3, it advantageous to include the h_0 mode when the IMU quality is sufficient. Note that the h_0 mode does apply the zero velocity update (ZVU) measurement if a sensor indicates that this is appropriate.

4.4.5 Other Modeling Considerations

It is important to note here other miscellanious modelling differences that exist between the unconstrained models and the constrained models. First, the pedestrian and ground vehicle models are constrained to the surface of the local DTED map. This is based on the obvious assumption that people and ground vehicles espcially do not leave the surface of the earth. This constraint is applied by setting the altitude of the reference trajectory equal to the DTED height at the prior best estimate of the latitude and longitude. Additionally a pseudomeasurement is created using the DTED altitude and applied to the pedestrian and ground vehicle particle sets. The model for the DTED pseudomeasurement is given by Equations (4.45- 4.48).

$$z^{DTED} = h_{DTED} - \hat{h} + \eta_{DTED} \tag{4.45}$$

$$\hat{z}^{DTED} = \delta\hat{x}_3 \tag{4.46}$$

$$\eta_{DTED} = \mathcal{N}(0, 100) \tag{4.47}$$

$$H^{DTED}_{(:,3)} = [\ \cdots \quad 1 \quad \cdots\ ] \tag{4.48}$$

The model accommodates for errors in the DTED altitude by assuming a 10 m standard deviation in the measurement error. Multistory buildings can complicate this situation in the case of pedestrians, but these situations do not arise in the data used in this work. It

would be reasonable to accommodate this situation by increasing the value of the standard deviation used in the pseudomeasurement for pedestrians. Alternatively this constraint could be heuristically disabled if a barometric altimeter is present.

The DTED pseudomeasurement is only used to weight the particles of the pedestrian set relative to themselves and of the ground vehicle set relative to themselves; it does not affect the overall weighting of the pedestrian and ground vehicle sets with respect to the other modes. This is done by normalizing the weights computed from the DTED relative likelihoods for each particle set before applying them. It adjusts which particles within the pedestrian and ground vehicle sets have the highest weight, but it does not adjust the overall weight of the set. This is because the aircraft and unconstrained particle sets don't receive this measurement, and so normalizing the DTED likelihoods of all of the sets all together would diminish the weight of the pedestrian and ground vehicle sets relative to the aircraft and unconstrained. This would be inaccurate and undesirable. The goal of applying this pseudomeasurement is to constrain the altitudes of the two earth-bound sets to the surface of the earth, thereby making them more accurate (if the true mode is earth-bound of course), and thus giving higher weight to the particles within the sets that fit this constraint. Normalizing the likelihood of this measurement *relative to each individual set* prior to applying it accomplishes this.

Another difference between the unconstrained models and the constrained models is that the constrained models impose limits on the minimum values which are allowed for inertial error statistics, which are used to set the process noise matrix for the filter. Whereas the unconstrained model accepts the values contained in the IMU metadata, the constrained models have preset values for many process noise elements dependent on the platform type. This is because the constrained models implicitly assume that the IMU quality is poor enough to merit using constraints instead of free inertial navigation, reflecting the tradeoff discussed in Chapter 2. It is also important to use larger values of process noise to account for the approximations made in the state propagation model for the constrained cases.

4.5 Measurement Models

The first measurement type considered is the loosely coupled GPS position solution [23], which is the stand alone position solution provided by a GPS receiver (as opposed to raw pseudo-ranges). This sensor signal is modeled as the true position values plus Gaussian noise, although it is possible to augment the filter with a Markov bias state to estimate any time correlated errors in the GPS solution. Equations (4.49 - 4.51) gives the GPS measurement model.

$$z^{GPS} = \left(\begin{bmatrix} \lambda \\ \Phi \\ h \end{bmatrix}_{GPS} - \begin{bmatrix} \lambda \\ \Phi \\ h \end{bmatrix}_{ref} \right) \cdot \begin{bmatrix} R_\lambda \\ \cos(\lambda) R_\Phi \\ -1 \end{bmatrix} + \eta_{GPS} \tag{4.49}$$

$$\eta_{GPS} = \mathcal{N}(0, R_{GPS}) \tag{4.50}$$

$$R_{GPS} = \left[\eta_{GPS} \eta_{GPS}^T \right] \tag{4.51}$$

Note that this includes the reference trajectory in the model, because the filter states are the errors in the inertial solution. By subtracting the reference position from the GPS signal, the measurement model is parametrized as a measurement of the inertial position error, which is a filter state. The predicted measurement and measurement matrix (assuming no additional state augmentation) are given in Equations (4.52) and (4.53).

$$\hat{z}^{GPS} = \delta \hat{x}_k \tag{4.52}$$

$$H^{GPS}_{(:,1:3)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.53}$$

For simplicity the subscript denoting the current time (k) is dropped. Additionally, only the columns which are non-zero are shown for the Jacobian H of each measurement model.

The columns (which correspond to the state vector) are given as subscripts using MATLAB matrix indexing notation.

The GPS velocity solution model is given in Equations (4.54 - 4.58).

$$z^v = \underline{v}_{NED} - \underline{v}^{ref}_{NED} + \eta_v \tag{4.54}$$

$$\hat{z}^v = \delta\hat{x}_{4:6} \tag{4.55}$$

$$\eta_v = \mathcal{N}(0, R_v) \tag{4.56}$$

$$R_v = [\eta_v \eta_v^T] \tag{4.57}$$

$$H^v_{(:,4:6)} = [\ \cdots \quad I_{3x3} \quad \cdots\ ] \tag{4.58}$$

The GPS provides measurements of NED frame velocities which are modeled as having zero mean Gaussian errors. The measurement is computed as the difference between the GPS velocity solution and the reference estimate of the NED velocity. This gives a measurement of the velocity error plus noise. The predicted measurement is given as the *apriori* estimate of the NED velocity error. This gives a measurement matrix of identity, in the rows and columns corresponding to the velocity error states, and zeros elsewhere.

Instead of using the position and velocity solutions from the GPS receiver, the navigation filter can process the raw pseudo-range and delta-range measurements directly as measurements in the filter. This is commonly referred to as "close coupling" or "tight coupling." The GPS pseudorange model is given in Equations (4.59-4.64), where the superscript

SV_i indicates satellite i, and the superscript u indicates the user [23].

$$z^{pr_i} = \left\| r_E^{SV_i} - r_E^u \right\| + c\delta t_u + \eta_{pr} \tag{4.59}$$

$$\hat{z}^{pr_i} = \left\| r_E^{SV_i} - \hat{r}_E^u \right\| + c\delta \hat{t}_u \tag{4.60}$$

$$H_{(1:3)}^{pr} = \begin{bmatrix} H^{SV_1} \\ \vdots \\ H^{SV_n} \end{bmatrix} \tag{4.61}$$

$$H^{SV_i} = -R_E^N \frac{r_e^{SV_i} - \hat{r}_e^u}{\left\| r_e^{SV_i} - \hat{r}_e^u \right\|} \tag{4.62}$$

$$\eta_{pr} = \mathcal{N}(0, R_{pr}) \tag{4.63}$$

$$R_{pr} = \left[\eta_{pr} \eta_{pr}^T \right] \tag{4.64}$$

The scalar c is the speed of light, which is multiplied by the estimated clock bias and clock drift of the user receiver. The pseudorange model used here assumes that the atmospheric effects have been accounted for using atmospheric models. The ionospheric effects can be eliminated if L1 and L2 frequencies are available. The satellite clock errors are also assumed corrected using the ephemeris information. The predicted measurement, Equation (4.60), is computed using the reported satellite position and the estimated user position and clock bias. The measurement matrix is composed simply of the unit vectors from each SV to the user, expressed in the NED frame (as the error states are kept in the NED frame). It is assumed that the remaining errors in the pseudorange measurement are Gaussian. This includes errors in the atmospheric corrections and SV ephemeris, in addition to remaining noise on the signal. While it is true that SV related errors from the ephemeris will be time correlated by nature, and thus violate the assumptions of the Kalman filter, it is found that this does not hinder the results in this work. Note that H^{SV_i} in Equation (4.62) is a 3×1 vector.

The GPS delta range model is given by Equations (4.65) - (4.72) [23].

$$z^{dr_i} = s^{SV_i}\lambda_{(L1,L2)} + \delta\dot{t}_u \tag{4.65}$$

$$\hat{z}^{dr_i} = \hat{v}_{u,E}^{SV_i} \cdot H^{SV_i} + \delta\dot{\hat{t}}_u \tag{4.66}$$

$$H_{(1:3)}^{dr} = \begin{bmatrix} H_{dr}^{SV_i} & H^{SV_i} \\ \vdots & \vdots \\ H_{dr}^{SV_n} & H^{SV_n} \end{bmatrix} \tag{4.67}$$

$$H_{dr}^{SV_i}(1) = -v_{u,N}^{SV}(1)\frac{\left(\left(r_{u,N}^{SV}(2)\right)^2 + \left(r_{u,N}^{SV}(3)\right)^2\right)}{\left(\hat{z}^{pr_i}\right)^3} \tag{4.68}$$

$$H_{dr}^{SV_i}(2) = -v_{u,N}^{SV}(2)\frac{\left(\left(r_{u,N}^{SV}(1)\right)^2 + \left(r_{u,N}^{SV}(3)\right)^2\right)}{\left(\hat{z}^{pr_i}\right)^3} \tag{4.69}$$

$$H_{dr}^{SV_i}(3) = -v_{u,N}^{SV}(3)\frac{\left(\left(r_{u,N}^{SV}(1)\right)^2 + \left(r_{u,N}^{SV}(2)\right)^2\right)}{\left(\hat{z}^{pr_i}\right)^3} \tag{4.70}$$

$$\eta_{dr} = \mathcal{N}(0, R_{dr}) \tag{4.71}$$

$$R_{dr} = \left[\eta_{dr}\eta_{dr}^T\right] \tag{4.72}$$

The measured Doppler shift between the user and the satellite is s, and $\lambda_{(L1,L2)}$ corresponds to either the $L1$ or $L2$ frequency, whichever carrier frequency is being used. The receiver clock drift is given by $c\delta\dot{\hat{t}}$. Equation (4.65), like Equation (4.59), assumes that the satellite clock error has been corrected using the ephemeris information. The predicted delta range measurement is computed using the satellite velocity and the estimated receiver velocity and clock drift rate. Note that $H_{dr}^{SV_n}$ in Equations (4.67-4.70) is a 3×1 vector.

The measurement model for the inclinometer is given by Equations (4.74) - (4.82).

$$z^{(\phi,\theta)} = \begin{bmatrix} \phi \\ \theta \end{bmatrix}_{meas} + \eta_{(\phi,\theta)} \tag{4.73}$$

$$\hat{z}^{(\phi,\theta)} = \begin{bmatrix} \hat{\phi} \\ \hat{\theta} \end{bmatrix} \tag{4.74}$$

$$H^{(\phi,\theta)}_{(:,7:8)} = \begin{bmatrix} H^{(\phi,\theta)}_{(1,7)} & H^{(\phi,\theta)}_{(1,8)} \\ H^{(\phi,\theta)}_{(2,7)} & H^{(\phi,\theta)}_{(2,8)} \end{bmatrix} \tag{4.75}$$

$$\eta_{(\phi,\theta)} = \mathcal{N}(0, R_{(\phi,\theta)}) \tag{4.76}$$

$$R_{(\phi,\theta)} = \left[\eta_{(\phi,\theta)}\eta^T_{(\phi,\theta)}\right] \tag{4.77}$$

$$c = \left(1 + \left(\frac{-R^B_{N(2,1)}\delta\hat{x}_8 + R^B_{N(2,2)}\delta\hat{x}_7 + R^B_{N(2,3)}}{-R^B_{N(3,1)}\delta\hat{x}_8 + R^B_{N(3,2)}\delta\hat{x}_7 + R^B_{N(3,3)}}\right)^2\right) \tag{4.78}$$

$$H^{(\phi,\theta)}_{(1,7)} = c\frac{\left(R^B_{N(2,1)}R^B_{N(3,2)} - R^B_{N(2,2)}R^B_{N(3,1)}\right)\delta\hat{x}_8 + \left(R^B_{N(2,2)}R^B_{N(3,3)} - R^B_{N(2,3)}R^B_{N(3,2)}\right)}{\left(R^B_{N(3,2)}\delta\hat{x}_7 - R^B_{N(3,1)}\delta\hat{x}_8 + R^B_{N(3,3)}\right)^2} \tag{4.79}$$

$$H^{(\phi,\theta)}_{(1,8)} = -c\frac{\left(R^B_{N(2,1)}R^B_{N(3,2)} - R^B_{N(2,2)}R^B_{N(3,1)}\right)\delta\hat{x}_7 + \left(R^B_{N(2,1)}R^B_{N(3,3)} - R^B_{N(2,3)}R^B_{N(3,1)}\right)}{\left(-R^B_{N(3,2)}\delta\hat{x}_7 + R^B_{N(3,1)}\delta\hat{x}_8 - R^B_{N(3,3)}\right)^2} \tag{4.80}$$

$$H^{(\phi,\theta)}_{(2,7)} = -\frac{R^B_{N(1,2)}}{\left(1 - \left(R^B_{N(1,2)}\delta\hat{x}_7 - R^B_{N(1,1)}\delta\hat{x}_8 + R^B_{N(1,3)}\right)^2\right)^{0.5}} \tag{4.81}$$

$$H^{(\phi,\theta)}_{(2,8)} = \frac{R^B_{N(1,1)}}{\left(1 - \left(R^B_{N(1,2)}\delta\hat{x}_7 - R^B_{N(1,1)}\delta\hat{x}_8 + R^B_{N(1,3)}\right)^2\right)^{0.5}} \tag{4.82}$$

The inclinometer provides measurements of roll and pitch via internal processing involving accelerometers. The roll and pitch measurements are intended for low dynamic situations due to their dependence on measuring the gravitational force. Therefore the roll and pitch measurements that the inclinometer produces are less accurate during high dynamic motion.

In order to deal with this, the measurement uncertainty variable $R_{(\phi,\theta)}$ is treated as time varying and is conditional upon a moving average of the pitch measurement. If the moving average is high, then the uncertainty value is set to a high value which renders it almost invalid. If the moving average is low, then the filter uses a nominal tuning value obtained via extensive testing across multiple ASPN data sets. The moving average of the measured pitch value is chosen for this purpose, instead of the filter estimate of pitch, in order to eliminate any undesirable feedback in this process. The measurement Jacobian is a function of the estimated rotation angles. It is given in Equations (4.78-4.82) [40].

The measurement model for the barometric altimeter is given by Equations (4.83) - (4.91) [23].

$$z^h = -\left(\left(1+\kappa_h\right)\left(h_{meas}\right) - h_{ref}\right) + b_h + \eta_h + \gamma_h \tag{4.83}$$

$$\hat{z}^h = \delta\hat{x}_3 + \hat{b}_h + \hat{\kappa}_h\hat{h} \tag{4.84}$$

$$\dot{b}_h = -\frac{1}{\tau_h}b_h + \eta_{gmh} \tag{4.85}$$

$$\gamma_h \sim \frac{1}{2}\mathcal{U}(-s_\gamma, s_\gamma) \tag{4.86}$$

$$\eta_h = \mathcal{N}(0, R_h) \tag{4.87}$$

$$R_h = \left[\eta_h\eta_h^T\right] \tag{4.88}$$

$$\eta_{gmh} = \mathcal{N}(0, R_{gmh}) \tag{4.89}$$

$$R_{gmh} = \left[\eta_{gmh}\eta_{gmh}^T\right] \tag{4.90}$$

$$H^h_{(:,[3,i_1,i_2])} = \begin{bmatrix} 1+\hat{\kappa}_h & \cdots & 1 & \cdots & \hat{h} \end{bmatrix} \tag{4.91}$$

$$\tag{4.92}$$

The measurement is computed as the difference between the measured altitude and the reference altitude. The measured altitude is modeled as having a bias b_h, scale factor κ_h, Gaussian noise η_h, and uniform noise γ_h. The bias and scale factor are added as augmented states in the filter and are estimated. The uniform noise violates the Kalman filter assumptions,

but is most simply dealt with by increasing the value of R_h used by the filter. It is worth considering the use of particles for the altitude state to better handle the non-Gaussian error model of the altimeter. However, it is found in experience that this would be an unnecessary complication as increasing the measurement noise has proven sufficient.

The measurement model for the magnetic compass is given by Equations (4.93) - (4.100) [23].

$$z^{\psi} = \psi_{meas} - \psi_{ref} + b_{\psi} + \eta_{\psi} \tag{4.93}$$

$$\hat{z}^{\psi} = \delta\hat{x}_9 + \hat{b}_{\psi} \tag{4.94}$$

$$\dot{b}_{\psi} = -\frac{1}{\tau_{\psi}} b_{\psi} + \eta_{gm\psi} \tag{4.95}$$

$$\eta_{\psi} = \mathcal{N}(0, R_{\psi}) \tag{4.96}$$

$$R_{\psi} = \left[\eta_{\psi}\eta_{\psi}^{T}\right] \tag{4.97}$$

$$\eta_{gm\psi} = \mathcal{N}(0, R_{gm\psi}) \tag{4.98}$$

$$R_{gm\psi} = \left[\eta_{gm\psi}\eta_{gm\psi}^{T}\right] \tag{4.99}$$

$$H^{\psi}_{(:,[9,i_1])} = \begin{bmatrix} \cdots & 1 & \cdots & 1 \end{bmatrix} \tag{4.100}$$

It is simpler than the altimeter, and only contains a Markov bias term as an additional element. There are no non Gaussian noise terms or scale factors.

