Initialization of a Pedestrian Navigation System Using a Transfer Alignment Approach

by

Archit Sudarsan Thopay

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 8, 2020

Keywords: Pedestrian Navigation, Transfer Alignment, Wahba's Problem, Kalman Filtering

Abstract

Pedestrian navigation techniques are used to provide positioning information in lieu of or to supplement more traditional systems such as GNSS. A challenge facing users of pedestrian navigation algorithms is the acquisition of initialization data. Existing initialization methods for pedestrian navigation methods assume the availability of GNSS position and velocity and of standstill periods that can be used to provide attitude information. This thesis presents a method that performs the attitude initialization for a pedestrian navigation system (PNS) while the pedestrian is riding inside a vehicle. An algorithm is presented which uses data from a vehicle-mounted IMU and a pedestrian-mounted IMU to detect when the two sensors form a rigid body. Once rigidity has been established, a solution to Wahba's Problem is used to solve for the misalignment between the two systems. The misalignment is used in conjunction with the vehicle's attitude to initialize the PNS before the pedestrian leaves the vehicle. Performance analyses of the rigidity detector and misalignment estimation are shown using both simulated and real data, while a performance analysis for the PNS initialization procedure is shown using real data. The method is implemented using IMU data gathered using a Vectornav VN-300 GPS/INS and a Lincoln MKZ instrumented with a KVH 1750 IMU. Results show that the proposed method can initialize a PNS with a mean initial heading error of 7.3°, with the error being attributable to violations of the rigidity condition.

Acknowledgments

First and foremost, I would like to thank my parents Raji and Sudarsan for their constant love and support and words of wisdom. Without them, the completion of this degree would not have been possible. I hope that one day I will be able to repay them for the love they have shown me. This thesis is dedicated to them. I would also like to thank my younger brother Arjun for helping shape me into the person I am today and helping me keep perspective throughout my time in graduate school.

Two people who deserves huge thanks are my high school band director Ryan Fitchpatrick and my AP US History teacher Richard Sturgeon. They are huge sources of inspiration to me because of their dedication to their students, and their passion for teaching. Both of these men changed my life for the better when they fostered enjoyment of learning in me.

I must also recognize the many coworkers I have had the pleasure of working with during my time in the GAVLAB. These men and women were directly responsible for enriching my graduate school experience. The first person I must thank is Dr. Howard Chen, as he was the first person to work with me when I entered the lab. His mentorship has been absolutely critical in my development as a researcher. A debt of gratitude goes to David Bell for his many hours of help with data collection. Without it, this work would surely not have been possible. Similarly, I would like to thank Brendan Schretter for his help with the Lincoln MKZ, especially for all those times I forgot the login credentials. I would also like to mention Stephanie Meyer and Christian Campos for being true friends who I have come to value considerably. The other folks (Trenton Hilyer, Sterling Thompson, Tyler Flegel, and Louis Buckalew) that made up the ‘Dungeon’ office also deserve thanks for the many interesting discussions, helpful input, and occasional pranks that made work fun.

One person that deserves special recognition is Ryan McWilliams. Since my underclassman days in college, he has been one of my closest friends and most reliable sources of support. I will cherish the many memories we have created together over the years, including inside jokes, insightful conversations, and trips, for a lifetime. Another person that needs to be recognized is Lakshmi Krishnaprasad. Throughout my time in Auburn, she has been a consistent source of reassurance and support, with many valuable insights about people and life. I will not soon forget the inspiration and wisdom that she has imparted on me.

A special thanks goes out to the Chick-fil-A and Chipotle locations on Magnolia Avenue for providing me tasty and easy-to-access nourishment when I was on campus. I would also like to thank the Auburn University Swing Dance Association for providing me a creative outlet that helped me de-stress take my mind off research once every week.

I would also like to thank the members of my committee Dr. Luke Oeding and Dr. Scott Martin for reviewing this document and advising throughout my time in graduate school.

Finally, I would like to thank Dr. David Bevly for the opportunity to work in the GAVLAB, as it has changed the course of my life for the better. I have had chances to work on many interesting projects and learn about various topics that I would not have been able to otherwise.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The problem of navigating, or determining *where we are*, is one that stretches back ages [1]. The Concise Oxford Dictionary defines navigation as "any of several methods of determining or planning a ship's or aircraft's position and course by geometry, astronomy, radio signals, etc." This definition has two major components. The latter component refers to the planning of a trajectory from one location to another, avoiding obstacles and collisions along the way. This is known as *guidance* . The former is the determination of a moving object's position with respect to a known reference. This is referred to as *positioning* [2].

Accurate and reliable positioning information is critical for many applications, including surveying, asset and animal tracking, vehicular navigation, and pedestrian navigation [2]. Starting in the early modern period, driven by the demands of commerce and colonization, traders and sailors carefully and painstakingly calculated their positions using observations of celestial bodies, mechanical clocks, and dead reckoning techniques [4]. Since then, advances in timekeeping, digital signals, computing, microelectronics, and spacecraft have culminated in the development of global navigation satellite systems (GNSS) in the mid to late 20th century. These systems, including American GPS and Russian GLONASS, transmit positioning information from satellites to users on Earth, providing anyone with a receiver a precise, instantaneous, externally-referenced position anywhere on the planet at any time [4].

GNSS is an attractive source of position information for many applications due to its accuracy, reliability, worldwide availability, ease of use, and ubiquity i.e. GNSS receivers are small and cheap enough to be in smartphones [5]. However, due to GNSS's reliance on signal transmission and reception, it suffers from several inherent weaknesses. The signals broadcast by the satellites are low power, so they are easily blocked or attenuated by objects such as buildings or trees, which makes their usefulness for indoor and forested areas limited. In urban areas, signals from some satellites will be blocked while others will be received after they have been reflected off buildings. This multipath effect leads to degraded position accuracy. Finally, GNSS signals are open to attacks such as spoofing and jamming, which can result in false navigation solutions or even navigational outages [2, 5]. These weaknesses can make it difficult to use GNSS in certain contexts.

One such context is the navigation of pedestrians. Pedestrian navigation is a problem with many useful applications, which include geolocation of dismounted soldiers, navigation for the blind and visually impaired, and the tracking of first responders [5]. However, the problem is complicated by the fact that pedestrians often operate in environments that are challenging for GNSS signals. These include indoor, urban, and forested areas. Since the accuracy and reliability of GNSS is compromised in such environments, it has been necessary to develop alternative navigation systems for pedestrians that operate independently of GNSS.

Several alternative systems have been developed that provide pedestrians with externally-referenced positioning information independently of GNSS. As with GNSS, these systems rely on pre-existing knowledge of the environment in the form of physical infrastructure or maps and usually involve the transmission and reception of signals to provide positioning information. Technologies include camera systems, infrared receivers, ultrasound, and short-range radio networks such as wireless local area network (WLAN), ultra-wide-band (UWB) radios, bluetooth low energy (BLE), and radio-frequency identification (RFID) [6, 8–10]. These technologies can be used to identify when a pedestrian is located next to an infrastructural or mapped-out point, such as a wireless router or visual landmark [8], or they can use more sophisticated techniques, such as received signal strength (RSS), time of flight (ToF), or angle of arrival (AoA) to determine the location of the pedestrian in the environment via trilateration or triangulation [6, 10]. These alternative methods are potential supplements or even replacements for GNSS when it is unavailable. However, as with GNSS, these systems are subject to blockages, multipath, and other factors that contribute to a poor signal environment. Furthermore, these systems require the installation of physical infrastructure or generation of maps *before* being used for navigation. This process, especially for physical infrastructure, is generally difficult, time-consuming, and costly, which may make implementation prohibitive [5].

Due to inherent difficulties with infrastructure or map-based navigational techniques, there is a need for pedestrian navigation techniques that operate independently of externally-referenced information. Dead-reckoning is a technique that meets this need. In the pedestrian navigation context, dead reckoning involves propagating a pedestrian's position using acceleration, angular velocity, and magnetic field measurements from Inertial Measurement Units (IMUs) and magnetometers worn by the pedestrian, and supplements these sources of information by making assumptions about human motion [5, 7, 11, 12, 25]. Two recent developments make implementing this technique practical. The first is increases in processing power of mobile computers has made real-time implementation of pedestrian dead reckoning practical, as shown in [25]. The second is advances in sensor technology have resulted in inertial and magnetic sensors that are small, both in size and power consumption, allowing for them to be worn by people [2]. A pedestrian dead-reckoning system based on these technologies can provide a real-time position solution to a pedestrian when external navigation infrastructure or maps are unavailable or unusable. However, a prerequisite to using dead-reckoning for positioning is the establishment of initial position and attitude information. For pedestrians, acquiring this information can itself be a challenging problem.

In existing literature, this problem is simplified by assuming that the pedestrian is standing still in an outdoor environment. This allows GNSS to be used for initializing position and velocity, gravity measurements to be used for initializing pitch and roll, and magnetic field measurements or user inputs to be used for initializing heading [13, 15–18, 27, 28, 31] . Outdoor standstill initialization has been sufficient for researching different pedestrian dead reckoning algorithms. For real-world applications though, requiring a pedestrian to stand still to allow the dead-reckoning algorithm in their personal navigation system to initialize may be impractical.

Fortunately, recent developments have created an opportunity to develop a more practical means of acquiring initialization information. As vehicles become more automated, they are being equipped with navigation sensors, including IMUs, of a level of quality that is sufficient for practical use [19, 20]. To take advantage of this development, this thesis will explore an IMU-based technique to initialize a pedestrian dead-reckoning algorithm by transferring a navigation solution from the vehicle's navigation system while the pedestrian is riding inside the vehicle.

1.2 Thesis Contributions

In general, the objective of this thesis is to develop an implementation of transfer alignment for use on an intermittently rigid body. For the purposes of this thesis, "intermittently rigid" refers to random instances in which an otherwise rigid body experiences deformations. In the context of in-vehicle PDR initialization, deformation occurs when the the pedestrian's body moves independently of the vehicle's while riding inside of it. To develop an initialization system applicable in this setting, this thesis:

- Examines existing methods of misalignment and lever arm estimation and analyze their observability.
- Develops a method to detect when the vehicle and pedestrian IMUs form a rigid body.
- Integrates transfer alignment and rigidity detection into a combined system that can be used to initialize a pedestrian navigation system using a vehicle navigation system.
- Evaluates the performance of the initialization system.

1.3 Thesis Outline

This thesis has six remaining chapters. Chapter 2 provides a technical background for general navigation, and focuses on coordinate frames, rotation translation transformations, and inertial navigation techniques. Chapter 3 provides an overview of existing pedestrian dead reckoning techniques. Chapter 4 provides an overview of existing transfer alignment techniques, including methods used to estimate misalignment and lever arm, and covers an analysis of these quantities' observability. Chapter 5 discusses zero-velocity detection and outlines a method by which a relative zero velocity or rigidity state between two IMUs can be detected. Chapter 6 brings together the elements discussed in the previous two chapters to implement the in-vehicle initialization system and presents an evaluation of its performance. The final chapter summarizes the thesis, provides conclusions, and discusses possible directions for future work.

Chapter 2

Technical Background

2.1 Coordinate Frames

Coordinate frames are a foundational mathematical concept for navigation. Their purpose is to provide an origin and a set of axes to describe an object's linear and angular motion. Discussion of this concept is imperative, since all navigation problems involve determining an object's position and velocity relative to a chosen coordinate frame.

It is customary to use orthogonal three-dimensional coordinate systems when describing navigation problems, i.e. frames with an x, y, and z axis. This means that most navigation information, such as position, velocity, acceleration, angular velocity, orientation, etc. are described with a minimum of three components. It is also customary to use at least two frames when describing a navigation problem, these frames being the object frame and the reference frame. The object frame A describes an object whose position and orientation are desired, while the reference frame B describes a known body, such as the Earth or Sun, which provides a datum for the position and orientation of the object of interest. Therefore, a full description of the object's navigation information is given as "*the position and orientation of A with respect to B.*" Sometimes, the navigation information is expressed in a third frame, the resolving frame C. In that case the object's navigation information would be given as "*the position and orientation of A with respect to B in C.*" What follows is a discussion of some commonly used coordinate frames that will be referenced in this thesis.

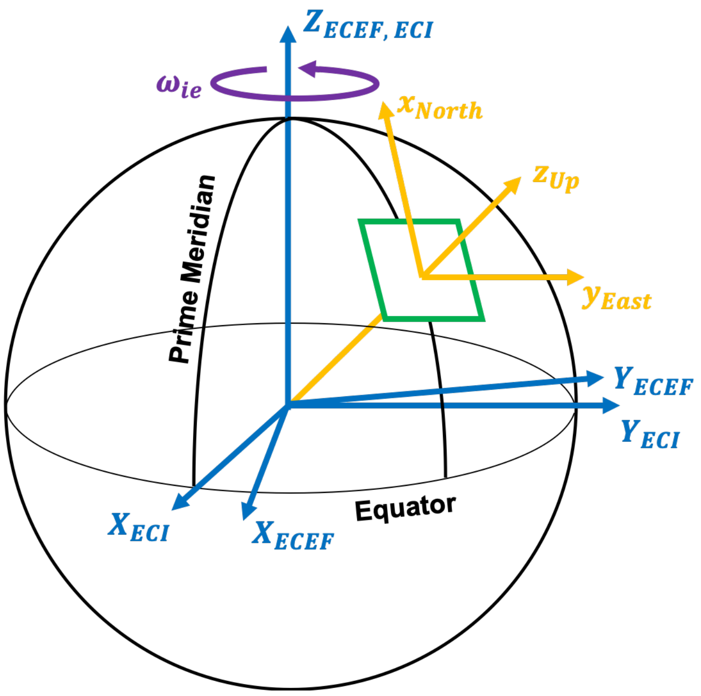

Figure 2.1: The ECI, ECEF, and ENU coordinate systems shown on a sphere representing the Earth.

2.1.1 Earth-Centered Inertial and Earth-Fixed Frames

The Earth-Centered Inertial Frame (ECI), denoted by i, and the Earth-Centered Earth-Fixed Frame (ECEF), denoted by e, are coordinate frames used to describe motion relative to the Earth. These coordinate frames are widely used as resolving or reference frames since users often want to know their position relative to the Earth. These two frames have similar coordinate axes. As shown in Figure 2.1, they both share a z-axis pointing from the center of the Earth to true North and perpendicular x and y-axes co-planar with the equator. Where they differ is in their movement. ECI is an inertial coordinate frame, which means that it does not accelerate or rotate with respect to the rest of the universe. By contrast, ECEF rotates with the planet at a rate of $\omega_{ie} = 7.292115 \times 10^{-5}\, rad/s$. The choice of which frame to use is highly application-dependent. For space-based applications such as GPS satellites, ω_{ie} is non-trivial, so ECEF must be used. For many ground-based applications, such as pedestrian navigation, the Earth's rotation rate can be safely assumed to be zero, so ECI can be used instead. Though this assumption is technically inaccurate, since the planet does in fact move, the error produced by ignoring the rotation rate is negligible for these applications.

2.1.2 Local Navigation Frame

For many navigation applications, it is also necessary to know the orientation of the navigating object with respect to the Earth. However, reading the orientation of an object resolved in ECI or ECEF can be counter-intuitive, especially when the orientation is expressed in Euler angles. For this reason, it is helpful to define a local navigation frame, denoted by n, as a small plane that coincides with the navigating object, as shown in Figure 2.1. A common axis convention is to point the z-axis down from the object towards the center of the Earth, the x-axis towards true north, and the y-axis towards the east, completing the orthogonal set. Another common convention is to use x (East), y (North), z (Up) coordinates. Reading the object's orientation resolved in either North-East-Down (NED) or East-North-Up (ENU) is much more intuitive than reading them resolved in ECEF.

2.1.3 Tangent Plane Frame

Reading positions and velocities resolved in ECI or ECEF can be similarly counter-intuitive. One solution to this problem is to take the positions resolved in ECEF, which is a cartesian coordinate system, and instead express them in a spherical coordinate system. Latitude, longitude, and altitude (LLA) is an example that is commonly used in navigation. Using LLA to express positions allows users to express positions on the Earth's surface using just two numbers, latitude and longitude, rather than needing a full set of three. Note that LLA is *not* a new coordinate system. It is simply a means of expressing vectors resolved in ECEF with a different set of numbers.

Another solution is to take the local navigation frame and affix it to a predetermined point on the earth's surface. This predetermined point, or point of tangency, is the origin for the tangent plane frame, also denoted by n. Using this new frame allows for an object's position and velocity to be expressed relative to a known landmark or point of interest, such as the starting point on a test track or a building entrance. Resolving vectors in such a frame makes position and velocity data easier to intuitively understand. However, using a flat tangent plane to represent the spherical surface of the Earth creates a positional error that is proportional to the position's distance from the tangent plane's origin. Therefore it is advisable to only resolve positions and velocities in a stationary tangent plane frame close to the chosen origin point.

2.1.4 Body Frame

Nearly all navigation application involve positioning some moving object. Therefore, it is imperative that there be a body frame attached to that object, as shown in Figure 2.2. This frame, denoted by b, is attached to the moving object much like the local navigation frame. However, unlike that frame, these axes remain fixed to the body. They are usually defined as x (Forward), y (Right), and z (Down) or x (Right), y (Forward), and z (Up). For angular motion, the x-axis is the roll axis, the y-axis is the pitch axis, and the z-axis is the yaw axis. For objects using inertial-based navigation systems, the axes of the inertial sensor on the object can be set to correspond to the body frame axes.

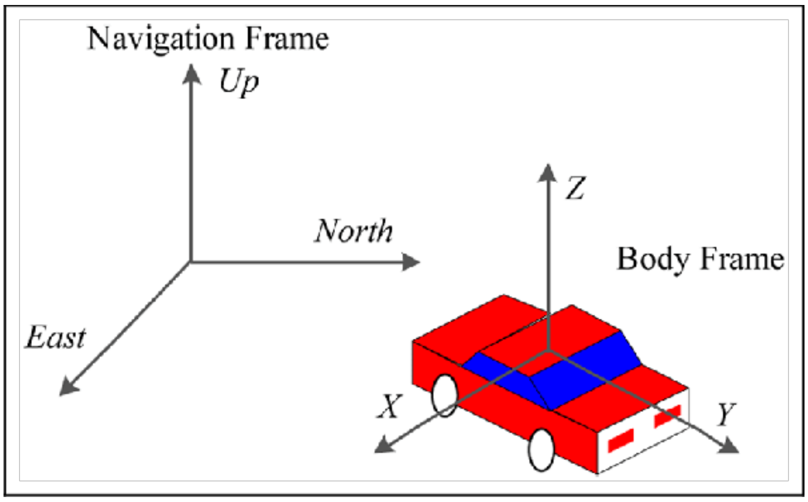

Figure 2.2: A body frame affixed to a vehicle.

2.2 Coordinate Frame Transformations

More often than not, a navigation problem with data resolved in one frame will require that information be resolved in another frame. This process is called coordinate frame transformation. One reason to perform this is simple convenience, i.e. transforming from ECEF to a local tangent plane to make position and orientation data easier to understand. Another is to resolve all available data in a problem into a common frame to make mathematical operations possible. For these reasons it is essential to possess a means to transform vector information from one coordinate frame to another.

2.2.1 Direction Cosine Matrices

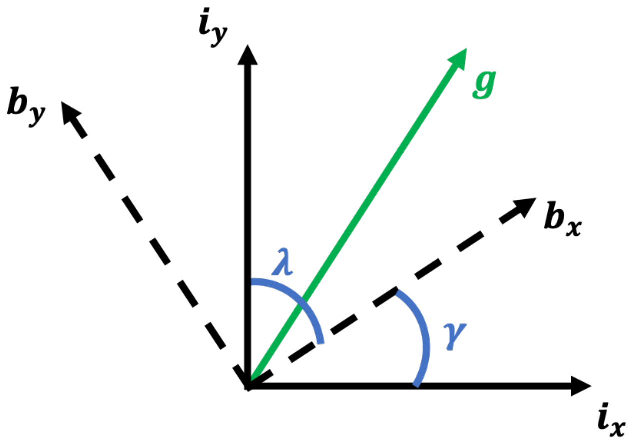

Figure 2.3: A vector g resolved in two coordinate frames

Consider the scenario shown in Figure 2.3. A vector g of arbitrary length can be resolved in two different coordinate frames, i and b. The question is, what relationship is there between the vector components resolved in each frame? Using the direction angles given, the component of g along the b_x axis is mapped onto the i frame by,

$$g_x^b = g_x^i cos(\gamma) + g_y^i cos(\lambda). \tag{2.1}$$

The $cos(\gamma)$ and $cos(\lambda)$ terms are known as direction cosines. Equation (2.1) can be extended to g_{b_y} as well. For neatness, the equations can be packaged together as,

$$\begin{bmatrix} g_x^b \\ g_y^b \end{bmatrix} = C \begin{bmatrix} g_x^i \\ g_y^i \end{bmatrix} = \begin{bmatrix} cos(\gamma) & cos(\lambda) \\ -cos(\lambda) & cos(\gamma) \end{bmatrix} \begin{bmatrix} g_x^i \\ g_y^i \end{bmatrix}. \tag{2.2}$$

The matrix C, made up of direction cosine terms, is termed a *Direction Cosine Matrix* (DCM). It transforms the vector g from the i frame to the b frame. Note that the DCM is not dependent on the length of g. It depends only the relative orientation between the frames. Though Equation (2.2) applies only to a 2D transformation, a similar process can be followed for deriving an 3-dimensional DCM. C is usually appended with a subscript and superscript for the sake of specificity, i.e. C_i^b. The notation signifies a matrix that takes a vector quantity $\vec{x}$ in the i resolving frame and transforms it to the b resolving frame, i.e.

$$\vec{x}^b = C_i^b \vec{x}^i. \tag{2.3}$$

To transform the vector from frame b back to frame i, Equation (2.3) must be multiplied by the inverse of C_i^b, which is notated C_b^i. If the process used to originally develop C was used to develop C_b^i, the resulting matrix would be equivalent to taking the transpose of C_i^b. In other words,

$$C_b^i = (C_i^b)^{-1} = (C_i^b)^T. \tag{2.4}$$

This fact is especially useful in computer implementations, since it allows a DCM's transpose to be taken instead of its inverse, an operation that is far more computationally expensive. Performing a transformation and then reversing the process must return the original vector so,

$$C_b^i (C_b^i)^{-1} = C_b^i (C_b^i)^T = C_b^i C_i^b = I_3. \tag{2.5}$$

Thus, direction cosine matrices are orthonormal. A property of such matrices is that

$$\det(C_b^i) = 1, \tag{2.6}$$

which is helpful as a computation-checking tool. Stringing together multiple transformations is straightforward. Consider the two transformations,

$$\vec{x}^c = C_b^c \vec{x}^b \quad \text{and} \quad \vec{x}^b = C_a^b \vec{x}^a. \tag{2.7}$$

Substitution yields

$$\vec{x}^c = C_b^c C_a^b \vec{x}^a = C_a^c \vec{x}^a. \tag{2.8}$$

2.2.2 Euler Angles

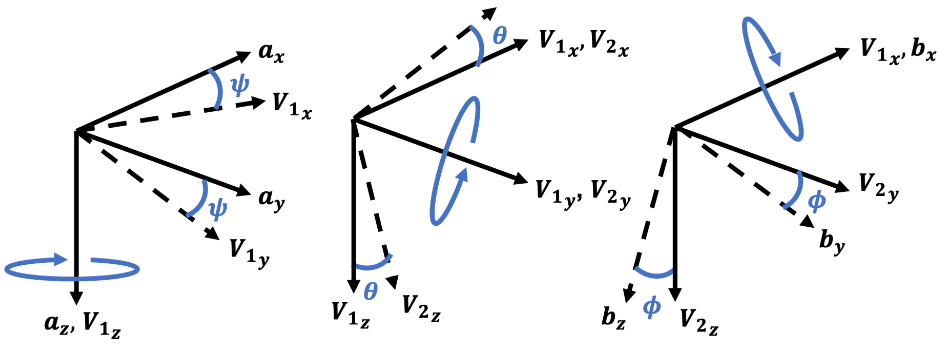

Figure 2.4: A series of rotations about the z, y, and x axes

Though DCMs are useful in transforming vectors, they are not the most intuitive means of representing a coordinate frame transformation. A more visually friendly representation can be found in the direction angles between the coordinate frames. For a 2D transformation, one of the two angles in Figure 2.3 can be used to represent the entire transformation. For a 3D transformation, a minimum of three unique angles are needed [2, 21]. A rotation sequence widely used for navigation, particularly in aerospace applications, is shown in Figure 2.4. To start off, the reference frame i is rotated around the z-axis by some angle ψ. This yawing motion makes ψ the angle between the reference frame i and the intermediate frame V_1. Then, the V_1 frame is pitched forward about the y-axis by an angle θ to create a second intermediate frame V_2. The rotation sequence is finished by rolling the V_2 frame by angle ϕ about the x-axis to the object frame b. This particular sequence of ZYX rotations, or *Euler angles*, is given as,

$$\epsilon_i^b = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \tag{2.9}$$

There are three distinctive aspects of the approach. The first aspect is the performance of the second and third rotations from the axes of the intermediate frames V_1 and V_2. This sort of rotation sequence is known as a body-fixed rotation sequence [21]. The other option is to perform the rotations around the axes of the base frame i. This alternative is known as a space-fixed rotation sequence [21]. Though this difference may seem significant, it is shown in [21] that the result of a sequence of body-fixed rotations leads to the same final orientation as the reverse sequence of space-fixed rotations, and vice-versa. Since body-fixed rotations are easier to visualize, they will be used in this thesis.

The second aspect is the number of unique axes used to perform the rotation sequence. Though it is helpful to use all three axes to perform a 3D rotation, as done in Figure 2.4, it is not a hard requirement. 3D rotations can be performed using just two axes. Using Figure 2.4 as an example, the b frame can be formed by rotating the V_2 frame by ϕ around its z-axis instead of its x-axis. This makes it a ZYZ sequence rather than a ZYX sequence. A ZYZ sequence is known as a symmetric set due to the repetition of the first rotation axis in the third rotation. Any two axes can be used in any order, giving a total of 6 possible symmetric angle sets, as shown in [22]. By contrast, a sequence that uses all three axes, such as the ZYX sequence in Figure 2.4, is known as an asymmetric angle set. The three axes can be arranged in any order, giving a total of 6 possible asymmetric angle sets, as shown in [22]. Due to their widespread use in navigation applications [2], the ZYX asymmetric set will be used in this thesis.

The third aspect is the specific order of the axial rotations used to create the intermediate frames. Generally, if anything about the generation of an intermediate frame is changed, such as the rotation axis or angle, then it follows that the final orientation will also change. This fact has some interesting effects. One effect is the criticality of the rotation order. The sequence shown in Figure 2.4, which uses a ZYX order, will not give the same final orientation were it to be done as an XYZ sequence. Another effect is the non-triviality of inversion. Simply changing the sign on the angles does not rotate from b back to i. A new set of Euler angles must be used, or the three rotations must be individually reversed in the opposite order they were performed. A further effect is the complication of successive transformations. Adding the angles and performing one rotation set with the sum will not give the desired result. Instead, the two rotation sequences must be performed individually.

A difficulty, specific to ZYX rotations, is the singularity that occurs at pitch angles of $\theta = \pm 90°$. A visual example of this phenomenon is given by the fighter jet in Figure 2.5. The jet can arrive to the shown attitude by either yawing and then pitching, or by pitching and then rolling, i.e. the roll and yaw become indistinguishable at $\pm 90°$. This effect, known

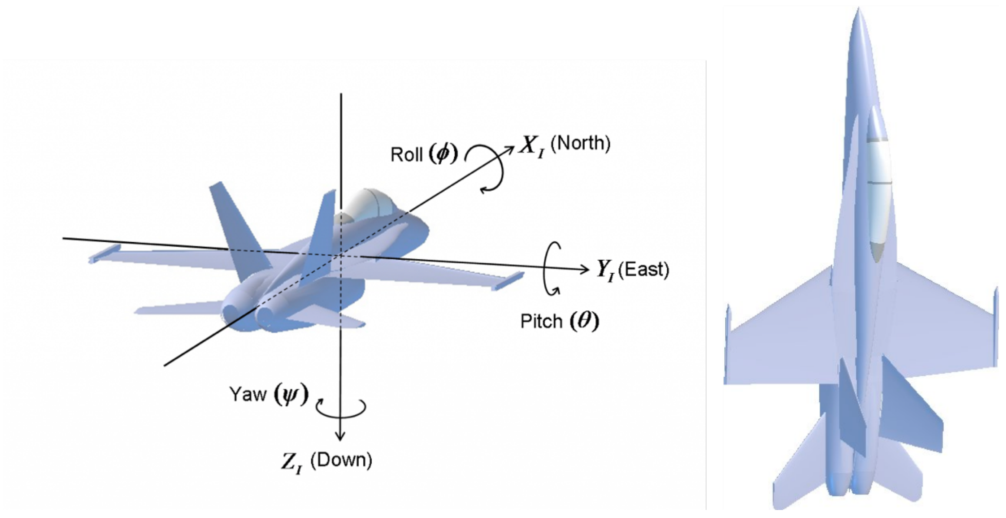

Figure 2.5: A fighter jet illustrating the problem of gimbal lock

as gimbal lock, limits the use of Euler angles to applications where the pitching is within the $90°$ bound. Despite these mathematical difficulties, Euler angles remain a popular choice for communicating coordinate frame transformations due to their intuitiveness. However, their inherent difficulties mean they are rarely used to mathematically implement coordinate frame transformations.

Since the coordinate frame transformation can be represented as a series of rotations, they are sometimes referred to as rotations or rotation transformations. The resulting orientation after the series of rotations is often referred to as an *attitude*, a term that will be frequently used in this thesis.

2.2.3 Euler Angles and Direction Cosine Matrices

Euler angles can instead be indirectly used to implement a coordinate frame transformation by writing a DCM in terms of the angles. Consider the original rotation transformation in Figure 2.3. The rotation can be described using just γ, as shown in Figure 2.6. The DCM developed in Equation (2.2) can be similarly updated by incorporating sine functions. In matrix form, this is given by,

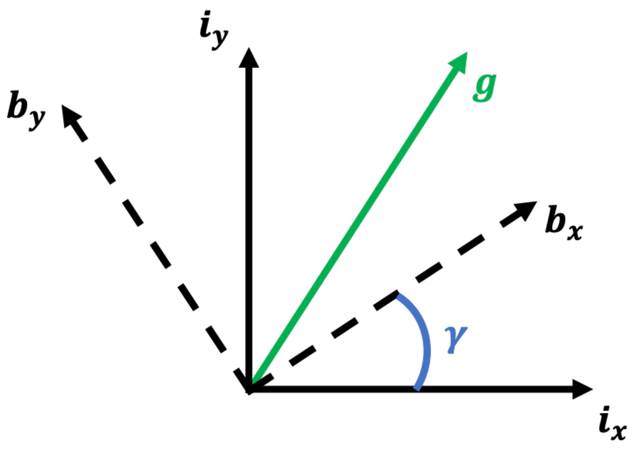

Figure 2.6: Two coordinate frames seperated by a single angle

$$\begin{bmatrix} g_x^b \\ g_y^b \end{bmatrix} = C \begin{bmatrix} g_x^i \\ g_y^i \end{bmatrix} = \begin{bmatrix} cos(\gamma) & sin(\gamma) \\ -sin(\gamma) & cos(\gamma) \end{bmatrix} \begin{bmatrix} g_x^i \\ g_y^i \end{bmatrix}. \tag{2.10}$$

Equation (2.10) shows that a single-angle DCM can be used to implement coordinate frame transformation about one axis . This idea can be extended to the sequence in Figure 2.4. Each of the three rotations can be implemented by a single-angle DCM written in terms of the Euler angles ϕ, θ, and ψ. These three DCMs are,

$$\text{the rotation about the } z\text{-axis:} \quad C_i^{V_1}(\psi) = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.11}$$

$$\text{the rotation about the } y\text{-axis:} \quad C_{V_1}^{V_2}(\theta) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix}, \tag{2.12}$$

and the rotation about the x-axis: $$C^b_{V_2}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} . \tag{2.13}$$

The general DCM of the combined rotation transformation from i to b is found by multiplying Equations (2.11) - (2.13) in the order the rotations are performed in. This gives,

$$\begin{aligned} C^b_i &= C^b_{V_2}(\phi)C^{V_2}_{V_1}(\theta)C^{V_1}_i(\psi) \\ &= \begin{bmatrix} c(\psi)c(\theta) & c(\theta)s(\psi) & -s(\theta) \\ c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & c(\theta)s(\phi) \\ s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) & c(\phi)c(\theta) \end{bmatrix} , \end{aligned} \tag{2.14}$$

where c and s represent cosine and sine respectively.