The measurement model for the odometer is given by Equations (4.101) - (4.105).

$$z^{\Delta r} = \frac{\Delta r}{\Delta t}\left(1 + \kappa_{\Delta r}\right) + \eta_{\Delta r} \tag{4.101}$$

$$\hat{z}^{\Delta r} = \left\| \begin{bmatrix} \hat{V}_N & \hat{V}_E \end{bmatrix} \right\| \tag{4.102}$$

$$\eta_{\Delta r} \sim \mathcal{U}(-s_{\Delta r}, s_{\Delta r}) \tag{4.103}$$

$$R_{\Delta r} \approx \left[\eta_{\Delta r}\eta_{\Delta r}^{T}\right] \tag{4.104}$$

$$H^{\Delta r}_{(:,[4:5,i_1])} = \begin{bmatrix} \cdots & \frac{(1+\hat{\kappa})\hat{V}_N}{\hat{z}^{\Delta r}} & \frac{(1+\hat{\kappa})\hat{V}_E}{\hat{z}^{\Delta r}} & \cdots & \hat{z}^{\Delta r} \end{bmatrix} \tag{4.105}$$

The measurement provides a delta position measurement Δr with an associated time interval Δt. The delta position measurement is modeled with a scale factor $\kappa_{\Delta r}$, and a uniform noise source $\eta_{\Delta r}$. The predicted measurement is formed using the filter speed estimate given in Equation (4.102). This assumes that the velocity is constant over the time interval Δt, which is found to be an acceptable approximation in practice. The measurement noise model includes a non Gaussian noise source, which has a uniform distribution, but this is still approximated with a Gaussian model inside the filter (Equation (4.104)). The value of R is simply set large enough to cover the range of the uniform variable. The Jacobian matrix of the measurement model is given in Equation (4.105) [40].

The measurement model for the camera is given by Equations (4.106) - (4.110) [40].

$$z^{\dot{\psi}} = \dot{\psi}_{meas} + \eta_{\dot{\psi}} \tag{4.106}$$

$$\hat{z}^{\dot{\psi}} = \hat{\dot{\psi}} + \delta\hat{x}_{18} + \delta\hat{x}_{27} \tag{4.107}$$

$$\eta_{\dot{\psi}} = \mathcal{N}(0, R_{\dot{\psi}}) \tag{4.108}$$

$$R_{\dot{\psi}} = \left[\eta_{\dot{\psi}}\eta_{\dot{\psi}}^T\right] \tag{4.109}$$

$$H^{\dot{\psi}}_{(:,[18,27])} = \begin{bmatrix} \cdots & 1 & \cdots & 1 \end{bmatrix} \tag{4.110}$$

The camera is used for visual odometry to provide a measurement of the change in rotation from frame to frame. The sample rate of the camera is used to convert this into a rate measurement, and only the yaw rate is used. This is compared with the estimated yaw rate in order to estimate the yaw rate gyro bias (both static and Gauss-Markov). The dynamic model for the gyro bias estimate is given by Equation (1.13) in Section 1.5.

The measurement model for the RFID is a nonlinear function of the magnitude of the three dimensional distance to the reported beacon location from the estimated user position. The RFID measurement is a binary signal which, if detected, simply means the user is within the reported range of the RFID beacon. A simple weighting function would assign zero weight to all of the particles outside of the reported range, and a weight of one to

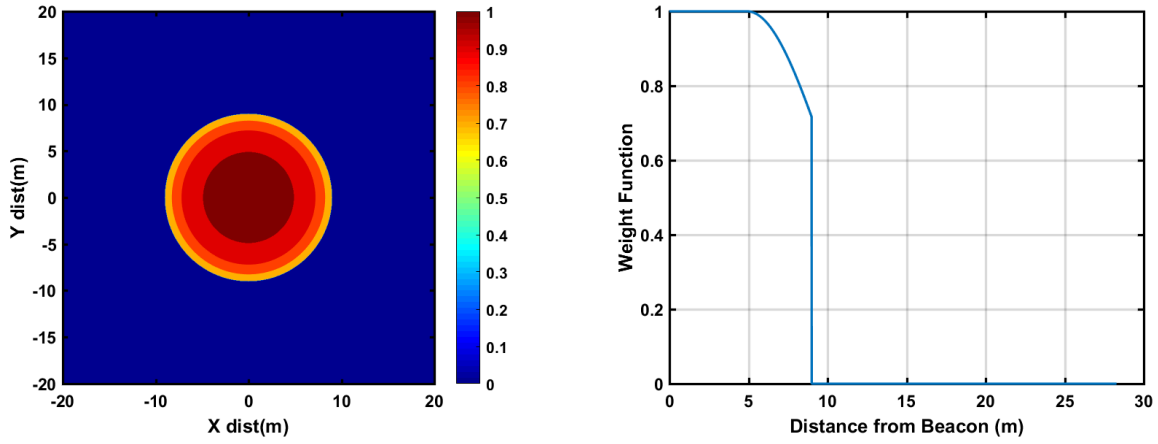
all within range (effectively unchanging these). However it was experimentally determined that it is better to allow for some error in the signal and to gently taper off the weights for particles outside of the RFID range. This is done via a truncated normal distribution. The weighting function is given by Equations (4.111 - 4.113).

$$r_{zone} = (1 + 0.1)(r_{RFID} + \sigma_{RFID}) \tag{4.111}$$

$$p(y_{RFID}|X_k^n, Y_{k-1}) = \begin{cases} 1, & \hat{r} \leq r_{zone} \\ \mathcal{N}(\hat{r}|r_{zone}, 5), & r_{zone} < \hat{r} \leq r_{max} \\ 0 & r_{max} < \hat{r} \end{cases} \tag{4.112}$$

$$r_{max} = 2.5 r_{RFID} \tag{4.113}$$

The potential error sources in the signal could come from anything that interferes with the RF signal propagation or timing latency. The RFID measurement is processed as a nonlinear measurement type according to Equation (4.112), which is a specific form of Equation (4.24). Figures 4.3a and 4.3b show a plot of the weighting function. Note that the range shown is $3.6m$, which is representative of the RFID beacons used in this work. Many of the



(a) (b)

Figure 4.3: RFID Weighting Visualization (range = 3.6 m, accuracy = 0.5 m 1σ) (a). 2D View (b). RFID Weighting as a Function of Distance from Beacon.

measurement models discussed in this section include augmented states for estimating errors on the sensors inside the filter. Below is a list of all of the states which are estimated for each sensor:

- GPS pseudoranges - clock bias
- GPS delta ranges - clock drift
- Barometric altimeter - bias, scale factor
- Magnetic compass - bias
- Odometer - scale factor

4.6 Conclusion

The JNCPF functions as a Rao-Blackwellized Particle Filter which incorporates regularization when resampling and uses locally linearized proposal distributions. Additionally, the nonlinear partition of the filter includes the discrete mode state m, which represents the unknown platform type. This discrete state makes the nonlinear partition a type of multiple model particle filter which is most similar to the JTC particle filter presented by Gordon et al. [21] and described in Section 3.6. This is because the number of particles for each mode is fixed (until the mode dies off), in order to mitigate the problem of "mode" degeneracy. One iteration of the JNCPF is given in Algorithm 21. First, at each time step, the reference trajectory for each mode is propagated according to the respective platform model type. Each model is subject to different constraints depending the dynamics particular to each, as described in Section 4.4. Next, all of the measurements available at the current time step are processed. The vector z_k, which is the measurement vector at the current time step, is computed according to the models given in Section 4.5. In order to apply the measurement update for both the linear and nonlinear states the predicted measurement must be computed in addition to the linearized measurement model H_k. Both the predicted measurement and the linearization are calculated for each particle and the Kalman update is applied. The

Algorithm 10 Single Iteration of JNCPF

At time t_k:
for $m \in M$ **do**
  Propagate mode conditioned reference trajectory to current time step:
  $x_k^{ref,m} = f(x_{k-1}^{ref,m}, m, a_{k-1}^B, \omega_{k-1}^B)$
end for
Process current received measurements z_k.
for all particles x_i **do**
  Compute measurement Jacobian H_k and predicted measurement.
  Kalman filter measurement update.
  Particle weighting update.
end for
for $m \in M$ **do**
  Calculate total probability of mode m (4.114).
  Particle resampling if needed (Algorithm 1).
end for
Compute state estimates.
Filter reset (if needed).
for all particles x_i **do**
  Propagate nonlinear error state vector to next time step.
  Propagate linear error state vector to next time step.
end for

weights of the nonlinear states are also updated with the measurement information using Equations (4.15), (4.24), and (4.25). All measurement information available at the current time step has been incorporated at this point in the algorithm. The probability of each mode is now evaluated as the sum of the weights of the individual particles in each mode set:

$$p(m^i|\mathbf{Y}_k) = \sum_{j=1}^{N_i} w_k^{i,j} \tag{4.114}$$

Next the effective sample size of each mode set is calculated according to Equation (3.21). This is done for each mode set, so N_p is the number of particles in each mode instead of the total number of particles. Any mode which fails the degeneracy check is then resampled. It is important that each mode set is resampled independently, as in Gordon et al. [21], to avoid sample impoverishment of unlikely modes. At this point the best estimate of the navigation state is computed from the reference and error state estimates, where the most likely mode

at the current time is selected as the best estimate. If the enough time has passed since the previous reset, then a reset is performed as described in Section 4.2. The reference trajectory is reset to the value of the current state estimate and the error state is reset to zero. Note that each mode specific reference trajectory is reset to the current state estimate for that particular mode, not the overall best estimate at the current time (which may be from a different mode). There is no mixing of the modes during the filter reset. Finally both the linear and nonlinear error states must be propagated forward to the next time step using the dynamic error models for each mode. The propagation equations for the linear state are given by Equations (4.19-4.23), and the nonlinear propagation equations are given by Equations (4.26). The dynamic models used in these equations, specific to each platform type, are those given in Section 4.4.

Chapter 5

Experimental Validation

5.1 Introduction

The performance of the MMPF is tested against several data sets collected on three different platforms: pedestrian, ground vehicle, and aircraft. These data sets were collected in various locations as part of the ASPN program and contain recordings of inertial measurement units and multiple other sensors in each case, in addition to ground truth. Case 17 was recorded in Athens, Ohio on the campus of Ohio University and includes data from three IMU's, an odometer, a barometric altimeter, a magnetic compass, GPS, and an inclinometer. Case 19 was recorded over Athens, Ohio, using a Douglas DC-3 belonging to Ohio University. The DC-3 is used by the engineering department as a cost efficient and flexible test bed for various programs. For the ASPN program, the aircraft was equipped with 3 IMU's, a barometric altimeter, GPS, an airspeed sensor, two terrain referenced altimeters, an inclinometer, and a magnetic compass. Case 11 is a pedestrian case where the dismount is equipped with an HG1700 IMU, a GPS receiver, a barometric altimeter, and an RFID reader. Case 10 is another pedestrian case with an Intersense IMU (MEMS grade), camera, and an inclinometer. The pedestrian data sets were recorded at facilities located at Wright-Patterson Air Force Base in Dayton, Ohio. These data sets include both indoor and outdoor scenarios. Finally, case 216 is an aircraft case (beginning from prior to takeoff) recorded at during a field demonstration at Ohio University on the DC-3. Table 5.1 provides a comparison of the sensors used in all 7 cases examined in this chapter.

Two IMU's were recorded for both cases 17 and 19: a Honeywell HG1700 AG58 and a Systron Donner MMQ-50. The former qualifies as a tactical grade IMU based on common

Table 5.1: Case sensor configurations.

Case:	17Bj	17Bj*	19Bj	11A	17A	19A	10A	216A
HG1700 IMU				X	X	X		X
MMQ IMU	X	X	X					
Intersense IMU							X	
Magnetic Compass	X	X	X					X
Odometer	X	X						
GPS Position	X	60s			X			X
GPS Velocity								X
GPS Pseudorange				97s		X		
Altimeter			X	Xx				X
Camera							X	
Air Data Computer			X					
Inclinometer			X				X	
RFID				X				

criteria, while the latter could be considered a high quality MEMS grade IMU [18]. Table 5.2 gives the parameters for each of these.

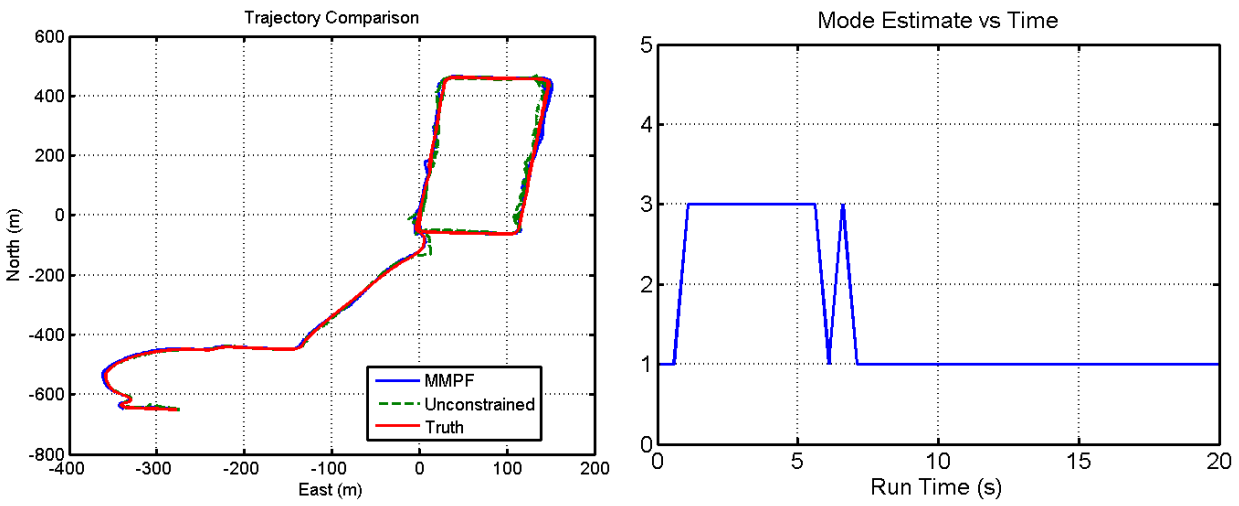
Table 5.2: IMU Performance Characteristics

	HG1700	MMQ50	Navigation [18]	Tactical [18]
Gyroscopes				
Bias Repeatability (deg/hr)	$1(1\sigma)$	100 (1σ)		
Bias Stability (deg/hr)	$1(1\sigma)$	$\leq$15	0.005 - 0.001	1 - 10
Angle Random Walk $(deg/\sqrt{hr})$	$\leq$0.125	0.3		
Accelerometers				
Bias Repeatability (m-g)	$1(1\sigma)$	$2.5(1\sigma)$		
Bias Stability (m-g)	$1(1\sigma)$	$\leq$3	5 - 10 ug	200 - 500 ug
Velocity Random Walk $((m/s)/\sqrt{hr})$	$\leq$0.22	$0.5mg/\sqrt{Hz}$		

5.2 Adaptively Constrained Navigation

5.2.1 Ground Vehicle: Case 17

First, the case with the aided MMQ-50 IMU is considered. Here the IMU is aided with GPS, an odometer, and a magnetic compass. The odometer is treated as a generic speed measurement. Figures 5.1 - 5.4 show the classification and navigation results for this test.



(a) (b)

Figure 5.1: Case 17Bj MMPF and nominal trajectory estimates (a). MMPF mode estimate (b). Mode 1 (ground vehicle) is the correct mode.

The trajectory is plotted in meters in North and East Position in Figure 5.1a. Note that the vehicle begins at point $(0, 0)$ and travels counter clockwise twice before traveling down the southward and final leg. The two trips around the loop take 10 minutes in all. The mode estimate is shown in Figure 5.1b, where it is seen that the mode estimate converges to the proper platform at just after 5 seconds. The speed measurement is the primary discriminator in classification at this point. The platform is fully identified before the first GPS measurement even arrives, which is just after 10 s. Recall that once classification is achieved, the filter shuts down the classification portion of the algorithm. Thus the mode can no longer change and the figure plot is zoomed in to the initial time period. Figures 5.2 and 5.3 show the comparison between the constrained and unconstrained version of the filter in the horizontal and vertical channels, respectively. In this case the presence of strong aiding from position, speed, and heading measurements keeps the unconstrained solution in check very well. The constraints do not add significant accuracy. Nevertheless, the platform is identified correctly and the accuracy is improved via additional constraints.

While the model identification and constraints didn't add much accuracy to the very strongly aided cases, it is important to consider the scenario wherein GPS is not available

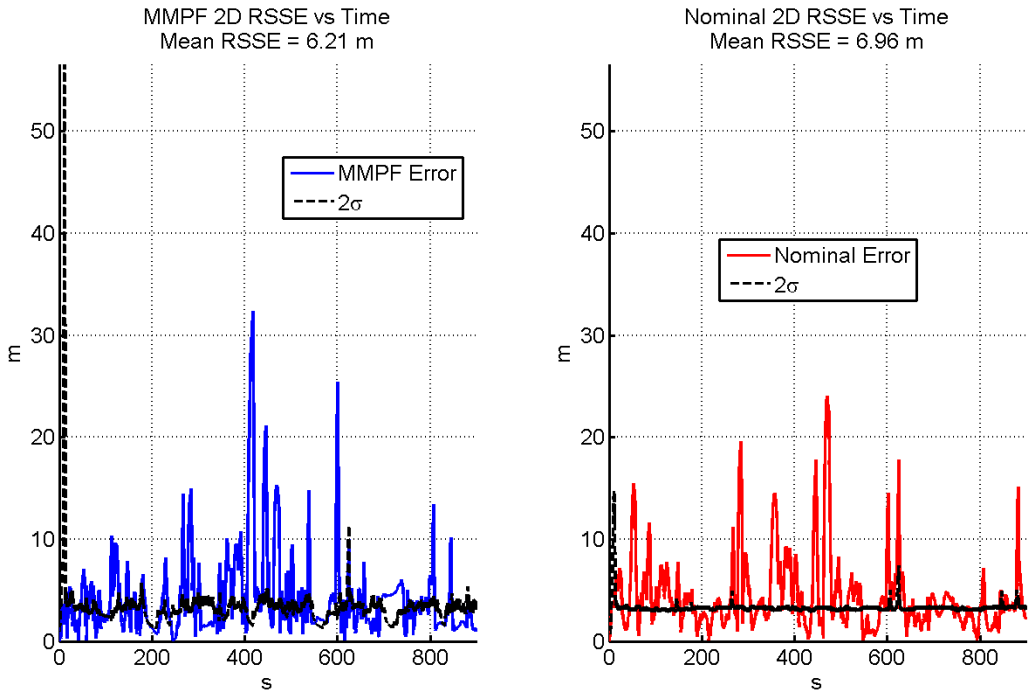


Figure 5.2: Case 17Bj horizontal navigation error comparison, MMPF vs. nominal.

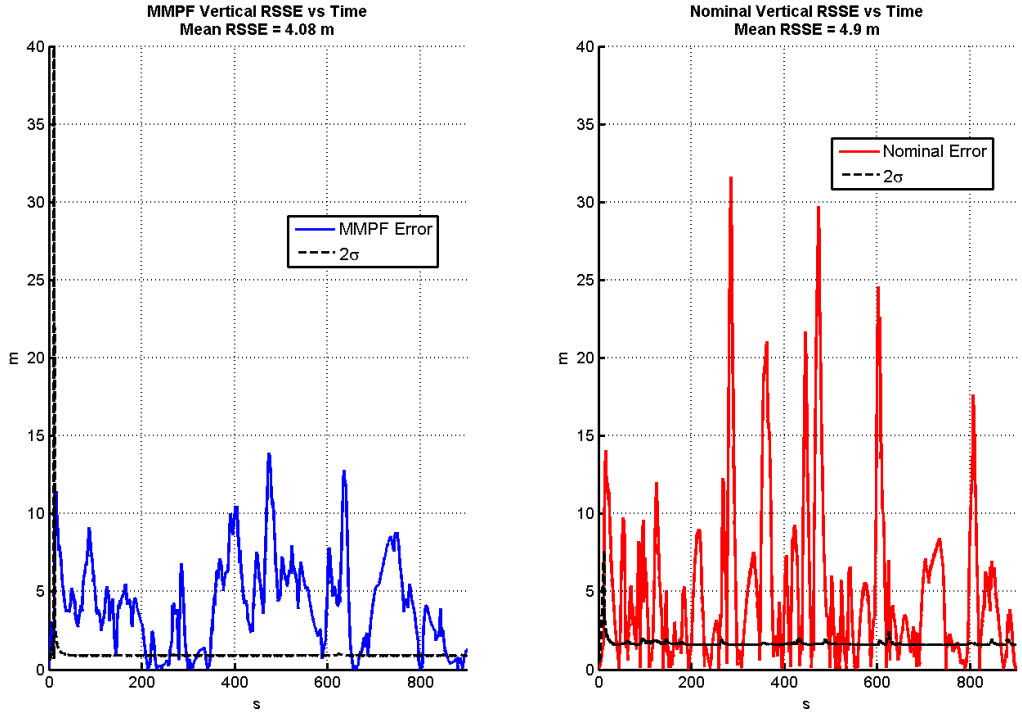


Figure 5.3: Case 17Bj vertical navigation error comparison, MMPF vs. nominal.

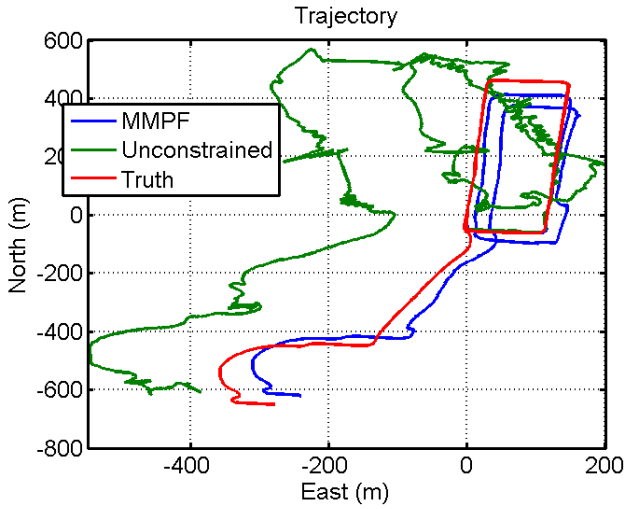


Figure 5.4: Case 17Bj MMPF and nominal trajectory estimates. GPS is turned off after 60 seconds.

after initialization. This may occur due to jamming or urban canyon environments. The following comparison considers the same scenario as before, except that GPS is turned off after the initial 60 seconds of operation. Figure 5.4 shows the trajectory comparison between the MMPF and the nominal unconstrained filter. Figure 5.5 shows the horizontal errors and uncertainties for each filter. As before, the two filters perform very similarly while GPS is available. However once GPS becomes unavailable the benefits of the vehicle constraints become apparent. The drift from the pure inertial solution cannot be sufficiently contained by the compass and odometer alone, as the unconstrained trajectory shows. The ground vehicle constraints are adequate for this purpose. The primary errors in the constrained MMPF solution are from heading drift (within the bounds of the compass error), and speed magnitude drift (within the bounds of the odometer accuracy). Figure 5.5 shows that both the predicted error and the actual error from the nominal solution drift over time, confirming what was stated above. There is inevitable drift in the MMPF also, but the drift rate is much reduced by comparison.

Figure 5.6 shows the altitude error comparison for the two filters. The primary difference here is that because the MMPF has identified the vehicle as a ground vehicle, then it is able to constrain the altitude to DTED. The nominal filter has no knowledge of the vehicle

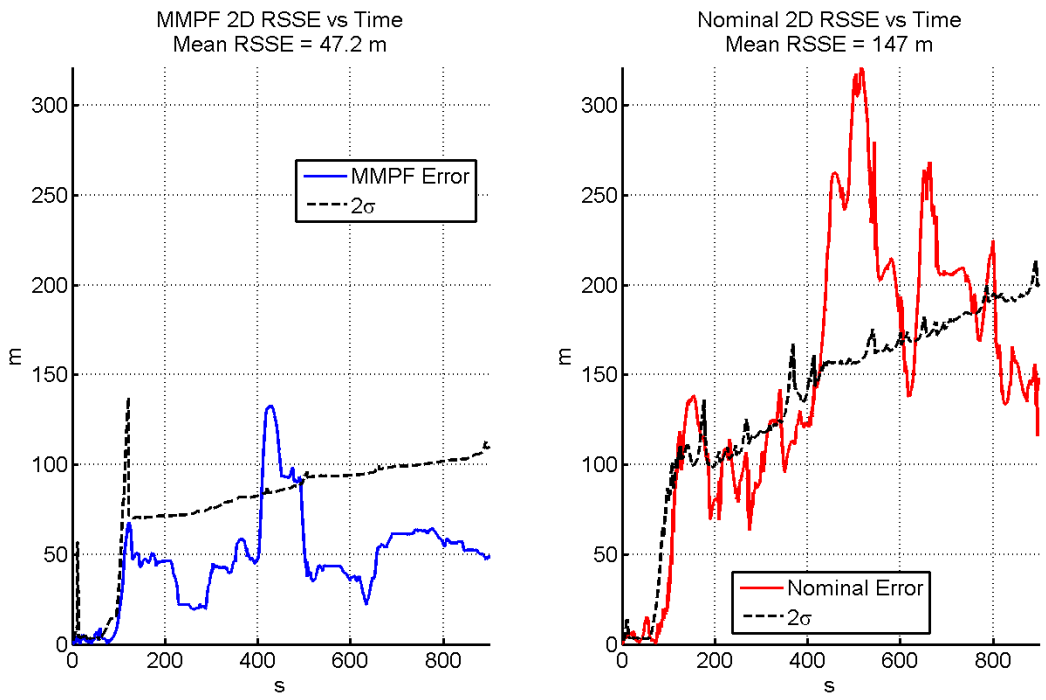


Figure 5.5: Case 17Bj horizontal error comparison, MMPF vs. nominal. GPS is turned off after 60 seconds.

platform and cannot make this assumption. With no other aiding information in the vertical channel, the altitude runs away. It is very likely that adding an altimeter would correct this problem, and that both filters would have similar performance in the vertical channel in this case. This is exactly what is seen in the previous case in Figure 5.3, where GPS measurements (including the vertical channel) are available the entire time. The vertical channel performance of the MMPF and the nominal filter are very similar.

5.2.2 Aircraft: Case 19

Case 19 is an aircraft case having the MMQ IMU, barometric altimeter, magnetic compass, inclinometer, and wind speed sensor. Figure 5.7a shows the trajectories for the constrained and unconstrained cases. Figure 5.7b shows that the aircraft is properly identified immediately. This is due to the barometric altimeter measurements. The pedestrian and ground vehicle particle sets are constrained to the DTED surface, and it is then trivial for the

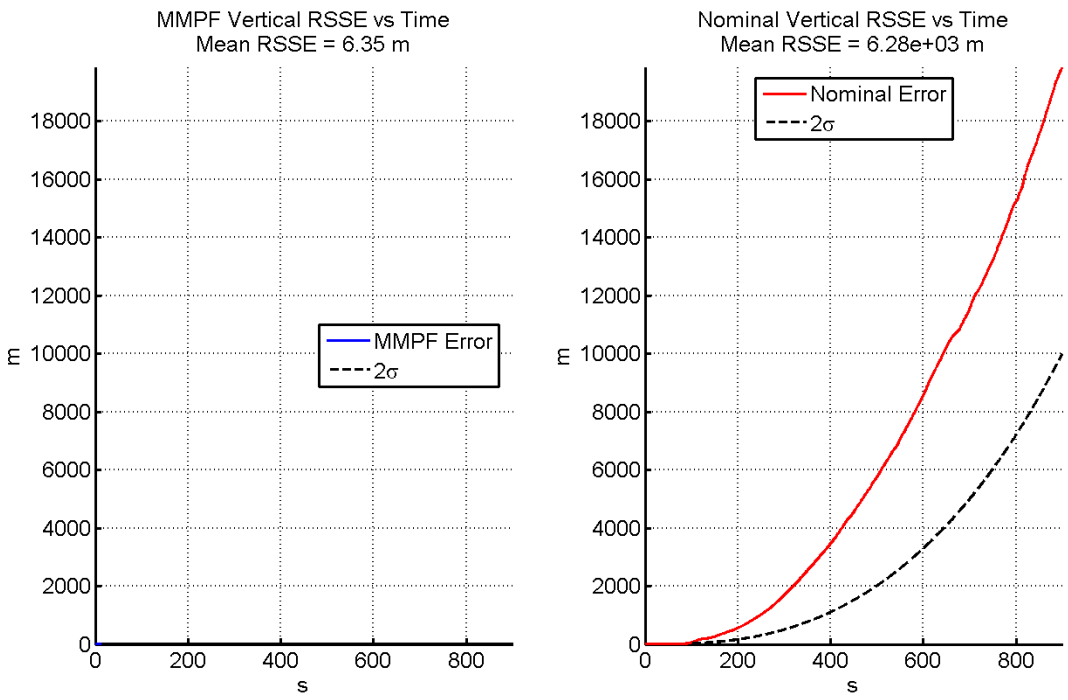
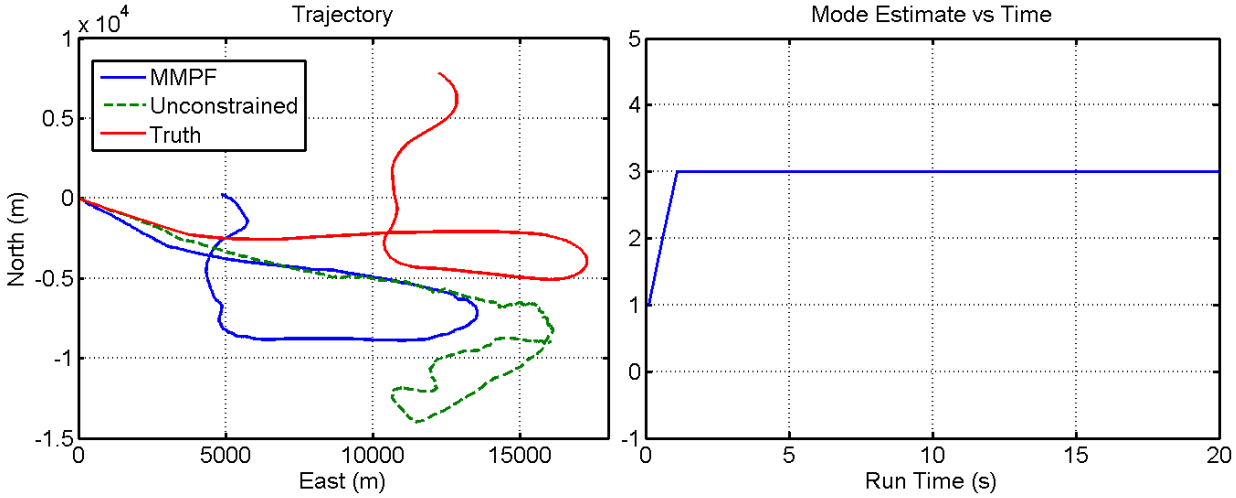


Figure 5.6: Case 17Bj altitude error comparison, MMPF vs. nominal. GPS is turned off after 60 seconds.

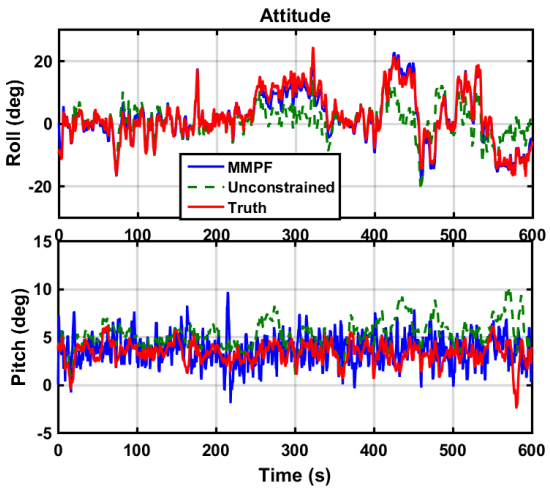
filter to discriminate models given the altimeter readings. While the unconstrained model offers a better estimate of the distance traveled in the east direction, the constrained model clearly provides a better representation of the true trajectory. The constrained model mostly maintains the shape of the trajectory. The primary error is in the heading, which unfortunately is a function only of the IMU and compass accuracies. The aircraft model does not apply any constraints to heading, and even sideslip cannot be constrained because of wind effects.

The primary benefit of the aircraft model is the constrained roll and pitch, which are subjected to a complimentary filter. These offer a substantial improvement over the free inertial solution. Roll and pitch errors are the primary source of positioning error in the unconstrained solution. Figure 5.8a shows a comparison of the roll and pitch states for the constrained and unconstrained filter, while Figure 5.8b shows a comparison of the roll and pitch errors for each. Figure 5.9 shows the horizontal error comparison of the two filters.

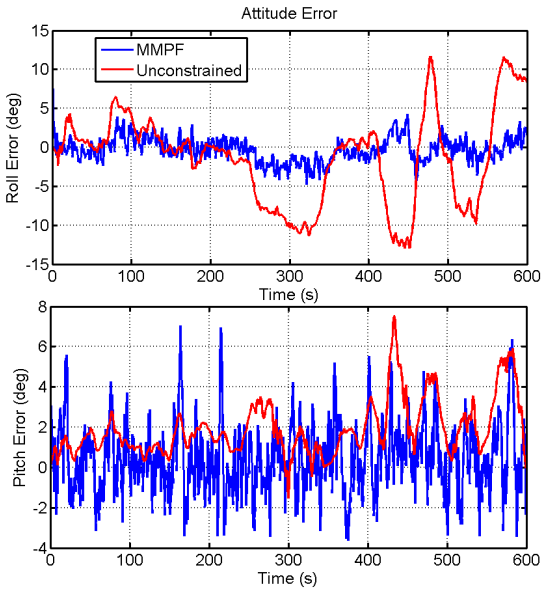


(a) (b)

Figure 5.7: Case 19Bj MMPF and nominal trajectory estimates (a). MMPF mode estimate (b). Mode 3 (aircraft) is the true mode.



(a) Case 19Bj roll and pitch state comparison, MMPF vs. nominal.



(b) Case 19Bj roll and pitch estimation error comparison, MMPF vs. nominal.