The non-commutative nature of Euler rotations is demonstrated in Equation (2.14). If Equations (2.11) - (2.13) are multiplied in a different order, then the product would not have the same terms as the matrix in Equation (2.14). To invert the transformation, the rotation sequence is reversed. Each of the constituent DCMs are supplied with the negative of their rotation angles and multiplied in the opposite order the original rotations were applied. This gives,

$$\begin{aligned} C^i_b &= C^{V_1}_i(-\psi)C^{V_2}_{V_1}(-\theta)C^b_{V_2}(-\phi) \\ &= \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) \\ c(\theta)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\phi)c(\theta) \end{bmatrix} . \end{aligned} \tag{2.15}$$

As expected, Equation (2.15), which is the inverse of Equation (2.14), shows up as its transpose. Since these DCMs ultimately implement rotations, the term *rotation matrix* is sometimes used to refer to them. Given a rotation matrix, it is possible to extract Euler angle information. This is done for C_i^b by,

$$\begin{aligned} \phi &= \arctan 2(C_i^b(2,3), C_i^b(3,3)), \\ \theta &= -\arcsin(C_i^b(1,3)), \\ \psi &= \arctan 2(C_i^b(2,1), C_i^b(1,1)). \end{aligned} \tag{2.16}$$

For C_b^i, this is done by,

$$\begin{aligned} \phi &= \arctan 2(C_i^b(3,2), C_i^b(3,3)), \\ \theta &= -\arcsin(C_i^b(3,1)), \\ \psi &= \arctan 2(C_i^b(1,2), C_i^b(1,1)). \end{aligned} \tag{2.17}$$

Equation (2.14) - (2.17) bridge the gap between Euler angles and DCMs, and combine the intuitiveness of the former with the ease of implementation of the latter. Equations (2.14) and (2.15) allow for rotation transformation in either direction, e.g. from reference(i) to object(b) or vice-versa. However, it is critical to note that this thesis has designated the Euler angles themselves as physical rotations starting from the reference frame and ending in the body frame. A user can reverse the convention for other applications, so long as they are aware of the direction that the rotation transformations and physical rotations are being performed in.

2.3 Translation Transformations

The ability to transform vector information from one coordinate frame to another is extremely useful in solving navigation problems. However, applying a coordinate frame transformation to navigation information does not by itself guarantee its validity in the new coordinate frame. Consider for example the position of an object resolved in ECIe somewhere on the Earth's equator. To resolve the object's position in local navigation frame it would be necessary to apply a coordinate frame transformation to bring the object's position into alignment with the earth frame. However, to ensure the validity of the object's position in the local navigation frame, the origin of the local frame relative to the Earth's center must also be considered. Intuitively, the same problem arises with velocity, acceleration, and derivatives of these quantities. Since knowledge of position and velocity is crucial to navigation, it is therefore essential to possess a means to translate these quantities between coordinate frames whose origins are spatially separated.

2.3.1 Position and Velocity Transformations

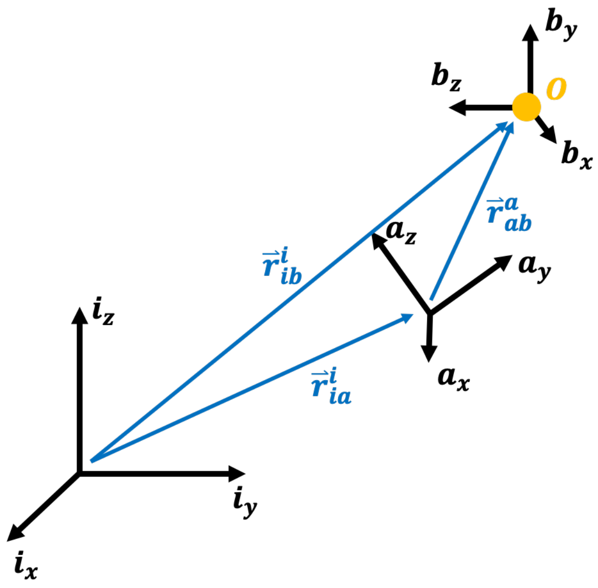

Figure 2.7: A general position translation

Consider the scenario shown in Figure 2.7. The inertial frame i is fixed in space while the secondary frame a arbitrarily translates and rotates through space. The tertiary frame b also arbitrarily translates and rotates through space, and has an object O attached to its origin. This makes that the distance between O and b, $\vec{r}_{bO}$, zero at all times in any resolving frame. What is far more interesting, is the distance from O/b to either a or i. Calculating these distances requires knowledge of the distances between each frame. The object O, whose location coincides with the origin of coordinate frame b, is separated from frame a by a distance $\vec{r}^{a}_{ab}$, which is the known distance between a and b resolved in a. At the same time, the inertial reference frame i is separated from a by $\vec{r}^{i}_{ia}$. Successful navigation of the object often requires knowledge of the object's position in the inertial frame. In terms of known quantities, the object's position relative to the inertial frame is given by,

$$\vec{r}^{\,i}_{ib} = \vec{r}^{\,i}_{ia} + C^i_a \vec{r}^{\,a}_{ab}. \tag{2.18}$$

Note the inclusion of the rotation matrix C^i_a in the equation. It makes position determination of O possible by orienting $\vec{r}^{\,a}_{ab}$ in the i frame. In addition to their orientation, the directions of the vectors are critical. If $\vec{r}^{\,i}_{ia}$ is measured starting from a rather than i, then it must be negated, i.e.

$$\vec{r}^{\,i}_{ia} = -\vec{r}^{\,i}_{ai}. \tag{2.19}$$

Another oft desired quantity in navigation is the object's velocity, which is the first derivative of its position with respect to time. The object's velocity relative to i is found by differentiating Equation (2.18) once, which gives

$$\vec{v}^{\,i}_{ib} = \vec{v}^{\,i}_{ia} + C^i_a \vec{v}^{\,a}_{ab} + \dot{C}^i_a \vec{r}^{\,a}_{ab}. \tag{2.20}$$

It is worth stating that C^i_a and $\dot{C}^i_a$ are the identity if a) frame a is known to be in alignment with frame i and if b) frame a is known to not rotate. Since these terms' values depend on the nature of the resolving frame, this means that the composition of the velocity (and later acceleration) equations is dependent on the choice of resolving frame. Choosing a resolving frame in this problem is straightforward, since there are only two choices, only one of which is inertial. However, for more complicated problems with several possible choices for a resolving frame, the choice may not be as straightforward.

2.3.2 Angular Velocity and Acceleration Transformation

Taking the time derivative of the $\vec{r}$ terms in Equation (2.18) is straightforward, since velocity is the time derivative of position. However, the differentiation of C^i_a produces $\dot{C}^i_a$, a term that is not as clear-cut. To more closely examine $\dot{C}^i_a$, the definition of the derivative is invoked.

Formally, the time derivative of C_a^i is,

$$\dot{C}_a^i = \lim_{dt \to 0} \frac{C_a^i(t_0+dt)C_a^i(t_0) - C_a^i(t_0)}{dt} = \lim_{dt \to 0} \frac{C_a^i(t_0+dt) - I_3}{dt} C_a^i(t_0). \tag{2.21}$$

The first term in the numerator of Equation (2.21) represents the fact that an infinitesimal rotation occurs over the infinitesimal time dt. $C_a^i(t_0+dt)$ is the rotation transformation that occurs over dt, while $C_a^i(t_0)$ is the rotation transformation at t_0. To find the rotation transformation after dt has elapsed, $C_a^i(t_0+dt)$ and $C_a^i(t_0)$ must be multiplied, as per Equation (2.8). To more closely examine this product, it must be expanded in terms of its constituent DCMs. After several algebraic manipulations, this expansion is given by,

$$\begin{aligned} C_a^i(t_0+dt) &= C_{V_2}^i(t_0+dt)C_{V_1}^{V_2}(t_0+dt)C_a^{V_1}(t_0+dt) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(d\phi) & sin(d\phi) \\ 0 & -sin(d\phi) & cos(d\phi) \end{bmatrix} \begin{bmatrix} cos(d\theta) & 0 & -sin(d\theta) \\ 0 & 1 & 0 \\ sin(d\theta) & 0 & cos(d\theta) \end{bmatrix} \begin{bmatrix} cos(d\psi) & sin(d\psi) & 0 \\ -sin(d\psi) & cos(d\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{2.22}$$

Application of the small angle assumption and the elimination of second-order differential terms yields,

$$\begin{aligned} C_a^i(t_0+dt) &\approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & d\phi \\ 0 & -d\phi & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -d\theta \\ 0 & 1 & 0 \\ d\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & d\psi & 0 \\ -d\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & d\psi & -d\theta \\ -d\psi & 1 & d\phi \\ d\theta & -d\phi & 1 \end{bmatrix}. \end{aligned} \tag{2.23}$$

Incorporation into Equation (2.21) gives,

$$\dot{C}_a^i = \begin{bmatrix} 0 & \frac{d\psi}{dt} & -\frac{d\theta}{dt} \\ -\frac{d\psi}{dt} & 0 & \frac{d\phi}{dt} \\ \frac{d\theta}{dt} & -\frac{d\phi}{dt} & 0 \end{bmatrix} C_a^i(t_0). \tag{2.24}$$

The off-diagonal terms of the matrix in Equation (2.24) represent the rate of change of the infinitesimal rotations. This quantity is known as angular velocity. In this case, the angular velocity is of frame i with respect to frame a, resolved in i. Written in terms of quantities typically measured by gyroscopes, Equation (2.24) is restated as,

$$\dot{C}_a^i = C_a^i \Omega_{ia}^a = C_a^i \begin{bmatrix} 0 & -\omega_{iaz}^a & \omega_{iay}^a \\ \omega_{iaz}^a & 0 & -\omega_{iax}^a \\ -\omega_{iay}^a & \omega_{iax}^a & 0 \end{bmatrix} \tag{2.25}$$

where Ω_{ia}^i is the skew-symmetric matrix (or cross-product matrix) of $\vec{\omega}_{ia}^a$.

Note that the order of the Euler angle sequence has no bearing on the angular velocity. Changing the order of the matrices in Equation (2.22) will not change their matrix product after making the small angle assumption. Stated another way, infinitesimal rotations are not subject to the same sequence dependency that Euler angles are. This means that both infinitesimal rotations and angular velocities can be added and negated like ordinary vectors.

Angular velocity can also be rotationally transformed like other vectors. Intuitively, a vehicle in a North-East-Down local navigation undergoes a positive yawing motion during a right turn. However, if the local frame is instead transformed to be an East-North-Up frame, then the vehicle undergoes a negative yawing motion during the right turn. It is a trivial matter to show that angular velocity, a derivative of the angular motion, is also affected by the coordinate frame transformation. In terms of the scenario in Figure 2.7,

$$\vec{\omega}_{ib}^i = C_b^i \vec{\omega}_{ib}^b = C_a^i C_b^a \vec{\omega}_{ib}^b. \tag{2.26}$$

However, unlike positions, velocities, and accelerations, angular velocities are not affected by translation. To see this, consider the Euler rotation sequence in Figure 2.4. Translating the origins of the starting, intermediate, or ending frames in that scenario would have had no effect on the rotation sequence or the Euler angle values. Therefore, it stands to reason that angular velocity is also unaffected by translational offsets.

With Equations (2.25) and (2.26), Equation (2.20) can be rewritten into,

$$\begin{aligned}\vec{v}^i_{ib} &= \vec{v}^i_{ia} + C^i_a\vec{v}^a_{ab} + C^i_a\Omega^a_{ia}\vec{r}^a_{ab} \\ &= \vec{v}^i_{ia} + \vec{v}^i_{ab} + \Omega^i_{ia}\vec{r}^i_{ab}.\end{aligned} \tag{2.27}$$

A third quantity widely used to navigate moving objects is acceleration. Since it is the second derivative of position with respect to time, it is found for the moving object b by differentiating Equation (2.27) once with respect to time. After the appropriate substitutions, this is given by,

$$\begin{aligned}\vec{a}^i_{ib} &= \vec{a}^i_{ia} + C^i_a\vec{a}^a_{ab} + C^i_a(\Omega^a_{ia})^2\vec{r}^a_{ab} + C^i_a\dot{\Omega}^a_{ia}\vec{r}^a_{ab} + 2C^i_a\Omega^a_{ia}\vec{v}^a_{ab} \\ &= \vec{a}^i_{ia} + \vec{a}^i_{ab} + (\Omega^i_{ia})^2\vec{r}^i_{ab} + \dot{\Omega}^i_{ia}\vec{r}^i_{ab} + 2\Omega^i_{ia}\vec{v}^i_{ab}.\end{aligned} \tag{2.28}$$

In summary, navigation information of a linear nature, such as position, velocity, and acceleration, require both rotation *and* translation transformations to ensure their validity in another coordinate frame. On the other hand, information that is angular in nature, such as angular velocity, need only be rotated to ensure its validity in another coordinate frame.

2.4 Inertial Navigation

A navigation system is a device or process that determines an object's position and velocity. To accomplish this task, a navigation system will employ several navigation sensors. These sensors measure properties, or navigation information, that can be used to compute position and velocity outputs [2]. In general, navigation systems can be classified in two different categories. One such category is position-fixing systems. These systems use identifiable external information in the form of physical infrastructure or maps to determine position and velocity directly [2]. Sometimes, these systems identify when an object is located next to an infrastructural or mapped-out point, such as a wireless router or visual landmark [8]. Alternatively, they can use more sophisticated techniques, such as received signal strength (RSS), time of flight (ToF), or angle of arrival (AoA) to precisley determine the object's position via trilateration or triangulation. GNSS is an example of a precision position-fixing system [2, 4]. The other category of navigation systems is dead-reckoning (DR) systems. These systems compute a moving object's linear and angular displacement by summing a moving object's positional and directional motion from a known starting point.

Both position-fixing and dead-reckoning have their respective pros and cons. Position-fixing has the advantage of being able to provide an absolute position for a navigating object. However, this ability depends on either pre-installed infrastructure or a priori mapping of the navigation environment, which can be difficult and costly. Dead-reckoning is advantageous from this standpoint in that it is self-contained. It can compute a moving object's position using only sensors on the object without using infrastructure or maps. However, due to its cumulative nature, an error in any of the sensor readings will cause the dead-reckoned solution's error to grow over time. In addition, since dead-reckoning computes displacement as opposed to absolute position, it requires initial absolute position and attitude values to be able to navigate in a global frame, values that may be difficult to obtain. Despite these disadvantages, DR is an oft-used navigation technique due to its usefulness in unfamiliar environments that lack infrastructure or map information.

2.4.1 Continuous-Time Inertial Navigation

An exceedingly common implementation of DR is inertial navigation. Its defining feature is its use of inertial sensor outputs. These sensors include a gyroscope, which measures angular velocity $\vec{\omega}_{ib}^{b}$, and an accelerometer, which measures total acceleration $\vec{a}_{ib}^{b}$. These two sensors are usually packaged together in a device known as an inertial measurement unit or IMU. Computing the three-dimensional inertial navigation solution of a moving object requires that the

IMU attached to it be able to measure angular velocity and acceleration in three mutually orthogonal directions. A particular process has to be followed to turn these measurement outputs into a navigation solution.

The first step in inertial navigation is to propagate the IMU's attitude. The exact mathematical equation depends on the convention used to express attitude. Since this thesis uses DCMs to express attitude, all of the inertial navigation equations presented will follow that convention. The continuous-time inertial navigation equation for propagating attitude in an inertial coordinate frame is given as a first-order differential equation by,

$$\dot{C}_b^n = C_b^n \Omega_{nb}^b \tag{2.29}$$

where Ω_{nb}^b is the skew-symmetric matrix of the three-dimensional angular velocity measurements from the IMU and C_b^n represents the IMU's attitude. Note the choice of the letter n to represent the inertial frame. This choice is a departure from the convention established earlier in this chapter. This choice was made because the local navigation frame n is assumed to have inertial properties in this thesis. Therefore, it will be referred to as a navigation frame for the remainder of this thesis. Once the moving object's attitude is known, the next step is to transform its acceleration measurements into the navigation frame. This is given by,

$$a_{nb}^n = C_b^n a_{nb}^b \tag{2.30}$$

where a_{nb}^n is the IMU's acceleration in the navigation frame. Note that the vector arrows have been excluded for neatness.

The following step is the gravity compensation. To understand the rationale being gravity compensation, consider the system shown in Figure 2.8.

The mass-spring system shown in this picture represents a basic accelerometer design [2]. To illustrate its operation, consider the scenario shown on the left, which shows the mass-spring system being accelerated downwards. This acceleration causes the mass to lag behind the box

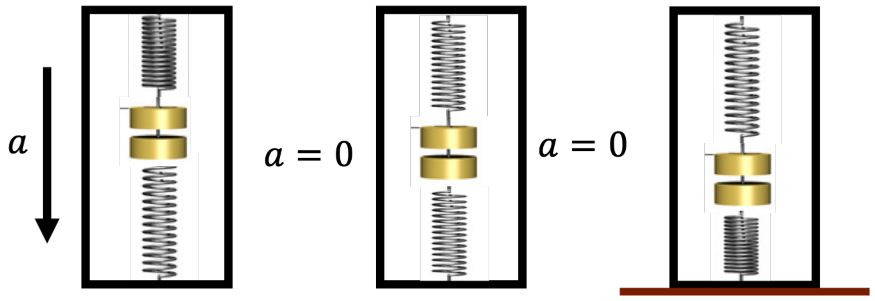

Figure 2.8: The mass-spring accelerometer system.

in movement since it wants to maintain a constant velocity. Since this spatial lag is proportional to the system's acceleration, it can be used as a measurement of acceleration.

By contrast, the center picture shows the mass-spring system in an empty section of outer space moving at zero velocity. In this scenario, there is no force acting on the system and therefore no acceleration. This results in the mass staying equidistant from the sides of the box with no spring compression.

Finally, on the right, the box is at rest on the Earth's surface. Between the Earth's gravitational force and the reaction force exerted on the box by the Earth's surface, the net force acting on the system is zero, just like the middle scenario. However, unlike that scenario, the mass is being pulled in one direction, in this case towards the Earth's center. This difference is accounted for by nature of the ground reaction force. The reaction force is able to push on the box containing the mass-spring system, and keep it from moving. It cannot however push on the mass itself, assuming the bottom spring is not fully compressed.

The difference in the accelerometer readings between the second and third scenarios shown in Figure 2.8 has implications for inertial navigation. In both scenarios, the mass-spring system experiences zero positional change (assuming a stationary Earth). However, a DR/inertial navigation solution based off each scenario's accelerometer readings will yield contrasting results due to differences in the same. The way to compensate for this effect is to account for the reaction force exerted by the Earth's surface. Since this reaction force equals the mass-spring system's weight, the accelerometer reading can be adjusted by the object's weight divided by its mass, which equals Earth's gravitational constant. This is the process that is referred to as gravity compensation. It is expressed mathematically by,

$$f^n_{nb} = a^n_{nb} - \gamma^n_{nb} \tag{2.31}$$

where γ^n_{nb} represents gravitational acceleration and f^n_{nb} represents the specific force, or non-gravitational acceleration. All quantities in the above equation are taken in the IMU's frame with respect to the navigation frame and resolved in the navigation frame. The exact value of gravitational acceleration to be used depends on where the IMU is on the Earth (or other celestial body). For applications in a local navigation frame on the Earth, it is sufficient to assume that gravitational acceleration is simply,

$$\gamma^n_{nb} = \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} m/s^2 \tag{2.32}$$

where the vertical or z-axis is perpendicular to the surface plane.

Once Equation (2.31) has been used for gravity compensation, what is left is the specific force f^n_{nb}, or non-gravitational acceleration. This value represents the motion that the IMU experiences resolved in the navigation frame. This specific force value is equal to the time derivative of the IMU's velocity, given by,

$$\dot{v}^n_{nb} = f^n_{nb}, \tag{2.33}$$

while the time derivative of the IMU's position is its velocity, given by,

$$\dot{r}^n_{nb} = v^n_{nb}, \tag{2.34}$$

where v^n_{nb} and r^n_{nb} are the IMU's velocity and position. Computation of the IMU's actual position, velocity, and attitude (PVA) requires integration with respect to time. For the attitude, this is given by,

$$C_b^n = \int \dot{C}_b^n dt, \tag{2.35}$$

while for the velocity, this is,

$$v_{nb}^n = \int \dot{v}_{nb}^n dt = \int f_{nb}^n dt, \tag{2.36}$$

and for the position, this is,

$$r_{nb}^n = \int v_{nb}^n dt. \tag{2.37}$$

The process outlined above computes a position, velocity, and attitude (PVA) displacement in continuous time for an IMU in motion. This series of equations used in conjunction with inertial sensor outputs is referred to as an inertial navigation system or INS.

2.4.2 INS Errors

Due to its dependency on IMU measurements, the fidelity of an INS's PVA outputs is subject to IMU errors. A simple error model for an IMU is given by,

$$a_{nb}^b = \tilde{a}_{nb}^b + b_a + w_a, \tag{2.38}$$

$$\omega_{nb}^b = \tilde{\omega}_{nb}^b + b_g + w_g, \tag{2.39}$$

where $\tilde{a}_{nb}^b$ and $\tilde{\omega}_{nb}^b$ represent the true acceleration and angular velocity, b_a and b_g represent accelerometer and gyroscope biases, w_a and w_g represent zero-mean, normally distributed random noise on the accelerometer and gyroscope measurements. Other errors are possible, such as misalignment, cross-correlation, and scale factor errors. However, they are not considered in this thesis for the sake of simplicity.

Since C_b^n, v_{nb}^n, and r_{nb}^n are computed from IMU measurements that have error, it stands to reason that the errors in these quantities depends on the IMU errors. In continuous-time, the inertial navigation errors are given in [2] by,

$$\delta\dot{\psi}^n_{nb} = C^n_b(b_g + w_g), \tag{2.40}$$

$$\delta\dot{v}^n_{nb} = -[f^n_{nb}\wedge]\delta\psi^n_{nb} + C^n_b(b_a + w_a), \tag{2.41}$$

$$\delta\dot{r}^n_{nb} = \delta v^n_{nb}, \tag{2.42}$$

where $[f^n_{nb}\wedge]$ is the skew-symmetric matrix of the specific force and δr^n_{nb}, δv^n_{nb}, and $\delta\psi^n_{nb}$ are the errors on the IMU's position, velocity, and attitude. Note that $\delta\psi^n_{nb}$ is in fact a vector that represents the attitude error as a small angular offset. The relationship between the INS's true PVA and its errors are given by,

$$\tilde{C}^n_b = (I_3 - [\delta\psi^n_{nb}\wedge])C^n_b, \tag{2.43}$$

$$\tilde{v}^n_{nb} = v^n_{nb} + \delta v^n_{nb}, \tag{2.44}$$

$$\tilde{r}^n_{nb} = r^n_{nb} + \delta r^n_{nb}, \tag{2.45}$$

where $[\delta\psi^n_{nb}\wedge]$ is the skew-symmetric matrix of the angular errors and $\tilde{C}^n_b$, $\tilde{v}^n_{nb}$, and $\tilde{r}^n_{nb}$ represent the true attitude, velocity, and position of the INS.

2.4.3 Discrete-Time Inertial Navigation

The continuous-time INS equations (Equations (2.29) - (2.37) and the INS error equations (Equations (2.40) - (2.42)) offer a meaningful mathematical model for DR using the outputs from an IMU. However, they are not immediatly suitable for practical implementations for two reasons: 1) IMUs take measurements in discrete intervals and 2) Computers perform calculations in discrete intervals. Therefore, for practical implementations on digital computers, these equations must be discretized.

The attitude update, Equation (2.29 and 2.35, can be discretized by using the matrix exponential, which is given by,

$$C_b^n(t+\Delta t) = C_b^n(t)(\mathrm{expm}([\Omega_{nb}^b \Delta t \wedge]), \tag{2.46}$$

where t represents the time of the previous computation epoch and $t+\Delta t$ represents the time of the current computation epoch, assuming that Δt is a constant time step. The discrete attitude update shown in Equation (2.46) is often implemented on computers as an equivalent Taylor series expansion. To simplify its implementation, the series is sometimes represented as a first-order approximation, given by,

$$C_b^n(t+\Delta t) = C_b^n(t)(I_3 + \Omega_{nb}^b \Delta t). \tag{2.47}$$

The acceleration frame transformation and gravity compensation steps, Equations (2.30) and (2.31)), are essentially the same in discrete-time as they are in continuous-time. However, the accuracy of the acceleration frame transformation can be enhanced by averaging C_b^n over two epochs. This is given by,

$$a_{nb}^n = \frac{1}{2}(C_b^n(t+\Delta t) + C_b^n(t))a_{nb}^b. \tag{2.48}$$

The principle used to approximate the discretization of the attitude update can also be applied to the velocity and position updates. Accordingly, the velocity update, Equations (2.33) and (2.36), are given in discrete time by,

$$v_{nb}^n(t+\Delta t) = v_{nb}^n(t) + f_{nb}^n \Delta t, \tag{2.49}$$

while the position update, Equations (2.34) and (2.37), are given in discrete time by,

$$r_{nb}^n(t+\Delta t) = r_{nb}^n(t) + v_{nb}^n \Delta t. \tag{2.50}$$

Similar logic can be applied to the continuous-time error equations. The discrete time error equation for the attitude is given by,

$$\delta\psi^n_{nb}(t+\Delta t) = C^n_b(b_g + w_g)\Delta t + \delta\psi^n_{nb}(t), \tag{2.51}$$

the velocity by,

$$\delta v^n_{nb}(t+\Delta t) = \delta v^n_{nb}(t)\Delta t + (-[f^n_{nb}\wedge]\delta\psi^n_{nb})\Delta t + C^n_b(b_a + w_a)\Delta t, \tag{2.52}$$

and the position by,

$$\delta r^n_{nb}(t+\Delta t) = \delta r^n_{nb}(t) + \delta v^n_{nb}\Delta t. \tag{2.53}$$

2.5 INS Aiding With Error-State Kalman Filter

Discrete-time INS is an implementation of DR that proves its worth in environments where the availability of position-fixing systems such as GNSS and map-matching cannot be guaranteed. However, using INS as a standalone system has its own disadvantages. Consider an IMU that is perfect in every way except for a $0.1\ m/s^2$ measurement bias in one of its axes. Over 10 seconds, that bias will translate to a $1\ m/s$ velocity error and $10\ m$ position error. Though IMU errors are not always that large, the fact remains that no standalone INS can return reasonable navigation solutions in perpetuity. For this reason, it is often useful to aid the PVA outputs from INS with PVA outputs from position-fixing systems if they are available. Such aiding can compute a PVA solution that is more accurate and robust than could be achieved by either INS or the aiding system alone. This is due to the fact that each system has complementary benefits and drawbacks. A position-fixing system can keep an INS-computed PVA from drifting, while an INS can both smooth the PVA and bridge outages computed by the position-fixing system.

Finding an optimal solution to this aiding problem involves finding a PVA that minimizes the difference between the INS's PVA and the aiding system's PVA given the uncertainty in each system's output. Solving this kind of problem is the objective of Kalman filtering (KF), which is detailed in Appendix A. Therefore, it is common to use a KF framework to aid INS with another navigation system. This is known as INS-KF. Each of the two systems, the INS and aiding system, correspond to the two different phases of the KF, these being the process and measurement updates respectively. It is common to put the INS in the process update and the aiding source in the measurement update. This setup is especially advantageous if the aiding source runs at a lower update rate than the INS.

The first step in designing an INS-KF is selection of an appropriate state propagation model. Though PVA values are an intuitive choice for the KF's states, they are not used as the state values. Directly using PVA as states is problematic, especially in the case of the attitude state, C_b^n. This is because the KF requires the states be expressed as a vector of scalar values that are a linear combination of the state propagation equations. More advanced Kalman filtering methods exist that could potentially handle a matrix as a state [23], but they are beyond the scope of this thesis. To handle a rotation matrix in a KF, its nine terms can be each included in the state vector, as done in [91]. However, as [91] discusses, doing so does not guarantee the orthonormality of the resulting matrix. An option that would guarantee orthonormality would be to express the attitude using different means, such as quaternions or Euler angles. Though both of these structures are vector quantities, using them in the process update would make the resulting KF non-linear. This extra level of complexity is also beyond the scope of this thesis.

Rather than using the INS's PVA as states, the INS's PVA errors can instead be used as the KF's states. This is made possible by the fact that unlike PVA, the PVA errors are quantities that are linear combinations of their propagation equations. Using the PVA errors as states requires using an error-state Kalman filter (ESKF) to estimate the optimal PVA. The ESKF algorithm, detailed in Appendix A, maintains two sets of states, the full states and the error states. The incoming measurements from the aiding system are used to optimally estimate the errors in the full states given the uncertainty in the error states and measurements. These error estimates are then used to correct the full states. For an INS aiding implementation, known as INS-ESKF, the error states are given by

$$\delta x = \begin{bmatrix} \delta r_{nb}^n & \delta v_{nb}^n & \delta \psi_{nb}^n & b_a & b_g \end{bmatrix}^T, \tag{2.54}$$

noting the inclusion of the IMU biases in δx. These biases are included to enhance the accuracy of the full state estimates. These are propagated in discrete time by,

$$b_a(t + \Delta t) = b_a(t) + w_{b_a} \text{ and } b_g(t + \Delta t) = b_g(t) + w_{b_g}. \tag{2.55}$$

These error states are meant to correct the full states, which are,

$$x = \begin{bmatrix} r^n_{nb} & v^n_{nb} & C^n_b \end{bmatrix}^T, \tag{2.56}$$

which is computed by propagating the discrete INS equations through time (Equations (2.49), (2.47), and (2.50)). Note that the availability of accelerometer and gyroscope bias estimates allows for a modification of two of the discrete INS equations, specifically Equations (2.49) and (2.47). These equations can account for the IMU bias estimates by subtracting the estimates from from the sensor measurements before they are used to propagate velocity and attitude. This is given by,

$$v^n_{nb}(t + \Delta t) = v^n_{nb}(t) + (f^n_{nb} - b_a)\Delta t, \tag{2.57}$$

and

$$C^n_b(t + \Delta t) = C^n_b(t)(I_3 + (\Omega^b_{nb} - b_g)\Delta t). \tag{2.58}$$

With the INS error states and IMU biases making up the error state vector, the discrete-time state transition matrix for the INS-ESKF is given by,

$$F = \begin{bmatrix} I_3 & I_3\Delta t & 0_3 & 0_3 & 0_3 \\ 0_3 & I_3 & [-f^b_{nb} \wedge]\,\Delta t & C^n_b\Delta t & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 & C^n_b\Delta t \\ 0_3 & 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}. \tag{2.59}$$

The process noise vector is given by,

$$w = \begin{bmatrix} w_a & w_g & w_{b_a} & w_{b_g} \end{bmatrix}^T \tag{2.60}$$

and the matrix mapping these noises to the state domain given by,

$$G = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ C_r^n & 0_3 & 0_3 & 0_3 \\ 0_3 & C_r^n & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}. \tag{2.61}$$

The final piece of the process update is the process noise covariance matrix. This is given by,

$$Q = \begin{bmatrix} \sigma_a^2 & 0_3 & 0_3 & 0_3 \\ 0_3 & \sigma_g^2 & 0_3 & 0_3 \\ 0_3 & 0_3 & \sigma_{b_a}^2 & 0_3 \\ 0_3 & 0_3 & 0_3 & \sigma_{b_g}^2 \end{bmatrix}. \tag{2.62}$$

The second step of designing an INS-ESKF's is composing its measurement update, the exact makeup of which depends on the measurement quantities output by the aiding system. Commonly used sources of aiding information are positions and velocities computed by GNSS. Though not every application of INS involves GNSS, the simplicity of GNSS's outputs makes it a convenient example to illustrate the idea of INS aiding. This sort of system is known as loosely coupled GNSS-INS [2].

To compose the measurement update, the measurement quantities must be related to the full states. The relationship between the INS's position output and the aiding system's position output is an application of Equation (2.18) and is given by,

$$r_{nb}^n = r_{ng}^n + C_b^n r_{bg}^b. \tag{2.63}$$

where the subscript g represents a quantity measured by GNSS. Similarly, Equation (2.20) can be applied to relate the INS's velocity and the aiding system's velocity. This is given by,

$$v_{nb}^n = v_{ng}^n + C_b^n \Omega_{nb}^b r_{gb}^b. \tag{2.64}$$

Since position and velocity are both vector quantities, the measurement error of each quantity is,

$$\delta y_r = r^n_{ng} - r^n_{nb} + C^n_b r^b_{gb} \tag{2.65}$$

and,

$$\delta y_v = v^n_{ng} - v^n_{nb} + C^n_b \Omega^b_{nb} r^b_{gb}. \tag{2.66}$$

The measurement error vector is simply a stacking of both measurement errors,

$$\delta y = \begin{bmatrix} \delta y_r \\ \delta y_v \end{bmatrix}. \tag{2.67}$$

The measurement model is a linear mapping from the error state vector δx to the domain of the measurement errors. For a GNSS that outputs a PV solution, the measurement model is given by,

$$H = \begin{bmatrix} -I_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & -I_3 & 0_3 & 0_3 & 0_3 \end{bmatrix}, \tag{2.68}$$

with a measurement noise matrix of,

$$R = \begin{bmatrix} \sigma^2_{r_g} & 0_3 \\ 0_3 & \sigma^2_{v_g} \end{bmatrix}. \tag{2.69}$$

If either position or velocity is not returned by the GNSS system, then the corresponding rows can be removed from the measurement model and noise matrix. Note that the translational offset and rotational offset between INS and GNSS, also known as lever arm and misalignment, are part of the measurement errors. Although these may not always be known, in this chapter, they are assumed to be known quantities. Using the equations and matrices presented in this section in conjunction with the ESKF algorithm presented in Appendix A will allow for the computation of IMU biases and a position, velocity, and attitude solution that is an optimal mixture of the PVA outputs from an INS and an aiding system such as GNSS.