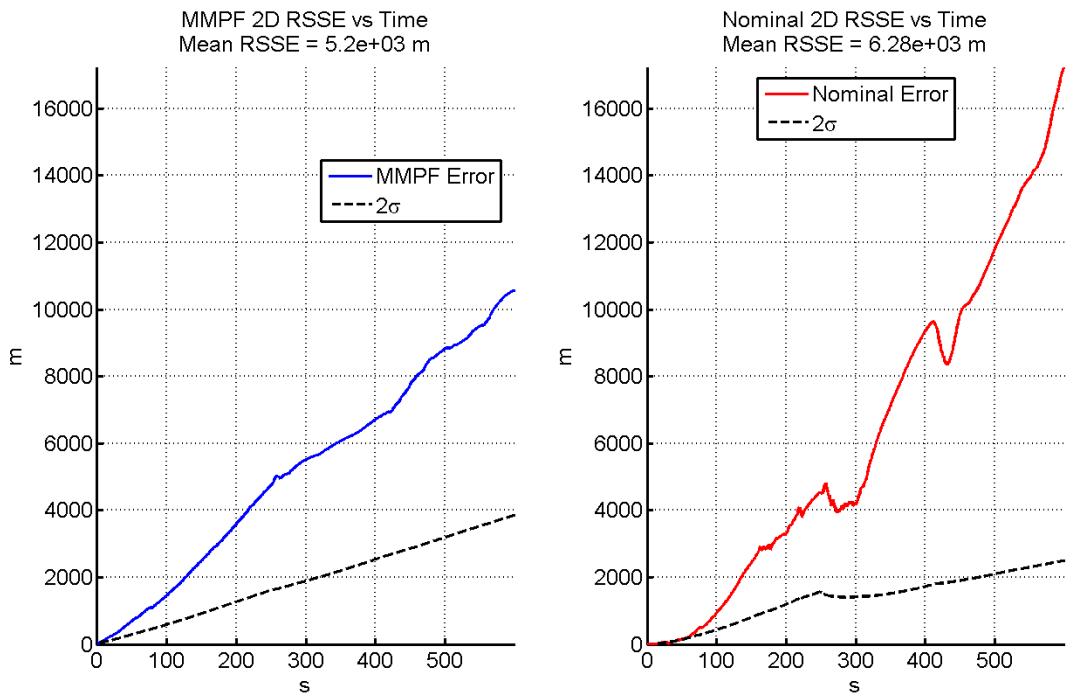
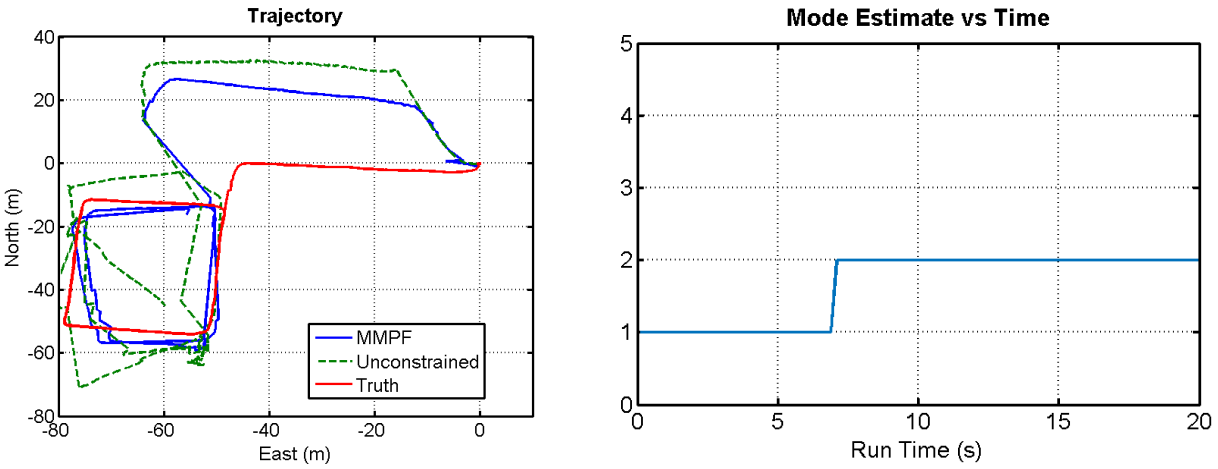


Figure 5.9: Case 19Bj horizontal error comparison, MMPF vs. nominal.

Both filters have unbounded error growth, but the constrained solution drifts at a slower rate. After 10 minutes, the unconstrained solution has 50% more error than the constrained solution.

5.2.3 Pedestrian: Case 11

Case 11 is a pedestrian case where the dismount is equipped with an HG1700 IMU, a GPS receiver (providing pseudorange measurements), a barometric altimeter, and an RFID reader. Figure 5.10a shows the trajectory comparison between the constrained, unconstrained, and true solutions. The GPS measurements are only available for the first 97 seconds (while the dismount is still outside the building). The unconstrained solution must freely integrate to find roll and pitch, with no direct aiding in these channels. This is the primary source of error for the unconstrained filter. The constrained filter, by contrast, uses a simple step detector developed by Leidos in order to estimate speed. The velocity direction is determined with the gyro (subject to roll and pitch estimates). The dismount remains



(a) (b)

Figure 5.10: Case 11A MMPF and nominal trajectory estimates (a). MMPF mode estimate (b). Mode 2 (pedestrian) is the true mode.

still for the first 50 seconds of this case. Once the dismount moves into the building the RFID reader is able to pick up signals broadcast by the transmitter. These provide non-linear measurements with respect to position, and the particle partition of the MMPF uses them to correct the position. The constrained solution is far more accurate than the unconstrained solution, and the general shape of the trajectory also much more closely matches the true path. The unconstrained solution does snap back to the RFID locations when beacon updates are available, but it drifts in between these updates resulting in a very poor trajectory.

Figure 5.10b shows the mode estimate of the MMPF for case 11. The pedestrian model drifts less than any of the other models (due to the speed constraints from step detection), so the GPS measurements effectively select the pedestrian model over all others. Figure 5.11 shows the error growth of both the constrained and unconstrained solutions. The constrained model is superior to the unconstrained model in this case. Both solutions benefit from the RFID measurements, but the unconstrained model continues to drift in between updates. The constrained model also exhibits more accurate covariance estimation than the unconstrained model near the end of the run.

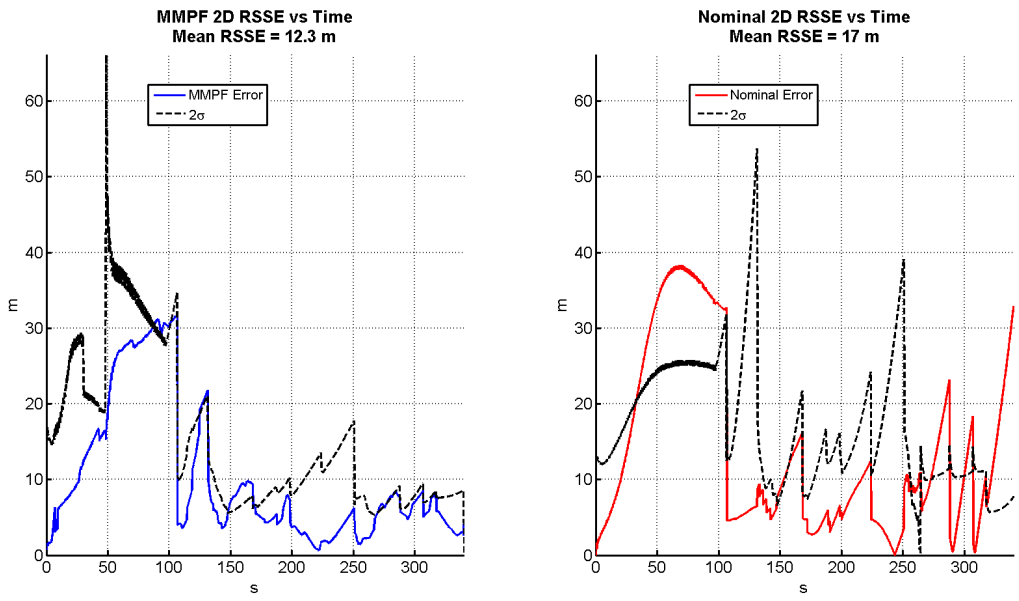


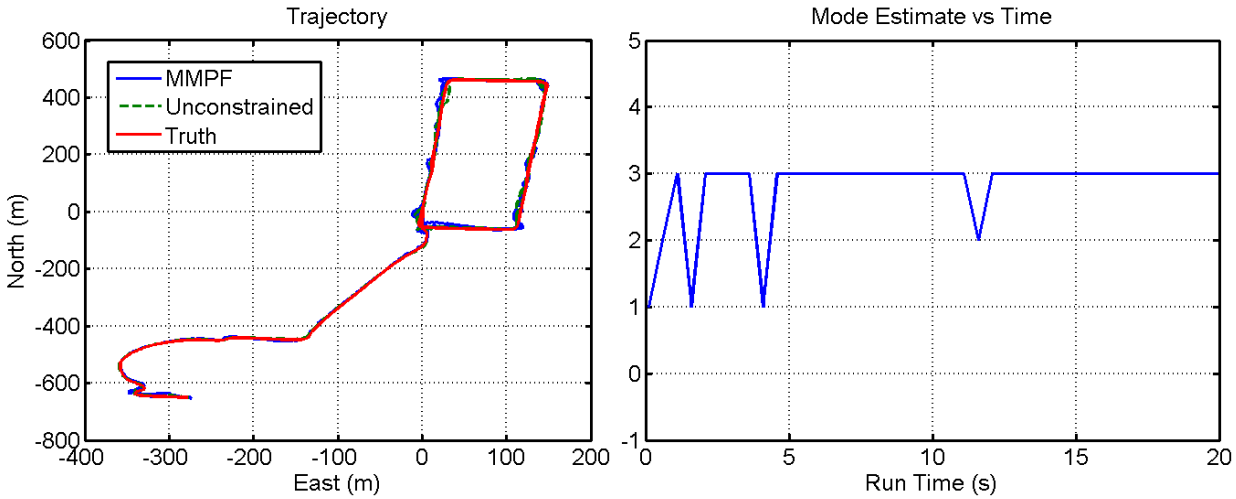
Figure 5.11: Case 11A horizontal error comparison, MMPF vs. nominal.

5.3 Adaptive Constraint Toggling

5.3.1 Ground Vehicle: Case 17

Previously case 17 was analyzed with the MMQ IMU, odometer, magnetic compass, and GPS. It was shown that the constraints improved the navigation solution over the nominal unconstrained filter. In this section case 17 is analyzed for the case where the higher quality IMU, the HG1700, is used in addition to GPS. The question is whether or not the constraints are actually needed now that a higher quality IMU is available. The MMPF was run using these sensors and compared with the nominal case. Figure 5.12a shows the trajectories for each, and Figure 5.13 shows the 2d error comparison.

The MMPF actually performs worse than the unconstrained solution. This implies what was concluded in Chapter 2, that constraints can actually hinder performance if the IMU quality is good enough. Given that a tactical grade IMU was used, it is unsurprising that the unconstrained solution is better. Another interesting side effect of "over constraining" is that the MMPF incorrectly identifies the vehicle platform type. Figure 5.12b shows the mode estimate vs. time for this case. The solution oscillates between the true platform



(a) (b)

Figure 5.12: Case 17A MMPF and nominal trajectory estimates (a). MMPF mode estimate (b). Mode 4 (unconstrained strapdown model) is the true mode.

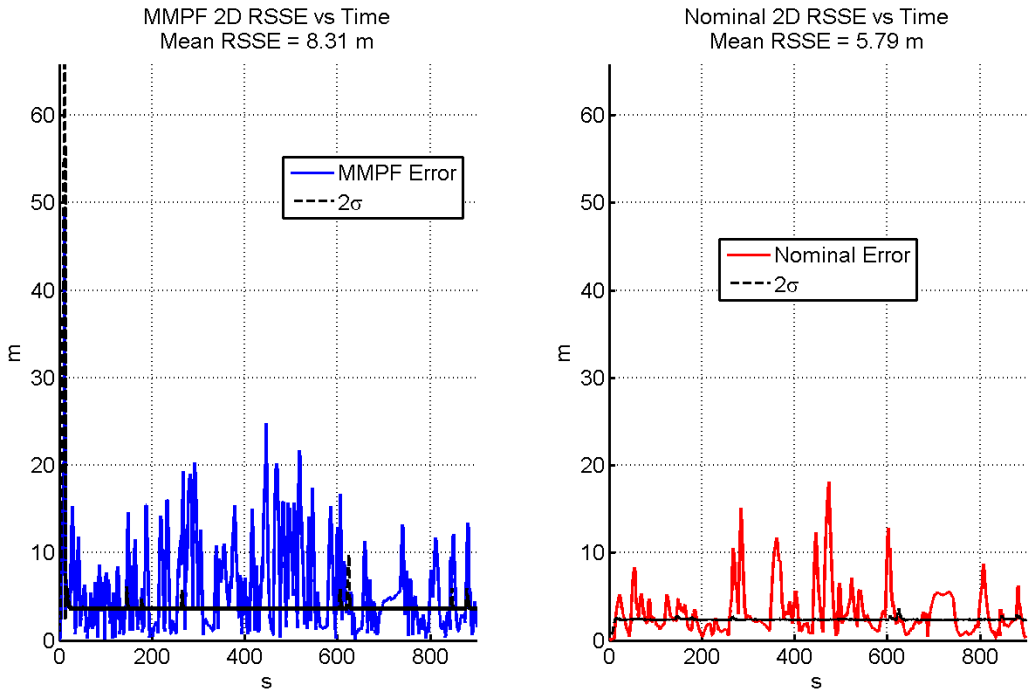


Figure 5.13: Case 17A horizontal error comparison, MMPF vs. nominal.

(ground vehicle) and aircraft, but eventually settles on the aircraft estimate. While this is incorrect, it is actually expected given the accuracy of the IMU. Speaking at a high level, the MMPF functions by simply giving more weight to platform types which most closely match the measurements. The aircraft model is the least constrained of the three platform options. Therefore it makes sense that if the unconstrained IMU is better than the ground vehicle model, then the MMPF should apply the most weight to the particle set with the least constraints. This is the aircraft set.

In order to overcome this problem, an additional particle set which is not subject to any dead reckoning constraints is added to the MMPF. This generates four possible platform hypotheses: ground vehicle (1), pedestrian (2), aircraft (3), and the "h0 hypothesis (4)." The h0 set is essentially the hypothesis that the IMU is of high enough quality so as not to require dead reckoning constraints, whatever the platform may be. In a sense this MMPF now also performs IMU classification, albeit indirectly. For the purpose of nomenclature, the MMPF which includes the h0 hypothesis is referred hereafter as the H0PF. The H0PF is tested on case 17 with the HG1700 aided with GPS. It is expected that, given the above findings. the H0PF will converge to the h0 hypothesis and that this solution will be more accurate than the baseline MMPF. Figure 5.14 shows the mode estimate vs. time for the H0PF. The filter does converge to the h0 hypothesis. Figure 5.15 shows the 2D RSSE vs. time for the H0PF compared with the nominal unconstrained filter. Given that the filter converges to the h0 hypothesis rather quickly, the navigation performance for the two filters will be very similar, as shown in Figure 5.15.

5.3.2 Aircraft: Case 19

Case 19 is run using the HG1700 IMU aided by GPS pseudo-ranges. The HG1700 was shown in previous cases to have error growth rates which are less than errors introduced by violating the navigation constraints. It is expected that in this case the same will be

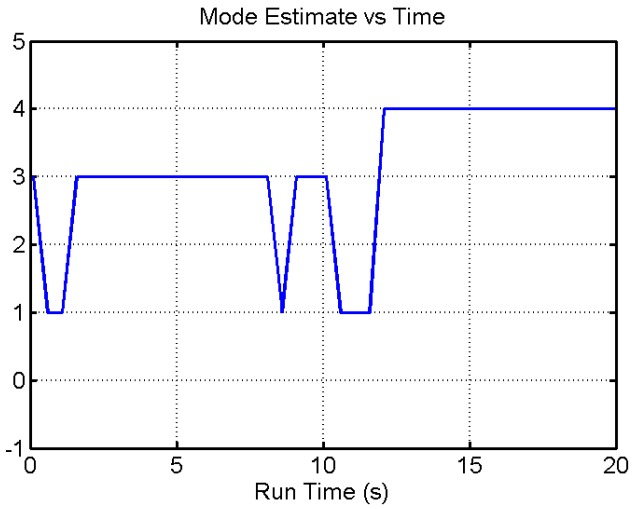


Figure 5.14: Case 17A HOPF mode estimate vs. time. Mode 4 (unconstrained strapdown model) is the true mode.

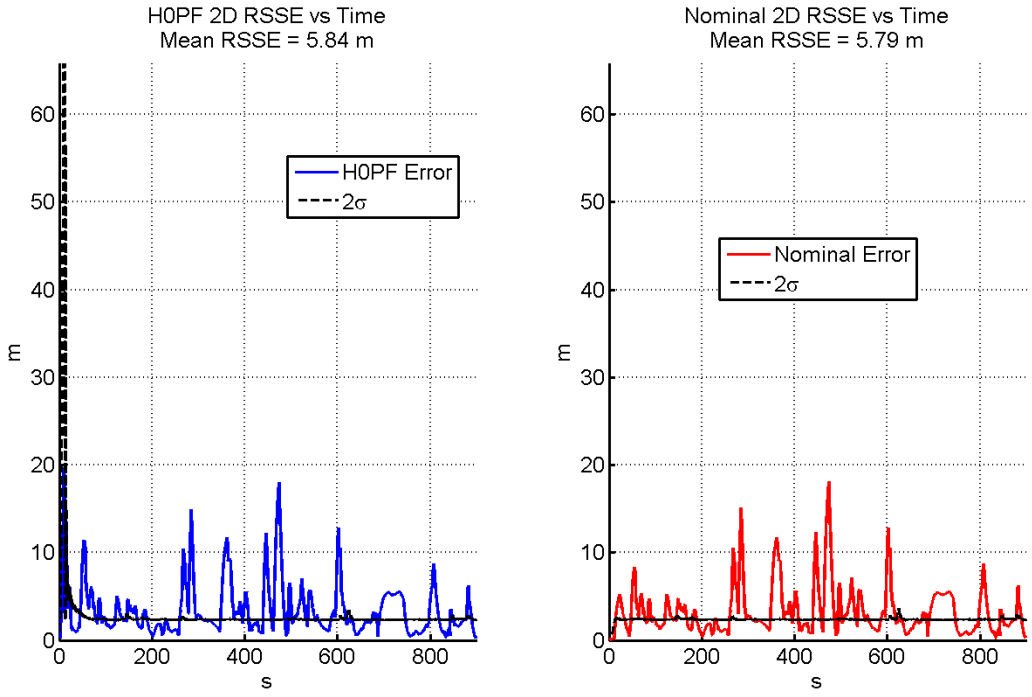


Figure 5.15: Case 17A horizontal error comparison, HOPF vs. nominal.