Chapter 3

Pedestrian Dead Reckoning

Pedestrian dead reckoning (PDR) is an application of the dead reckoning technique used to track the movement of pedestrians. In general, PDR propagates a pedestrian's position using outputs from Inertial Measurement Units (IMUs) and other sensors worn by the pedestrian, and supplements these sources of information by making assumptions about human motion [5, 7, 11, 12, 25]. The ability to implement PDR has been made possible by recent advances in sensor and computing technology. These devices are now sufficiently accurate and small enough to practically track pedestrians. As such, PDR has recently become a topic of wide research exploration. PDR algorithms themselves come in two different varieties, these being the INS approach and step-detection approach. To provide context for the transfer alignment content in the rest of this thesis, this chapter will elaborate on these two approaches.

3.1 Human Gait Pattern

3.1.1 Gait Analysis

The main function of PDR is to track a pedestrian as they travel. To successfully track pedestrians, all PDR implementations must take into account an individual user's walking motion. Therefore, in order to understand PDR, it is essential that the normal walking motion be described. Normal human walking is defined in [26] as "a method of locomotion involving the use of the two legs, alternately, to provide both support and propulsion." It is differentiated from running by the requirement that at least one foot be in contact with the ground at all times. Another word commonly used to refer to walking is gait. Though the two terms are often used interchangeably, walking is in fact distinct from gait, which is defined as the “manner or style of walking” [26]. In general, walking refers to method of movement, while gait refers to an individual’s specific style of walking.

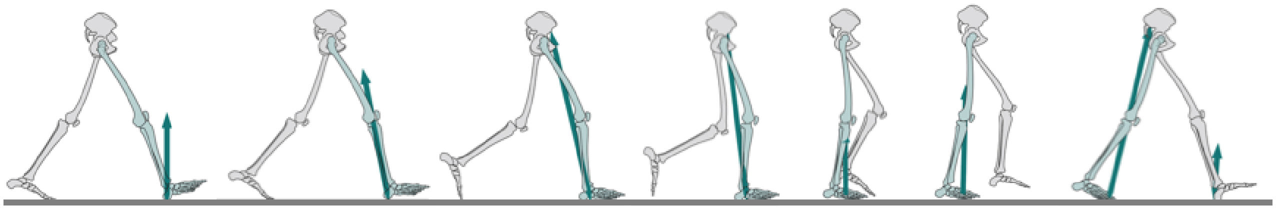
Figure 3.1: A half gait cycle is shown starting on the right foot. The right foot (higlighted in green) is in a stance phase while the left foot is in a swing phase.

Since walking is a repetitive motion, it can be thought of as a repeating cycle. The term used to describe this is gait cycle, which is defined as “the time interval between two successive occurrences of one of the repetitive events of walking” [26]. Note that a gait cycle can start and end from any event in a walking cycle. For convenience, the start and ending point for a gait cycle in this thesis will begin at the moment the heel of a foot comes into contact with the ground. A full cycle, from right heelstrike to right heelstrike is known as a *stride*. A half gait cycle, or *step*, is shown in Figure 3.1. The figure highlights the two major phases in the gait cycle, known as “stance” and “swing.” Stance refers to the period in which some part of the foot is in contact with the ground. A stance can be broken down into the following events:

1. **Initial contact** - The foot’s first contact with the ground.
2. **Loading Response** - The ground reaction reaction force increases as rest of the foot is lowered to the ground.
3. **Mid-stance** - The foot is mostly flush with the ground while the opposite leg is swinging forward.
4. **Heel Rise** - The heel starts to peel off the ground.
5. **Toe Off** - The toe leaves the ground.

Intuitively, swing refers to the period in which the foot is swinging through the air. A swing can be further broken down as follows:

1. **Toe Off** - The toe leaves the ground.
2. **Feet Adjacent** - As the foot is moving forward, it passes the opposite leg, which is in the middle of its stance phase.
3. **Tibia Vertical** - The section of the leg between the knee and the ankle is perpendicular with respect to the ground
4. **Initial Contact** - The foot's first contact with the ground.

Generally, as one foot is swinging, the other foot is standing on the ground. To transition between feet, the ending of a stance phase for one foot overlaps with the beginning of the stance for the other foot. The general pattern can be seen in Figure 3.2. The periods marked "Double support" are the transition periods in which both feet are contacting the ground, while the "Single Support" periods signify the periods in which only one foot is on the ground.

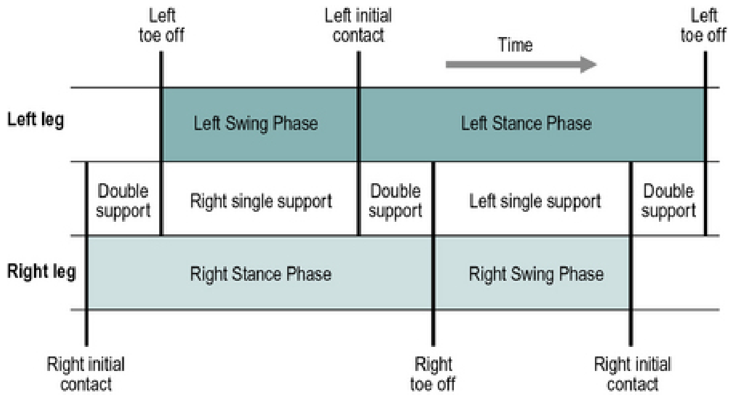

Figure 3.2: The progression of stance and swing phases in a gait cycle [26].

The graphics and terminology discussed so far are from [26], a biomechanics text. In pedestrian navigation literature, there is some inconsistency in terminology, especially in the usage of the words step and stride. Therefore, when reading pedestrian navigation papers it is important to keep track of the intended meanings behind a particular author's chosen terminology.

The goal of any PDR implementation is to use the dead reckoning principle to track the described walking pattern. Since dead reckoning is a process subject to accumulating position error, nearly all PDR algorithms described in the literature take advantage of distinctive features in the gait cycle to reduce positioning error.

3.2 An INS-Based Approach to Pedestrian Dead Reckoning

One approach to PDR is tracking the pedestrian's motion using INS techniques, as described in [7, 14, 18, 25, 27, 29]. Broadly speaking, this approach propagates the pedestrian's angular and positional displacement by integrating acceleration and angular velocity outputs from an IMU mounted on the pedestrian at the IMU's update rate. Such an approach is attractive, since it makes no assumptions about a pedestrian's specific manner of style of walking. However as discussed in Chapter 2, the positioning results returned by this method accumulate error in proportion to the errors in the IMU used to collect inertial measurements. For the MEMS IMUs that are often used to track pedestrians, an INS-computed position would quickly become useless, as IMU errors will cause the integrated position and attitude to diverge. Therefore any INS-based approach must contain a mechanism to reduce positioning error. Such a mechanism becomes available if the IMU is placed on the pedestrian's foot. Specifically, the output from a foot-mounted IMU can be used to detect stance events. Stance detection is useful, since the foot's near zero-velocity state during a stance can be used to reduce dead reckoning error. For this reason, INS-based PDR methods use information from foot-mounted IMUs to track a pedestrian's motion. This technique, which is known as PDR-INS in this thesis, is made up of two phases. These are the propagation of the pedestrian's position, velocity, and attitude (PVA) and the reduction of the PVA errors. These two phases are discussed in detail in the following subsections.

3.2.1 Propagation of the Pedestrian's Position, Velocity, and Attitude

The first phase of PDR-INS is the tracking of the pedestrian's PVA. This is done by propagating the foot-mounted IMU's position, velocity, and attitude (PVA). This can be performed in real time by applying inertial navigation equations to its acceleration and angular velocity outputs. Inertial navigation was described in detail in Chapter 2 of this thesis, but an abridged discussion is given here for convenience. In discrete-time, the foot-mounted IMU's attitude update is given by,

$$C_b^n(t + \Delta t) = C_b^n(I_3 + \Omega_{nb}^b \Delta t), \tag{3.1}$$

where C_b^n represents the IMU's attitude, t represents the time of the previous computation epoch and $t + \Delta t$ represents the time of the current computation epoch. The acceleration frame transformation and gravity compensation are given by,

$$a_{nb}^n = \frac{1}{2}(C_b^n(t + \Delta t) + C_b^n(t))a_{nb}^b, \tag{3.2}$$

$$f_{nb}^n = a_{nb}^n - \gamma_{nb}^n, \tag{3.3}$$

where a_{nb}^n represents the IMU's (total) acceleration, f_{nb}^n represents its non-gravitational acceleration, and γ_{nb}^n represents gravitational acceleration. The exact value of gravitational acceleration to be used depends on where the IMU is on the Earth (or other celestial body). For applications in a local navigation frame, it is sufficient to assume that gravitational acceleration is,

$$\gamma_{nb}^n = \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} m/s^2. \tag{3.4}$$

Finally, the velocity (v_{nb}^n) and position (p_{nb}^n) updates are given in discrete time by,

$$v_{nb}^n(t + \Delta t) = v_{nb}^n(t) + f_{nb}^n \Delta t, \tag{3.5}$$

$$r_{nb}^n(t + \Delta t) = r_{nb}^n(t) + v_{nb}^n \Delta t. \tag{3.6}$$

As discussed in Chapter 2, the accuracy of inertial navigation solution is dependent on the errors in the IMU measurements. A simple error model of the IMU's accelerometer and gyroscope is given by,

$$a^n_{nb} = \tilde{a}^n_{nb} + b_a + w_a, \tag{3.7}$$

$$\omega^n_{nb} = \tilde{\omega}^n_{nb} + b_g + w_g, \tag{3.8}$$

where $\tilde{a}^n_{nb}$ and $\tilde{\omega}^n_{nb}$ represent the true acceleration and angular velocity, w_a and w_g represent zero-mean, normally distributed noises on the IMU noises, and b_a and b_g represent accelerometer and gyroscope biases.

Since C^n_b, v^n_{nb}, and r^n_{nb} are computed from IMU measurements with error, it stands to reason that the errors in these quantities depend on the IMU errors. The evolution of these errors in discrete time is given by,

$$\delta\psi^n_{nb}(t + \Delta t) = C^n_b(t)(b_g + w_g)\Delta t, \tag{3.9}$$

$$\delta v^n_{nb}(t + \Delta t) = \delta v^n_{nb}(t)\Delta t + (-[f^n_{nb}\wedge]\delta\psi^n_{nb})\Delta t + C^n_b(b_a + w_a)\Delta t, \tag{3.10}$$

$$\delta r^n_{nb}(t + \Delta t) = \delta r^n_{nb}(t) + \delta v^n_{nb}\Delta t, \tag{3.11}$$

which are the attitude, velocity, and position errors respectively. They are related to the foot-mounted INS's PVA values by,

$$\tilde{C}^n_b = (I_3 - [\delta\psi^n_{nb}\wedge])C^n_b, \tag{3.12}$$

$$\tilde{v}^n_{nb} = v^n_{nb} - \delta v^n_{nb}, \tag{3.13}$$

$$\tilde{r}^n_{nb} = r^n_{nb} - \delta r^n_{nb}, \tag{3.14}$$

where the quantities under tildes represent the true PVA of the foot-mounted INS. [2]

3.2.2 PVA Error Reduction

3.2.2.1 Stance Detection

The second phase of PDR-INS is the reduction of the foot-mounted INS's PVA errors. There are several mechanisms with which to do this, but the most important one is the zero-velocity update or ZUPT. For pedestrian navigation, a stance can be considered a zero-velocity event, since the foot experiences little translation while it is in contact with the ground. During a stance, the foot-mounted INS's PVA error can be reduced by resetting its velocity to zero. In addition, a ZUPT allows for the calibration of the foot-mounted INS's x and y-axis gyroscope biases. In order to accomplish this, stance events must first be detected. These events can be detected by considering two quantities derived from the measurements taken by the IMU. These quantities are the Euclidean norms of the acceleration and angular velocity measurements, given by,

$$||a^b_{nb}|| = \sqrt{(a^b_{nb,x})^2 + (a^b_{nb,y})^2 + (a^b_{nb,z})^2} \text{ and } ||\omega^b_{nb}|| = \sqrt{(\omega^b_{nb,x})^2 + (\omega^b_{nb,y})^2 + (\omega^b_{nb,z})^2}. \tag{3.15}$$

The general principle is, if these quantities indicate near-zero motion, then the IMU can be considered to be experiencing an instance of zero velocity. A qualitative version of this approach, which considers only evolution of the acceleration norm over time, is shown in Figure 3.3. Stances and swings can be identified during the gait cycle by applying this general principle. The goal of a stance detector is to take this qualitative approach shown in Figure 3.3 and perform it on a quantitative basis.

The occurrence of a stance event can be ascertained by applying two conditions to the acceleration norm and one condition to the angular velocity norm. The presented conditions are based upon methods developed by Pierce [15], Ray [24], and Jimenez et al [7, 14]. They are:

1. A window of acceleration norms must be between an upper and lower bound ($th_{a_{min}} = 8.5\, m/s^2$ and $th_{a_{max}} = 11\, m/s^2$). This is expressed mathematically as,

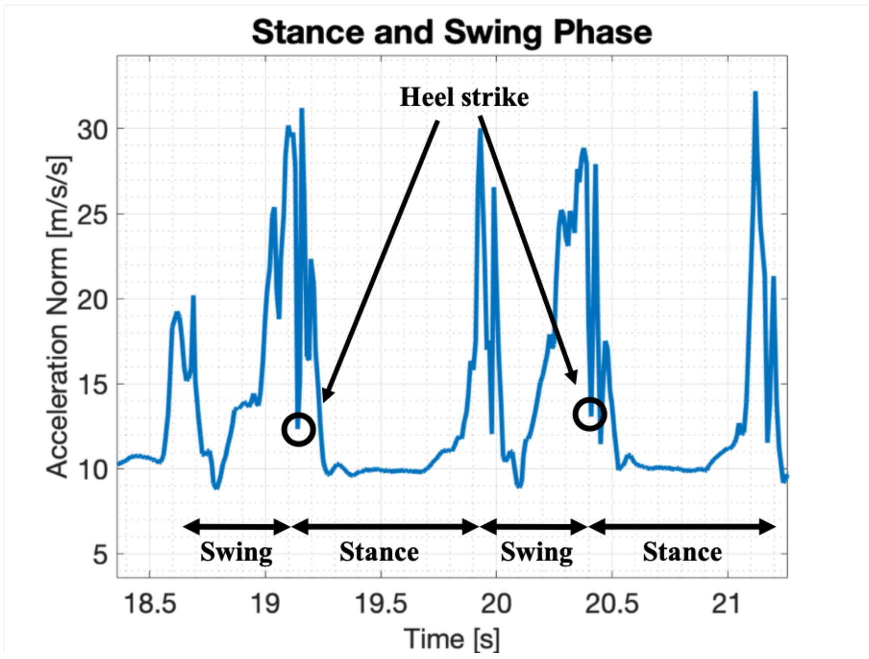

Figure 3.3: Euclidean norm of acceleration data showing stance and swing phases.

$$C1 = \begin{cases} True & \text{if } \sim\text{any } (th_{a_{min}} < ||a^b_{nb,window}|| < th_{a_{max}}) \\ False & \text{otherwise.} \end{cases} \tag{3.16}$$

where the term $\sim$ is a logical NOT and the function any() determines if any of elements in the window fail to meet the condition. The conditional statement can be more intuitively thought of as: if none of the acceleration norm measurements in the window are above or below the bounds, then the conditional statement is true. A window length of 21 samples was used with a sample rate of 100 Hz in this thesis, resulting in a 0.21 second window.

2. The local acceleration standard deviation must be below a threshold. This is expressed mathematically as,

$$C2 = \begin{cases} True & \text{if } ||a^b_{nb}|| - \mu_{window} < 1.5 \cdot \sigma_{window} \\ False & \text{otherwise,} \end{cases} \tag{3.17}$$

where μ_{window} and σ_{window} are the mean and standard deviation of the current window of acceleration norms. A window length of 21 samples was also used, with a sample rate of

100 Hz. The result of applying these two conditions to the signal shown earlier in Figure 3.3 is shown in Figure 3.4.

3. A window of angular velocity norms must be below an upper bound ($th_{\omega_{ZUPT}} = 0.7\, rad/s$). This is expressed mathematically as,

$$C3 = \begin{cases} True & \text{if } \sim\text{any } (||\omega^b_{nb,window}|| > th_{\omega_{ZUPT}}) \\ False & \text{otherwise.} \end{cases} \tag{3.18}$$

   A window length of 21 samples was also used, with a sample rate of 100 Hz.

If $C1$, $C2$, and $C3$ are all true, then a stance event is occurring.

3.2.2.2 Still Detection

As useful as ZUPTs are, there is one critical value they cannot estimate, which is the z-axis gyroscope bias. Leaving this quantity unestimated will result in significant heading drift, so it is important that it be estimated. A mechanism that can be used to estimate the IMU's z-axis gyroscope bias is still detection. Still detection is so named since it involves detecting periods in which the pedestrian is completely still, i.e. not walking. Besides offering a chance to perform a ZUPT, a still event also offers a chance to perform a zero angular-rate update or ZARU. This is due to the fact that the foot experiences near-zero rotation during a still event. This is contrasted with a stance event, in which the foot, though not translating, is still experiencing significant rotation. Applying a ZARU to an INS-computed PVA solution allows the calibration of the gyroscope's biases in all three axes, since $\tilde{\omega}^n_{nb}$ in Equation (3.8) equals zero.

As with stance events, a still event can be detected using the measurements taken by a foot-mounted IMU and following a principle similar to the one used for stance detection. A difference between stance and still detection is the fact that still detection need only consider the angular velocity norm. The occurrence of a still event can be ascertained by applying two conditions to this quantity. As with the stance detection conditions, the conditions for still detection are based upon methods developed by Pierce [15], Ray [24], and Jimenez et al [7, 14] They are:

1. A window of angular velocity norms must be below an upper bound ($th_{\omega_{max}} = 0.4\,rad/s$). Note that this is a lower threshold than what is needed for stance detection. The lowering of the threshold is driven by the fact that a completely still foot experiences less rotation than a foot experiencing a mid-gait stance. Therefore, to ensure that the foot is completely still, a lower threshold is needed. The condition is expressed mathematically as,

$$C4 = \begin{cases} True & \text{if } \sim\text{any } (||\omega^b_{nb,window}|| < th_{\omega_{max}}) \\ False & \text{otherwise.} \end{cases} \tag{3.19}$$

   The conditional statement can be more intuitively thought of as, if none of the angular norm measurements in the window are above the bound, then the conditional statement is true. A window length of 21 samples was used for a sample rate of 100 Hz for still detection.

2. The local angular velocity standard deviation must also be below a threshold. This is expressed mathematically as,

$$C5 = \begin{cases} True & \text{if } ||\omega^b_{nb}|| - \mu_{window} < 2 \cdot \sigma_{window} \\ False & \text{otherwise.} \end{cases} \tag{3.20}$$

   where μ_{window} and σ_{window} are the mean and standard deviation of the current window of angular velocity norms. A window length of 21 samples was again used, resulting in a 0.21 second window. The result of applying these two conditions the signal shown earlier in Figure 3.3 is shown in Figure 3.4.

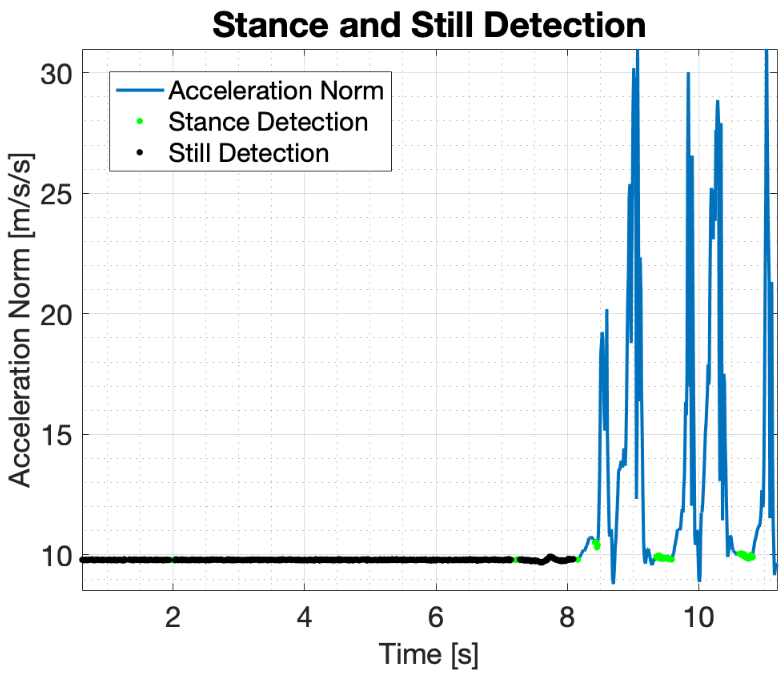

Figure 3.4: Euclidean norm of acceleration data showing stance and still detection.

3.2.2.3 Heuristic Drift Reduction

A major weakness of ZARUs is that they can only be applied during still periods. If no still periods occur, then the ZARU mechanism will be unable to estimate the z-axis gyroscope bias. To calibrate this bias and reduce heading drift while on the move, an alternative bounding mechanism must be used. Such a mechanism is heuristic drift reduction or HDR, which is described in [24] and [7]. HDR takes advantage of periods of straight walking to reduce heading drift and in turn calibrate the z-axis gyroscope bias. The general principle is as follows: If there is only a small change in the pedestrian's heading between steps ($\delta\psi_s$), it can be assumed to be zero. This general principle is given mathematically by,

$$C6 = \begin{cases} \text{Straight Walking} & |\delta\psi_s| \leq th_{\delta\psi} \\ \text{Non-Straight Walking} & |\delta\psi_s| > th_{\delta\psi} \end{cases} \tag{3.21}$$

where the pedestrian's heading change is given by,

$$\delta\psi_s = \psi_s - \psi_{s-1}, \tag{3.22}$$

where s is the index value that corresponds to the beginning of a stance and ψ_s is the heading of the foot at the current stance s, computed by $\psi_s = \arctan(C_b^n(1,2), C_b^n(1,1))$.

3.2.2.4 Heading Measurements from Magnetometer

The ability of HDR to bound heading drift during walking is incredibly useful. However, this ability is only available during straight walking periods. If no such period occurs, then HDR cannot bound heading drift. For heading drift bounding to be continuously available, a heading measurement must be provided by an external source. The earth's magnetic field can function as such a source. Since the earth's magnetic field generally points northward everywhere on the planet (apart from local variations), measurements of the magnetic field can be used to provide a continuously available and independent source of heading information for the foot-mounted IMU, as implemented in [18] and [7]. To fully utilize this source of information, magnetic field readings must be taken by a triaxial magnetometer that is co-located with the foot-mounted IMU. An example of such a device is the Vectornav VN-300 [92]. Heading is computed from magnetometer measurements by first leveling the measurements into the navigation frame using

$$B_{nb}^n = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} B_{nb}^b. \tag{3.23}$$

Once the magnetometer measurements have been leveled, a heading can be computed by,

$$\psi_n = \psi_{mag} + \alpha_m = \arctan(B_{nb,y}^n, B_{nb,x}^n) + \alpha_m, \tag{3.24}$$

where α_m is the angle between magnetic and true north. By applying a magnetometer-computed heading to C_b^n, the drift in the pedestrian's heading error can be bounded and the foot-mounted IMU's z-axis gyroscope bias can be estimated.

3.2.3 INS-EKF-ZUPT

In pedestrian navigation, there are two possible approaches that can be used to incorporate these four methods of error reduction into the PVA solution. Skog et al. refers to these approaches as ”hard” and ”soft” updates [17]. A hard update, for example, incorporates ZUPTs by setting v^n_{nb} to zero when a stance is detected. This approach is used in [27], [14], and [29]. By contrast, a soft update incorporates this information using an optimal estimation framework. This alternative approach is taken by [25], [7], and [15]. Though this second approach is more complicated, the optimal estimation framework has the advantage of providing a means to optimally weight and incorporate information from other, non-inertial sensors. These include not only magnetometers, but GNSS as well [25].

Due to this inherent advantage, the optimal estimation framework for PDR-INS presented in [25] and [7] will be described at length in this subsection. The goal of optimal fusion is to compute a PVA that minimizes the difference between the foot-mounted INS's PVA and the inputs provided by bounding methods. This sort of problem is ideally suited for solution with a Kalman filtering (KF) approach. A description of Kalman filtering is given in Appendix A, while Kalman filtering in the context of inertial navigation was summarized in Chapter 2. The goal of this subsection therefore, is to fit the PDR-INS problem described in the previous two subsections into the KF algorithm.

Each of the two components of PDR-INS, the foot-mounted INS and the error reduction methods, correspond to the two different phases of the KF, these being the process and measurement updates respectively. It is common to put the foot-mounted INS in the process update and the error reduction in the measurement update. This setup is especially advantageous, since all of the described error reduction xsmethods, save for the magnetometer, occur at a rate that is slower than the INS's update rate.

3.2.3.1 State Selection

Though PVA values are an intuitive choice for the KF's states, they are not used as the state values. Instead, the foot-mounted INS's PVA errors are used as the KF states. Using the

PVA errors as states makes this an implementation of an error-state Kalman filter (ESKF). The ESKF algorithm, detailed in Appendix A, maintains two sets of states, the full states and the error states. The incoming measurements are used to optimally estimate the errors in the full states given the uncertainty in the error states and measurements. These error estimates are then used to correct the full states. In the context of PDR-INS, this means that the information provided by the bounding methods are used to correct errors in the foot-mounted INS. In [7], Jiminez refers to this approach as INS-EKF-ZUPT or IEZ.

The justification for using the ESKF for PDR-INS as opposed to a full-state KF lies in the nature of the inertial navigation equations used to propagate the foot-mounted INS's PVA. (Equations (3.1) - (3.6)) Neither the PVA values nor its propagation equations fit the profile required the full state KF algorithm, namely that it be a vector of scalar values that are a linear combination of the state propagation equations. This is especially true in the case of the attitude, C_b^n. However, PVA errors and their propagation equations fit this linear profile, which justifies maintaining the PVA errors as states inside an ESKF framework for applications that use INS as a process model.

Other states that are estimated are the foot-mounted INS's IMU biases, since using the four discussed bounding methods allow for the calibration of the foot-mounted IMU. They are propagated in discrete time by,

$$b_a(t + \Delta t) = b_a(t) + w_{b_a} \text{ and } b_g(t + \Delta t) = b_g(t) + w_{b_g}, \tag{3.25}$$

where w_{b_a} and w_{b_g} are zero-mean, normally distributed driving noises. Since accelerometer and gyroscope bias estimates are available in a typical implementation, they can be subtracted from their respective sensor measurements before they are used to propagate velocity and attitude. This is given by,

$$v_{nb}^n(t + \Delta t) = v_{nb}^n(t) + (f_{nb}^n - b_a)\Delta t, \tag{3.26}$$

and

$$C_b^n(t+\Delta t) = C_b^n(t)(I_3 + (\Omega_{nb}^b - b_g)\Delta t), \tag{3.27}$$

which are modifications of Equations (3.5) and (3.1) respectively. Given the error state approach, and the inclusion of the foot-mounted IMU's biases, the error state vector is given by,

$$\delta x = \begin{bmatrix} \delta r_{nb}^n & \delta v_{nb}^n & \delta \psi_{nb}^n & b_a & b_g \end{bmatrix}^T, \tag{3.28}$$

which is meant to correct the full states,

$$x = \begin{bmatrix} r_{nb}^n & v_{nb}^n & C_b^n \end{bmatrix}^T. \tag{3.29}$$

3.2.3.2 Process Update

Since the INS equations are responsible for propagating the full states, the primary purpose of IEZ's process update is to propagate the uncertainty in the error states. With the foot-mounted INS's PVA errors and IMU biases making up the state vector, the discrete-time state transition matrix for IEZ is given by,

$$F = \begin{bmatrix} I_3 & I_3\Delta t & 0_3 & 0_3 & 0_3 \\ 0_3 & I_3 & \left[-f_{nb}^b \wedge\right]\Delta t & C_b^n\Delta t & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 & C_b^n\Delta t \\ 0_3 & 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}. \tag{3.30}$$

The process noise vector is given by,

$$w = \begin{bmatrix} w_a & w_g & w_{b_a} & w_{b_g} \end{bmatrix}^T \tag{3.31}$$

,

and the matrix mapping these noises to the state domain is given by

$$G = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ C_r^n & 0_3 & 0_3 & 0_3 \\ 0_3 & C_r^n & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}. \tag{3.32}$$

The final part of the process update is the process noise covariance matrix. This is given for IEZ by

$$Q = \begin{bmatrix} \sigma_a^2 & 0_3 & 0_3 & 0_3 \\ 0_3 & \sigma_g^2 & 0_3 & 0_3 \\ 0_3 & 0_3 & \sigma_{b_a}^2 & 0_3 \\ 0_3 & 0_3 & 0_3 & \sigma_{b_g}^2 \end{bmatrix}. \tag{3.33}$$

3.2.3.3 Measurement Update

The measurement update for IEZ serves two purposes. The first is the actual optimal fusion of the error reduction information with the foot-mounted INS's PVA, while the second is the adjustment of the error state covariance matrix proportional to the relative weight given to the error reduction information. The measurement update's exact composition depends on the available error reduction information, but it can be broken down into three basic structures. These are the velocity update, the gyroscope bias update, and the heading update.