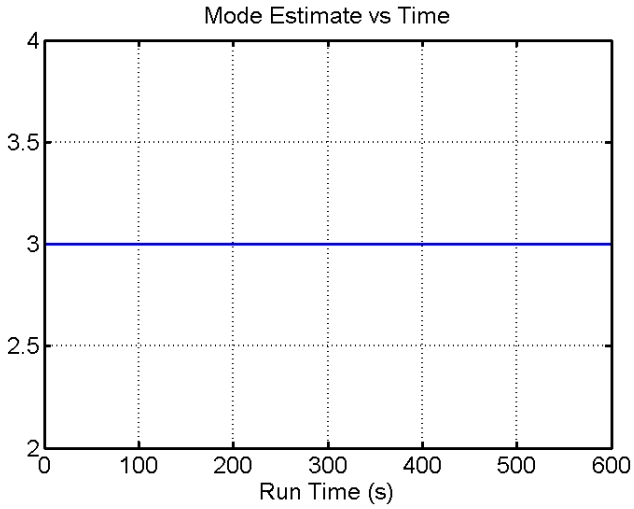


Figure 5.16: Case 19A MMPF mode estimate vs. time. True mode is 3 (aircraft).

true. Figure 5.16 shows the mode estimate of the baseline MMPF for this case. The MMPF correctly estimates the platform, in contrast with case 17 when the HG is used. This is because the DTED constraints make it very easy to discriminate amongst the particle sets using the altimeter. Yet even though the platform is properly identified the navigation results are worse than the unconstrained case. This is expected. Figure 5.17 shows the horizontal errors of the baseline MMPF and the nominal filter.

The H0PF is again tested using this data set in order to mitigate the problem of over constraining the high quality IMU. Figure 5.18 shows the mode estimate vs. time for the H0PF. The filter correctly converges to the h0 hypothesis, as expected. Figure 5.19 compares the 2D errors for the H0PF and the nominal filter. The error profiles are almost identical, as expected, after the initial convergence period. The difference in uncertainties during the first few seconds are also due to the transients of the mode estimation for the H0PF. The H0PF successfully determines that constraints are not needed for the IMU in use, and the h0 hypothesis is selected for optimal navigation.

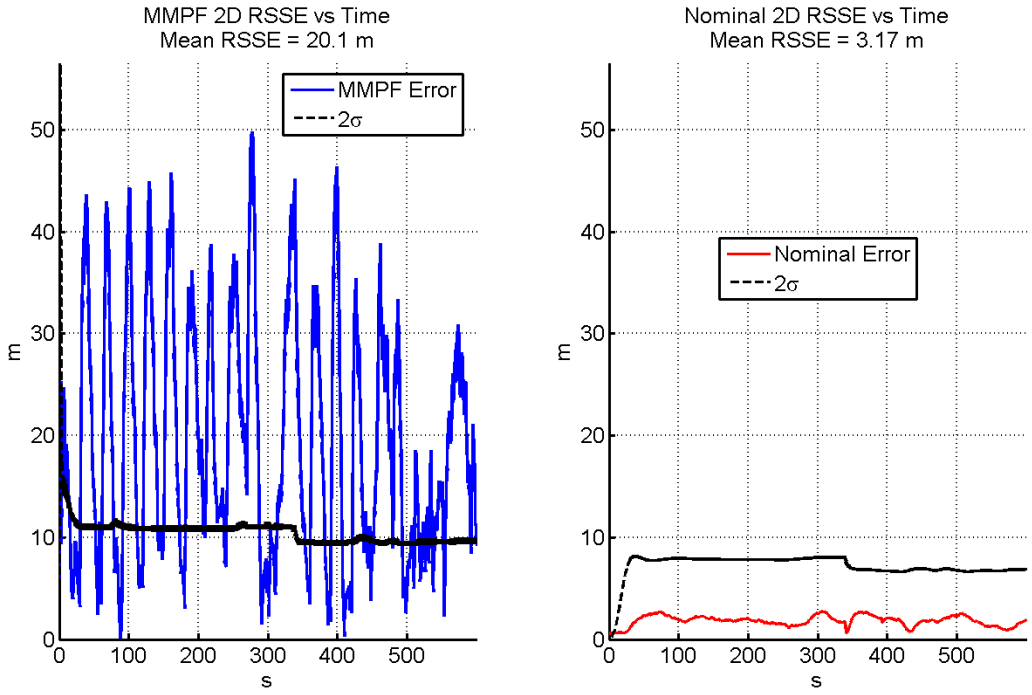


Figure 5.17: Case 19A horizontal error comparison. MMPF vs. nominal.

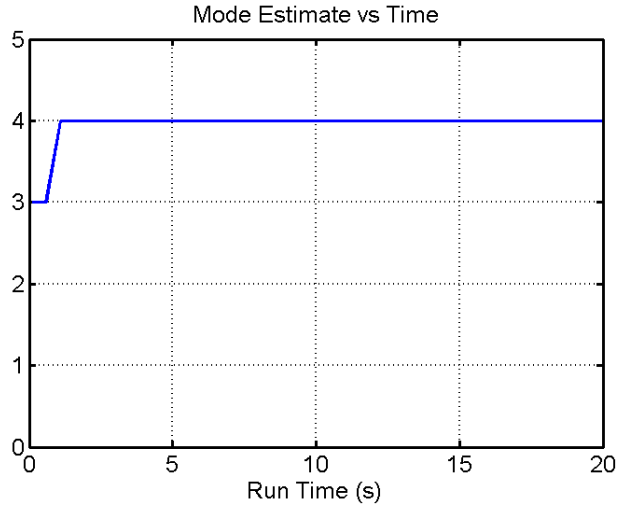


Figure 5.18: Case 19A HOPF mode estimate vs. time. True mode is 4 (unconstrained strapdown model).

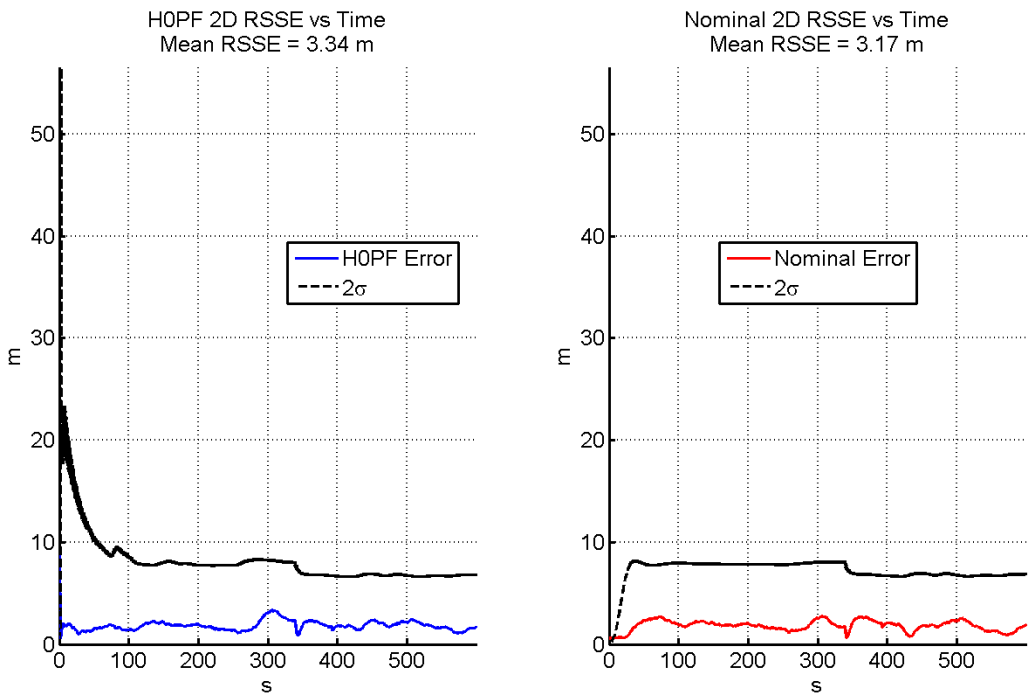
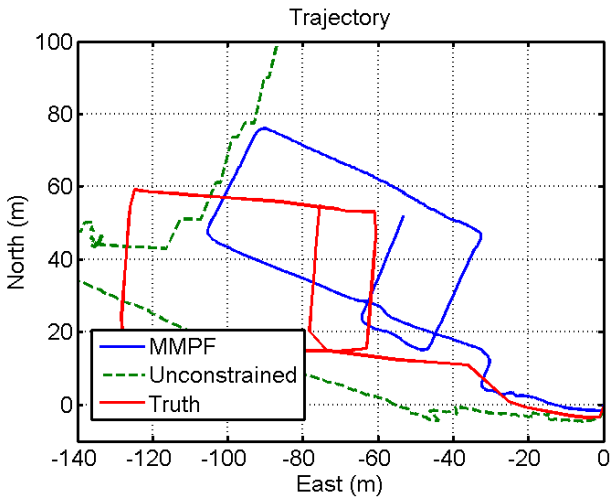


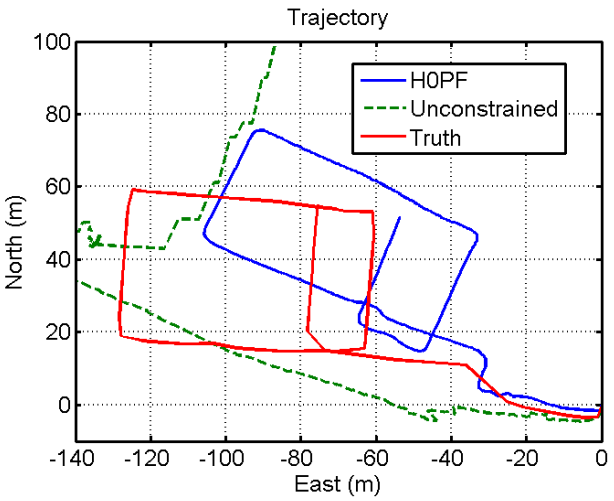
Figure 5.19: Case 19A horizontal error comparison. H0PF vs. nominal.

5.3.3 Pedestrian: Case 10

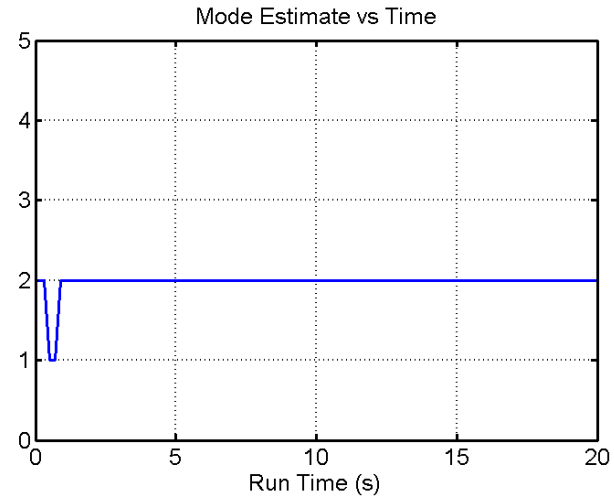
The previous results show the advantages of the H0PF over the MMPF. However it is important to demonstrate that the extra hypothesis doesn't interfere with the mode estimation when the proper mode is one of the three constrained sets. This is because ideally the H0PF should be the standard filter for normal operation, simultaneously deciding online whether or not constraints are needed, and if so, which model to apply to the filter. In order to demonstrate successful operation of the H0PF in a situation where the h0 hypothesis is not the correct one, case 10 is evaluated using the H0PF. Case 10 is a pedestrian case where a MEMS grade IMU is aided by an altimeter and an inclinometer. Figures 5.20a and 5.20b show the trajectories of the MMPF and H0PF (each compared with the nominal). The H0PF matches the MMPF very closely, and the resulting RSSE of the H0PF and the MMPF are within 1 m of each other.



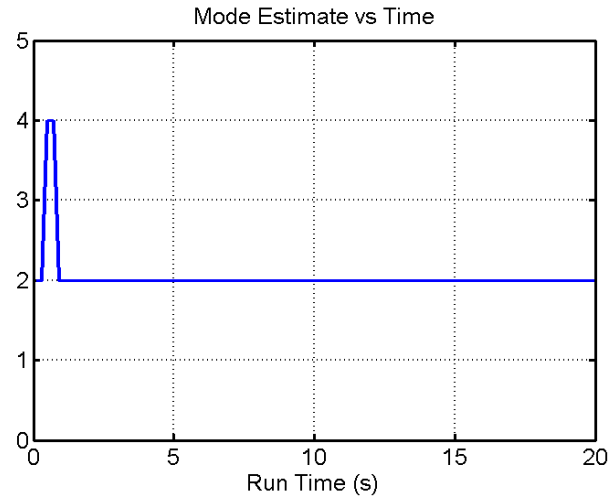
(a) Case 10A MMPF trajectory estimate.



(b) Case 10A H0PF trajectory estimate.



(a) Case 10A MMPF mode estimate vs. time.

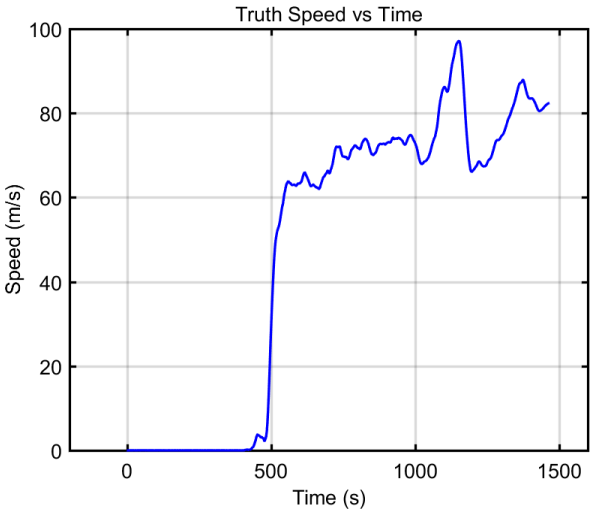


(b) Case 10A H0PF mode estimate vs. time.

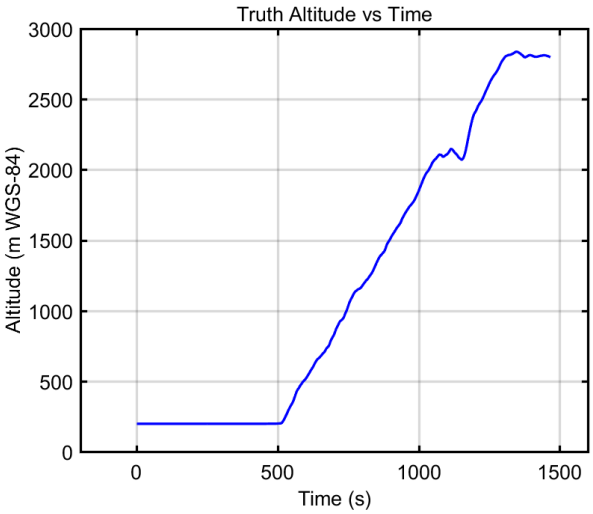
Figures 5.21a and 5.21b show the comparison of the mode estimates vs. time for the MMPF and H0PF, respectively. It can be seen that the H0PF properly converges to the pedestrian mode, thereby confirming proper operation with the additional h0 hypothesis.

5.3.4 Aircraft Takeoff: Case 216A

The results from cases 19 A and Bj show that the JNC filter can correctly classify an aircraft or switch to an unconstrained model if it is appropriate given the quality of the IMU. However in that case the data set began with the aircraft already in flight. An important question is whether the JNC filter can properly classify the aircraft before takeoff, or at

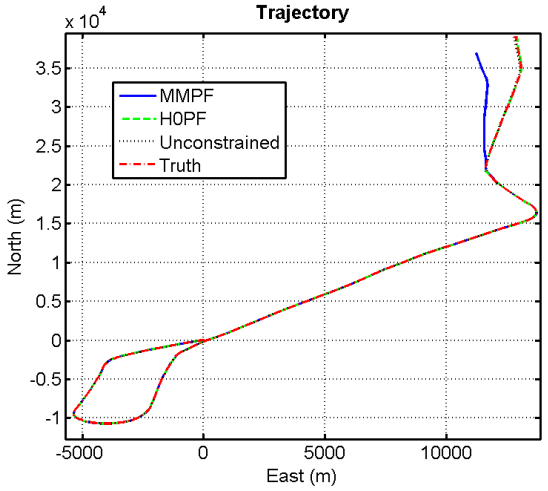


(a) Case 216 truth system speed profile.