I. Velocity Update: ZUPT

A velocity update is used to apply a ZUPT to the foot-mounted INS's velocity. It is invoked whenever $C1$ and $C2$ are both true. Its measurement error vector is given by

$$\delta y = \delta y_{ZUPT} = v_{nb}^n - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.34}$$

and mapped onto the states by the measurement model,

$$H_{ZUPT} = \begin{bmatrix} 0_3 & I_3 & 0_3 & 0_3 & 0_3 \end{bmatrix}, \tag{3.35}$$

with a measurement covariance matrix of,

$$R_{ZUPT} = \max(\mathrm{trace}(H_{ZUPT} \cdot P \cdot H_{ZUPT}^T)). \tag{3.36}$$

II. Gyroscope Bias Update: ZARU

A gyroscope bias update is used to apply a ZARU to calibrate the foot-mounted INS's gyroscope. It is invoked whenever $C3$ and $C4$ are both true. Its measurement error vector is given by,

$$\delta y = \delta y_{b_g} = \omega_{nb}^n - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.37}$$

and mapped onto the states by the measurement model,

$$H_{ZARU} = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}, \tag{3.38}$$

with a measurement covariance matrix of,

$$R_{ZARU} = \max(\mathrm{trace}(H_{ZARU} \cdot P \cdot H_{ZARU}^T)). \tag{3.39}$$

III. Heading Update: HDR

A heading update is used to apply HDR to the foot-mounted INS's heading. It is invoked whenever $C5$ indicates straight walking. Its measurement error vector is given by,

$$\delta y = \delta y_{\delta\psi} = \delta\psi_s, \tag{3.40}$$

and mapped onto the states by the measurement model,

$$H_\psi = \begin{bmatrix} 0_3 & 0_3 & \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} & 0_3 & 0_3 \end{bmatrix}, \tag{3.41}$$

with a measurement covariance matrix of,

$$R_\psi = \max(\text{trace}(H_\psi \cdot P \cdot H_\psi^T)). \tag{3.42}$$

IV. Heading Update: Magnetometer Measurements

A heading update is also used to apply magnetometer measurements to the foot-mounted INS's heading. It is invoked whenever readings from the foot-mounted magnetometer are available. Its measurement error vector is given by,

$$\delta y = \delta y_{\delta\psi} = \psi_{nb}^n - \psi_n, \tag{3.43}$$

and mapped onto the states by the measurement model,

$$H_\psi = \begin{bmatrix} 0_3 & 0_3 & \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} & 0_3 & 0_3 \end{bmatrix}. \tag{3.44}$$

V. Multiple Measurements

To apply more than one kind of measurement update at the same time, the three discussed measurement updates can be stacked together. For example, a simultaneous ZUPT and ZARU has a measurement error of,

$$\delta y = \begin{bmatrix} \delta y_{ZUPT} \\ \delta y_{b_g} \end{bmatrix}, \tag{3.45}$$

which is mapped onto the error state vector by a combined measurement model of,

$$H = \begin{bmatrix} H_{ZUPT} \\ H_{ZARU} \end{bmatrix}, \tag{3.46}$$

with a measurement covariance of,

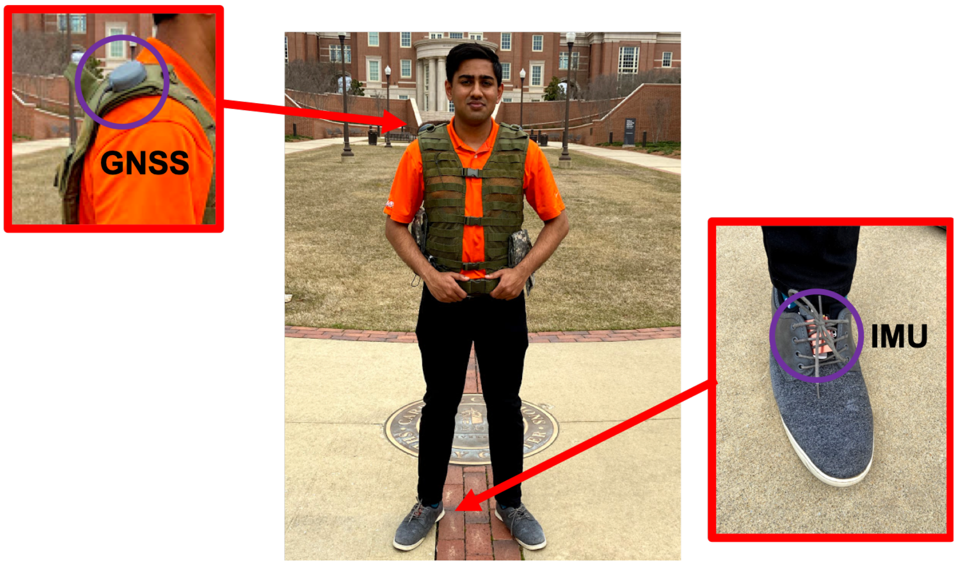

Figure 3.5: Sensor setup used to collect data for IEZ implementation.

$$R = \begin{bmatrix} R_{ZUPT} & 0_3 \\ 0_3 & R_{ZARU} \end{bmatrix}. \tag{3.47}$$

3.2.4 Implementation of IEZ

To examine its performance, the IEZ approach was implemented on real-world data gathered by a pedestrian walking a rectangular path in a parking lot. The pedestrian performed the walk while wearing a Vectornav VN-300 dual antenna GNSS/INS [92]. As shown in Figure 3.5, the IMU was attached under the shoelaces on the pedestrian's right shoe while the two GPS antennas were attached to the pedestrian's shoulders. The outputs from the foot-mounted IMU were fed into the IEZ algorithm to track the pedestrian trajectory, while GPS was used as truth. To see the effects of using different error reduction methods, the IEZ algorithms was run twice with two different measurement update structures: once with just the velocity update (ZUPT) and once with both the velocity and gyroscope bias updates (ZUPT+ZARU). The trajectory and error results are shown in Figures 3.6 and 3.7.

The z-axis gyroscope bias is known to be unobservable when ZUPTs are the only measurement source [24], a fact that is seen in Figure 3.6 by the significant positioning drift experienced during the velocity update-only trajectory. Applying a gyroscope bias update in addition to the

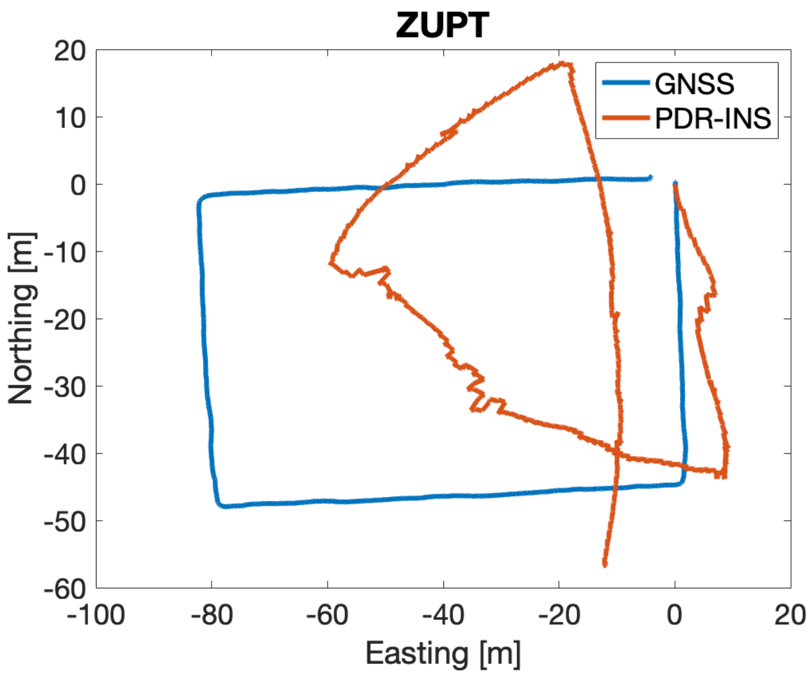

Figure 3.6: PDR-INS trajectory results using just a zero-velocity update.

velocity update allowed IEZ to estimate the z-axis gyroscope bias during the minute-long still period preceding the walk. As evidenced in Figure 3.7, having this bias estimate significantly reduced the amount of heading and positioning drift, though some drift in the positioning solution remained. It should be noted that these results are preliminary and should be treated as such. A thorough analysis of each of these methods is beyond the scope of this thesis. However, these preliminary results were included since they provide a sense of the accuracy that can be expected from the transfer alignment-initialized PDR system shown later in this thesis.

3.2.5 Summary of PDR-INS

In summary, PDR-INS is an approach to pedestrian dead reckoning that tracks pedestrian motion in real time by integrating the angular velocity and acceleration outputs from an IMU attached to the person. To address the accumulating error inherent to this approach, the IMU is attached to pedestrian's foot. This setup allows for the detection of stance and still events, which can be used to reduce the foot-mounted IMU's PVA errors in a Kalman filter framework. This general approach has the advantage of being able to track any movement, since it does not make assumptions about the pedestrian's manner of style of walking. Another advantage specific to

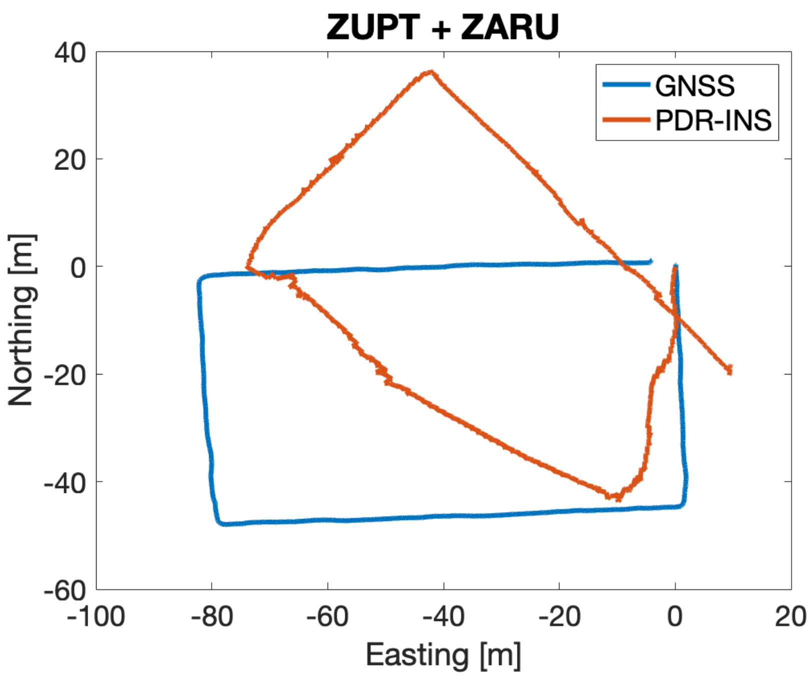

Figure 3.7: PDR-INS trajectory results using a zero-velocity and zero-angular rate update.

the IEZ algorithm is that it provides an optimal estimation framework. This framework makes it possible to incorporate other measurement sources while taking into account their uncertainties.

3.3 A Step Detection Approach to Pedestrian Dead Reckoning

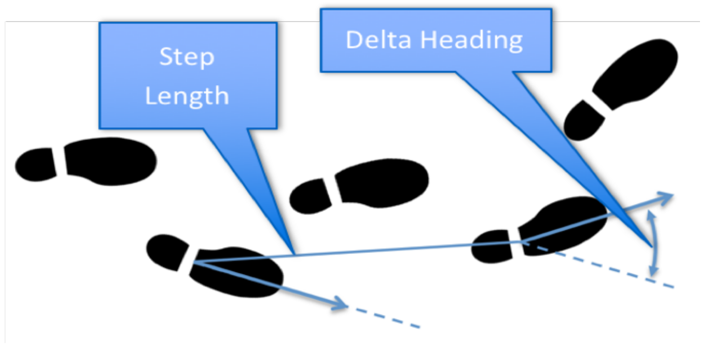

Figure 3.8: A pictoral representation of the PDR-SD approach.

An alternative approach to PDR considers each step as a discrete unit of motion with its own translational and angular displacement, as conveyed in Figure 3.8. Presumably, by summing each step's positional and angular displacement, the pedestrian's trajectory can be tracked over a significant distance. This approach is known as the step detection approach in [2] and [5], and is described at length in [11, 12, 14, 16, 30–36]. In performing summation at every step as opposed to every IMU measurement epoch, the step detection approach offers a method of PDR that is conceptually simpler than the INS-based approach described in the previous section. Known as PDR-SD in this thesis, this method is made up of four components: step detection, step length estimation, attitude determination, and the position update. Each of these components are described in detail in the following subsections.

3.3.1 Step Detection

The goal of the step detection process is to discern the occurrence of steps during regular walking motion. Accurately determining the occurrence of steps is critical, since step events are the basis from which the pedestrian's navigation solution is propagated. The sensor of choice for this task is an accelerometer, since it offers a self-contained means of determining an object's motion. This approach is used in several works [2, 6, 30, 31, 35, 36, 38].

3.3.1.1 IMU Placement

The challenge for PDR-SD is mounting the IMU on the pedestrian in a way that maximizes the percentage of steps detected while also minimizing false step detections. Mounting the IMU on the pedestrian's foot and running its output through the stance detection algorithm discussed in the previous section allows for the detection of steps [12, 28, 33, 37]. However, a pitfall of that approach lies in the fact that the foot can experience extraneous swinging motion that is not characteristic of a normal gait pattern. In other words, the foot can move while the pedestrian does not. This can lead to false step detection if the IMU is mounted on the pedestrian's foot. To better reflect the movement of the pedestrian's whole body, the IMU must instead be mounted on or near the pedestrian's torso. This can be on a backpack [5, 34], on the shoulder [36], on the back [30], or on the chest [28]. When PDR-SD is used in this thesis, the IMU is mounted on the pedestrian's chest. For further discussion of IMU locations used to perform step detection, readers are directed to [6].

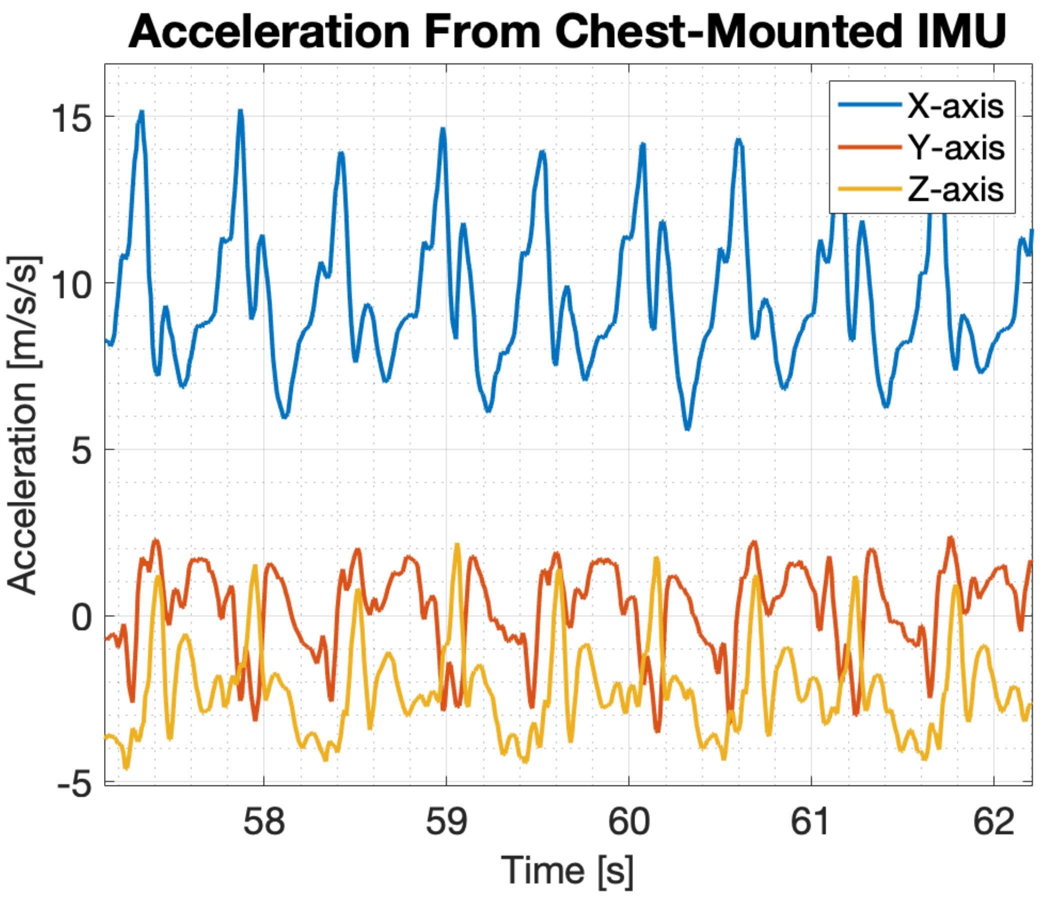

Figure 3.9: Acceleration measurements taken from a triaxial IMU mounted on the chest of a walking person.

3.3.1.2 Detection Algorithm

To detect the occurrence of steps, many different signals can be used, including acceleration, angular velocity, magnetic field, and pressure [6, 14]. Due to its relative straightforwardness, acceleration will be used in this thesis. To use acceleration for step detection, the evolution of the signal on the pedestrian's chest must be considered. This is shown in Figure 3.9, which depicts readings taken from a chest-mounted triaxial IMU mounted on a pedestrian. For this dataset, the IMU's x-axis was roughly perpendicular with the plane of the ground, which means that it was able to capture nearly all of the vertical bounce that a pedestrian normally experiences during walking [26, 38]. The large magnitude and periodicity of the x-axis' measurements allow it to be used as the primary source of information for step detection. This can be implemented in one of two ways. If a triaxial IMU is available, then the norm of the acceleration should be used [2, 28, 34]. The norm is given by,

$$||a_{nb}^b|| = \sqrt{(a_{nb,x}^b)^2 + (a_{nb,y}^b)^2 + (a_{nb,z}^b)^2}. \tag{3.48}$$

If a triaxial IMU is unavailable, a uniaxial IMU can be used instead. If this is the case, then the axis should be aligned roughly perpendicular with the plane of the ground to capture as much of the vertical bounce as possible.

Several methods exist that use the acceleration signal to detect steps. These include peak detection [28, 30, 36], thresholding [33, 35], and more advanced methods such as spectral analysis and correlation [6]. Due to its simplicity, robustness, and reliability, thresholding was chosen for use in this thesis.

For the thresholding method, a new step is detected when there is an acceleration zero crossing. For the most basic threshold detection method, acceleration zero crossings occur when the magnitude of the acceleration readings cross the magnitude of the acceleration due to gravity, which is $g = 9.81\ m/s^2$. This is based upon the assumption that when a step begins at footfall, the non-gravitation acceleration experienced at the torso is nearly zero [30].

A more advanced thresholding method calculates a maximum, minimum, and average threshold for a window of samples [35]. The size of the window is determined by the IMU sample rate. For example, if the IMU was sampled at 100 Hz, then the thresholds would be updated every 100 samples. Four conditions have to be met for there to be a detected step. Qualitatively, these are:

1. The acceleration must have a negative slope. This means the current measurement must be less than the previous measurement.

2. The previous epoch must be above the average threshold and the current epoch must be below the the average threshold. These are called acceleration crossings.

3. There must be a minimum time between each detected step. The minimum time between each step varies based on the type of motion (e.g. walking, running), the type of IMU, and the specific algorithm implementation.

4. The minimum and maximum thresholds must be values that indicate pedestrian movement. This condition was added by Ray in [24] to prevent false step detections during quasi-stationary periods.

The above thresholding method was applied to the dataset shown earlier in Figure 3.9, with the results shown in Figure 3.10. By applying this method, a pedestrian's steps can be reliably detected, which enables further implementation of the PDR-SD approach.

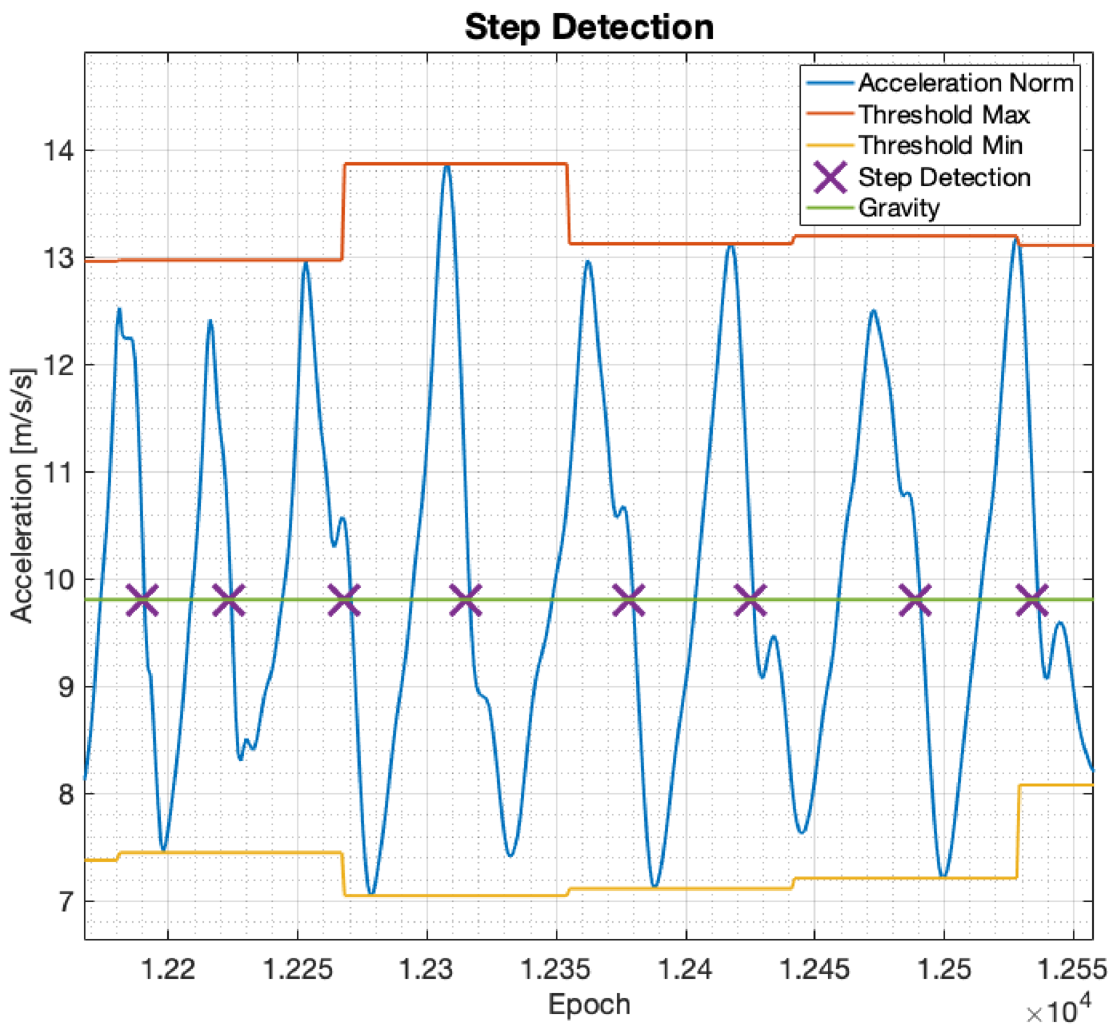

Figure 3.10: The thresholding step detection method applied to a set of data from a chest-mounted accelerometer.

3.3.2 Step Length Estimation

The second component of PDR-SD is the estimation of the pedestrian's planar (2D) translation, or step length. There are a wide variety of step length estimation algorithms with varying levels of sophistication. The algorithms found in the literature can be broken up into four categories: ones that assume a fixed step length, ones that relate step length to user height, ones that relate step length to user hip bounce, and ones that relate step length to signal characteristics.

3.3.2.1 Fixed and Height-Based Step Length

The most simplistic algorithms assume a constant step length (SL), as naturally walking pedestrians have an almost constant step length. This is an easy approach to implement as only the average step length for a user needs to be identified. A pedestrian's average step length can be computed by measuring distance traveled using GNSS and dividing it by the number of steps taken. This is given by,

$$SL_1 = \frac{x}{n}, \tag{3.49}$$

where x is the distance traveled and n is the number of steps taken. Some pedometer manufacturers relate step length to a pedestrian's height [39]. This relationship is given by,

$$SL_2 = h \cdot K, \tag{3.50}$$

where h is pedestrian's height and K is a constant based on or related to height. An often quoted value for the pedestrian height constant is $K = 0.413$ inches. This constant was created using a survey of multiple pedestrians walking naturally. Unfortunately, step length can very as much as 40% from person to person, as a person's step length is largely dependent on the length of their leg. In addition, a person's step length can vary as much as 50% as a pedestrian changes their gait (i.e. walking slower and faster than a natural pace) [38]. Unless the pedestrian walks at a natural pace for the entire distance traveled, an assumed fixed step length approach is not adequate and will introduce additional error into the PDR solution.

A slightly more sophisticated method is presented by Zhao in [35], which uses a simple tabular method of finding step length based upon step frequency and height. Note that the author calls a step length a stride in his work. The table of values for this method is presented in Table 3.1.

Table 3.1: SL_3 Definition

Steps per 2 seconds	SL_3 (m)
0-2	$h/5$
2-3	$h/4$
3-4	$h/3$
4-5	$h/2$
5-6	$h/1.2$
6-8	$h/4$
≥ 8	$1.2 \cdot h$

3.3.2.2 Step Length from Hip Bounce

To make PDR-SD available for widespread use, some researchers have implemented it using IMUs and other sensors included in smartphones. Since smartphones are usually carried in a pant or jacket pocket, this has spurred the development of step length formulas that use sensor information taken from a pedestrian's hip [40]. Since these formulas use the vertical component acceleration of the pedestrian to compute the step length, the pose of the smartphone must either be known or aligned with the vertical direction. Weinburg's work created a dynamic step length estimation formula based upon the maximum vertical displacement of the hip [28, 38]. It was shown that stride length was a function of the vertical displacement and the angle between the maximum and minimum hip position. This angle is assumed to be constant even though it is typically not. When modified to estimate step length instead of stride length, it is given by the empirical relationship,

$$SL_4 = K\sqrt[4]{a_{max} - a_{min}}, \tag{3.51}$$

where a_{max} and a_{min} are the maximum and minimum measured values of the vertical acceleration during a step. Note that implementing SL_4 requires detection of peak and valley values in the acceleration signal. This method reports step lengths to be within 8% of their true values [28, 38].

The second step length formula for hip located sensors is given by,

$$SL_5 = 0.1 \sqrt[2.7]{\frac{\sum_{i=1}^{k=N} ||a_{vert}(i)||}{N} \sqrt{\frac{K}{\sqrt{T \cdot a_{peak}}}}}, \tag{3.52}$$

where T is the step duration and a_{vert} is the vector of measured vertical accelerations during a single step. The term a_{peak} is the difference between maximum and minimum vertical accelerations during the step and N is the number of samples during one step. Similar to Equation (3.51), K in Equation (3.52) is a calibration constant. The empirical relation in Equation (3.52) was created in [39] and is based upon step length formula in [12]. This formula is given by,

$$SL_6 = 0.98 \sqrt[3]{\frac{\sum_{i=1}^{k=N} ||a_{vert}(i)||}{N}}. \tag{3.53}$$

Another step length formula uses a simple biomechanical model of a pedestrian, based upon work by [41] and presented in [42], which models a kneeless biped as an inverted pendulum. The relationship is given by,

$$SL_7 = K\sqrt{2LY - Y^2}, \tag{3.54}$$

where Y is the vertical displacement of the pedestrian's hip and L is the length of the pedestrian's leg.

3.3.2.3 Step Length from Signal Characteristics

The last category of methods of step length determination models the step length as a function of certain signal characteristics such as step frequency and variance of specific force. Step frequency has been shown to be strongly related to step length [43, 44]. Gusenbauer et al. chose to relate the step frequency to step length using the linear function,

$$SL_8 = K_1 + K_2 f, \tag{3.55}$$

where f is step frequency and K_1 and K_2 are the model coefficients [45]. Other works add terms to the above model. Ladetto models step length as a function of both step frequency and variance of the specific force, resulting in the function,

$$SL_9 = K_1 + K_2 f + K_3 \sigma_f^2, \tag{3.56}$$

where σ_f^2 is the variance of the acceleration magnitude and K_3 is an additional model coefficient [30]. Groves et al. also includes a grade term in their work, resulting in,

$$SL_{10} = K_1 + K_2 f + K_3 \sigma_f^2 + K_4 S, \tag{3.57}$$

where S is the slope term and K_4 is an additional model coefficient [5]. Note that Groves et al. found that K_2 and K_4 were both poorly estimated from their data. Therefore, the authors used the equation in a reduced form, given by,

$$SL_{11} = K_1 + K_3 \sigma_f^2. \tag{3.58}$$

For SL_8, SL_9, SL_{10}, and SL_{11}, the step frequency is computed by,

$$f_k = \frac{1}{t_{k_{stop}} - t_{k_{start}}}, \tag{3.59}$$

where $t_{k_{stop}}$ and $t_{k_{start}}$ are the times of the beginning and end of the step respectively. In order to calculate the variance of the acceleration magnitude, the magnitude is first found by computing its Euclidean norm, over the duration of a step. This is given by,

$$||\tilde{a}^b_{nb,k_{start}:k_{stop}}|| = \sqrt{(a^b_{nb,x,(k_{start}:k_{stop})})^2 + (a^b_{nb,y,k_{start}:k_{stop}})^2 + (a^b_{nb,z,k_{start}:k_{stop}})^2}. \tag{3.60}$$

From there, the variance can be computed by,

$$\sigma_f^2 = \frac{\left(\sum_{k=k_{start}}^{k_{stop}} (\tilde{a}^b_{nb,k}) - \mu_{(k_{start}:k_{stop})}\right)^2}{k_{stop} - k_{start}}, \tag{3.61}$$

where $\mu_{(k_{start}:k_{stop})}$ is the mean of the acceleration computed in Equation (3.60).

In [42] and [24], a comparison of SL_8, SL_9, and SL_{11} is given. A comprehensive comparison of all the above step length formulas is beyond the scope of this thesis. By utilizing any of the formulas outlined in this subsection, a pedestrian's step length can be estimated, allowing for the estimation of a pedestrian's translational displacement. In this thesis, SL_1 will be used to implement PDR-SD, since the pedestrian gathering the inertial made sure to walk at a steady pace.

3.3.3 Attitude Determination

To navigate in a navigation frame, a pedestrian's step length must first be mapped into that frame using a computation of the chest-mounted accelerometer's attitude. There are three possible methods for computing this attitude. These methods are 1) dead-reckoning angular velocity, 2) computing heading and tilt using magnetic and gravitational fields, and 3) fusing these two methods using an attitude heading reference system.

3.3.3.1 Dead-Reckoning Angular Velocity

The accelerometers that are used for step detection and step length estimation are often packaged together with gyroscopes in devices known as inertial measurement units (IMU). This additional sensor, which measures the the rotation rate of the gyroscope's coordinate frame, offers an opportunity for attitude determination. To determine attitude using angular velocity outputs, the outputs need to be integrated with respect to time. If only single-axis gyroscope is available, its axis must point perpendicular to the ground plane to estimate yaw. If this is done, the pedestrian's heading can be computed by integrating its angular velocity outputs. This is given in discrete-time by,

$$\psi_n(t + \Delta t) = \psi_n(t) + \omega(t)\Delta t. \tag{3.62}$$

The possession of a triaxial gyroscope opens up the possibility of tracking the IMU's attitude in three dimensions. If the attitude is parametrized as a rotation matrix, a three-dimensional attitude solution can be approximated in discrete time by,

$$C_b^n(t + \Delta t) = (I_3 + \Omega_{nb}^b \Delta t) C_b^n(t). \tag{3.63}$$

A major pitfall of this approach has to do with the quality of the angular velocity measurements. As discussed in [2], measurements taken by gyroscopes are subject to noise, bias, and other errors. These errors will cause any attitude solution computed by Equation (3.63) to drift over time. The occurrence of this drift is not guaranteed however, since a chest-mounted IMU, like its foot-mounted counterpart, does have a few opportunities to use motion assumptions to reduce the growth of attitude error. One possibility is to use use ZUPTs and ZARUs during still periods. As well, straight walking periods can be used to provide an additional means of error reduction. However, the occurrence of these still and straight walking periods are not as consistent as ZUPTs are for a foot-mounted IMU, which means that a chest-mounted IMU will experience relatively few opportunities to check the growth of error in the computation of attitude. Therefore, angular velocity measurements alone cannot be used to determine attitude for PDR-SD.

3.3.3.2 Heading and Tilt Using Magnetic and Gravitational Fields

As an alternative to dead reckoning angular velocity, the chest-mounted IMU's attitude can instead be computed with respect to Earth's magnetic and gravitational fields. The first step in this process is computation of the IMU's pitch and roll relative to a local navigation frame using measurements of gravity. This is given in [2] by,

$$\theta_A = \frac{a_{nb,x}^b}{\sqrt{{a_{nb,y}^b}^2 + {a_{nb,z}^b}^2}}, \ \phi_A = \arctan 2(-a_{nb,y}^b, -a_{nb,z}^b), \tag{3.64}$$

where the subscript A refers to "accelerometer." The second step in this process is the computation of the IMU's heading using measurements of Earth's magnetic field. These measurements can be taken by a magnetometer that is co-located with the inertial sensors. Note that the magnetometer measurements must be calibrated before they can be used for attitude determination [46]. For a magnetometer that measures only in two axes, a heading can be computed by,

$$\psi_M = \psi_{mag} + \alpha_m = \arctan(B^n_{nb,y}, B^n_{nb,x}) + \alpha_m, \tag{3.65}$$

where the subscript M refers to "magnetometer." α_m is the angle between magnetic and true north. The above equation assumes that the x and y-axes form a plane that is parallel with the navigation frame. If they do not, then the magnetometer measurements must first be leveled into a coordinate frame that is co-planar with the local navigation frame. This is given in [2, 7] by,

$$B^n_{nb} = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} B^b_{nb}. \tag{3.66}$$

Note that implementing Equation (3.66) requires the possession of a triaxial magnetometers. A major advantage that this approach has over the dead-reckoning of angular velocity is its relative resistance to drift. This resistance is derived from the fact that the IMU's attitude is referenced to external sources as opposed to being deduced internally. However, using Earth's magnetic and gravitational fields references for attitude comes with its own problems.

One problem specific to the pitch and roll computation is the assumption that the only force reflected in the accelerometer measurements $(a^b_{nb,x}, a^b_{nb,y}, a^b_{nb,z})$ is gravity. This assumption can only be true if the chest-mounted IMU is not accelerating. As shown earlier in Figures 3.9 and 3.10, this is not true for a chest-mounted IMU. The presence of significant non-gravitational acceleration violates the assumption made in Equation (3.64) and introduces error into the pitch and roll computation. This error affects the heading computation in turn through Equation (3.66).

However, even if the pitch and roll values were error-free, the computed heading value is still subject to error from magnetic disturbances. These disturbances could be anything that manipulates Earth's magnetic field, including but not limited to power lines, cars, underground pipes, motors, and generators [2]. A pedestrian that walks by any of these objects while relying on magnetic field readings for heading will compute erroneous heading values. In summary, even though there are advantages to using external field forces to determine the chest-mounted IMU's attitude, difficulties can arise in reading these field forces.

3.3.3.3 Attitude Heading Reference System

Each of the two aforementioned approaches to perform attitude determination carries its respective benefits and drawbacks. On one hand, an attitude computed from dead-reckoned angular velocity measurements is stable in the short term, but will drift in the long term. On the other hand, an attitude derived from measurements of Earth's magnetic and gravitational fields will be stable in the long term, but will be affected by local perturbations from magnetic disturbances and acceleration. Clearly, the best overall approach would be one that could draw from both approaches while maximizing their respective benefits and minimizing their drawbacks. Such approaches exist and are known as attitude heading reference systems (AHRS) [2].

In general, AHRS computes an attitude by first dead-reckoning angular velocity measurements to compute heading, pitch and roll. The dead-reckoned pitch and roll is then corrected using gravity measurements from an accelerometer. Subsequently, the corrected pitch and roll is used to rotate the magnetic field measurements into the navigation frame, which are used in turn to correct the dead-reckoned heading. To minimize the effects of short term errors in the gravity and magnetic field measurements, smoothing filters with low gains are used to incorporate these measurements into the dead-reckoned attitude. Many AHRS also incorporate maneuver and magnetic disturbance rejection routines to further shield the attitude estimate from errors. Sometimes, a Kalman filter is used to implement an AHRS, since this enables the smoothing gains to be dynamically optimized and allows for sensor errors such as gyroscope biases to be estimated.

As established earlier, computation of an attitude using a KF approach can be implemented by using the KF to estimate attitude error, and then using the error to correct the attitude estimate. The implementation of an AHRS KF presented in [3] follows a similar approach, where the author presents an AHRS KF that tracks the full state,

$$x = \begin{bmatrix} C_b^n \end{bmatrix} \tag{3.67}$$

by estimating the error states

$$\delta x = \begin{bmatrix} \delta\psi_{nb}^n & b_g \end{bmatrix}^T . \tag{3.68}$$

This is accomplished by propagating (in discrete time) the IMU's attitude using Equation (3.27) (which is Equation (3.63) after accounting for gyroscope bias), its attitude error using Equation (3.9), and its gyroscope bias using Equation (3.25). The resulting state transition matrix for this approach is given by

$$F = \begin{bmatrix} I_3 & C_b^n \Delta t \\ 0_3 & I_3 \end{bmatrix} , \tag{3.69}$$

while its process noise vector is,

$$w = \begin{bmatrix} w_g & w_{b_g} \end{bmatrix}^T \tag{3.70}$$

.

The matrix mapping these noises to the state domain is given by,

$$G = I_2 , \tag{3.71}$$

while the covariances for these noises is given by,

$$Q = \begin{bmatrix} \sigma_g^2 & 0_3 \\ 0_3 & \sigma_{b_g}^2 \end{bmatrix} . \tag{3.72}$$

The attitude and gyroscope biases propagated by this process are then updated using pitch and roll computed using Equation (3.64 and heading using Equation (3.65). This is implemented using a measurement error vector,

$$\delta y = \begin{bmatrix} \phi_A - \phi(t + \Delta t) \\ \theta_A - \theta(t + \Delta t) \\ \psi_M - \psi(t + \Delta t) \end{bmatrix}, \tag{3.73}$$

where the $\phi(t + \Delta t)$, $\theta(t + \Delta t)$, and $\psi(t + \Delta t)$ refer to the roll, pitch, and yaw computed from the rotation matrix propagated using Equation (3.27). These measurement errors are mapped onto the error states using the measurement matrix

$$H = \begin{bmatrix} I_3 & 0_3 \end{bmatrix}, \tag{3.74}$$

with a noise covariance of

$$R = \begin{bmatrix} \sigma_A^2 & 0 & 0 \\ 0 & \sigma_A^2 & 0 \\ 0 & 0 & \sigma_M^2 \end{bmatrix}. \tag{3.75}$$

By implementing the described KF, or any of the approaches described in [2, 47–50], a PDR-SD system can estimate a dynamically fused attitude that has a degree of immunity to both gyroscope errors and perturbations due to acceleration and magnetic disturbances.

3.3.4 Position Update

Once the steps have been detected, their lengths estimated, and the chest-mounted IMU's attitude determined, the three sources of information can be put together to propagate the pedestrian's position in the navigation frame. In PDR-SD, two simplifying assumptions are made. These assumptions are

1. The pedestrian has only three degrees of freedom. These are North, East, and rotation about the vertical axis. This is contrasted with PDR-INS, which considers six degrees of freedom.
2. The pedestrian is facing the direction they are walking in.

These assumptions inform the equations that are used to propagate the pedestrian's position. Since the pedestrian can experience only planar motion, this means that only two equations are needed for positional propagation. In addition, the planar motion assumption means that only the heading is needed to map the step length into the navigation frame. From [5, 11, 24] the position propagation equations are,

$$x_{k+1} = x_k + SLsin(\psi_n) \tag{3.76}$$

$$y_{k+1} = y_k + SLcos(\psi_n) \tag{3.77}$$

where x and y represent East and North respectively. In addition, since the pedestrian is facing the direction of travel, no further angle needs to be added to ψ_n. By following the procedure outlined in this section, a pedestrian's position can be propagated using the output from a chest-mounted IMU and the PDR-SD method.

3.3.5 Implementation of PDR-SD

To examine its performance, the PDR-SD method was implemented on real-world data gathered by a pedestrian walking a rectangular path in a parking lot. The pedestrian performed the walk while wearing a Vectornav VN-300 dual antenna GPS/INS [92]. As shown in Figure 3.11, the IMU was attached to vest on the right of the pedestrian's sternum while the two GPS antennas were attached to the pedestrian's shoulders. True position was provided by GPS, while the acceleration outputs from the chest-mounted IMU were used to detect steps using the thresholding method. Since the pedestrian maintained a steady pace, a constant step length

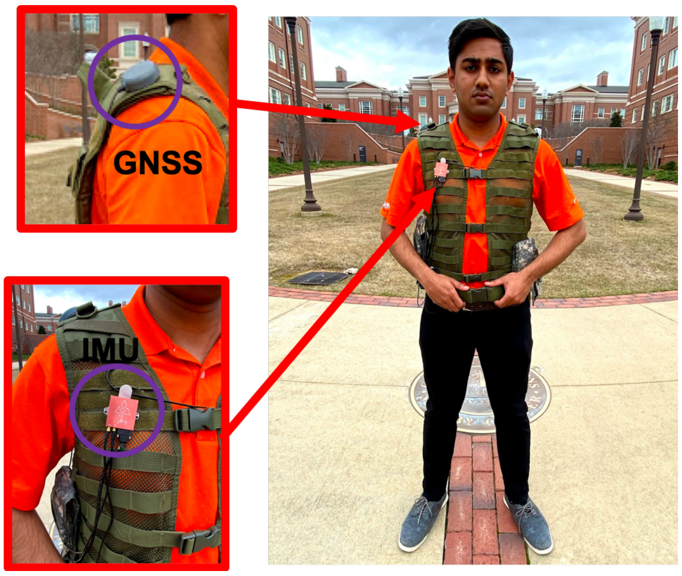

Figure 3.11: Sensor setup used to collect data for PDR-SD implementation.

of was assumed for the data run. To see the effects of using different attitude determination methods, three different methods were used, which were:

1. **Corrected pitch and roll only**: Computing a smoothed pitch and roll by running measurements of gravity and angular velocity through the filter described in [50]. This filter uses corrected pitch and roll to level the angular velocity measurements, and then computes heading by dead-reckoning the vertical-axis angular velocity without any correction using magnetic field measurements.

2. **Corrected heading, pitch, and roll**: Using the attitude output by the Vectornav's onboard proprietary AHRS. Broadly speaking, the proprietary AHRS computes an attitude by dead-reckoning angular velocity measurements and then corrects the full attitude using gravity measurements and magnetic field measurements.

PDR-SD was run once with each attitude determination method, resulting in two different trajectories. The two PDR-SD trajectories, along with GPS truth, are shown in Figure 3.12. Both trajectories follow GPS fairly closely. A key difference between the two PDR-SD trajectories is the amount of drift each experiences. The mag-less trajectory (in orange) deviates from GPS slightly more than the mag-corrected trajectory (in yellow), particularly in the bottom-most leg

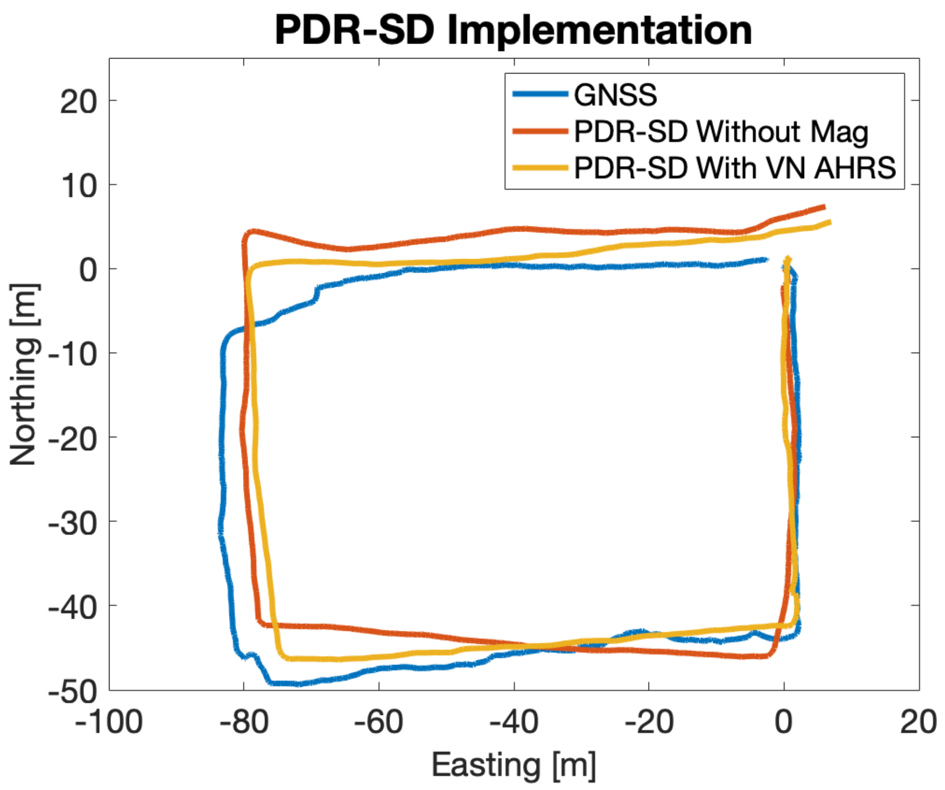

Figure 3.12: PDR-SD trajectories resulting from two different implementations of AHRS.

of the rectangle. This is due the fact the magnetic field measurements used by Vectornav's proprietary AHRS provide an external reference for the pedestrian's heading. Without this heading reference, the PDR-computed trajectory experiences drift as a result of noise, bias, and other errors in the gyroscope. It is notable that the orange mag-less trajectory, experiences far less drift than the ZUPT+ZARU implementation of PDR-INS previously shown in Figure 3.7. This is likely due to the bounds that the gravity measurements placed on pitch and roll errors in both AHRS implementations, which is an information source not available in the implementation of PDR-INS in the previous section. It should be noted that these results are preliminary and should be treated as such. A thorough analysis of each of these methods is beyond the scope of this work. However, these preliminary results were included since they provide a sense of the accuracy that can be expected from the transfer alignment-initialized PDR system shown later in this thesis.

3.3.6 Summary of PDR-SD

In summary, PDR-SD is an approach to pedestrian dead reckoning that tracks pedestrian motion by treating each step as a discrete unit of movement.By summing the positional and angular displacement of each step, it is possible to track a pedestrian's motion over significant distances. This operating principle assumes that the pedestrian's motion is primarily planar, and that the pedestrian is facing the direction they are walking. These simplifying assumptions make it possible to implement PDR-SD using less sophisticated sensors than what is required for the implementation of PDR-INS.

Chapter 4

Transfer Alignment as a Solution to the PDR Initialization Problem

As described in Chapter 2, all dead-reckoning systems, including PDR, face two challenges which must be overcome before use. These challenges are

1. **Accumulation of Error:** An error in a sensor reading taken at one point in time will affect the rest of the navigation solution.
2. **Initialization:** Since dead-reckoning computes displacement as opposed to absolute position, it requires initial absolute position and attitude values to be able to navigate in a global frame.

Since PDR is a particular implementation of dead-reckoning, these two challenges each play roles in the real-world implementation of any PDR system.

The challenge of error accumulation is seen in the drift experienced by PDR-computed solutions, examples of which are shown in Figures 3.6, 3.7, and 3.12. One strategy for mitigating this drift is to use higher-quality IMUs, which have smaller errors, i.e. less noise and slower bias drift rates. However, using higher quality sensors is not always desirable since they come at a high price and often impose a significant penalties in size, weight, and power-consumption. For this reason, various error bounding methods such as ZUPT [7], ZARU [7], magnetic field and gravity measurements [7, 28], GNSS-fusion [11], and feature/map matching [15, 24] have been used to bound PDR error.

To meet the challenge of acquiring initialization information, a few assumptions have to be made. Consider PDR-SD. At a minimum, initial position, heading, pitch, and roll values are needed. Additionally, if an AHRS is used, an initial attitude error covariance matrix may also be needed. GNSS can be used for initializing position, gravity measurements can be used for initializing pitch and roll, and calibrated magnetic field measurements can be used for initializing heading [13, 15–18, 27, 28, 31]. The weakest point in this initialization procedure is the assumption of the availability of GNSS and unperturbed readings of Earth's magnetic field. The weakness arises from the fact that these two information sources are not readily available in all environments, such as urban canyons and indoors. If PDR-SD is to be initialized in such an environment, an alternative source of initialization information must be used.

Similar problems arise with implementations of PDR-INS. As with PDR-SD, initial position, heading, pitch, and roll values are needed. Since INS tracks velocity as well, this quantity must also be initialized. If the implementation of PDR-INS uses the IEZ algorithm, then the IMU biases and the error covariance matrix must also be initialized. As with PDR-SD, GNSS can be used for initializing position and velocity, gravity measurements can be used for initializing pitch and roll, and calibrated magnetic field measurements can be used for initializing heading. [13, 16–18, 27, 28, 31]. This makes one extra assumption over the initialization routine used for PDR-SD. Specifically, it assumes that the pedestrian is standing still [15, 24]. This assumption is necessitated by the fact that IMU is mounted on the pedestrian's foot as opposed to the chest. Since the foot experiences far greater acceleration than the chest, using acceleration measurements taken at the foot to compute pitch and roll will yield highly erroneous pitch and roll values unless the pedestrian is standing still. This extra assumption is yet another weakness in the initialization routine for PDR-INS. On top of the GNSS and magnetic field weaknesses pointed out for PDR-SD previously, this standstill requirement goes one step further and requires users to perform a particular action in order to be able to use their PDR system. Requiring the user to stand still even for a couple of minutes can be problematic for users with time-sensitive missions, such as first responders. For these users, an initialization routine that cuts out required actions such as standstill periods is of interest.

Fortunately, recent developments have created an opportunity to develop a more practical means of acquiring initialization information for PDR systems. As vehicles become more automated, they are being equipped with navigation sensors, including IMUs, of a level of quality that is sufficient for navigational use [19, 20]. Given that some PDR users, such as first responders, ride in vehicles, this development creates an opportunity to use the vehicle's navigation system as part of a system that can determine the PDR system's initial position, velocity, and attitude values.

An instrumented vehicle potentially has two navigation sensors that are helpful in computing these initial values. These sensors are the vehicle's IMU and GNSS receiver. The vehicle's IMU measures the vehicle's specific force and angular velocity, much like its pedestrian-mounted counterpart, while the GNSS receiver measures the vehicle's position and velocity. The information collected by both of these vehicle-mounted sensors can be related to measurements taken by the pedestrian's IMU through the kinematic equations discussed in Chapter 2. These relationships make it possible to initialize the pedestrian navigation system by transferring the navigation solution from the vehicle in a process known as transfer alignment. This chapter first discusses existing transfer alignment approaches, and follows the discussion with the presentation of novel approach which uses solutions to Wahba's Problem to handle large misalignment angles. The chapter closes with a discussion of these algorithms' observability requirements.

4.1 Rigid Body Kinematics

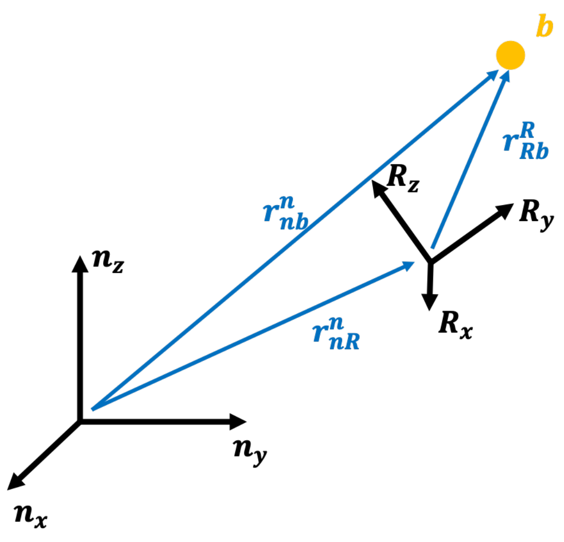

Figure 4.1: A coordinate-frame depiction of a pedestrian inside a moving vehicle

Relating the navigation information from the vehicle and pedestrian navigation systems requires applying the transformations presented in Chapter 2 to the scenario of a pedestrian riding inside a moving vehicle. The scenario is shown in Figure 4.1, noting that the vector arrows from Chapter 2 have been omitted for neatness. The navigation frame n is fixed in space while the vehicle carrying the pedestrian, represented by frame R, arbitrarily translates and rotates through space. The pedestrian, represented by frame b, also arbitrarily translates and rotates. Note that the sensors mounted on the pedestrian (IMU) and vehicle (IMU and GNSS) are assumed to be coincident with the origins of b and R respectively. From Figure 4.1, the rotation transformation from b to n is given by,

$$C_b^n = C_R^n C_b^R \tag{4.1}$$

Any piece of vector information, including angular velocity (ω), position (r), velocity (v), and acceleration (a) can be resolved in another frame using Equation (4.1). Relating angular velocities in different coordinate frames is straightforward, i.e. simply multiply by a rotation matrix [2, 21]. However, relating position, velocity, and acceleration in different coordinate frames is more involved, since the translational offsets between the frames play more of a role. According to Figure 4.1, the position of b relative to n, resolved in both n and R, is given by,

$$\begin{aligned} r_{nb}^n &= r_{nR}^n + C_R^n r_{Rb}^R \\ r_{nb}^R &= r_{nR}^R + r_{Rb}^R. \end{aligned} \tag{4.2}$$

Following the process detailed in Chapter 2, the time derivative of the first line of Equation (4.2) is taken. Doing this once gives the velocity of b relative to n, [2, 21]

$$\begin{aligned} v_{nb}^n &= v_{nR}^n + C_R^n v_{Rb}^R + C_R^n \Omega_{nR}^R r_{Rb}^R \\ v_{nb}^R &= v_{nR}^R + v_{Rb}^R + \Omega_{nR}^R r_{Rb}^R. \end{aligned} \tag{4.3}$$

Taking the time derivative of velocity gives the acceleration of b relative to n, [2, 21]

$$\begin{aligned} a^n_{nb} &= a^n_{nR} + C^n_R a^R_{Rb} + C^n_R(\Omega^R_{nR})^2 r^R_{Rb} + C^n_R \dot{\Omega}^R_{nR} r^R_{Rb} + 2C^n_R \Omega^R_{nR} v^R_{Rb} \\ C^R_b a^b_{nb} &= a^R_{nR} + a^R_{Rb} + (\Omega^R_{nR})^2 r^R_{Rb} + \dot{\Omega}^R_{nR} r^R_{Rb} + 2\Omega^R_{nR} v^R_{Rb} \end{aligned} \tag{4.4}$$

The kinematic equations, which are Equations (4.1) - (4.4), relate measurements taken by the navigation sensors attached to the pedestrian and vehicle. The practical use of these equations is complicated by the fact that only some of their terms are known. The vehicle's position and velocity, along with all of the acceleration and angular velocity terms, are known. However, the pedestrian's position and velocity are not known, and must be derived through applying inertial navigation equations.

Further complicating these equations' practical use is the presence of several offset terms. These terms include relative position r^R_{Rb}, relative velocity v^R_{Rb}, relative acceleration a^R_{Rb}, and rotational offset C^R_b. Determining these offsets is not a trivial problem in the context of vehicle-to-pedestrian transfer alignment, since an IMU mounted on a pedestrian's body is free to move independently of the vehicle. In such a scenario, the relative position, velocity, acceleration, and misalignment between the two IMUs are extremely difficult to determine. Fortunately, the complexity of this problem can be reduced by performing transfer alignment only when the pedestrian's IMU is still with respect to the vehicle. This state of relative rigidity is known as pedestrian-rigid (PR).

When PR is true, there is no relative motion between the two IMUs, i.e v^R_{RB} and a^R_{Rb} go to zero. This leaves just the static offsets (r^R_{Rb} and C^R_b) as unknowns. When this condition is true, Equation (4.3) becomes,

$$C^R_b v^b_{nb} = v^R_{nR} + \Omega^R_{nR} r^R_{Rb}, \tag{4.5}$$

while Equation (4.4) becomes,

$$C_b^R a_{nb}^b = a_{nR}^R + (\Omega_{nR}^R)^2 r_{Rb}^R + \dot{\Omega}_{nR}^R r_{Rb}^R. \tag{4.6}$$

In addition, the angular velocities measured in R and b become related by a constant C_b^R, which is given by,

$$\omega_{nb}^R = C_b^R \omega_{nb}^b \text{ and } \omega_{nR}^b = C_R^b \omega_{nR}^R. \tag{4.7}$$

As well, the positions of R and b become related by a constant r_{Rb}^R, the form of which is given by Equation (4.2).

As discussed earlier, the ultimate goal of initializing a pedestrian navigation system is to know its position r_{nb}^n, velocity v_{nb}^n, and attitude C_b^n (PVA) at the moment the pedestrian starts navigating. In the context of in-vehicle PDR initialization, this moment is no later than when the pedestrian dismounts from the vehicle. Using outputs from the IMU and GNSS on the vehicle and the IMU on the pedestrian in conjunction with the kinematic equations simplified by PR, it becomes possible to initialize the pedestrian navigation system's PVA with a transferal of PVA from the vehicle's navigation system. Algorithms that perform this function are known as transfer alignment algorithms.

Transfer alignment (TA) dates back to at least the 1960s [54]. It is used to initialize, align, and calibrate an IMU in motion using a reference navigation system [2]. Applications of transfer alignment include the navigation of cars, planes, ship, spacecraft, and guided weapons. In general, there are two major parts to a TA system. They are the aligning (pedestrian) IMU b and the reference system (vehicle) R. Reference systems used for TA can take on different forms, though two commonly used systems are GNSS and other INSs [2]. Due to the wide range of possible applications for TA, there are a plethora of different TA algorithms in use. What follows in is an overview of the major TA algorithms in existence.

4.2 INS-Based Transfer Alignment

Most TA systems perform real-time calibration and alignment of an IMU in motion by comparing the position, velocity, and attitude (PVA) output by its INS against those of a reference system's. Such systems are explored in [51-66]. This approach is known in this thesis as TA-INS. TA-INS is the oldest and most established approach for transfer alignment that can be found in the literature. In fact, most papers that use the term transfer alignment are actually referring to this approach. In the literature, TA-INS is often discussed in the context of aligning and calibrating munitions attached to military aircraft, though the fundamentals of TA-INS can be used for any application that requires optimal fusion between two sets of PVA values. In general, TA-INS estimates the PVA and calibration parameters of the aligning IMU by 1) integrating its acceleration and angular velocity outputs using inertial navigation techniques and 2) optimally fusing the aligning INS's PVA with that of the reference system. The result is an estimate of the aligning INS's navigation states that is more accurate than what could be computed from the aligning INS or reference system individually. The following subsections give an overview of the technique as presented in the literature.

4.2.1 Reference Information

One major component of TA-INS is the establishment of the reference information that will be used to align and calibrate the aligning INS. This step is critical, since the exact makeup of the reference information depends on the system used as reference.

A reference system used in early applications of TA was simply another, higher quality INS, as described in [51–55, 84–87, 90]. The higher quality is what allows the reference INS to act as a reliable alignment standard. Typically, TA-INS utilizes only velocity and (if available) attitude measurements from the reference INS, and does not include position measurements. The reason behind the exclusion of position measurements lies in the fact that position and velocities output by an INS are both integrals of the IMU's acceleration output. Since all three of of these quantities are essentially the same, using more than one simultaneously offers no additional information. Of these three, velocity is usually used for reference, since maintaining the reference INS velocity requires integrating its acceleration output. The integration averages out accelerometer noise, leading to a velocity output that is less noisy than the accelerometer output. This ultimately results in faster state estimation than could be achieved by using acceleration output [2]. These benefits can be taken further by performing the second integration of acceleration, and using the reference INS position instead of its velocity [54]. However, the slight increase in speed that results usually does not justify the added complexity and computational cost of a second integration.

GNSS is a reference system used in more recent TA applications, as described in [57–66]. The accurate externally-referenced positioning information it provides makes it an excellent alignment standard. Since GNSS is at its core, a positioning system, TA-INS implementations that use GNSS as a reference typically utilize position, velocity, and (if multiple antennas are available) attitude as the measurement update. Even though velocity is the time derivative of position, it is still included as a separate measurement since 1) GNSS computes velocity independently of position and 2) GNSS-measured velocity is typically a cleaner measurement than GNSS-measured position [4].

Regardless of the system used as reference, the PVA outputs (or some combination thereof) of the reference system are related to the PVA outputs of the aligning INS by Equations (4.2), (4.5), and (4.7).

4.2.2 Inertial Navigation

The other major component of TA-INS is the computation of the aligning IMU's PVA. This is done in real time by applying inertial navigation equations to its acceleration and angular velocity outputs. Inertial navigation was described in detail in Chapter 2 of this thesis, but an abridged discussion is given here for convenience. In discrete-time, the aligning IMU's attitude update is given by,

$$C_b^n(t+\Delta t) = C_b^n(t)(I_3 + \Omega_{nb}^b \Delta t), \tag{4.8}$$

where t represents the time of the previous computation epoch and $t + \Delta t$ represents the time of the current computation epoch. The acceleration frame transformation and gravity compensation are given by,

$$a^n_{nb} = \frac{1}{2}(C^n_b(t + \Delta t) + C^n_b(t))a^b_{nb}, \tag{4.9}$$

$$f^n_{nb} = a^n_{nb} - \gamma^n_{nb}, \tag{4.10}$$

where γ^n_{nb} represents gravitational acceleration. The exact value of gravitational acceleration to be used depends on where the IMU is on the Earth (or other celestial body). For applications in a local navigation frame, it is sufficient to assume that gravitational acceleration is simply,

$$\gamma^n_{nb} = \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} m/s^2. \tag{4.11}$$

Finally, the velocity and position updates are given in discrete time by,

$$v^n_{nb}(t + \Delta t) = v^n_{nb}(t) + f^n_{nb}\Delta t, \tag{4.12}$$

$$r^n_{nb}(t + \Delta t) = r^n_{nb}(t) + v^n_{nb}\Delta t. \tag{4.13}$$

As discussed in Chapter 2, the accuracy of an inertial navigation solution is highly dependent on the errors in the IMU measurements. A simple error model of the IMU's accelerometer and gyroscope is given by,

$$a^n_{nb} = \tilde{a}^n_{nb} + b_a + w_a, \tag{4.14}$$

$$\omega^n_{nb} = \tilde{\omega}^n_{nb} + b_g + w_g. \tag{4.15}$$

where $\tilde{a}_{nb}^n$ and $\tilde{\omega}_{nb}^n$ represent the true acceleration and angular velocity, w_a and w_g represent IMU noises, and b_a and b_g represent accelerometer and gyroscope biases.

Since C_b^n, v_{nb}^n, and r_{nb}^n are computed from IMU measurements in error, it stands to reason that the errors in these quantities depend on the IMU errors. The evolution of these errors in discrete time is given by,

$$\delta\psi_{nb}^n(t+\Delta t) = \delta\psi_{nb}^n(t) + C_b^n(t)(b_g + w_g)\Delta t, \tag{4.16}$$

$$\delta v_{nb}^n(t+\Delta t) = \delta v_{nb}^n(t)\Delta t + (-[f_{nb}^n \wedge]\delta\psi_{nb}^n)\Delta t + C_b^n(b_a + w_a)\Delta t, \tag{4.17}$$

$$\delta r_{nb}^n(t+\Delta t) = \delta r_{nb}^n(t) + \delta v_{nb}^n\Delta t, \tag{4.18}$$

which are the attitude, velocity, and position errors, respectively. They are related to the aligning INS's PVA values by,

$$\tilde{C}_b^n = (I_3 - [\delta\psi_{nb}^n \wedge])C_b^n, \tag{4.19}$$

$$\tilde{v}_{nb}^n = v_{nb}^n + \delta v_{nb}^n, \tag{4.20}$$

$$\tilde{r}_{nb}^n = r_{nb}^n + \delta r_{nb}^n, \tag{4.21}$$

where the quantities under tildes represent the true PVA of the INS [2].

4.2.3 TA-INS-ESKF

The conventional method of using reference PVA information to align an INS is to optimally fuse the reference PVA values with those from the aligning INS. The goal of optimal fusion is to compute a PVA that minimizes the difference between the aligning INS's and reference system's PVA given the uncertainty in each system's output. This sort of problem is ideally suited for a Kalman filtering (KF) approach. A description of Kalman filtering is given in Appendix A, while Kalman filtering in the context of inertial navigation was summarized in

Chapter 2. The goal of this subsection therefore, is to fit the TA-INS problem described in the previous two subsections into the KF algorithm.

Each of the two systems in TA-INS, the aligning INS and the reference system, correspond to the two different phases of the KF, these being the process and measurement updates respectively. It is common to put the aligning INS in the process update and the reference system in the measurement update. This setup is especially advantageous if the reference system runs at a lower update rate than the aligning INS.

4.2.3.1 State Selection

Though PVA values are an intuitive choice for the TA KF's states, they are not used as the state values. Instead, the aligning INS's PVA errors are used as the KF states. Using the PVA errors as states makes this an implementation of an error-state Kalman filter (ESKF). The ESKF algorithm, detailed in Appendix A, maintains two sets of states, the full states and the error states. The incoming measurements are used to optimally estimate the errors in the full states given the uncertainty in the error states and measurements. These error estimates are then used to correct the full states. In the context of TA-INS, this means that PVA measurements from the reference system are used to correct errors in the aligning INS. This is known as TA-INS-ESKF.

The justification for using the ESKF for TA as opposed to a full-state KF lies in the nature of the inertial navigation equations used to propagate the aligning INS's PVA. (Equations (4.8) - (4.13)) Neither the PVA values nor its propagation equations fit the profile required for the full state KF algorithm, namely that it be a vector of scalar values that are a linear combination of the state propagation equations. This is especially true in the case of the attitude, C_b^n. However, PVA errors and their propagation equations fit this profile, which justifies maintaining the PVA errors as states inside an ESKF framework for applications that use INS as a process model.

At a bare minimum, most implementations of TA-INS-ESKF maintain the aligning INS's velocity and attitude along with their respective errors as full and errors states respectively. Though its inclusion might seem intuitive, the estimation of the aligning INS's position is not useful unless GNSS is used as reference, as described previously. Other states that often are estimated are the aligning INS's IMU biases, since applying PVA from a reference system allows for the calibration of the aligning IMU. The evolution of these biases over time is given by,

$$b_a(t + \Delta t) = b_a(t) + w_{b_a} \text{ and } b_g(t + \Delta t) = b_g(t) + w_{b_g}, \tag{4.22}$$

where w_{b_a} and w_{b_g} are zero-mean, normally distributed driving noises. Since accelerometer and gyroscope biase estimates are both available in a typical implementation, they can be subtracted from from their respective sensor measurements before they are used to propagate velocity and attitude. This is given by,

$$v^n_{nb}(t + \Delta t) = v^n_{nb}(t) + (f^n_{nb} - b_a)\Delta t, \tag{4.23}$$

and

$$C^n_b(t + \Delta t) = C^n_b(t)(I_3 + (\Omega^b_{nb} - b_g)\Delta t), \tag{4.24}$$

which are modifications of Equation (4.12 and 4.8 respectively. Given the error state approach, and the inclusion of aligning IMU's biases, the error state vector is given by,

$$\delta x = \begin{bmatrix} \delta r^n_{nb} & \delta v^n_{nb} & \delta\psi^n_{nb} & b_a & b_g \end{bmatrix}^T, \tag{4.25}$$

which is meant to correct the full states,

$$x = \begin{bmatrix} r^n_{nb} & v^n_{nb} & C^n_b \end{bmatrix}^T. \tag{4.26}$$

4.2.3.2 Process Update

Since the INS equations are responsible for propagating the full states, the primary purpose of TA-INS-ESKF's process update is to propagate the uncertainty in the error states. With the aligning INS's PVA errors and IMU biases making up the state vector, the discrete-time state transition matrix for TA-INS-ESKF is given by,

$$F = \begin{bmatrix} I_3 & I_3\Delta t & 0_3 & 0_3 & 0_3 \\ 0_3 & I_3 & \left[-f_{nb}^{b} \wedge\right]\Delta t & C_b^n\Delta t & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 & C_b^n\Delta t \\ 0_3 & 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}. \tag{4.27}$$

The process noise vector is given by,

$$w = \begin{bmatrix} w_a & w_g & w_{b_a} & w_{b_g} \end{bmatrix}^T, \tag{4.28}$$

and the matrix mapping these noises to the state domain is given by,

$$G = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ C_r^n & 0_3 & 0_3 & 0_3 \\ 0_3 & C_r^n & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}. \tag{4.29}$$

The final part of the process update is the process noise covariance matrix. This is given for TA-INS-ESKF by,

$$Q = \begin{bmatrix} \sigma_a^2 & 0_3 & 0_3 & 0_3 \\ 0_3 & \sigma_g^2 & 0_3 & 0_3 \\ 0_3 & 0_3 & \sigma_{b_a}^2 & 0_3 \\ 0_3 & 0_3 & 0_3 & \sigma_{b_g}^2 \end{bmatrix}. \tag{4.30}$$

4.2.3.3 Measurement Update

The measurement update for TA-INS-ESKF serves two purposes. The first is the actual optimal fusion of the reference system's PVA with the aligning INS's PVA, while the second is the adjustment of the error state covariance matrix proportional to the relative weight given to the reference information. Since the measurement update includes outputs from the reference system, this means that it will be composed of some combination of PVA measurements. The measurement update's exact composition is how different implementations of TA-INS-ESKF are distinguished from each other. Three common implementations include traditional transfer alignment, rapid transfer alignment, and loosely coupled GNSS/INS fusion.