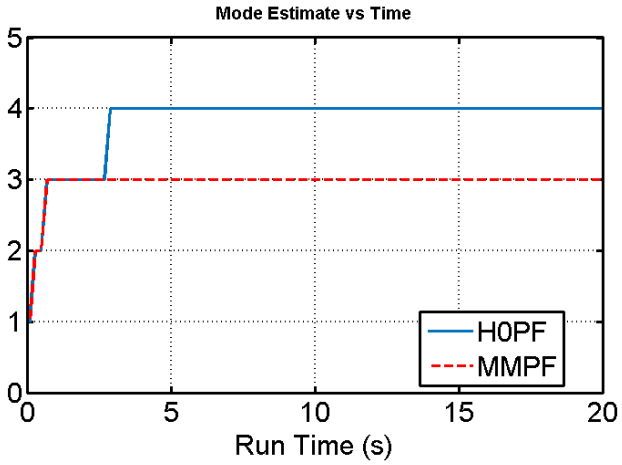


(b) Case 216 truth system altitude profile.

least avoid making an erroneous classification of a ground vehicle during taxiing. Case 216 is evaluated in order to answer this question. In case 216, the aircraft is equipped with a tactical grade IMU (from a Novatel SPAN system), GPS position and velocity measurements, a barometric altimeter, and a magnetic compass. The speed and altitude profiles of the flight are shown in Figures 5.22a and 5.22b, respectively. The first 400 seconds of data are recorded while the aircraft is not moving on the runway, so the case is evaluated beginning at 450 seconds into the run. This is because there is very little actual signal in the reference trajectory, and it is mostly noise. The purpose of this work is focused on classification under more dynamic conditions, but future work should consider how to take long static periods into account. The aircraft slowly begins to move at 400 seconds and is at taxi speeds when the run begins. The aircraft does not begin acceleration into takeoff until 480 seconds, and liftoff does not occur until just after 500 seconds. Figure 5.23a shows the trajectory comparison of the MMPF, the H0PF, the unconstrained filter, and the truth data. Figure 5.23b shows the mode estimate vs. time. The H0PF correctly determines that a tactical grade IMU is in use and selects the unconstrained model. The MMPF does not have this option and it identifies the constrained aircraft dead reckoning model. However the errors introduced by violations of the constraint assumptions outweigh the errors from inertial propagation



(a) Case 216 trajectory estimates compared with truth.



(b) Case 216 mode estimates vs. time.

and the unconstrained model performs better than the constrained model. This result is just like the result from case 19A, where the MMPF also classified the aircraft correctly but performed more poorly than the H0PF which selected the unconstrained model. In this case the classification occurs before the aircraft ever leaves the ground. This is primarily due to the very accurate GPS velocity measurements, which are sensitive (accurate) enough to capture the difference between the error propagation of the different models in only a few filter iterations. Case 216 provides another example of proper classification of the H0PF, but it doesn't answer the potential problem mentioned previously pertaining to aircraft taxiing. It is clear from this example that this is only a problem when a low quality IMU is used on the aircraft, and so data from such a scenario is needed to determine the effect of ground taxiing on aircraft platform classification.

5.4 Conclusion

This chapter has presented several experimental data sets collected on various platforms and an analysis of the MMPF and H0PF across these sets. The IMU's used in each case were described and categorized according to common navigation classifications. First, three

cases which use the MMQ, a high grade MEMS IMU, were analyzed. This was followed by an analysis of results from four cases using the tactical grade HG1700 IMU.

Table 5.3 gives the 3D RMSE values for all cases presented in this work. First, it is seen in Case 17Bj that the MMPF does improve the overall navigation performance. However, as this case has strong aiding from GPS throughout the entire run, there isn't much room for improvement. A more practical example of the benefits of the MMPF is to consider the case of GPS drop out. Case 17Bj* does just this, where GPS is shut off after 60 seconds. As expected, the MMPF vastly outperforms the nominal filter case when GPS is denied. The MMPF also outperforms the nominal filter on Case 19Bj. In this case there is no absolute position information, thus there is nothing to prevent filter drift. However the MMPF is able to properly identify the platform (aircraft) and slow the drift. Case 11A is a pedestrian case which uses an HG1700 IMU. Here also the MMPF is able to succesfully classify the platform and improve performance compared with the unconstrained solution. Cases 17A and 19A demonstrate the importance of the h0 hypothesis in the H0PF. In both of these cases the IMU used was the tactical grade HG1700, as opposed to the MEMS grade MMQ. The results from Chapter 2 demonstrate that constrained navigation can be inferior to unconstrained when using high quality IMUs. Therefore it is expected that the H0PF, which allows for the selection of an unconstrained model as a possible hypothesis, would outperform the MMPF in the cases with higher quality IMUs. Cases 17A and 19A show that this is true. Case 10A is included to demonstrate the robustness of the H0PF. It is not always advantageous to choose the unconstrained model when higher quality IMUs are used, therefore it is important to demonstrate that the inclusion of the h0 hypothesis doesn't detract from the MMPF operation otherwise. In case 10A the unconstrained solution performs very poorly, and the H0PF is quick to select the pedestrian mode, as the MMPF also does. This is why the H0PF has a very similar error metric for this run. Additionally, the H0PF results are given for the first four cases in order to show that the inclusion of the h0 hypothesis doesn't qualitatively degrade the navigation performance. Finally, case 216A

Table 5.3: Mean RSS errors (2D / 3D) (m).

	Nominal	MMPF	H0PF
Case 17Bj	4.3 / 7.0	4.0 / 6.2	4.0 / 6.2
Case 17Bj*	147 / 6286.9	47.2 / 48.1	48.9 / 49.8
Case 19Bj	6281.0 / 6281.8	5204.4 / 5204.7	4747.3 / 4747.7
Case 11A	17.0 / 17.2	12.3 / 12.7	11.8 / 12.2
Case 17A	3.2 / 5.8	5.7 / 8.3	3.4 / 5.8
Case 19A	1.8 / 3.2	19.0 / 20.1	1.8 / 3.3
Case 10A	417.5 / 497.2	21.0 / 21.4	20.6 / 21.0
Case 216A	6.6 / 10.2	258.2 / 259.5	2.7 / 5.8

is included to demonstrate that the platform classification works for aircraft even before the vehicle takes off. The results from this case show that this is not a problem when a tactical grade IMU is used. However, hard conclusions cannot be drawn about the MEMS grade IMU case without data for this situation. It has been shown that the H0PF is able to successfully identify the proper platform model for best navigation adaptively and apply the appropriate constraints to improve navigation (if any).

5.5 A Word of Acknowledgement

It is important to acknowledge two major contributions of outside parties to this work. The first is the DARPA ASPN team who performed the data collection for the data used in Chapter 5 and made it available for this work. The author was not involved in any way with the hardware setup or test planning or execution of any of these data collects. The depth, volume, and quality of the data has been invaluable in developing the algorithms presented here, and the author is greatly indebted to the DARPA personnel for making this available for this study.

The second party is the Leidos team in Huntsville, AL, who implemented the sensor and measurement interfacing system. The base RBPF algorithm which this work makes use of was presented in [36] and discussed in Section 4.3, but it is built on top of a complicated sensor handling interface designed by the Leidos team. This system allows sensors to arbitrarily

come online or go offline and makes it easy to use data for the many various sensors used in Chapter 5. Furthermore, the capability to dynamically switch which filter states are in the linear or nonlinear partitions on the fly was developed by Leidos [40]. The multiple model particle filter used in the JNCPF presented in this work naturally takes advantage of this partion switching capability. Finally, Leidos also developed several measurement models discussed in Section 4.5 [40].

The author is very grateful to all of these individuals.

Chapter 6

Conclusion

This dissertation has developed an algorithm which is capable of solving the joint navigation and classification problem. That is, the algorithm is able to determine the host platform type online without any apriori knowledge and then use this information to improve the navigation solution via kinematic constraints specific to the platform type.

6.1 Kinematically Constrained Navigation

First, the motivation for why kinematic constraints are needed was presented in Chapter 2. Inertial navigation systems drift over time, and while it is true that aiding sensors can mitigate this drift, adding constraints can diminish this drift even further. The idea is to compute the best inertial trajectory possible before applying aiding from external sensors, so that when the external measurements arrive there is less error that needs correcting. In this way the filter is able to converge to an even finer accuracy in steady state than would be possible if the inertial trajectory were not as good. Therefore Chapter 2 presented a case study in which an unaided IMU is improved via kinematic ground vehicle constraints. It was shown that adding the constraints provides a substantial improvement over the otherwise unaided solution, therefore motivating the need for including these constraints. These results were proven via Monte Carlo simulations and also with experimental data obtained from an ATV outfitted with a tactical grade IMU and also a Novatel SPAN GPS/INS system for ground truth comparison. The results also showed that is possible to actually degrade the navigation solution with these same constraints, if they are applied when the quality of the IMU is high enough and there are sufficient dynamics. This effectively means that in these cases the amount of error introduced by small violations in the constraint assumptions is

actually larger than the amount of drift present in the IMU. For lower quality IMUs, the inertial error is greater than the constraint violations. Nevertheless this finding introduces a challenge when trying to achieve the navigation improvement from the constraints while simultaneously maintaining the flexibility provided by the online platform classification. It is not enough to classify the platform online, because if the IMU is high grade then applying the constraints even with a properly identified platform can degrade the result in very dynamic situations. So the joint navigation and classification (JNC) filter must also be able to determine on the fly if the constraints should even be applied.

6.2 Online Platform Classification

Chapter 3 discussed the fundamentals of particle filtering in general and also presented a trade study of various multiple model (or hybrid) particle filter designs which are relevant to the JNC problem. Particle filtering is chosen as a starting place for JNC development for several reasons. First, it provides the highest level of flexibility when considering how to apply the kinematic constraints. However, it was determined that for the lowest quality IMUs it is better to apply the constraints directly to the inertial propagation as in [27], and not as pseudomeasurements as in Chapter 2. So particles are not needed for this purpose. Yet particle filtering also allows for nonlinear and non Gaussian measurement types, such as the RFID beacon measurements which are used in the pedestrian cases. Furthermore, particle filtering is very compatible conceptually with the concept of treating the platform type as a mode state in a hybrid state vector setup. The basics of particle filtering are presented in Chapter 3, followed by a discussion of important considerations regarding sample impoverishment and how to choose the best proposal density for resampling. Sample impoverishment occurs after repeated resampling of a particle set which has a low process noise. The particles do not have time to "spread out" across the state space in between resampling, and the result is that most of the particles end up occupying a very small portion of the state space. The method of particle regularization was used to solve this problem. Effectively,

regularization amounts to adding Gaussian noise to the resampled vector. The kernel used in regularization is proportional to a Gaussian approximation of the density prior to resampling. The end result is that the variance collapse due to resampling is diminished, resulting in a better coverage of the state space. Following the particle filtering preliminaries, a detailed trade study of multiple model hybrid state particle filters was given. There are three main categories that can be used to categorize each of the algorithms analyzed. The first discriminator is, can the mode state change over time or not (i.e. static mode vs. dynamic mode)? Second, if the mode can change, is the transition probability strictly Markov or not? Third, what measures are taken to ensure that the mode state does not suffer unwanted sample impoverishment. This phenomenon occurs when modes which have low probability can be "resampled out" erroneously, not because they have been definitively ruled out via the measurements but because there aren't enough particles to cover the entire state space. The method presented by Gordon et al. [21] is chosen over the others as a starting place for the JNC filter for several reasons. First, it sufficiently mitigates the mode impoverishment problem by maintaining a constant number of particles for all modes with non-zero probabilities. Therefore modes can only die out if the measurement likelihoods cause it, as desired. Second, the filter by Gordon et al. treats the mode state as static, which is more appropriate for the problem of JNC than a dynamic mode. Chapter 4 develops the joint navigation and classification filter algorithm which is ultimately presented as the solution to the JNC problem. First, some initial filtering concepts are described which help frame the overall structure of the JNC filter. The indirect filtering concept is presented, followed by Rao-Blackwellization of the particle filter. The Rao-Blackwellization method chosen is that of [36]. This approach divides the state vector into a linear and nonlinear partition. The linear partition is solved via extended Kalman filtering, while the nonlinear partition is solved via particle filtering. In the JNC filter, only 2 of 27 states are treated as particles, thus minimizing the computational load. Following this, the constrained kinematic models for each of the platforms is presented in addition to the null hypothesis. There are three

constrained models, one for a pedestrian, ground vehicle, and aircraft. The null hypothesis assumes no constraints. The measurement models for all sensors used as filter measurement sources are also presented. This includes the measurement equation, the partial derivative of the measurement with respect to the state, and the error model of each measurement type.

Finally, Chapter 5 presented the validation of the joint navigation and classification particle filter (JNCPF) using various experimental data sets. First, three data sets are considered which should be well conditioned for kinematic constraints to improve navigation. The first two data sets are ground vehicle and aircraft cases which do not use a tactical grade IMU. So it is expected that kinematic constraints should improve the navigation solution, if the online platform classification can be achieved. The JNCPF succesfully classified the platforms and improved the navigation solution for these scenarios. The third case is a pedestrian case using a tactical grade IMU. It should be noted that based on the results of Chapter 2 one would expect the unconstrained tactical grade IMU to outperform the constrained tactical grade IMU. However the JNCPF classified the model as a pedestrian, and this constrained result was superior to the unconstrained result. Recall that chapter 2 showed that there is a point at which constraint violations introduce more error growth than the IMU drift. This point is of course a function of the IMU quality, but it is also a function of the kind of dynamics seen in the given application. The pedestrian data sets which were used are fairly benign in terms of how much they deviate from the kind of motion assumed in the pedestrian mode (e.g. there is no crawling or walking sideways). Whereas in the data presented in Chapter 2, the ATV experienced a lot of sideslip due to the aggressive dynamic maneuvers. So the general principle of this tradeoff curve between constraints and inertial drift still stands, but the kind of dynamics experienced by the host platform will effectively shift the curve in one direction or another. In these benign pedestrian cases, the curve is expected to shift slightly in favor of the constraints. There is another element which makes strapdown inertial navigation different on a pedestrian platform than a ground or aerial vehicle. Compared with ground vehicle or aircraft motion, the signal to noise ratio

of the pedestrian motion (in particular acceleration) is much lower. So it is likely that this also "shifts the curve" of optimality towards the kinematic constraints in a way that doesn't occur for ground vehicle and aircraft.

Next, the case where the ground vehicle and aircraft platforms use tactical grade IMUs was considered. For these scenarios it was expected that the JNCPF would actually degrade performance if the null hypothesis isn't included. The results presented showed that this was true. In fact, not only did the JNCPF degrade navigation performance in the tactical grade aircraft case, but it actually failed to properly classify the platform. However by including the null hypothesis (which is a candidate hypothesis with an unconstrained model), the JNCPF was able to successfully avoid this pitfall. This is seen in that the JNCPF *with* the H0 hypothesis achieved basically the same error as the nominal model in the ground vehicle and aircraft cases using tactical grade IMUs. Additionally, the JNCPF with the H0 hypothesis was tested on all scenarios to validate that inclusion of this extra hypothesis didn't degrade performance when other hypotheses were correct. It was shown that the JNCPF correctly selected the most accurate model each time.

There is no concrete rule which can be used to determine if the IMU is high enough quality to warrant not using the constraints, as discuessed in Section 5.2.3. This is because the type or severity of the host platform dynamics also affects this. So a filter designer cannot simply make a decision apriori based on the IMU quality (unless the IMU is of exceedingly high or low quality). Yet the JNCPF with H0 can handle this implicitly, and the results from Chapter 5 validate this. This is an important benefit of the JNCPF framework.

This work has shown the capability and benefit of an algorithm which can solve the JNC problem. The JNCPF is able to classify the host platform type online and improve the navigation solution using kinematic constraints appropriate to the classified platform type. This solution offers a way to provide a platform optimized solution without compromising the high degree of flexibility that is set up by the ASPN goals and requirements. This best of both worlds combination of accuracy and flexibility could benefit a wide range of applications

beyond the ASPN application (although the set of potential end users of ASPN is very broad already). Anyone in the commercial world who seeks to reduce time-to-market without compromising accuracy would benefit from the JNCPF. The JNCPF would also apply to anyone seeking a navigation solution which can be switched between multiple vehicles (or IMUs) in the field, as the JNCPF would require virtually no reconfiguration to handle this.

6.3 Future Work

What if the host platform is a pedestrian riding in a vehicle who ultimate exits the vehicle at some point during the mission? In this case, the host platform type could be considered to be dynamic. The ideal scenario would be if the JNCPF could handle switching platform modes, such that the h0 hypothesis is identified and used while the pedestrian is in the vehicle and the pedestrian hypothesis is selected once the dismount exits the vehicle. So the JNCPF would have to be handle dynamic platform modes as opposed to static modes, and also be sensitive enough to detect this transition. Note that while the dismount is in the vehicle, the optimal model choice would still likely be the h0 mode as opposed to the ground vehicle mode because the orientation of the IMU with respect to the vehicle body frame would be changing in some arbitrary manner. Especially in a military application it would be a stretch to assume that the passenger is seated in a certain orientation and that they maintain that position until they exit the vehicle. Regardless, the assumptions about the mode state being static would have to change. This author believes the best course would be to move from a MMPF framework similar to [21], which is what is used now, to a framework similar to [30]. This framework is beneficial in this case for several reasons. First, the mode state is dynamic and the framework is set up to minimize the problem of mode impoverishment. Mode impoverishment would mean complete failure of the algorithm in the dynamic case, so this point is critical. Any mode which has a non-zero transition probability is guaranteed to be represented by some particles in Narasimhan et al.'s framework. Furthermore, they do not assume that the mode transition probabilities are

Markovian. This is important, because it allows for the mode transition probability to be a function of other parts of the state vector. For example, in the case of the dismount riding in the truck, it is unlikely that he or she will exit the truck if the speed is very high. But as the speed approaches zero, the likelihood of an exit would increase. Multiple model frameworks which make the Markov transition assumption cannot account for this, but the Narasimhan et al. approach can. This author believes that in order to account for a scenario of dynamic mode switching, a version of Narasimhan et al.'s [30] framework would be optimal to perform JNC.

Another important avenue of future work would be to properly apply integrity monitoring of the measurements within the JNC framework. If enough redundant measurements are available it would be possible to apply a "measurement only" integrity monitoring scheme, where the integrity checks are not a function of the state estimate. Receiver autonomous integrity monitoring (RAIM), which is common when solving the GPS pseudorange problem, is a good example of this kind of integrity monitoring. However the more challenging problem is what to do when there isn't enough redundant information in the measurements alone, and the state estimate must be used as in classical chi-squared measurement rejection techniques. State dependent measurement rejection techniques rely on accurate error statistics of both the measurement and the state vector. In the case of the JNCPF, the incorrect models (which haven't already been eliminated) will by definition have inaccurate error models because they don't match the true kinematics. Therefore chi squared cannot be used in the measurement rejection and integrity monitoring process, and more sophisticated methods must be developed. These methods would also have to account for the non-Gaussian nature of the horizontal position posterior distribution, because these states are in the nonlinear partition and may not be Gaussian.