I. Traditional Transfer Alignment

Traditional transfer alignment (TTA) is described by Sutherland and Gelb in [54] and uses velocity measurements from a reference INS to correct the PVA errors and estimate the biases of the aligning INS. The measurement error is given by,

$$\delta y = \delta y_v = v^n_{nR} - v^n_{nb} + C^n_b \Omega^b_{nb} r^b_{Rb}, \tag{4.31}$$

and mapped onto the states by the measurement model,

$$H_{TTA} = \begin{bmatrix} 0_3 & -I_3 & 0_3 & 0_3 & 0_3 \end{bmatrix}, \tag{4.32}$$

with a measurement covariance matrix of,

$$R_{TTA} = \begin{bmatrix} \sigma^2_{v_r} \end{bmatrix}. \tag{4.33}$$

Since the reference system for TTA is another INS, the filter presented in [54] excludes position states. This means that the state transition model, process noise vector, process noise matrix, process noise covariance, and measurement models presented in [54] and [2] exclude the rows and columns that correspond to the position error state. For the sake of consistency and brevity, this reduction has not been shown.

II. Rapid Transfer Alignment

An extension of TTA was first described in [53] by Kain and Cloutier and expanded upon by various authors in [51] and [66]. The extension is known as rapid transfer alignment (RTA). It is so named since alignment and calibration of the aligning INS generally takes less time with

RTA when compared with TTA. RTA achieves this by adding an attitude measurement from the reference INS. The small angle attitude measurement error is given by,

$$I_3 + [\delta y_\psi \wedge] = C_R^n C_b^R C_n^b. \tag{4.34}$$

The measurement error vector for RTA includes both velocity and attitude errors and is given by,

$$\delta y = \begin{bmatrix} \delta y_v \\ \delta y_\psi \end{bmatrix}, \tag{4.35}$$

which is mapped onto the error states by,

$$H_{RTA} = \begin{bmatrix} 0_3 & -I_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & -I_3 & 0_3 & 0_3 \end{bmatrix}, \tag{4.36}$$

with a covariance matrix of,

$$R_{RTA} = \begin{bmatrix} \sigma_{v_r}^2 & 0_3 \\ 0_3 & \sigma_{\psi_r}^2 \end{bmatrix}. \tag{4.37}$$

As with TTA, the RTA filter presented in [53] also excludes the position states.

III. Loosely Coupled GNSS/INS Fusion

Loosely coupled GNSS/INS fusion is an implementation of TA-INS-ESKF that uses GNSS outputs as the reference, as described in [57–66]. It is critical to note that the term transfer alignment is not conventionally used to discuss loosely coupled GNSS/INS. Despite this difference in nomenclature, loosely coupled GNSS/INS fusion is included in this subsection since it is both mathematically and functionally similar to TTA and RTA. The biggest difference between TTA/RTA and GNSS/INS fusion is the nature of the two systems used to align and calibrate the aligning INS. GNSS, unlike the reference INS used in RTA and TTA, is capable of computing independent position and velocity solutions. Therefore, both position and velocity are used together in the measurement update for GNSS transfer alignment. The measurement error for position is given by,

$$\delta y_r = r^n_{nR} - r^n_{nb} + C^n_b r^b_{Rb}, \tag{4.38}$$

which is combined into a measurement error vector of,

$$\delta y = \begin{bmatrix} \delta y_r \\ \delta y_v \end{bmatrix}. \tag{4.39}$$

The measurement error is mapped onto the state vector by,

$$H_{LC} = \begin{bmatrix} -I_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & -I_3 & 0_3 & 0_3 & 0_3 \end{bmatrix}, \tag{4.40}$$

with a measurement noise covariance matrix of,

$$R_{LC} = \begin{bmatrix} \sigma^2_{r_g} & 0_3 \\ 0_3 & \sigma^2_{v_g} \end{bmatrix}. \tag{4.41}$$

If either position or velocity is not returned by the GNSS system being used, then the corresponding rows can be removed from the measurement model and noise matrix.

4.2.4 The Misalignment Problem

Up until this point, the discussion has included both the lever arm r^R_{Rb} and misalignment C^R_b between the aligning INS and reference system in the measurement errors. Inclusion of this information requires that it be determined beforehand. This information can be determined using physical measurements or drawings of the vehicle carrying the two systems. In the event that drawings or measurements are unavailable, the lever arm and misalignment can instead be estimated as states in the TA-INS-ESKF framework. The topic has been addressed in [2, 57–62].

In the context of the in-vehicle PDR initialization problem presented in this thesis, r_{Rb}^{R} and C_b^R are unknown quantities that are impractical to measure, which means that they must be estimated online, as they are in [2, 57–62]. However, a problem arises in using the KF framework presented in [57, 58, 62] to estimate the misalignment C_b^R. Specifically, using a KF framework to estimate misalignment requires that it be constituted by small angles ($\leq 10°$). This requirement is driven by the fact that the misalignment C_b^R, which is a non-linear relationship, must be linearized to be estimated by a KF. This linearization is only valid if the error resulting from non-linearity is small [2]. If the offset angle is large, which would be the case if the misalignment angles were large, any KF, including the ones presented in [57, 58, 62, 77], would not be able to solve for the misalignment angles. For the in-vehicle PDR initialization problem, there is no basis upon which to make assumptions about the size of the misalignment angles, meaning they could be quite large. Therefore, estimating these angles using a framework based on a linear KF would be unproductive. Instead, alternative means must be used to solve for the full misalignment between the aligning and reference systems.

4.3 Misalignment Computation

To begin looking for such an alternative, consider Equations (4.6) and (4.7). Assuming that the aligning and reference systems are each IMUs, their raw outputs (assuming negligible errors) are related to each other by the misalignment C_b^R through these two equations. This means that either of these equations can be used to compute C_b^R. However a quick glance at these equations' terms shows that the misalignment is the only quantity that relates the angular velocity measurements from the two IMUs. By contrast, both the misalignment and lever arm stand between the two IMUs' acceleration measurements. This means that solving for the misalignment using angular velocity measurements in conjunction with Equation (4.7) is a far simpler task than using acceleration measurements in conjunction with Equation (4.6) . Therefore, for the sake of simplicity, the former will be used to solve for the misalignment.

Solving for a rotation matrix C in an equation with the form of 4.7, given two sets of k vector measurements a and b is a problem of computing an optimal rotation matrix that minimizes the cost function,

$$L(C) = \frac{1}{2}\sum_k \frac{1}{\sigma_a^2 + \sigma_b^2}\|a_k - Cb_k\|^2. \tag{4.42}$$

This problem was proposed in 1965 by Grace Wahba in [67]. Solutions to Wahba's problem have typically been used to determine the attitude of spacecraft [73]. However, the problem and its many solutions are also relevant in the transfer alignment context since both attitude determination and transfer alignment aim to find a rotation matrix that relates two sets of vector measurements. In the context of Equation (4.7), the two sets of vector measurements are the angular velocity measurements from the two IMUs, while the rotation matrix is the misalignment between the two. Written in this context, Equation (4.42) becomes,

$$L(C_b^R) = \frac{1}{2}\sum_k \frac{1}{\sigma_{\omega_R}^2 + \sigma_{\omega_b}^2}\|\omega_{nR_k}^R - C_b^R \omega_{nb_k}^b\|^2. \tag{4.43}$$

On first glance, solving the above cost function this seems straightforward. A matrix that maps ω_{nb}^b onto ω_{nR}^R can be computed by,

$$C_b^R = \omega_{nb}^b {\omega_{nR}^R}^T (\omega_{nR}^R {\omega_{nR}^R}^T)^{-1}. \tag{4.44}$$

This approach is taken in [68–70]. However, the drawback of this approach is that it computes each term in the rotation matrix individually, with no consideration given to the relationship between the terms. As will be seen later in this chapter, not considering the interdependencies in a rotation matrix makes it more difficult to compute the misalignment between two IMUs. Therefore, this section will focus on solutions to Wahba's problem that preserve these interdependencies, as covered in [71–76]. Note that the rotation matrix C_b^R will be substituted with the more general rotation matrix C for the following discussion.

4.3.1 First Solutions

Authors of solutions to Wahba's problem have found it convenient to rewrite the problem in the following manner:

$$L(C) = \lambda_0 - \mathrm{trace}(CB^T), \tag{4.45}$$

where

$$\lambda_0 = \sum_k a_k, \tag{4.46}$$

and

$$B = \sum_k \frac{1}{\sigma^2_{\omega_R} + \sigma^2_{\omega_b}} \omega^R_{nR_k} \omega^b_{nb_k}{}^T. \tag{4.47}$$

The matrix B is known as the *attitude profile matrix*. It is a weighted sum of all k vector quantities that are used to solve for C. This matrix is the basis of a whole family of solutions to Wahba's problem, the goal of which is to maximize $\mathrm{trace}(CB^T)$. In the original problem statement, Wessner and Brock proposed a solution, given by,

$$C = B(B^T B)^{-1/2}. \tag{4.48}$$

This solution requires that B be non-singular. In parallel, Farrell and Stuelpnagel noted that B has a polar decomposition of,

$$B = WR, \tag{4.49}$$

where W is orthogonal and R is symmetric and positive semidefinite. R can be diagonlized by,

$$R = VDV^T, \tag{4.50}$$

where V is orthogonal and D is diagonal. The optimal rotation matrix C is given by,

$$C = WV\,\mathrm{diag}[1 \quad 1 \quad \det(W)]V^T. \tag{4.51}$$

4.3.2 SVD and FOAM

A solution equivalent to the Farrell and Stuelpnagel approach was proposed by Landis Markley in [71] nearly a decade after the initial problem statement. It involves taking the singular value decomposition (SVD) of B, which is,

$$B = U\Sigma V^T = U\,\mathrm{diag}[\Sigma_{11} \quad \Sigma_{22} \quad \Sigma_{33}]V^T, \tag{4.52}$$

where the U and V are orthogonal and $\Sigma_{11} > \Sigma_{22} > \Sigma_{33}$. $\mathrm{trace}(CB^T)$ is computed by,

$$tr(U^TCV\,\mathrm{diag}[\Sigma_{11} \quad \Sigma_{22} \quad \Sigma_{33}]), \tag{4.53}$$

the value of which is maximized when,

$$U^TCV = \mathrm{diag}[1 \quad 1 \quad det(U)det(V)], \tag{4.54}$$

which gives the rotation matrix,

$$C_{SVD} = U\,\mathrm{diag}[1 \quad 1 \quad det(U)det(V)]V^T. \tag{4.55}$$

The estimation error is characterized by a small angle rotation error vector ϕ_{err}, which is defined by,

$$\mathrm{expm}[\phi_{err} \wedge] = C_{true}C_{SVD}^T. \tag{4.56}$$

The terms in ϕ_{err} correspond to small angle errors in roll, pitch, and yaw relative to the the reference IMU's coordinate frame. The error covariance of this angular error is given in,

$$P_{SVD} = U \;\; \mathrm{diag}[(\Sigma_2 + \Sigma_3 \det(U)\det(V))^{-1} \; (\Sigma_1 + \Sigma_3 \det(U)\det(V))^{-1} \; (\Sigma_2 + \Sigma_1)^{-1}] \; U^T. \tag{4.57}$$

The equivalence to the solution by Farrell and Stuelpnagel can be seen if $U = WV$. Though the SVD method is incredibly stable from a numerical standpoint, it has not been widely applied since it is a computationally intense method when compared to other available methods [73]. To address this problem, Markley developed the Fast Optimal Attitude Matrix (FOAM) approach in [72]. For FOAM, the rotation matrix C is rewritten as,

$$C = [(\kappa + \|B\|^2)B + \lambda \operatorname{adj}(B^T) - BB^T B]/\xi, \tag{4.58}$$

where,

$$\|B\| = \Sigma_{11}^2 + \Sigma_{22}^2 + \Sigma_{33}^2. \tag{4.59}$$

The scalar coefficients κ and ξ are defined in terms of λ as,

$$\kappa = \frac{1}{2}(\lambda^2 + \|B\|^2) \tag{4.60}$$

$$\xi = \kappa\lambda - \det(B). \tag{4.61}$$

The variable λ can be computed by iteratively solving,

$$(\lambda^2 + \|B\|^2)^2 - 8\lambda \det(B) - 4adj(\|B\|^2) = 0. \tag{4.62}$$

Once λ has been solved for, then κ and ξ become available, which allows for the computation of C_{FOAM}. Once C_{FOAM} has been computed, its error covariance can be computed by performing,

$$P_{FOAM} = (\kappa\lambda - \det(B))^{-1}(\kappa + BB^T). \tag{4.63}$$

The strength of FOAM is that it is less computationally intense than most of the available methods for finding C [73].

4.3.3 Q-Method and QUEST

Since a rotation matrix R can be related to a four-component quaternion, C can be found by solving for the equivalent quaternion q. The first useful solution of Wahba's problem that solved for q was devised by Davenport in 1977 in [74]. The centerpiece of his Q-Method is the matrix

$$K = \begin{bmatrix} S - I\,\mathrm{trace}(B) & z \\ z^T & \mathrm{trace}(B) \end{bmatrix}, \tag{4.64}$$

where,

$$S = B + B^T, \tag{4.65}$$

and,

$$z = \begin{bmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{21} \end{bmatrix}. \tag{4.66}$$

This K-matrix lets Equation (4.45) be written as

$$L = 1 - q^T K q. \tag{4.67}$$

From this equation, it is easy to see that Equation (4.67) is minimized when $q^T K q$ is maximized. This occurs when q is equal to the eigenvector associated with the largest eigenvalue of K, known as λ_{max}. The attitude determination problem is a problem of solving for that eigenvalue. The most basic method of solving for that eigenvalue would be to take the characteristic polynomial of K and solve for the zeros. Unfortunately, this method is almost as computationally intense as the SVD method discussed earlier [73]. Therefore, most implementations of the Q-method take other approaches.

A widely used approach, called Quaternion Estimator (QUEST) by its creators in [75], involves applying an iterative approach to solve for q. By applying the Cayley-Hamilton Theorem to S, the characteristic equation for λ_{max} is given by,

$$\lambda^4 + (a + B)\lambda^2 - c\lambda + (ab + \frac{c}{2}\,\mathrm{trace}(S) - d) = 0, \tag{4.68}$$

where,

$$a = (\frac{1}{2}\,\mathrm{trace}(S))^2 - tr(\mathrm{adj}(S)), \tag{4.69}$$

$$b = (\frac{1}{2}tr(S))^2 + Z^T Z, \tag{4.70}$$

$$c = \det(S) + Z^T S Z, \tag{4.71}$$

$$d = Z^T S^2 Z. \tag{4.72}$$

Numerical algorithms, such as Newton-Raphson method, can be applied on Equation (4.68) with $\lambda = 1$ as a starting point. Based on Equation (4.67) and assuming that λ_{max} is close to unity, its value is easily computed by a very few number of iterations (generally just a single iteration) with reasonable accuracy. Once λ_{max} has been found, the corresponding q can be found with,

$$Kq = \lambda_{max} q. \tag{4.73}$$

QUEST is a popular approach for finding C due to its speed. It is the fastest of the Wahba's problem solutions discussed here, and is able to achieve this speed with no loss in accuracy [73]. For the sake of brevity, several more approaches for minimizing the cost function in Equation (4.45) have been excluded from the thesis. These include the ESOQ family of methods. For further information on these approaches, readers are directed to [73].

4.3.4 Gyroscope Bias Computation

Any one of the aforementioned solutions to Wahba's problem is an easy-to-implement solution that can solve for the optimal rotation matrix that satisfies both Equations (4.43) and (4.7). However, both of these equations assume that the gyroscope measurements have negligible errors. Oftentimes, this is not the case. The biases on the gyroscope measurements can significantly affect the angular velocity values received by a user, especially for the types of IMUs used in PDR. Therefore, accounting for the biases b_g of the aligning IMU, as TA-INS-ESKF does, will lead to a more accurate computation of the misalignment in theory. The biases are related to the two sets of angular velocity measurements and misalignment through,

$$\omega_{nR}^R = C_b^R(\omega_{nb}^b - b_g). \tag{4.74}$$

Adding b_g as an extra set of unknowns changes the nature of the problem. This new problem is no longer equivalent to Wahba's problem since two quantities must now be solved for, namely C_b^R and b_g. This means that none of the aforementioned Wahba's problem solution methods are applicable. Instead, a different solution method must be developed. One approach, used by Rogers et al. in [55], is to use least-squares to simultaneously solve for the misalignment and the bias vector. The goal of using least squares is to estimate the misalignment and the bias vector such that the sum of the squares of the residuals over k measurement epochs is minimized. This is given by the cost function,

$$\sum_k = ||r_k||^2 = \sum_k ||y_k - h_k(x)||^2 = \sum_k \frac{1}{\sigma_{\omega_R}^2 + \sigma_{\omega_b}^2} ||\omega_{nR_k}^R - C_b^R(\omega_{nb_k}^b - b_g)||^2. \tag{4.75}$$

In Equation (4.75), x is the vector that represents the states to be estimated, which are the rotation matrix and bias vector. Representing b_g in x is straightforward since it is a vector. However, difficulty arises when representing the matrix C_b^R in a vector. One option might be to representing it in the state vector using all 9 of the rotation matrix's terms individually. However, as discussed earlier, this is problematic, since it does not guarantee the orthonormality of the resulting matrix. [55] addresses this problem by assuming that the misalignment angles are small, which allows the rotation matrix to be represented as a vector ψ_{bR}^{R}. For the PDR initialization problem presented in this thesis however, assumptions about the size of the misalignment cannot be made. A vector representation of C_b^R that both maintains its integrity as a rotation matrix and represents large misalignment angles is an Euler angle representation. As discussed in Chapter 2, the Euler angle sequence used in this thesis is a ZYX body-fixed sequence. This means that the misalignment can be written in terms of its constituent Euler angles by,

$$C_b^R(\psi,\theta,\phi) = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) \\ c(\theta)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\phi)c(\theta) \end{bmatrix}. \tag{4.76}$$

Since existing literature does not cover the possibility of simultaneously estimating Euler angles and gyroscope biases, an original least-squares estimator must be developed that does the same. Writing C_b^R in terms of its constituent Euler angles means that the least-squares state vector is,

$$x = \begin{bmatrix} \phi & \theta & \psi & b_{g,x} & b_{g,y} & b_{g,z} \end{bmatrix}^T. \tag{4.77}$$

Note that this makes the relationship between x and the two sets of angular velocity measurements non-linear. In other words, $h_k(x)$ in Equation (4.75) cannot be written as a linear combination of these states. To solve for x, a non-linear least squares approach, such as Newton-Raphson or Guass-Newton, must be used [4].

To solve for the misalignment and gyroscope biases using the Gauss-Newton method, an initial guess for the states x_i must be formed. With a guess for x_i the residual can be computed for all k angular velocity values. This is given by,

$$y_{1:k} - h_{1:k}(x_i) = \begin{bmatrix} \omega^R_{nR,x_1} \\ \omega^R_{nR,y_1} \\ \omega^R_{nR,z_1} \\ \omega^R_{nR,x_2} \\ \omega^R_{nR,y_2} \\ \omega^R_{nR,z_2} \\ \vdots \\ \omega^R_{nR,x_k} \\ \omega^R_{nR,y_k} \\ \omega^R_{nR,z_k} \end{bmatrix} - \begin{bmatrix} C^R_b(\psi_i, \theta_i, \phi_i) \\ C^R_b(\psi_i, \theta_i, \phi_i) \\ \vdots \\ C^R_b(\psi_i, \theta_i, \phi_i) \end{bmatrix} \begin{bmatrix} \omega^b_{nb,x_1} - b_{g,x_i} \\ \omega^b_{nb,y_1} - b_{g,y_i} \\ \omega^b_{nb,z_1} - b_{g,z_i} \\ \omega^b_{nb,x_2} - b_{g,x_i} \\ \omega^b_{nb,y_2} - b_{g,y_i} \\ \omega^b_{nb,z_2} - b_{g,z_i} \\ \vdots \\ \omega^b_{nb,x_k} - b_{g,x_i} \\ \omega^b_{nb,y_k} - b_{g,x_i} \\ \omega^b_{nb,z_k} - b_{g,x_i} \end{bmatrix}, \tag{4.78}$$

where $y_{1:k} - h_{1:k}(x_i)$ is $3k \times 1$ vector. The initial guess can be improved by iteratively updating the state estimate. This iteration is given by,

$$x_{i+1} = x_i + (J_i^T J_i)^{-1} J_i^T (y_{1:k} - h_{1:k}(x_i)). \tag{4.79}$$

where J_i is the Jacobian matrix, or the partial derivative of the residuals with respect to the states. It is a $n \times 3k$ matrix given by,

$$J_i = \frac{(\partial y_{1:k} - h_{1:k}(x_i))}{\partial x_i}. \tag{4.80}$$

Given enough iterations, the state estimates will eventually converge to the true values of ψ, θ, ϕ, and b_g. The error covariance of x is given by,

$$P_{GN} = (\sigma^2_{\omega_R} + \sigma^2_{\omega_b})(J_i^T J_i)^{-1}, \tag{4.81}$$

where J_i is the latest available Jacobian matrix. As with the P_{SVD} and P_{FOAM}, the angular error terms in P_{GN} correspond to small angle errors in roll, pitch, and yaw relative to the reference IMU's coordinate frame. The aim is that simultaneously solving for the misalignment and bias vector allows for a more accurate computation of the misalignment between the aligning (pedestrian) and reference (vehicle) IMUs than can be achieved with a Wabha's problem solution.

4.4 Lever Arm Computation

Once the misalignment has been estimated using one of the aforementioned methods, the lever arm r^R_{Rb} between the aligning and reference IMUs can be estimated. This estimation can be performed in a number of different ways. One option is to estimate the lever arm as a state in a TA-INS-ESKF framework. This approach is taken in [59–61]. The other option is to independently estimate the lever arm using Equation (4.6), which assumes that the IMU's errors are negligible. Since C^R_b has already been estimated, the only unknown terms in 4.6 are the lever arm terms. The aim is to estimate the lever arm such that the sum of the squares of the residuals over k measurement epochs is minimized. This cost function is given by,

$$\sum_k ||y_k - M_k x|| = \sum_k \frac{1}{\sigma^2_{a_R} + \sigma^2_{a_b}} ||C^R_b a^b_{nb_k} - a^R_{nR_k} - (\Omega^R_{nR_k})^2 r^R_{Rb} - \dot{\Omega}^R_{nR_k} r^R_{Rb}||^2. \qquad (4.82)$$

The above cost function can be minimized using a least-squares approach. Upon inspection of Equation (4.6), it can be seen that the lever arm is a linear combination of the angular velocity and angular acceleration terms. This means that the lever arm can be computed through a linear least-squares approach, as done in [81] and [70]. To solve for the lever arm using least-squares, Equation (4.6) must be written in matrix form with the unknown quantities arranged in a vector. This is given for k inertial sensor measurements by,

$$y_{1:k} = M_{1:k}x$$
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_k \end{bmatrix} \begin{bmatrix} r^R_{Rb,x} \\ r^R_{Rb,y} \\ r^R_{Rb,z} \end{bmatrix}, \tag{4.83}$$

where the acceleration measurement vector y_k is,

$$y_k = C^R_b \begin{bmatrix} a^b_{nb,x_k} - a^R_{nR,x_k} \\ a^b_{nb,y_k} - a^R_{nR,y_k} \\ a^b_{nb,z_k} - a^R_{nR,z_k} \end{bmatrix}. \tag{4.84}$$

and where the mapping matrix M_k is,

$$M_k = \begin{bmatrix} -{\omega^R_{nR,y_k}}^2 - {\omega^R_{nR,z_k}}^2 & -\dot{\omega}^R_{nR,z_k} - \omega^R_{nR,x_k}\omega^R_{nR,y_k} & \dot{\omega}^R_{nR,y_k} + \omega^R_{nR,x_k}\omega^R_{nR,z_k} \\ \dot{\omega}^R_{nR,z_k} + \omega, x^R_{nR_k}\omega^R_{nR,y_k} & -{\omega^R_{nR,x_k}}^2 - {\omega^R_{nR,z_k}}^2 & -\dot{\omega}^R_{nR,x_k} + \omega^R_{nR,y_k}\omega^R_{nR,z_k} \\ -\dot{\omega}^R_{nR,y_k} + \omega, x^R_{nR_k}\omega^R_{nR,z_k} & \dot{\omega}^R_{nR,x_k} + \omega^R_{nR,y_k}\omega^R_{nR,z_k} & -{\omega^R_{nR,x_k}}^2 - {\omega^R_{nR,y_k}}^2 \end{bmatrix}. \tag{4.85}$$

Since this problem is linear, the least-squares estimate can be performed in one computational step, as opposed to the iteration required for the misalignment plus bias solution presented earlier. To solve for x, the pseudoinverse of M must be multiplied by y. This is given by,

$$x = (M^T_{1:k}M_{1:k})^{-1}M^T_{1:k}y_{1:k}, \tag{4.86}$$

with the estimate error covariance given by,

$$P_{la} = (\sigma^2_{a_R} + \sigma^2_{a_b})(M^T_{1:k}M_{1:k})^{-1}. \tag{4.87}$$

Given enough IMU data, this least squares approach can solve for the lever arm between the aligning (pedestrian) and reference (vehicle) IMUs.

4.4.1 Accelerometer Bias Computation

As with the misalignment estimation, the accuracy of the lever arm estimation can be enhanced with the simultaneous estimation of the aligning IMU's accelerometer biases. Since the simultaneous estimation of the lever arm and accelerometer bias has not been addressed in the literature, the algorithm proposed by [81] must be extended to include these biases. These accelerometer biases are related to the aligning IMUs outputs by,

$$C_b^R a_{nb}^b - a_{nR}^R - b_a = (\Omega_{nR}^R)^2 r_{Rb}^R + \dot{\Omega}_{nR}^R r_{Rb}^R. \tag{4.88}$$

Solving for the biases requires only slight modification of x and M. The new state vector is,

$$x = \begin{bmatrix} r_{Rb,x}^R & r_{Rb,y}^R & r_{Rb,z}^R & b_{a,x} & b_{a,y} & b_{a,z} \end{bmatrix}^T, \tag{4.89}$$

and the new mapping matrix is,

$$M_k^{bias} = \begin{bmatrix} M_k & I_3 \end{bmatrix}. \tag{4.90}$$

The other steps, such as the state estimation and covariance matrix computation are the same as for lever arm-only computation.

4.5 Observability and Maneuvering Requirements

The idea of a system's observability is of great importance in many navigation problems, since a system's observability indicates whether or not it is possible to estimate its states based on given measurement inputs. At first, this problem may seem trivial, since a moving object's PVA states can usually be directly measured. For example, if an airplane's heading needs to be determined, it can be measured by a compass. However, there are situations in which direct measurement of some states will not be possible. If the airplane's compass was non-functional for instance, its heading would have to be determined from another source. Certainly heading could not be determined if the only measurement available was a reading of the pressure in the plane's cabin, but what if East and North velocity were available from GNSS? This information is not heading, but is much more related to heading than cabin pressure. Potentially, the velocity measurements from GNSS could be used to infer or observe the airplane's heading. The question is, under what conditions is this information enough? These questions of relating the information from the sensors to knowledge of the desired states, particularly unmeasured states, are at the heart of observability. A basic definition of observability is given in [82]:

Definition 1 *A system is said to be observable if the initial state $x(t_0)$ can be determined from the output $y(t)$ over the finite time interval $[t_0, t_f]$.*

This definition serves as a general definition of what is meant by observability for all types of systems. In the context of transfer alignment, the goal is to estimate the aligning INS's PVA and IMU calibration parameters (x) using measurements from a reference navigation system (y). Some of these states are explicitly provided as measurements by the reference system, such as velocity and attitude in rapid transfer alignment. However, other critical states such as misalignments, lever arms, and sensor biases, are not. The idea of observability becomes most relevant in the computation of these "hidden" quantities, since the TA algorithm must infer or observe them from the information that is provided. As with the airplane heading example, the question that must be asked about any TA approach is, under what circumstances is there enough information to determine these hidden quantities? The rest of this section attempts to answer this question by describing the observability properties of the different quantities estimated by the various transfer alignment algorithms presented in this thesis.

4.5.1 Observing the PVA and IMU Biases

The observability properties of the aligning INS's PVA and IMU biases have been extensively explored in the literature. The findings in [58, 59, 62–65, 84–87] all offer valuable insight into the kind of circumstances that make all of the PVA and IMU bias states observable. To analyze the observability properties of these quantities, these works consider them in the context of the TA-INS-ESKF algorithm. This ESKF algorithm can be formulated as an equivalent least squares problem. For a 12 state ESKF that excludes position, this is given by,

$$\delta y_v = O \begin{bmatrix} \delta v_{nb}^{n} \\ \delta \psi_{nb}^{n} \\ b_a \\ b_g \end{bmatrix}, \tag{4.91}$$

where O maps the error states to the velocity measurements from the reference systems. This matrix is known as the total observability matrix [2, 84], and is given over k measurement epochs by,

$$O = \begin{bmatrix} H_1 \\ H_2 F_1 \\ H_3 F_2 F_1 \\ \vdots \\ H_k F_{k-1} \dots F_2 F_1 \end{bmatrix}. \tag{4.92}$$

In traditional transfer alignment, the only quantity explicitly provided to the Kalman filter from the reference system is the velocity error, as expressed by the one-row measurement model H_{TTA}. The other 9 states, which include the aligning INS's attitude errors and IMU biases, must be inferred from this information. In order to solve for these hidden states using least-squares, O must be non-singular. This in turn requires it to be full-rank, which in this case is rank 12. A full-rank matrix indicates that the ESKF has been able to form a unique mathematical link between the attitude and bias states and the velocity measurements from the reference system. Since the non-velocity states are not initially known, the rank of O is initially less than 12 at $k = 1$. This means, in order to achieve full rank, the values in O must be manipulated over the k epochs that the ESKF is running. Since the structure of H_{TTA} is set by the reference input's composition, the only way to affect the rank of O is to manipulate the values in the state transition model F. The makeup of F is in turn affected by the dynamic information in F, these being the specific force, angular velocity, and the body-to-nav rotation matrix. This means that a certain sequence of accelerations and/or rotations must be performed by the body carrying the aligning and reference systems to ensure that O is full-rank and that the full state vector is observable. In the case of TTA, high-g horizontal turns [54, 87] have been shown to be sufficient for full state observability.

Using a rapid transfer alignment measurement model changes the nature of the problem. The addition of attitude information from the reference system reduces the states the ESKF has to infer down to the aligning INS's IMU biases. This additional information from the reference is embodied in a new two-row matrix H_{RTA}. Altering the structure of H from H_{TTA} to H_{RTA} consequently alters that of O as well. As a result, the maneuvering requirements for full state observability in RTA are different from those for TTA. In [86], Zhilan Xiong determined that RTA needs only angular acceleration and angular jerk in one axis to achieve full state observability. For military aircraft trying to align an attached munition, this is attained with a few wing rocks [53] and generally takes less time than the required maneuvers for TTA [53, 54]. In general, since RTA provides one more state (attitude) as a reference input than TTA provides, its maneuvering requirements are more relaxed than those for TTA.

The findings in these works bring up an important point. What allows a TA algorithm to estimate hidden states is the performance of certain maneuvers. As such, the goal of analyzing TA's observability properties is not so much to evaluate the observability of the system, as much as it is to evaluate the observability of the system along a certain trajectory or maneuver. Determining the observability of a certain maneuver sequence or trajectory is a very useful endeavor as an analysis or design tool, since it allows a designer to draw conclusive and defined operating boundaries outside of which a TA algorithm is guaranteed to fail. For this reason, existing literature has focused on a maneuver-centric approach to analyzing the observability of the aligning INS's PVA and IMU biases [58, 59, 62–65, 84–87]. In all of these works, the researchers explore the exact combinations of states that are made observable by certain maneuver sequences by analyzing the structure of O.

4.5.2 Observing the Misalignment

An additional state that can be estimated is the misalignment between the reference system and aligning INS. This state's observability requirements has been the subject of research in the context of multi-antenna loosely-coupled GNSS/INS in [57] and [58]. In both of these papers, Hong et al., using methods similar to those described in the previous subsection, showed that the misalignment between an INS and a multi-antenna GNSS array can be observed if the vehicle carrying both devices experiences time-varying linear acceleration in at least two different directions, while there is no angular motion.

An alternative perspective on this finding can be seen by inspecting Equation (4.6). In a situation where the vehicle is experiencing purely linear motion, there is zero angular motion, which makes the angular velocity and angular acceleration terms in Equation (4.6) go to zero. This makes the relationship between the accelerations experienced by the vehicle's INS (b) and its GNSS array (R),

$$C_b^R a_{nb}^b = a_{nR}^R. \tag{4.93}$$

As shown by Shuster and Oh in [75], a minimum of two linearly independent vectors are necessary to estimate the rotation matrix C_b^R in an equation such as Equation (4.93) using a Wahba's problem solution. This condition that is satisfied when the vehicle carrying the INS and GNSS antenna array experiences time-varying linear acceleration in at least two different directions, sans angular motion. Specifically, in [57] and [58], the authors were able to estimate the misalignment between the INS and antenna array by using gravitational acceleration while the vehicle was sitting still and by using forward acceleration when the vehicle started moving.

Note that gravitational acceleration is only available as a means to estimate misalignment when the vehicle is sitting still. If a period in which the vehicle is sitting still is not available, then purely linear acceleration data from periods in which the vehicle is moving must be used instead. Most vehicles, however, do not experience purely linear acceleration. Ground vehicles in particular experience significant angular motion during ordinary operation. If a misalignment is to be estimated on a ground vehicle with any non-zero angular motion, the angular velocity terms in Equation (4.6) become non-zero. This means that any estimate of the misalignment that is computed by applying a Wahba's problem solution to Equation (4.93) will be systemically corrupted in the presence of angular motion. This raises the question of whether or not it is possible to reformulate the misalignment estimation problem in a way that excludes this systemic corruption.

Systemic corruption in the misalignment computation can be eliminated if angular motion is used to compute the misalignment during vehicle motion instead of acceleration. This approach is included in [58] with the use of a three-antenna GNSS array as a reference system for the vehicle's INS. As described in [2], the measurements taken by an array of at least three GNSS antennas can be used to compute its attitude, which means that the three-antenna array in [58] implicitly provided an attitude update to the vehicle's INS. By building an observability matrix and examining its structure, Hong et. al. determined that the misalignment can be observed with angular motion in at least two perpendicular axes.

An alternative perspective on this finding can be seen by inspecting the relationship between angular velocities of the vehicle's INS and antenna array, which is given by,

$$C_b^R \omega_{nb}^b = \omega_{nR}^R. \tag{4.94}$$

As shown in [75], a minimum of two linearly independent vectors are necessary to estimate the rotation matrix C_b^R in an equation such as Equation (4.94) using a Wahba's problem solution, a condition that is satisfied when the vehicle carrying the INS and GNSS antenna array experiences angular velocity in at least two perpendicular axes.

Note that there are no motion terms in Equation (4.94) besides those that represent angular motion. The absence of other motion quantities indicates that using angular velocity to compute the misalignment eliminates the opportunity for systemic corruption that is present when acceleration (or linear motion in general) is used instead. It is for this reason that this thesis prefers to use angular motion to estimate the misalignment rather than linear motion. Since the computation of misalignment using angular motion in a Wahba's problem solution (as opposed to the ESKF in [58] and [57]) and its application on a ground vehicle has not been explicitly addressed in the literature (to the author's knowledge), the remainder of this subsection will address this topic and the rationale behind it.

4.5.2.1 Misalignment Computation Using Angular Velocities in Wahba's Problem

I. Observability Analysis

As discussed in the previous section, solving for a misalignment is a problem of computing an optimal rotation matrix that relates vector information resolved in two different coordinate frames. Several different techniques exist for estimating this rotation matrix, all of which attempt to infer/observe the rotation matrix from the given vector information. To this end, the vector information must meet one requirement. This requirement, detailed in [75], is that the vector information contain at least two linearly independent vectors. To illustrate the rationale behind this, recall from Chapter 2 that the product of a rotation matrix and its transpose is,

$$C_b^R C_R^b = I_3. \tag{4.95}$$

The above is only possible if a rotation matrix is made up of mutually orthogonal unit vectors. The implication of this fact is that any column or row in C_b^R can be computed by taking the cross product of the other two columns or rows, respectively. As a result, recovering two unit vectors making up a rotation matrix is enough to recover the whole. Recovering a column in rotation matrix is simple. Simply multiply the matrix by a vector. This is given for a one-component vector of arbitrary length by,

$$a \begin{bmatrix} C_{11} \\ C_{21} \\ C_{31} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix}. \tag{4.96}$$

Performing the above operation recovers of one column of the matrix's columns. To recover the whole matrix, just one more column needs to be recovered. This can be done by multiplying the rotation matrix by a second vector that is not parallel with the first. This is given by,

$$\begin{bmatrix} aC_{11} \\ aC_{21} \\ aC_{31} \end{bmatrix} \begin{bmatrix} bC_{12} \\ bC_{22} \\ bC_{32} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & b \\ 0 & 0 \end{bmatrix} . \tag{4.97}$$

There is now enough vector data on both sides of the equals sign to recover the entire rotation matrix. Though the b vector is perpendicular to the a vector, this is not necessary, as any non-parallel vector would have invoked one of the other columns. This non-parallel state is known as linear independence. With two linearly independent vectors resolved in two different coordinate frames, there exists enough information to estimate the rotation matrix that relates two sets of vectors. A quick litmus test for the required linear independence is the rank of the attitude profile matrix B, computed by Equation (4.52). Due to its structure, it will reach rank 3 when there are two sets of two linearly independent vectors. Therefore, an attitude profile matrix of rank 3 indicates that the rotation matrix is observable from the vector data.

In accordance with the steps outlined above, the body experiencing rotation must have at least two epochs of linearly independent angular velocity that occurs across at least two coordinate axes to be able to observe the rotation matrix using angular velocity measurements. A real-world example of this would be an airplane performing successive pitching and rolling motions. While such a maneuver would guarantee observability, successive rotations are not realistic for many vehicles, including cars or ships, since both of these vehicle types rarely experience significant angular velocity in several axes simultaneously. To ensure that these vehicles experience observable rotation, there must be different angular accelerations in each axis of rotation. This would guarantee that each successive vector of angular velocity measurements is linearly independent of the previous one. Realistically, any maneuver that has rotation in two axes, such as horizontal turns with slight rolling, [54, 86, 87], will likely meet this requirement.

Once this requirement is met, any of the Wahba's problem solutions described previously can be used to solve for the misalignment matrix. It is worth noting that these solutions are not the only method available to solve for the misalignment. As described earlier, there are other approaches covered in [68–70] that are centered on the idea in Equation (4.44). However, they face one key disadvantage that solutions to Wahba's problem do not face. Specifically, they compute each of the nine terms in C_b^R individually, without considering the rotation matrix's orthonormality. Therefore, the approaches outlined in [68–70] require angular motion in all three orthogonal axes to observe all nine terms of the misalignment matrix, which is one more axis of motion than what a Wahba's problem solution needs.

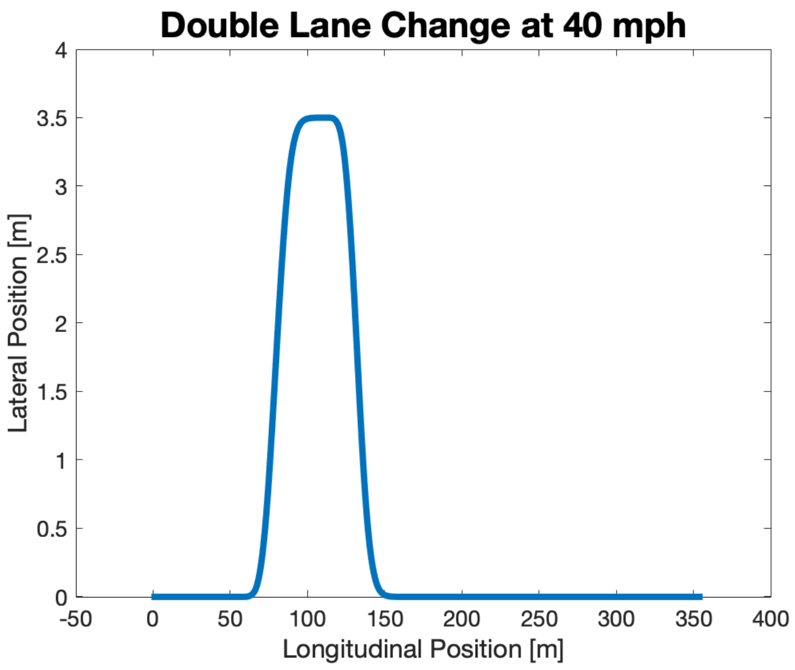

Figure 4.2: The trajectory taken by the simulated vehicle during the the double lane change maneuver.

II. Simulation Study

To verify this hypothesis that only two epochs of linearly independent angular velocity are necessary to estimate misalignment, a Wahba's Problem solution was implemented in a simulation study, the data for which was generated in Carsim. Carsim is a commercial vehicle simulation software package whose function is to realistically simulate the movement of ground vehicles. Carsim allows the user to select or build a vehicle profile and command the vehicle to perform a set of maneuvers. During a simulation, Carsim computes and returns several different types of dynamic data that describe the vehicle's motion. Some of these values include position, velocity, acceleration, attitude, angular velocity, and angular acceleration. This simulated data provides a cost-effective and safe-to-use tool for the development and testing of estimation and control algorithms.

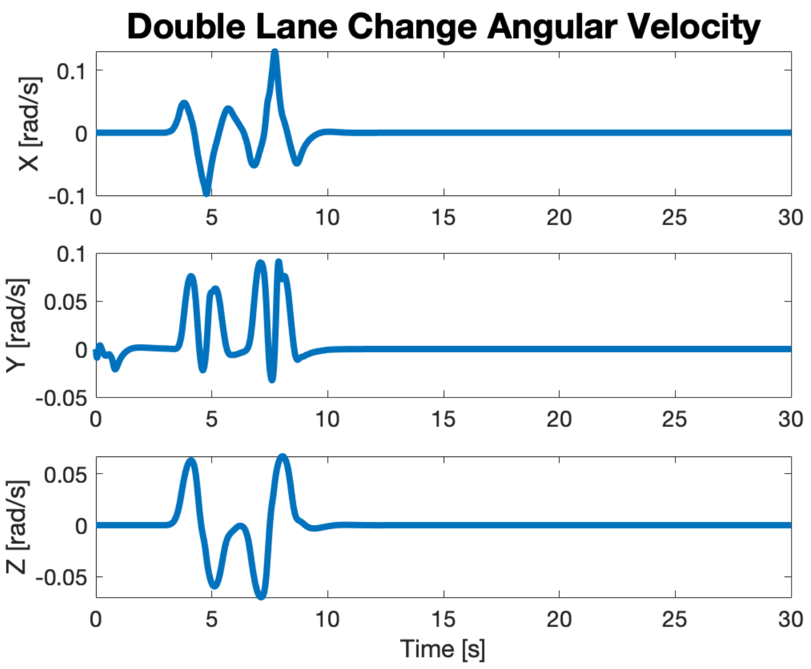

Figure 4.3: The angular velocity profile of the simulated vehicle during the the double lane change maneuver. The vehicles XYZ axes correspond to its forward, right, and downward directions respectively.

For this simulation study, a compact car was commanded to start traveling forward at 40 miles per hour and perform a 3.5 meter-wide double lane change (DLC) as shown in Figure 4.2. A DLC was chosen since it fits the profile described earlier, namely that the maneuver involve independent angular acceleration in least two axes. This can be seen in the x and z-axes of the DLC's angular velocity profile, which is given by Figure 4.3. Note that the data in Figure 4.3 is simulated such that there is no noise. This is due to the fact that Carsim does not model sensor errors by default. As a result, all of the dynamic data returned by Carsim represents the vehicle's true trajectory. The takeaway is that the Carsim-generated data can offer clear, unvarnished insight into the observability of the DLC.

To test observability, a known misalignment was applied to the Carsim-generated angular velocity data from Carsim. Once this was done, a cumulative attitude profile matrix B was computed at each epoch starting at the beginning of the data set in accordance with Equation (4.47). A misalignment was then estimated at each epoch using the SVD method outlined by Equations (4.52) and (4.55) and compared against the true misalignment. The results are shown in Figure 4.4, which details the estimation error in terms of Euler angles and the rank of B over

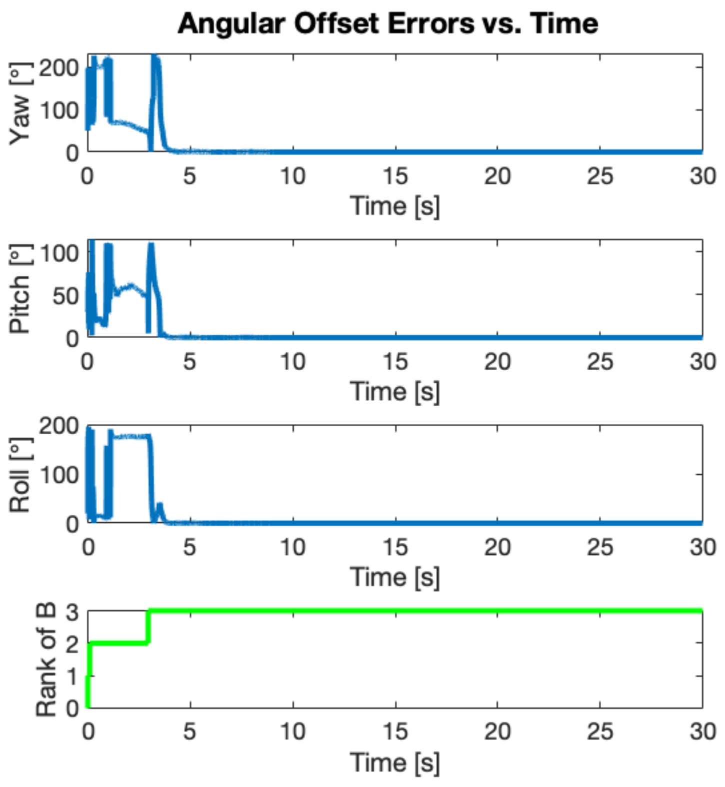

Figure 4.4: The angular offset errors as estimated by the SVD method.

the whole maneuver. At the beginning of the DLC, the simulated car only moves forward. However, as the car starts moving laterally at about the 3-second mark, B's rank increases to three and the angle estimate errors converge to zero. These occurrences indicate that there is sufficient data starting at that point in time to compute the misalignment. In other words, as the lateral movement starts, the misalignment becomes observable through a DLC.

4.5.2.2 Misalignment Computation Plus Bias

I. Observability Analysis

To make the observability analysis of misalignment computation more realistic, the aligning IMU's gyroscope biases should be estimated simultaneously with the misalignment. As discussed in the previous section, this can be done with with a non-linear least-squares estimator that estimates the three gyroscope bias components and the misalignment as a set of ZYX Euler angles. The key to analyzing this estimator's observability properties is the J matrix, shown in Equations (4.79) and (4.80). J is analogous to O in TA-INS-ESKF. Like O, J forms a map between two vector spaces, the states and the residual in Equation (4.79). Another similarity it shares with O is the inclusion of the reference IMU's motion. Specifically J includes the reference IMU's angular velocity. The final similarity between the two is the mathematical operation that must be performed as part of the state estimation. Both matrices need to be psuedoinverted in order to solve for their respective state vectors. This requires that both O and J be non-singular. This in turn requires them both to be full rank, which is rank 6 in J's case. All of these similarities suggest that J may be as strong of analysis tool as O is.

However, there is one critical difference between the two matrices. O is a linear mapping, while J is a linearization of a non-linear mapping. Were J linear, it being rank 6 would be necessary and sufficient condition for Euler angle and gyroscope bias observability. However, since it is non-linear, a rank of 6 cannot guarantee full state observability. Instead, it only guarantees full state observability near the current iteration of the state estimates. Even though a rank check divulges less information about J than it does for O, the information it offers about J is still meaningful, since full state observability near the current iteration is a necessary condition of being able to observe the actual state values. In other words, J being full rank is no guarantee of full state observability, but it at least allows the possibility. By contrast, J being rank deficient guarantees that the full state will not be observable. Therefore, this thesis will use the rank of J as an analysis tool.

Unlike non-biased misalignment estimation, the observability requirements of misalignment plus gyroscope bias estimation are not as clear-cut. Therefore, to analyze the observability of the misalignment plus gyroscope bias estimator, sample angular velocity scenarios were created. Each sample scenario was used to build its own J, after which J's rank was checked. These angular velocity scenarios and rank results are shown in Table 4.1. The first scenario is a repeat of the observability requirement for recovering rotation matrices, which is two sets of two linearly independent vectors. The result was a J matrix that was rank-deficient. In other words, estimating the gyroscope bias simultaneously with the misalignment requires more than just two linearly independent rotations. In the second and third scenarios, a third linearly independent rotation was added. In both of these scenarios, J was full rank. This suggests that three linearly independent rotations are sufficient to estimate both misalignment and gyroscope bias. Of particular note is the third scenario, which has zero angular velocity in the z-axis. The fact that the third scenario results in a full rank J shows that a set of three linearly independent angular velocity vectors does not need to have rotation in all three axes to achieve linear independence.

Table 4.1: The resulting ranks from the three rotation scenarios.

Scenario	**Rank**
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}1\\0\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\1\\0\end{bmatrix}$	5
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}1\\0\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\1\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_3} = \begin{bmatrix}0\\0\\1\end{bmatrix}$	6
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}1\\0\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\1\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_3} = \begin{bmatrix}0.1\\1\\0\end{bmatrix}$	6

Any maneuver that has different angular acceleration in two axes will likely produce at least three linearly independent angular velocity vectors. This qualitative profile is the same as what is required to observe a rotation matrix. So presumably, manuevers that make a misalignment observable, such as a DLC, will also make the gyroscope bias observable. However, there is a key difference between the two estimation scenarios. Estimating a gyroscope bias in addition to the misalignment requires more epochs of data than just estimating a misalignment does. This means that any implementation of the misalignment plus bias estimator will inherently take longer to estimate the misalignment than the misalignment-only estimator will. Additional factors may also lengthen the estimation time, such as high sensor noise or low dynamics, which may make it more productive to implement the misalignment-only estimator.

II. Simulation Study

To verify the hypothesis that maneuvers that make a misalignment observable, such as a DLC, will also make the gyroscope bias observable, the non-linear misalignment plus bias estimator was implemented using the same simulated DLC data used to test misalignment estimation. To test observability, a known misalignment was first applied to the Carsim-generated

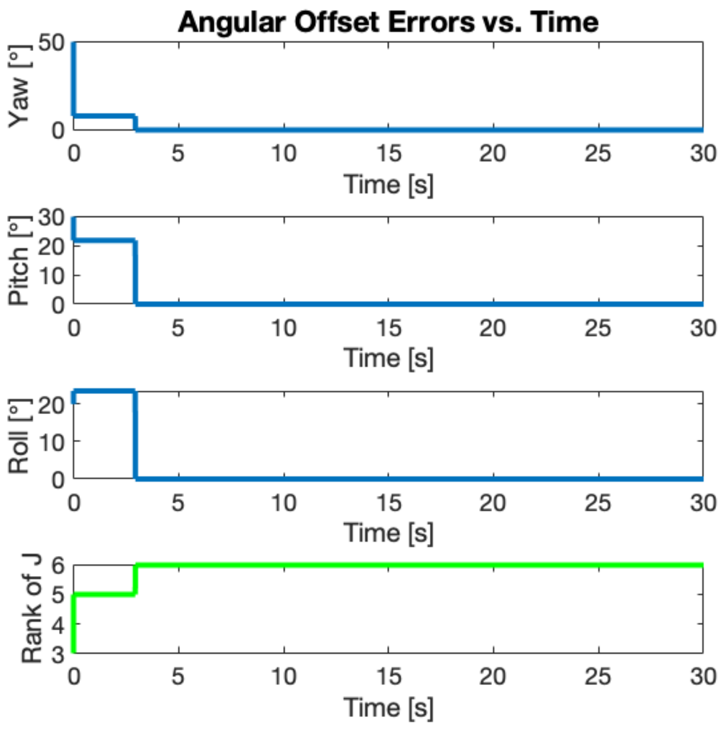

Figure 4.5: The angular offset errors as estimated by the Gauss-Newton algorithm.

angular velocity data. Then, a known bias was added to the misaligned angular velocity data. Once this was done, a residual vector (Equation (4.78)) and Jacobian matrix J (Equation (4.80)) was built at each epoch (i.e. the residual and jacobian at $k = 1$ had three rows, at $k = 2$ six rows, etc.). The residuals and jacobians were used to estimate the errors in the misalignment and gyroscope bias at each epoch, the results of which are shown in Figure 4.5. At the beginning of the DLC, the simulated car only moves forward. However, as the car starts moving laterally at about the 3-second mark, B's rank increases to three and the angle estimate errors along with the gyroscope bias errors converge to zero. These occurrences indicate that there is sufficient data in the DLC to compute the misalignment. These occurrences indicate that there is sufficient data starting at that point in time to compute the misalignment and bias. In other words, as the lateral movement starts, both the misalignment and gyroscope bias become observable through a DLC.

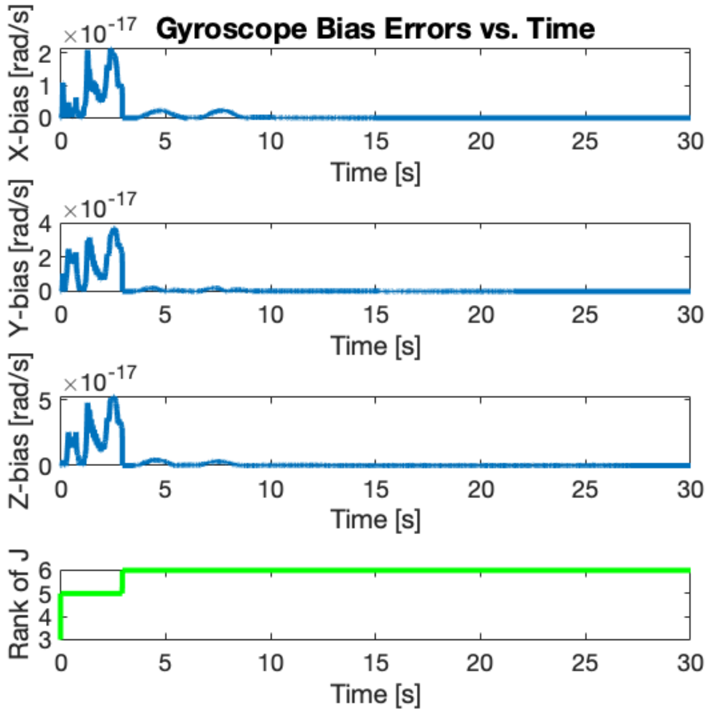

Figure 4.6: The bias errors as estimated by the Gauss-Newton algorithm.

4.5.3 Observing the Lever Arm

Yet another state that can be estimated by a TA algorithm is the lever arm between the reference system and aligning INS. As discussed previously, lever arm estimation has received attention in the context of loosely-coupled GNSS/INS. By analyzing the structure of the total observability matrix O, Hong et al. showed in [57–59, 65] that rotational motion in at least two orthogonal axes is necessary to be able to estimate a lever arm between an INS and a GNSS supplying an INS-KF with position updates, provided the INS and GNSS-computed positions are in the same coordinate frame.

Another approach to analyzing the observability of the lever arm would be to examine the least squares algorithm discussed earlier. Though this algorithm, presented in [81], is different from the ESKF presented in [57–59, 65], it is fundamentally similar. This similarity arises the fact that acceleration values used by the least squares algorithm in [81] are the second time derivative of position values used by the ESKF in [57–59, 65]. This means that both approaches use essentially the same information to estimate the lever arm. Therefore, it is expected that the conditions needed to make the lever arm observable for the least squares algorithm in [81] will be the same as what is needed to make the lever arm observable for the ESKF in [57–59, 65]. The observability can be analyzed by considering the structure of the mapping matrix M. Recall that M relates the lever arm to the difference between two IMUs' acceleration outputs. This makes M's role in lever arm computation analogous to O's role in TA-INS-ESKF.

In light of the similarity between the two approaches, performing an analysis on M to determine the lever arm's observability requirements may seem superfluous, since [58], [59], and [65] have already determined lever arm observability requirements by analyzing O. However, determining lever arm observability by analyzing M is still a meaningful exercise, since M is a smaller and fundamentally simpler matrix than O. Therefore, analyzing the structure of M offers an alternative perspective on lever arm observability that is easier to understand than analyses described in [58, 59, 65].

4.5.3.1 Lever Arm Computation With and Without Bias

I. Lever Arm Observability Analysis

Since M and O perform analogous roles, the condition for full state observability is the same one that was used for O, namely that M must be non-singular. Non-singularity results from a matrix being full rank, which in this case is rank 3. Since the only variables in M are angular velocity and angular acceleration, the only way to affect its rank is to rotate the body the that two IMUs are attached to. Due to M's complexity, it can be tough to get an impression of its observability properties by simply inspecting it. Therefore, to investigate these properties, M must be tested with sample angular velocities and accelerations. To this end, several two sample epoch scenarios of general angular velocity and angular acceleration were created.
Six such scenarios were created and inserted into a two-epoch version of M, after which each scenario's observability was checked by taking M's rank. The complete list of scenarios and rank results is given in Table 4.2.

Table 4.2: The resulting ranks from six different two-epoch rotation scenarios.

Scenario	**Rank**
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}\omega^n_{n\mathrm{R}_1}(x)\\0\\0\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix}\dot{\omega}^n_{n\mathrm{R}_1}(x)\\0\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}\omega^n_{n\mathrm{R}_2}(x)\\0\\0\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix}\dot{\omega}^n_{n\mathrm{R}_2}(x)\\0\\0\end{bmatrix}$	2
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\\omega^n_{n\mathrm{R}_1}(\mathrm{y})\\0\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\\dot{\omega}^n_{n\mathrm{R}_1}(\mathrm{y})\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\\omega^n_{n\mathrm{R}_1}(\mathrm{y})\\0\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\\dot{\omega}^n_{n\mathrm{R}_2}(\mathrm{y})\\0\end{bmatrix}$	2
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\0\\\omega^n_{n\mathrm{R}_1}(\mathrm{z})\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\0\\\dot{\omega}^n_{n\mathrm{R}_1}(\mathrm{z})\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\0\\\omega^n_{n\mathrm{R}_2}(\mathrm{z})\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\0\\\dot{\omega}^n_{n\mathrm{R}_2}(\mathrm{z})\end{bmatrix}$	2
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\\omega^n_{n\mathrm{R}_1}(y)\\0\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\0\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}\omega^n_{n\mathrm{R}_2}(x)\\0\\0\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\0\\0\end{bmatrix}$	3
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\\omega^n_{n\mathrm{R}_1}(y)\\0\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\0\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\0\\\omega^n_{n\mathrm{R}_2}(\mathrm{z})\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\0\\0\end{bmatrix}$	3
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\0\\\omega^n_{n\mathrm{R}_1}(\mathrm{z})\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix}0\\0\\0\end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix}\omega^n_{n\mathrm{R}_2}(x)\\0\\0\end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix}0\\0\\0\end{bmatrix}$	3

The rank checks done on M show that rotation around a single axis is insufficient for full lever arm observability. This result is somewhat intuitive. A rotation about one axis "exposes" only two components of the lever arm. To expose all three, there must be successive rotation around at least two axes, which is consistent with the rank results. What is missing from the scenarios shown in Table 4.2 is consideration for scenarios with simultaneous rotation in multiple axes. Consideration of such motion is important, since, as discussed earlier, simultaneous multi-axis rotation is far more likely to occur in practical transfer alignment scenarios than successive single axis rotations. To investigate the effect of simultaneous multi-axis rotation on lever arm observability, more scenarios were implemented and rank-checked. The complete list of scenarios and rank results is given in Table 4.3. These new results show that constant rotational motion in at least two axes simultaneously does not give full lever arm observability. To attain full observability, the rotation rate about at least one of those axes must be non-constant rate, i.e there needs to be angular acceleration. This result validates the findings in [58, 59, 65], which showed that the same rotation profile was necessary to observe a lever arm state in a TA-INS-ESKF. This result is promising, since there are are many easy-to-perform maneuvers that fit this profile, including banked turns and lane-changing. This suggests that the lever arm between two IMUs attached to a rigid body can be estimated using realistic vehicle movements, such as the DLC used for the misalignment and gyroscope bias computation.

Table 4.3: The resulting ranks from the six simultaneous rotation scenarios.

Scenario	**Rank**
$\omega_{n\mathrm{R}_1}^n = \begin{bmatrix} \omega_{n\mathrm{R}_1}^n(x) \\ \omega_{n\mathrm{R}_1}^n(y) \\ 0 \end{bmatrix}, \dot{\omega}_{n\mathrm{R}_1}^n = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	2
$\omega_{n\mathrm{R}_1}^n = \begin{bmatrix} \omega_{n\mathrm{R}_1}^n(x) \\ 0 \\ \omega_{n\mathrm{R}_1}^n(z) \end{bmatrix}, \dot{\omega}_{n\mathrm{R}_1}^n = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	2
$\omega_{n\mathrm{R}_1}^n = \begin{bmatrix} 0 \\ \omega_{n\mathrm{R}_1}^n(y) \\ \omega_{n\mathrm{R}_1}^n(z) \end{bmatrix}, \dot{\omega}_{n\mathrm{R}_1}^n = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	2
$\omega_{n\mathrm{R}_1}^n = \begin{bmatrix} \omega_{n\mathrm{R}_1}^n(x) \\ \omega_{n\mathrm{R}_1}^n(y) \\ 0 \end{bmatrix}, \dot{\omega}_{n\mathrm{R}_1}^n = \begin{bmatrix} \dot{\omega}_{n\mathrm{R}_1}^n(x) \\ 0 \\ 0 \end{bmatrix}$	3
$\omega_{n\mathrm{R}_1}^n = \begin{bmatrix} \omega_{n\mathrm{R}_1}^n(x) \\ 0 \\ \omega_{n\mathrm{R}_1}^n(z) \end{bmatrix}, \dot{\omega}_{n\mathrm{R}_1}^n = \begin{bmatrix} 0 \\ 0 \\ \dot{\omega}_{n\mathrm{R}_1}^n(z) \end{bmatrix}$	3
$\omega_{n\mathrm{R}_1}^n = \begin{bmatrix} 0 \\ \omega_{n\mathrm{R}_1}^n(y) \\ \omega_{n\mathrm{R}_1}^n(z) \end{bmatrix}, \dot{\omega}_{n\mathrm{R}_1}^n = \begin{bmatrix} 0 \\ \dot{\omega}_{n\mathrm{R}_1}^n(y) \\ 0 \end{bmatrix}$	3

II. Lever Arm Plus Bias Observability Analysis

To further expand the observability analysis, the aligning IMU's accelerometer biases were added to the state vector. With the addition of states, the mapping matrix for this scenario changes from M to M_{bias}. Making this addition is vital, since accounting for the accelerometer biases will result in a more realistic lever arm estimate. Since there are now 6 states to estimate, the full observability condition is for M_{bias} to be rank 6. To test the observability of this lever arm plus bias estimator, additional two-epoch rotation scenarios were created and rank-checked. The complete list of scenarios and rank results is given in Table 4.4.

As with the previously discussed lever arm-only estimator, the results show that angular velocity around a single axis is insufficient for full state estimation. Interestingly, the minimum requirement for the lever arm plus bias estimator to have full state observability is the same as the lever arm-only estimator's. Namely, there must be angular velocity in at least two axes with variable angular velocity in at least one axis. There is however, one key difference. In order for M_{bias} to be rank 6, the motion that causes observability must occur over two or

Table 4.4: The resulting ranks from the six simultaneous rotation scenarios.