Finally, it would be important to add additional platform model types such as rotary wing aircraft or skid steer ground vehicles. These in particular are increasingly relevant with

the rise of small unmanned aerial and ground vehicles. The pedestrian model should also be improved upon to account for more diverse motion such as crawling and climbing, etc.

Bibliography

[1] Defense Advanced Research Projects Agency. All Source Positioning and Navigation (ASPN) Phase 2. DARPA-BAA-12-45., July 2012.

[2] Eric C. Anderson. Monte Carlo Methods and Importance Sampling. Lecture Notes for Stat 578C, University of Washington, October 1999.

[3] Donka Angelova and Lyudmila Mihaylova. Joint Target Tracking and Classification with Particle Filtering and Mixture Kalman Filtering Using Kinematic Radar Information. *Digital Signal Processing*, 16(2):180 – 204, 2006.

[4] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, Feb 2002.

[5] I Bilik and J Tabrikian. Maneuvering Target Tracking in the Presence of Glint Using the Nonlinear Gaussian Mixture Kalman Filter. *Aerospace and Electronic Systems, IEEE Transactions on*, 46(1):246–262, 2010.

[6] Henk AP Blom, Edwin Bloem, et al. Exact Bayesian and Particle Filtering of Stochastic Hybrid Systems. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(1):55–70, 2007.

[7] Yvo Boers and Hans Driessen. Hybrid State Estimation: A Target Tracking Application. *Automatica*, 38(12):2153 – 2158, 2002.

[8] Subhash Challa and Graham W Pulford. Joint Target Tracking and Classification Using Radar and ESM Sensors. *Aerospace and Electronic Systems, IEEE Transactions on*, 37(3):1039–1055, 2001.

[9] John L Crassidis. Sigma-Point Kalman Filtering for Integrated GPS and Inertial Navigation. *Aerospace and Electronic Systems, IEEE Transactions on*, 42(2):750–756, 2006.

[10] N.J. Cutaia and J.A. O'Sullivan. Automatic Target Recognition Using Kinematic Priors. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 4, pages 3303 –3307 vol.4, Dec 1994.

[11] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte. The Aiding of a Low-Cost Strapdown Inertial Measurement Unit Using Vehicle Model Constraints for Land Vehicle Applications. *IEEE Transactions on Robotics and Automation*, 17(5):731–747, 2001.

[12] Randal Douc and Olivier Cappé. Comparison of Resampling Schemes for Particle Filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69. IEEE, 2005.

[13] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and computing*, 10(3):197–208, 2000.

[14] Arnaud Doucet and Adam M Johansen. A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

[15] H. Driessen and Y. Boers. Efficient Particle Filter for Jump Markov Nonlinear Systems. In *Radar, Sonar and Navigation, IEE Proceedings-*, volume 152, pages 323–326. IET, 2005.

[16] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel. A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 340–345. IEEE, 2008.

[17] W. Gao, Z. Xiong, Y. Hao, and F. Sun. Comparison of INS Transfer Alignments through Observability Analysis. In *2006 IEEE/ION Position, Location, And Navigation Symposium*, pages 823–829, 2006.

[18] D. Gebre-Egziabher, JD Powell, and PK Enge. Design and Performance Analysis of a Low-Cost Aided Dead Reckoning Navigation System. *Gyroscopy and Navigation*, 4(35):83–92, 2001.

[19] S. Gleason and D. Gebre-Egziabher, editors. *GNSS Applications and Methods*. Artech House Publishers, 685 Canton Street Norwood, MA 02062, 2009.

[20] S. Godha and ME Cannon. Integration of DGPS with a Low Cost MEMSbased Inertial Measurement Unit (IMU) for Land Vehicle Navigation Application. In *Proceedings of the 18th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS05)*, pages 333–345, 2005.

[21] N.J. Gordon, S. Maskell, T. Kirubarajan, et al. Efficient Particle Filters for Joint Tracking and Classification. In *Proceedings of SPIE*, volume 4728, pages 439–449, 2002.

[22] Paul D Groves, Graham W Pulford, C Aaron Littlefield, David LJ Nash, and Christopher J Mather. Inertial Navigation Versus Pedestrian Dead Reckoning: Optimizing the Integration. In *Proc. ION GNSS*, pages 2043–2055, 2007.

[23] P.D. Groves and Ebooks Corporation. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, 2008.

[24] Shawn Herman and Pierre Moulin. A Particle Filtering Approach to FM-Band Passive Radar Tracking and Automatic Target Recognition. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 4, pages 4–1789. IEEE, 2002.

[25] NovAtel Inc. Apn-062: Novatel Correct with Veripos. rev. c., April 10 2014.

[26] David Ronald Kincaid and Elliott Ward Cheney. *Numerical Analysis: Mathematics of Scientific Computing*, volume 2. American Mathematical Soc., 2002.

[27] I. Klein, S. Filin, and T. Toledo. Vehicle Constraints Enhancement for Supporting INS Navigation in Urban Environments. *Navigation*, 58(1):7–15, 2011.

[28] F Landis Markley. Attitude Error Representations for Kalman Filtering. *Journal of Guidance, Control, and Dynamics*, 26(2):311–317, 2003.

[29] S. McGinnity and G.W. Irwin. Multiple Model Bootstrap Filter for Maneuvering Target Tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 36(3):1006–1012, 2000.

[30] Sriram Narasimhan, Richard Dearden, and Emmanuel Benazera. Combining Particle Filters and Consistency-Based Approaches for Monitoring and Diagnosis of Stochastic Hybrid Systems. In *15th International Workshop on Principles of Diagnosis (DX04), Carcassonne, France*. Citeseer, 2004.

[31] Nadia Oudjane and Christian Musso. Progressive Correction for Regularized Particle Filters. In *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, volume 2, pages THB2–10. IEEE, 2000.

[32] Ihnsoek Rhee, Mamoun F. Abdel-Hafez, and Jason L. Speyer. Observability of an Integrated GPS/INS During Maneuvers. *Aerospace and Electronic Systems, IEEE Transactions on*, 40(2):526–535, April 2004. read.

[33] B. Ristic, N. Gordon, and A. Bessell. On Target Classification Using Kinematic Data. *Information Fusion*, 5(1):15–21, 2004.

[34] Jonathan Ryan and David Bevly. Robust Ground Vehicle Constraints for Aiding Stand Alone INS and Determining Inertial Sensor Errors. In *Proceedings of the 2012 International Technical Meeting of The Institute of Navigation*, pages 374–401, January 2012.

[35] Jonathan Ryan, Jianbo Lu, and David Bevly. State Estimation for Vehicle Stability Control: A Kinematic Approach Using Only GPS and VSC Sensors. In *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*, Sept. 2010.

[36] Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund. Marginalized Particle Filters for Mixed Linear/Nonlinear State-Space Models. *Signal Processing, IEEE Transactions on*, 53(7):2279–2289, 2005.

[37] R.F. Stengel. *Optimal Control and Estimation*. Dover Pubns, 1994.

[38] Siamak Tafazoli and Xuehong Sun. Hybrid System State Tracking and Fault Detection Using Particle Filters. *Control Systems Technology, IEEE Transactions on*, 14(6):1078–1087, 2006.

[39] William Travis and David M Bevly. Navigation Errors Introduced by Ground Vehicle Dynamics. *Proceedings of the ION GNSS 2005*, pages 302–310, 2005.

[40] M. Turbe, K. Betts, D. Stranghoener, P. OLeary, T. Mitchell, J. Ryan, J. Wetherbee, and M. Carroll. GPS-Denied Field Test Results for Real-Time Plug-and-Play Navigation Software. In *Joint Navigation Conference, ION*, 2015.

[41] Tae Suk Yoo, Sung Kyung Hong, Hyok Min Yoon, and Sungsu Park. Gain-Scheduled Complementary Filter Design for a MEMS Based Attitude and Heading Reference System. *Sensors*, 11(4):3816–3830, 2011.

[42] Neil Zhao. Full-Featured Pedometer Design Realized With 3-axis Digital Accelerometer. *Analog Dialogue*, 44(06), 2010.

Appendix A

Partial Derivatives

$$F_k = \begin{bmatrix} F_{11} & F_{12} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ F_{21} & F_{22} & F_{23} & F_{24} & F_{25} & 0_{3\times3} & 0_{3\times3} & F_{28} & 0_{3\times3} \\ F_{31} & F_{32} & F_{33} & 0_{3\times3} & 0_{3\times3} & F_{36} & F_{37} & 0_{3\times3} & F_{39} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{44} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & F_{66} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{A.1}$$

$$F_{11} = \begin{bmatrix} -\frac{v_N}{(R_\lambda+h)^2}\frac{\partial R_\lambda}{\partial \lambda} & 0 & -\frac{v_N}{(R_\lambda+h)^2} \\ \frac{v_E \sec(\lambda)}{(R_\Phi+h)^2}\frac{\partial R_\Phi}{\partial \lambda} + \frac{v_E \sec(\lambda)\tan(\lambda)}{R_\Phi+h} & 0 & -\frac{v_E \sec(\lambda)}{(R_\Phi+h)^2} \\ 0 & 0 & 0 \end{bmatrix} \tag{A.2}$$

$$F_{12} = \begin{bmatrix} \frac{1}{R_\lambda+h} & 0 & 0 \\ 0 & \frac{\sec(\lambda)}{R_\Phi+h} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{A.3}$$

$$F_{23} = -R_B^N \left[\hat{a}^B \times\right] \tag{A.4}$$

$$F_{21} = \begin{bmatrix} Y_{11} & 0 & Y_{13} \\ Y_{21} & 0 & Y_{23} \\ Y_{31} & 0 & Y_{33} \end{bmatrix} \tag{A.5}$$

$$F_{22} = \begin{bmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & 0 \end{bmatrix} \tag{A.6}$$

$$F_{24} = -R_B^N(I_{3\times3} - \hat{\kappa}_a) \tag{A.7}$$

$$F_{25} = -R_B^N(a^B - \hat{b}_a) \tag{A.8}$$

$$F_{28} = -R_B^N(I_{3\times3} - \hat{\kappa}_a) \tag{A.9}$$

$$F_{33} = -\left[(I_{3\times3} - \hat{\kappa}_g)(\omega_{BI}^B - \hat{b}_g)\times\right] \tag{A.10}$$

$$F_{31} = -R_N^B \frac{\partial \omega_{NI}^N}{\partial p} \tag{A.11}$$

$$F_{32} = -R_N^B \frac{\partial \omega_{NI}^N}{\partial v} \tag{A.12}$$

$$F_{36} = -(I_{3\times3} - \hat{\kappa}_g) \tag{A.13}$$

$$F_{37} = -(\omega_{BI}^B - \hat{B}_g) \tag{A.14}$$

$$F_{39} = -(I_{3\times3} - \hat{\kappa}_g) \tag{A.15}$$

$$Y_{11} = -\frac{v_E^2 \sec^2(\lambda)}{R_\Phi + h} + \cdots \tag{A.16}$$

$$\frac{v_E^2 \tan(\lambda)}{(R_\Phi + h)^2} \frac{\partial R_\Phi}{\partial \lambda} - \cdots \tag{A.17}$$

$$2\omega_e \cos(\lambda) - \frac{v_N v_D}{(R_\lambda + h)^2} \frac{\partial R_\lambda}{\partial \lambda} \tag{A.18}$$

$$Y_{13} = \frac{v_E^2 \tan(\lambda)}{(R_\Phi + h)^2} - \frac{v_N v_D}{(R_\lambda + h)^2} \tag{A.19}$$

$$Y_{21} = \frac{v_E v_N \sec^2(\lambda)}{R_\Phi + h} - \cdots \tag{A.20}$$

$$\frac{v_E v_N \tan(\lambda)}{(R_\Phi + h)^2} \frac{\partial R_\Phi}{\partial \lambda} + \cdots \tag{A.21}$$

$$2\omega_e \cos(\lambda) - \frac{v_E v_D}{(R_\Phi + h)^2} \frac{\partial R_\Phi}{\partial \lambda} - 2\omega_e v_D \sin(\lambda) \tag{A.22}$$

$$Y_{23} = -v_E \left[\frac{v_N \tan(\lambda) + v_D}{(R_\Phi + h)^2}\right] \tag{A.23}$$

$$Y_{31} = \frac{v_E^2}{(R_\Phi + h)^2} \frac{\partial R_\Phi}{\partial \lambda} + \cdots \tag{A.24}$$

$$\frac{v_N^2}{(R_\lambda + h)^2} \frac{\partial R_\lambda}{\partial \lambda} + \cdots \tag{A.25}$$

$$2\omega_e v_E \sin(\lambda) + \frac{\partial g}{\partial \lambda} \tag{A.26}$$

$$Y_{33} = \frac{v_E^2}{(R_\Phi + h)^2} + \frac{v_N^2}{(R_\lambda + h)^2} + \frac{\partial g}{\partial h} \tag{A.27}$$

$$\tag{A.28}$$

$$Z_{11} = \frac{v_D}{R_\lambda + h} \tag{A.29}$$

$$Z_{12} = -\frac{2v_E \tan(\lambda)}{R_\Phi + h} + 2\omega_e \sin(\lambda) \tag{A.30}$$

$$Z_{12} = \frac{v_N}{R_\lambda + h} \tag{A.31}$$

$$Z_{21} = \frac{v_E \tan(\lambda)}{R_\Phi + h} + 2\omega_e \sin(\lambda) \tag{A.32}$$

$$Z_{22} = \frac{v_D + v_N \tan(\lambda)}{R_\Phi + h} \tag{A.33}$$

$$Z_{23} = \frac{v_E}{R_\Phi + h} + 2\omega_e \cos(\lambda) \tag{A.34}$$

$$Z_{31} = -\frac{2v_N}{R_\lambda + h} \tag{A.35}$$

$$Z_{32} = -\frac{2v_E}{R_\Phi + h} - 2\omega_e \cos(\lambda) \tag{A.36}$$

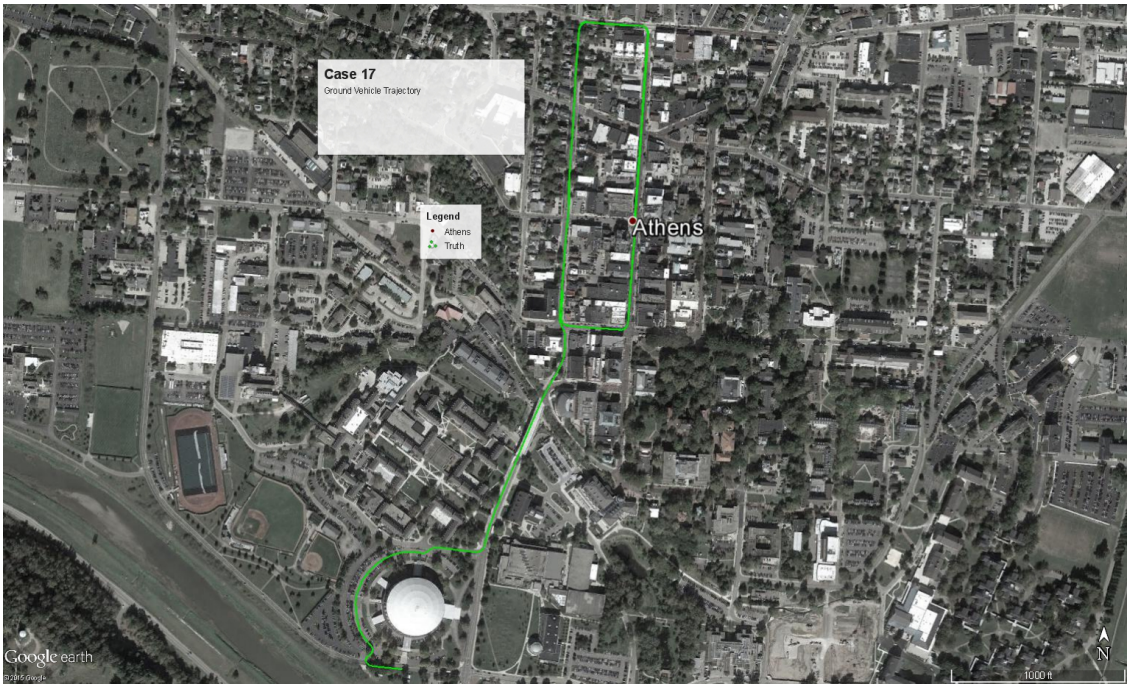
Appendix B

Trajectories of Various ASPN Data Sets

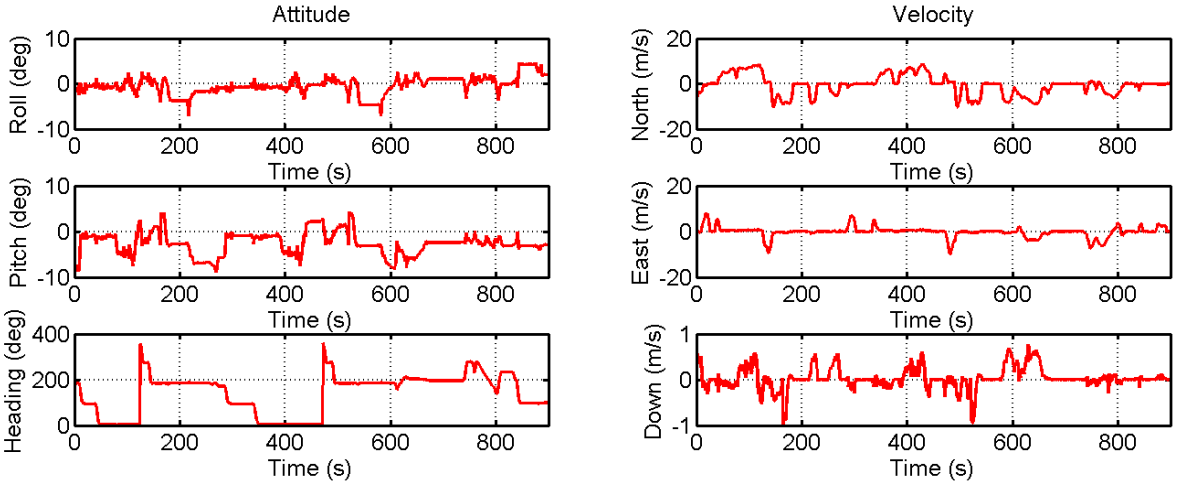
Figure B.1 shows the true navigation state trajectory of case 17 from the ASPN data sets. Figure B.1a gives the position overlaid onto Google Earth, Figure B.1b gives the attitude in Euler angles vs. time, and Figure B.1c gives the NED velocity vs. time. The vehicle makes two clockwise loops through downtown Athens, stopping in the normal flow of traffic, before proceeding south and looping around the circular building in the bottom of the image. The data set is 15 minutes long. The elevation stays within a 30 m range throughout the run, and the largest road slope is 6 degrees. The vehicle speed stayed below 25mph throughout the run.

Figure B.2 gives the state trajectory for the aerial case, case 19. For reference, the trajectory for case 17 is also shown in blue. The aircraft flies from west to east over Athens, turns south and then back west towards Athens, and then turns north over the city for the remainder of the flight. This run is 10 minutes long. The aircraft maintains altitude of between 665 m and 710 m during the run; the flight is pretty level. When making the largest turns the aircraft sustains bank angles of around 15 degrees, with spikes of up to 25 degrees. The absolute speed of the aircraft is between 110 mph and 170 mph during the run, or approximately 95 knots to 145 knots.

Figure B.3 gives the state trajectory for a pedestrian case, case 11. The dismount remains standing in one location for the first 50 seconds before heading west along the sidewalk and turning south into the building. After entering the building the dismount makes two loops in the clockwise direction before stopping at the same entrance. The total time elapsed is just over five and a half minutes. Once the dismount begins walking, he stops



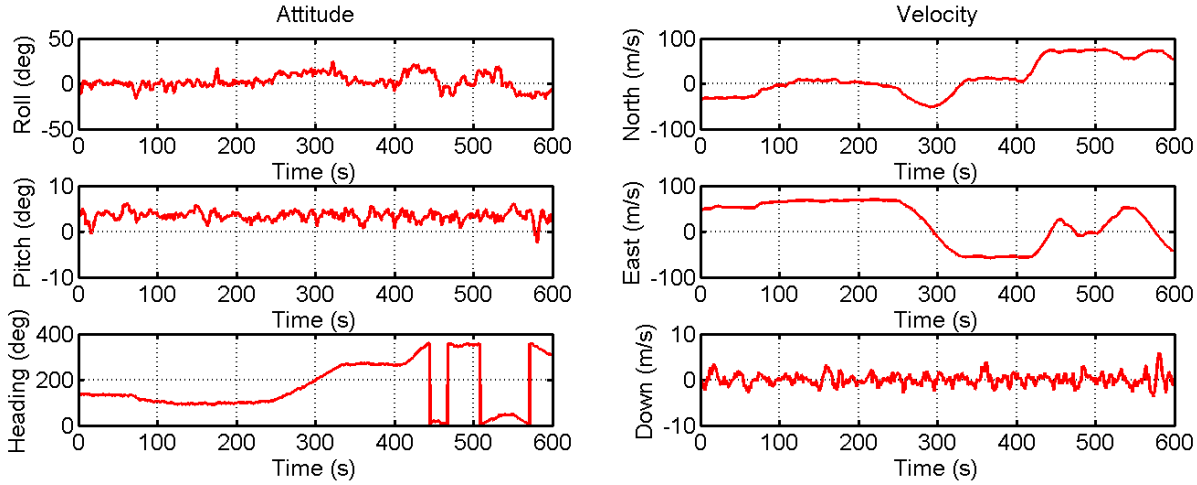
(a)



(b) (c)

Figure B.1: Case 17.

(a)



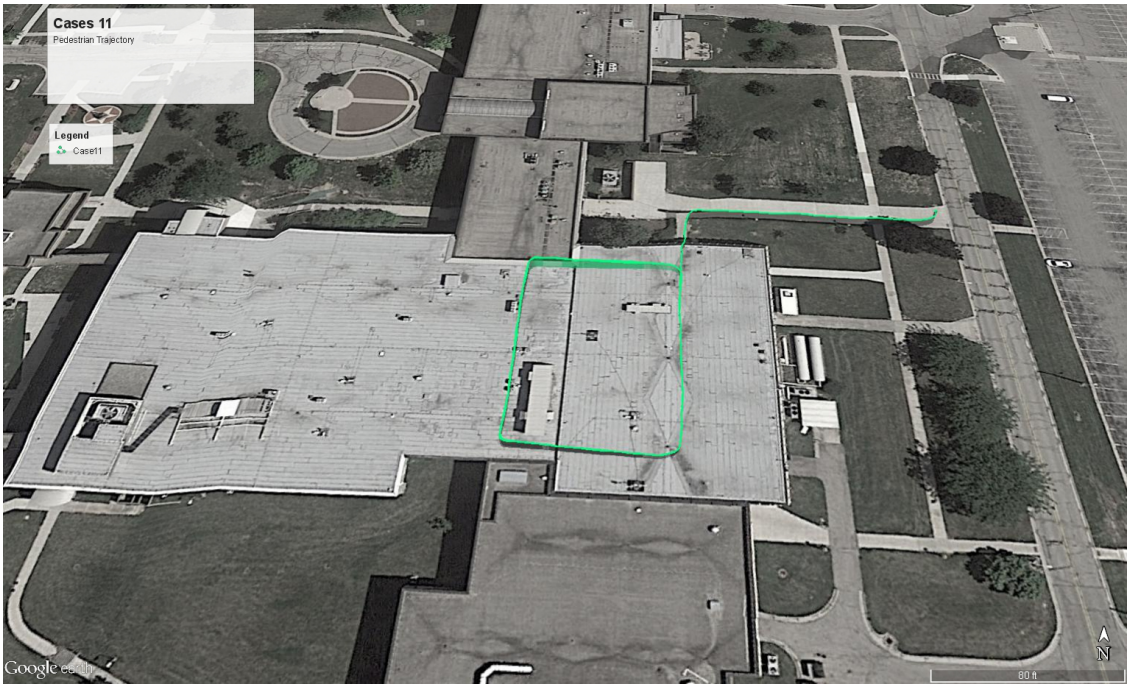
(b) (c)

Figure B.2: Case 19 (17 also shown).

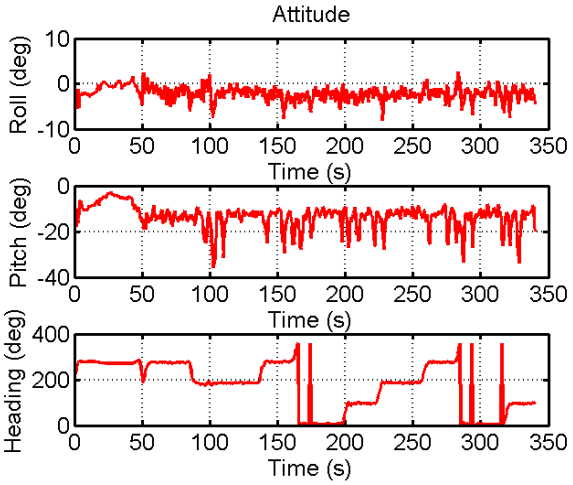
only to enter the building and remains in motion the rest of the time. The walking speed of the dismount between 1.75 mph and 3 mph consistently.

Figure B.4 gives the position of the dismount for case 10 overlaid onto Google earth. Note that the trajectory of case 11 is shown for reference. There is no truth attitude or velocity information available for this data set. In this case the dismount begins outside the southeast corner of the building and enters at this corner. The dismount proceeds west through the length of the building and completes one loop around the larger western portion of the building. Following this, the dismount again heads west but turns north in a nearer hallway before turning back east and ending the run at the northeast corner of the larger loop. This run lasts just over five and a half minutes.

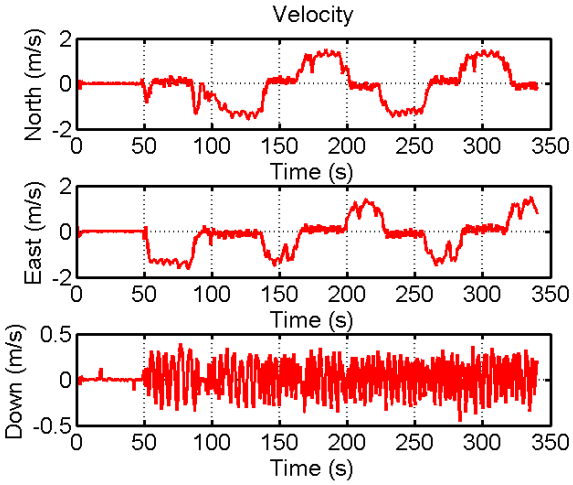
Figure B.5 gives the truth state trajectory of the aircraft for case 216. Case 216 is an aircraft case where the data collection begins on the ground before takeoff. The aircraft takes off from the Ohio University airport heading southwest before turning around and flying back over the airport. The aircraft then turns eastward to pass over Athens before continuing north. The aircraft is ascending for the most of this case until leveling off at an altitude of just over 9000 feet. The run is 1450 seconds long, but the first 450 seconds are just the aircraft sitting still prior to taxi and are omitted from the analysis done in this work. The aircraft reaches a maximum speed of 212 m/s in this case, or 184 knots. It is interesting to note during the intial static period of the data ($t < 450s$) that the pitch angle is approximately positive 10 degrees. This is because the DC-3 which hosted the data collect is a "tail dragger" in the old style of aircraft, and so it sits at an incline when at rest.



(a)



(b)



(c)

Figure B.3: Case 11.

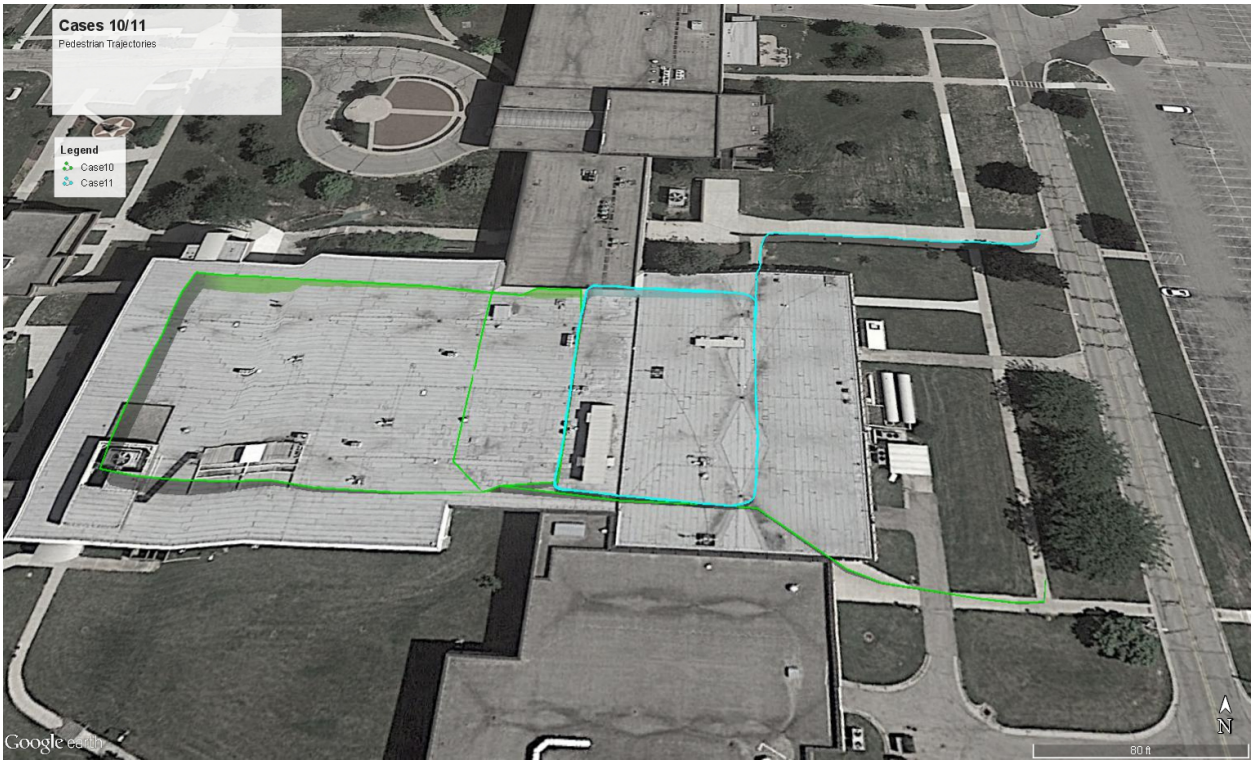
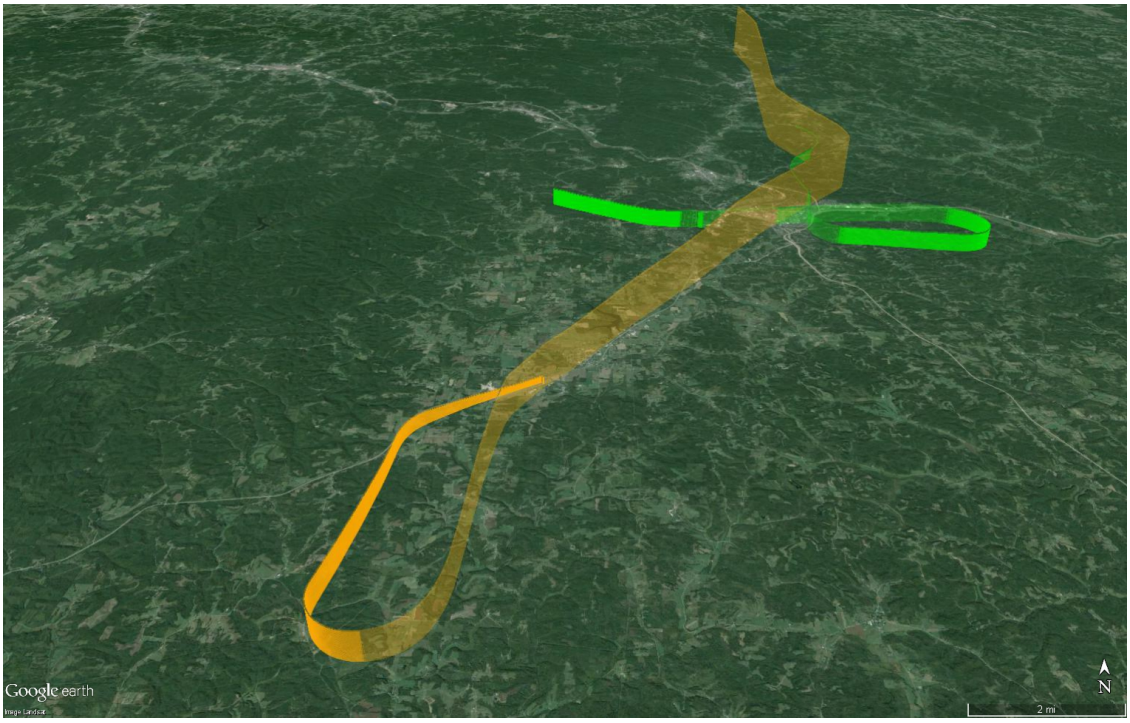
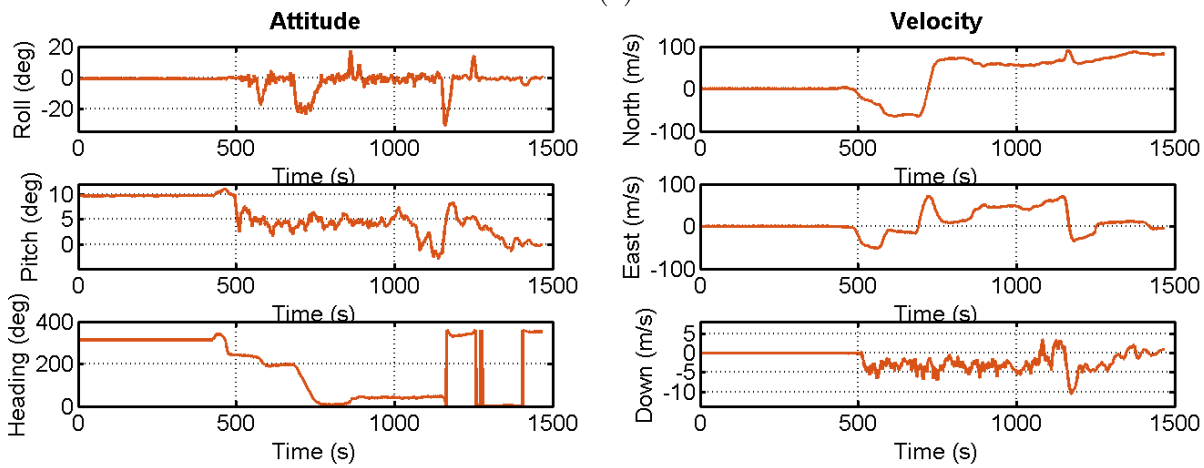


Figure B.4: Case 10 (11 also shown).



(a)



(b) (c)

Figure B.5: Case 216 (orange), 19 also shown (green)).