Scenario	**Rank**
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix} \omega^n_{n\mathrm{R}_1}(\mathrm{x}) \\ 0 \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_1}(x) \\ 0 \\ 0 \end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix} \omega^n_{n\mathrm{R}_1}(\mathrm{x}) \\ 0 \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_2}(x) \\ 0 \\ 0 \end{bmatrix}$	5
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix} \omega^n_{n\mathrm{R}_1}(\mathrm{x}) \\ 0 \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_1}(x) \\ 0 \\ 0 \end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix} \omega^n_{n\mathrm{R}_2}(\mathrm{x}) \\ 0 \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_1}(x) \\ 0 \\ 0 \end{bmatrix}$	5
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix} \omega^n_{n\mathrm{R}_1}(x) \\ \omega^n_{n\mathrm{R}_1}(\mathrm{y}) \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_1}(x) \\ 0 \\ 0 \end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix} \omega^n_{n\mathrm{R}_2}(x) \\ \omega^n_{n\mathrm{R}_2}(\mathrm{y}) \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_2}(x) \\ 0 \\ 0 \end{bmatrix}$	6
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix} \omega^n_{n\mathrm{R}_1}(x) \\ \omega^n_{n\mathrm{R}_1}(\mathrm{y}) \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_1}(x) \\ 0 \\ 0 \end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix} \omega^n_{n\mathrm{R}_2}(x) \\ \omega^n_{n\mathrm{R}_1}(\mathrm{y}) \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_2}(x) \\ 0 \\ 0 \end{bmatrix}$	6
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix} \omega^n_{n\mathrm{R}_1}(x) \\ \omega^n_{n\mathrm{R}_1}(\mathrm{y}) \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \omega^n_{n\mathrm{R}_2} = \begin{bmatrix} \omega^n_{n\mathrm{R}_2}(x) \\ \omega^n_{n\mathrm{R}_1}(\mathrm{y}) \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_2} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	6
$\omega^n_{n\mathrm{R}_1} = \begin{bmatrix} \omega^n_{n\mathrm{R}_1}(x) \\ \omega^n_{n\mathrm{R}_1}(\mathrm{y}) \\ 0 \end{bmatrix}, \dot{\omega}^n_{n\mathrm{R}_1} = \begin{bmatrix} \dot{\omega}^n_{n\mathrm{R}_1}(x) \\ 0 \\ 0 \end{bmatrix}$	3

more epochs. In other words, all of the states are not observable instantaneously as they were for the lever arm-only estimator. Rather they must be observed over time. This fact implies that any implementation of the lever arm plus bias estimator will inherently take longer to estimate the lever arm than the lever arm-only estimator. Additional factors may also lengthen the estimation time, such as high sensor noise or low dynamics, which may make it more productive to implement the lever arm-only estimator.

III. Simulation Study

To test the lever arm estimator, simulations were performed to verify the results from the observability analysis. As was done for the misalignment, clean data from a DLC simulated in Carsim was used for testing. First, a known lever arm (but not a misalignment) was applied to the Carsim-generated acceleration data using Equation (4.4). Once this was done, an acceleration residual vector (Equation (4.84)) and mapping matrix M (Equation (4.85)) was built at each epoch. (The residual and mapping matrix at $k = 1$ had three rows, at $k = 2$ six rows, etc.) The stacked residuals and mapping matrix were both used to estimate the lever arm and accelerometer bias at each epoch, the results of which are shown in Figure 4.7. At the beginning of the DLC, the simulated car only moves forward. However, as the car starts moving laterally at about the 3-second mark, M's rank increases to three and the lever arm estimate errors converge to zero. These occurrences indicate that there is sufficient data starting at that point in time to compute the lever arm. In other words, as the lateral movement start, the lever arm becomes observable from the DLC.

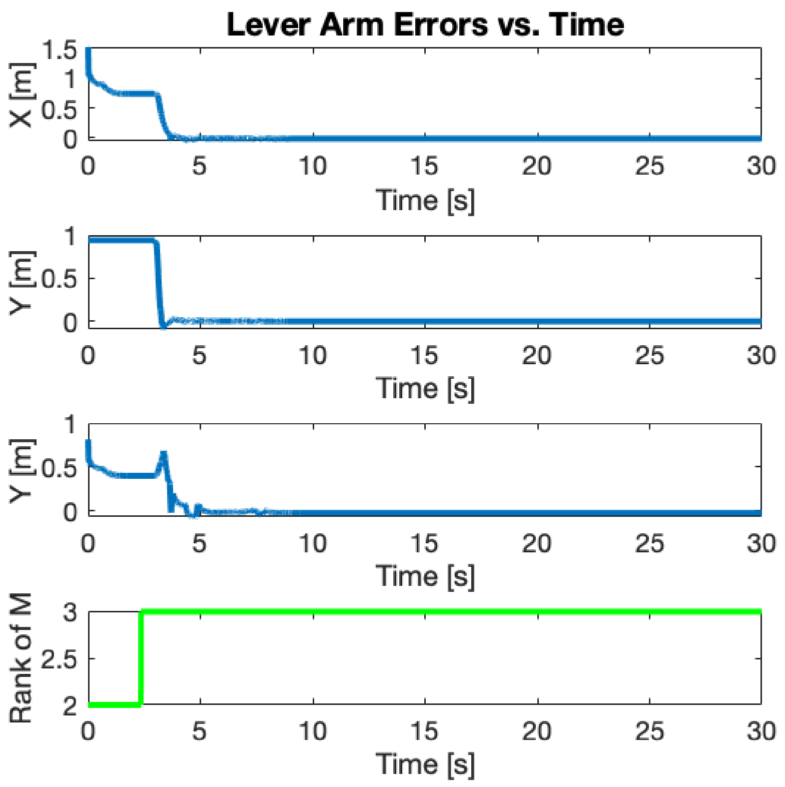

Figure 4.7: The lever arm error as estimated by least-squares.

This DLC simulation scenario was extended to the lever arm plus bias estimator by applying a known constant bias to the offset acceleration data and by using M^{bias} as the mapping matrix instead of M. The bias estimation errors of the lever arm plus bias estimator are shown in Figure 4.8. Note that the lever arm estimation errors are not shown for the sake of brevity. As with the lever arm estimator, M's rank increases to full (in this case rank 6) and the estimate errors converge to zero as the car starts moving laterally. This shows that aligning IMU's accelerometer bias is indeed observable from the DLC.

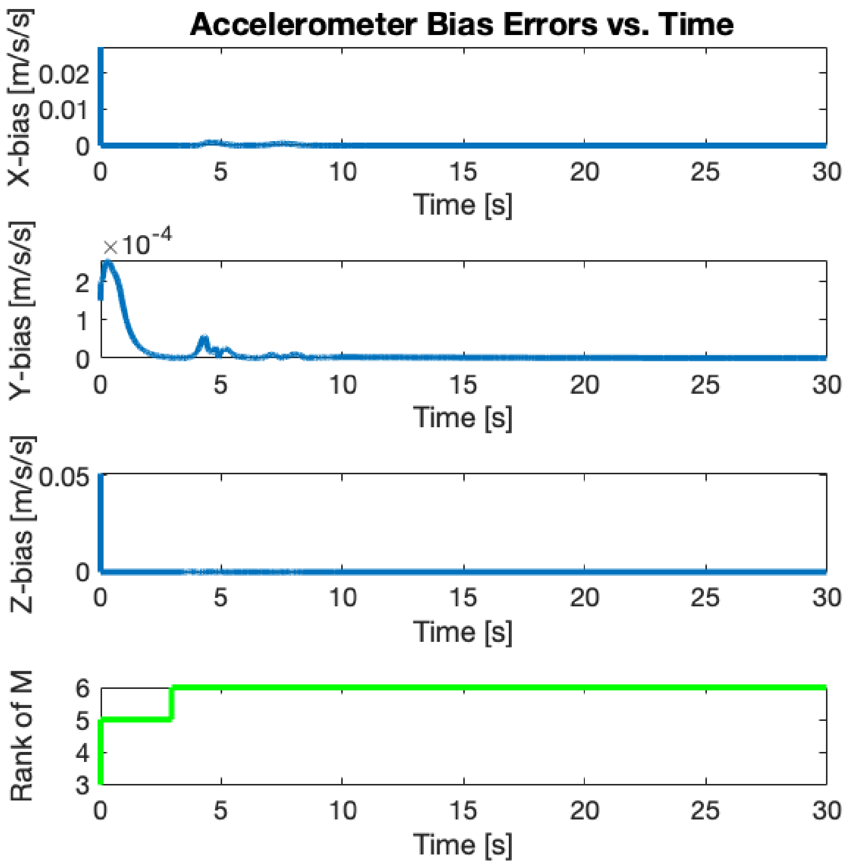

Figure 4.8: The accelerometer bias error as estimated by least-squares.

4.6 Observability Car Test

As a check on the validity of the observability analyses in the previous sections, the misalignment and lever arm were estimated using data gathered from a real-world driving test. The test involved a drive around Auburn University's campus in a AutonomouStuff Lincoln MKZ. The MKZ was instrumented with a KVH 1750 [93] in the trunk and a VectorNav VN-300 [92] on the roof above the front row of seats, as shown in Figure 4.9. The distance between the two IMUs was fixed at $r_{Rb}^{R} = [1.5240, 0.9398, 0.8128]\ m$ where R and b were the KVH 1750 and VN-300 respectively, while the misalignment was set to be $\psi = 90°$, $\theta = 90°$, and $180°$. Both of these quantities are resolved in the frame of the KVH 1750.

The MKZ was driven along the path shown in Figure 4.10, starting from inside a parking lot (the red X) and ending on a major roadway (the purple X). Using the angular velocity and

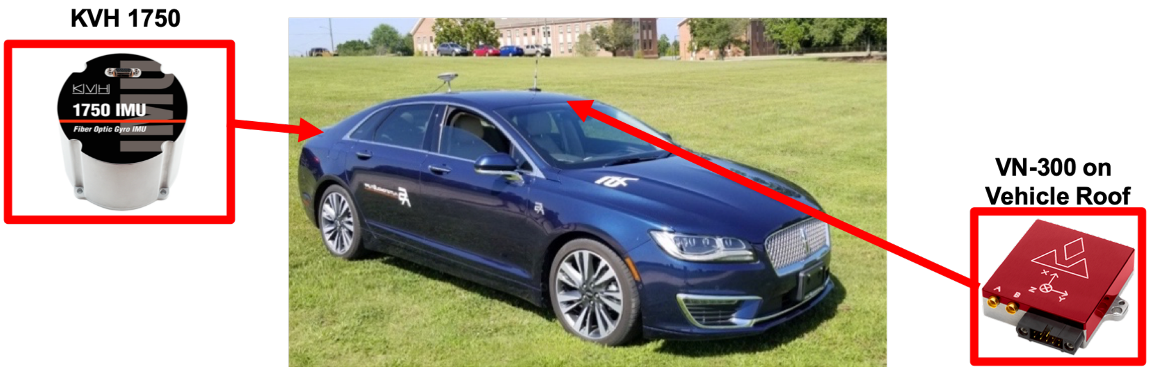

Figure 4.9: The setup used to collect data for the real-world transfer alignment test.

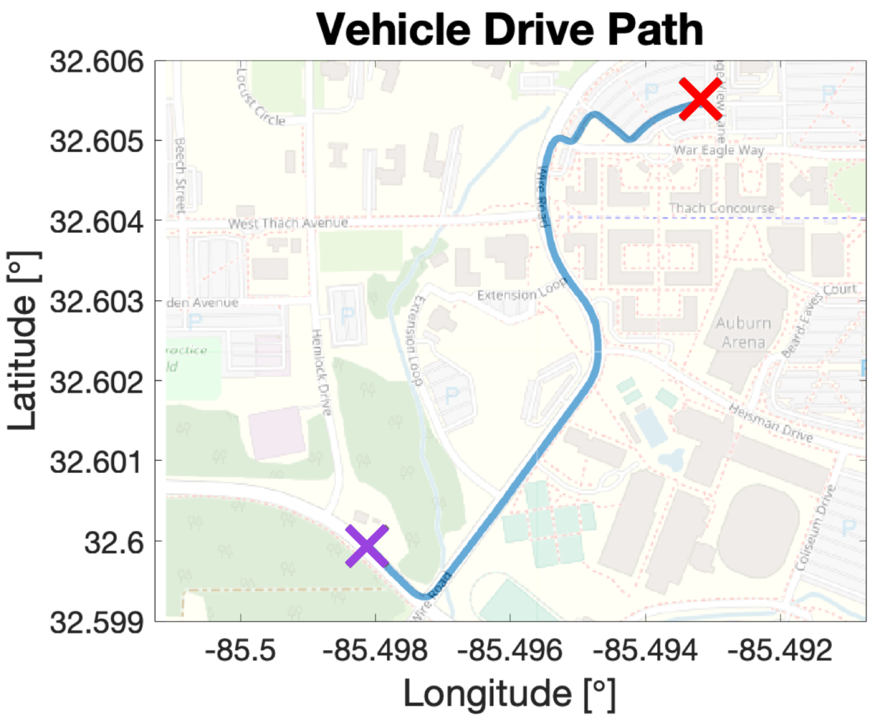

Figure 4.10: The path driven by the instrumented test vehicle.

linear acceleration output by both of the IMUs, the misalignment and lever arm between the two IMUs were estimated.

Figure 4.11 shows the misalignment error. Both methods of misalignment estimation were able to compute the angular offset to within a degree of their true values in less than 100 seconds, just after the MKZ exited the parking lot. Similar results can be seen in Figure 4.12 for the lever arm estimation. With the true misalignment known, the same maneuvers were able to estimate the lever arm to less than 0.4 meter-level accuracy over the same time period in the x and y-axes.

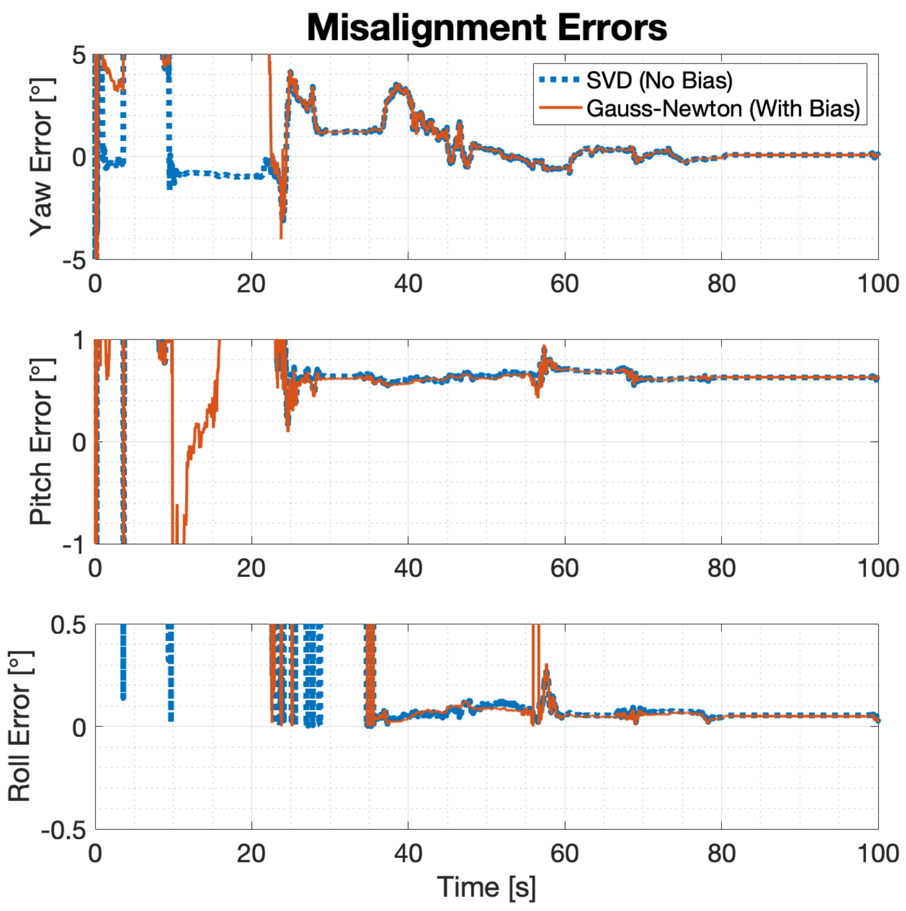

Figure 4.11: The errors in the estimation of misalignment.

However, both methods of lever arm estimation were unable to estimate the z-component of the lever arm with any reasonable level of accuracy. The reason that x and y-components are more observable than the z-component has to do with the nature of the vehicle's angular motion. Most of this angular motion is in the z axis (yawing), which means it lies in the plane formed by the x and y-axes. This makes these the lever arm components in these two axes highly observable. In comparison to the yawing motion however, the vehicle experiences very little pitching and rolling. Since these two modes are the only ones that are orthogonal to the z-axis, the vertical component of the lever arm has relatively worse observability in comparison to the other two axes. The vertical component of lever arm could be more accurately estimated if the signal-to noise ratio was higher. This would be achieved if a) the pitching and rolling motion was more intense, b) there was less sensor noise, or c) there was greater distance in the z-direction.

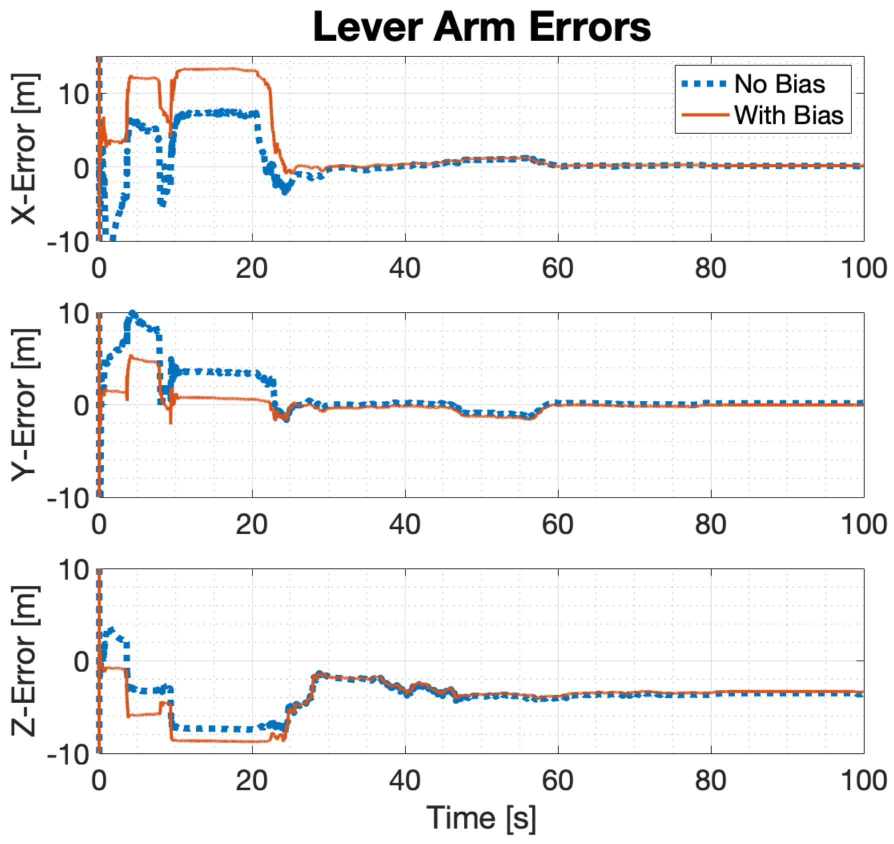

Figure 4.12: The errors in the estimation of lever arm.

Despite the poor observability in the z-axis, these results are encouraging since they show that both the misalignment and part of the lever arm are observable on ground vehicles through the performance of ordinary maneuvers.

Chapter 5

Rigidity Detection

The transferal of a PVA solution from an IMU mounted on a vehicle to an IMU attached to a pedestrian riding inside the vehicle can be accomplished using any of the algorithms described in the previous chapter. However, the ability to perform this transfer of alignment information requires knowledge of the kinematic relationships between the two IMUs. In other words, the relative position, velocity, and acceleration between the vehicle and pedestrian IMUs must all be known.

Determining these offsets is not a trivial problem in the context of vehicle-to-pedestrian transfer alignment, since an IMU mounted on a pedestrian's body is free to move independently of the vehicle. In such a scenario, the relative position, velocity, acceleration, and misalignment between the two IMUs are extremely difficult to determine. Fortunately, the complexity of this problem can be reduced by performing transfer alignment only when the pedestrian is holding their IMU still with respect to the vehicle. This state of relative rigidity is known as pedestrian-rigid (PR). When PR is true, the relative velocity and acceleration terms go to zero, which leaves the relative position and misalignment between the two IMUs as the only quantities to be determined. This reduction of terms greatly simplifies the transferal of a PVA solution from the vehicle to the pedestrian. However, to practically make this simplification and perform vehicle-to-pedestrian transfer alignment, there must be a means to detect a state of relative rigidity. This chapter uses existing zero-velocity detection methods to develop a method for the detection of relative rigidity between the vehicle and pedestrian IMUs.

5.1 Rigidity Detection

5.1.1 Thresholding

A rigid period is a period in which the relative velocity between the vehicle and pedestrian IMUs, $\vec{v}_{Rb}^{R}$, equals zero. This means that the problem of rigidity detection is a problem of zero-velocity determination. Zero-velocity determination has a long history in navigation, since periods of zero-velocity provide an easy, low-cost means of bounding the error growth and maintaining the calibration of an inertial navigation system [2]. One method of performing zero-velocity detection is checking to see if the values of an inertial navigator's velocity states are at or near zero. Stationarity can be further checked by considering the yaw rate. If its value is at or near zero, then a vehicle can be considered to have zero-velocity [2].

However, this process has potential pitfalls. Both velocity and yaw rate are computed by integrating raw IMU outputs, a process that is highly sensitive to sensor errors and noise. Relying on them for zero-velocity detection could lead to erroneous detections of zero-velocity events. Therefore, there is value in using zero-velocity detection methods that do not rely on integrating the IMU output, but instead use the raw outputs from the inertial sensors, i.e. acceleration or angular velocity measurements. The basic idea of this approach is that if inertial sensor outputs are below a threshold, then the velocity of the inertial sensor (and the object it is attached to) can be assumed to be zero [2].

One example of this approach in action is stance detection in pedestrian navigation. As detailed in Chapter 3, a pedestrian's foot is nearly stationary during a stance event, which is when the foot is planted on the ground. In theory, the foot's linear velocity will be near-zero during a stance. This means the measurements taken by an IMU on the foot will be ideally $9.81\,m/s^2$. Therefore, if a foot-mounted IMU measures acceleration magnitudes that is close to this value, then the foot can be assumed to be stationary on the ground. This method is used to detect stance events in [2, 7, 12, 14, 15, 17, 24, 27, 28, 88]. The logic is given by,

$$C1 = \begin{cases} True & \text{if } {\sim}\text{any } (8.5\,m/s^2 < ||a^{b}_{nb,window}|| < 11\,m/s^2) \\ False & \text{otherwise.} \end{cases} \tag{5.1}$$

In the context of in-vehicle PDR initialization, a similar approach can be applied to the difference in angular velocity magnitudes to detect rigid periods between the vehicle and pedestrian IMUs. The theory behind applying this logic is simple. As established in Chapter 2, all points on a rotating body experience the same rotation, i.e. the distance between any two points does not create a rotational offset between them. Since angular velocity is the time derivative of rotation, this idea holds for angular velocity as well. This means that all points in a rigid body experience the same angular velocity. Since angular velocity is a vector, the values of its components depend on the chosen resolving frame. However, its magnitude (and that of any vector) is unaffected by rotation transformation. This means that for any point on a rigid body,

$$||\omega_1|| = ||\omega_2||. \tag{5.2}$$

Applying the logic in Equation (5.1) to the rigidity detection problem gives,

$$C2 = \begin{cases} True & \text{if } ||\omega||_{diff}\Big|_k < th_{rigid} \\ False & \text{otherwise} \end{cases} \tag{5.3}$$

where $||\omega||_{diff} = ||\omega^p_{np}|| - ||\omega^c_{nc}||$ and th_rigid is the rigidity threshold. A "True" is returned when there is a state of rigidity between the vehicle and pedestrian IMUs, while a "False" is returned during a state of non-rigidity between the two IMUs. This algorithm will be referred to as the "Rigidity Detector."

5.1.2 Thresholding Tradeoff

Applying Equation (5.3) will allow for the detection of occurrences of rigidity on an epoch-by-epoch basis. The accuracy of the detector is related to the choice of th_{rigid}. To illustrate the effect of varying th_{rigid}, the rigidity detector was run on real-world data. The data is a 8000-epoch set of angular velocity measurements taken from two IMUs rigidly mounted to a moving vehicle. Between 2000 and 6000 epochs, random noise was added to represent non-rigidity. Both data sets were differenced to form $||\omega||_{diff}$, which was run through the rigidity detector. The rigidity detector was run on the data set using two test values of th_{rigid}, $0.2\,rad/s$ and $0.03\,rad/s$.

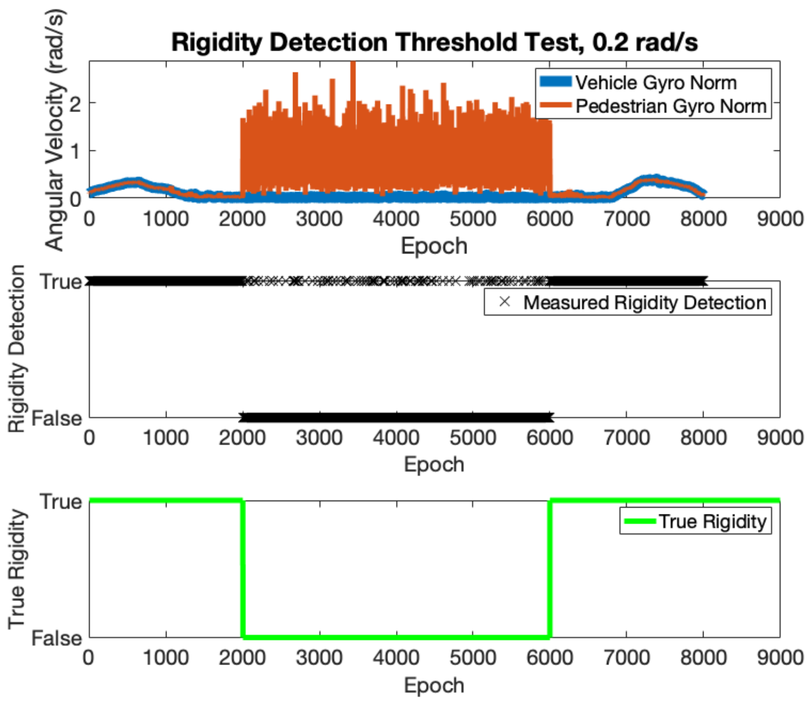

Figure 5.1: Rigidity Detector running with a threshold of 0.2 rad/s.

For the larger value of th_{rigid}, $0.2\,rad/s$, the rigidity detector is able to identify the rigid periods with near-perfect accuracy. However, it returns a significant number of true values during non-rigid periods (between 2000 and 6000 epochs), as shown in Figure 5.1. This has to do with the unpredictability of the pedestrian IMU's motion during the non-rigid periods. Since the motion is completely random, there will almost certainly be individual values of $||\omega||_{diff}$ during the rigid periods that fall below th_{rigid}.

To capture these values, a possible solution is to lower th_{rigid}. However, this is not a perfect solution. As shown in Figure 5.2, lowering th_{rigid} to $0.03\,rad/s$ results in the rigidity detector returning a significant number of false values during the rigid periods. This is due to the random noise present in the two IMUs' measurements. The randomness in the IMU measurements means that there are likely to be individual values of $||\omega||_{diff}$ during the non-rigid periods that fall above th_{rigid}.

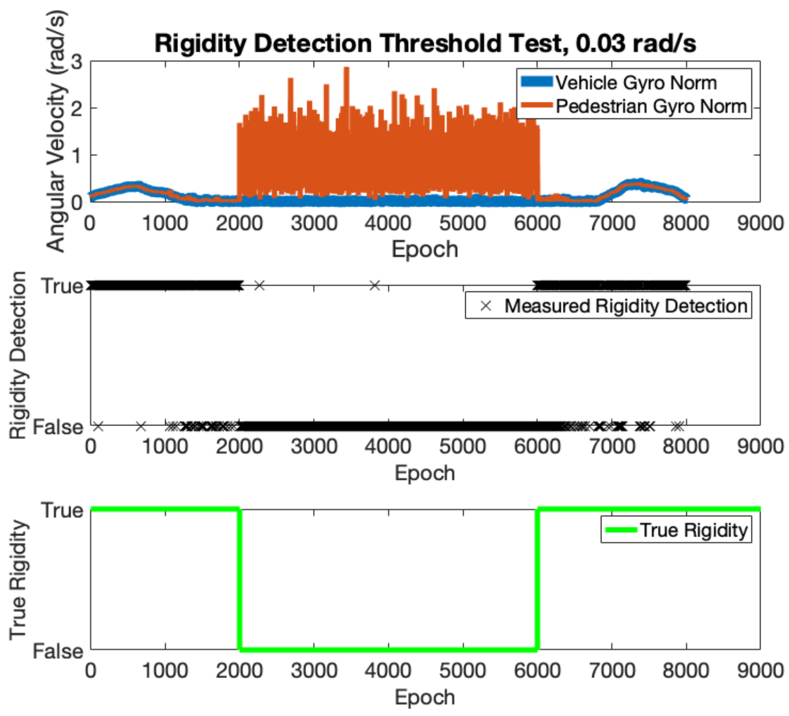

Figure 5.2: Rigidity Detector running with a threshold of 0.03 rad/s.

Either situation is problematic. On one hand, incorrect true flags during a non-rigid period will lead to the IMU data from those epochs to be used to compute r^R_{Rb} and C^R_b. On the other hand, incorrect false flags during a rigid period will lead to the IMU data from that epochs not being used to compute r^R_{Rb} and C^R_b. Either option opens the door to erroneous computations of r^R_{Rb} and C^R_b which in turn impedes successful transfer alignment and ultimately, initialization of the PDR algorithm.

To maximize the chances of a successful transfer alignment, the incorrect rigidity detection values arising from the randomness in $||\omega||_{diff}$ must be minimized. The problem can be addressed in one of two ways. One option is to adjust th_{rigid} such that no values of $||\omega||_{diff}$ fall below or above th_{rigid} during the rigid and non-rigid periods respectively. Turning th_{rigid} to find the optimal tradeoff between these two extremes, assuming it even exists, is potentially a time-consuming process. In addition, the location of the optimal tradeoff is highly dependent on the intensity of the pedestrian IMU's motion during a non-rigid period. The higher the intensity, the lower th_{rigid} can be. However, taking advantage of the intensity of the pedestrian

IMU's motion requires making assumptions about the motion. Since the pedestrian IMU's motion during the drive is being assumed to be completely random, such assumptions could easily break down.

5.1.3 Smoothing Window

Another option is to apply filtering or smoothing to $||\omega||_{diff}$. The desired result of filtering or smoothing would be to reduce the effect of the randomness $||\omega||_{diff}$ on the output of the rigidity detector. An exceedingly common method is to applying thresholding over a window of values rather than a single epoch. The idea is that a zero-velocity event will occur if and only if all the values in the window violate the threshold. Windowed thresholding can be applied to acceleration or angular velocity outputs as described in [2, 7, 14, 17, 24, 33]. A variation on this approach, presented in [7, 14, 17, 24, 33], is to apply a threshold to the variance of all the values in the window rather than the values themselves. In either case though, windowing is used to increase the reliability of the zero-velocity detection. In terms of the stance detector described in Equation (5.1), the logic of windowed thresholding is given by,

$$C3 = \begin{cases} True & \text{if } ||a^b_{nb}|| \Big|_{k-w}^{k} < 10.5\, m/s^2 \\ False & \text{if } ||a^b_{nb}|| \Big|_{k-w}^{k} > 10.5\, m/s^2. \end{cases} \tag{5.4}$$

For this thesis, a moving window w of adjustable size will be used to smooth the rigidity detector's output. The detector will consider the current value of $||\omega||_{diff}$ along with the previous $w-1$ values. If ***all*** of the values of $||\omega||_{diff}$ in the window fall below the threshold, the detector will return a true value at the current epoch. This logic is given by,

$$C4 = \begin{cases} True & \text{if } ||\omega||_{diff} \Big|_{k-w}^{k} < th_{rigid} \\ False & \text{otherwise.} \end{cases} \tag{5.5}$$

This approach places a high standard on a declaration of rigidity. Such a conservative approach was intentionally used since an incorrectly declared true value was judged to be worse than a an incorrectly declared false value. The reasoning was that the former event would result in erroneous computations r_{rB}^{R} and C_{b}^{R}, while the latter would give no computations of r_{Rb}^{R} and C_{b}^{R}. In the author's judgment, erroneous offset values are more counter-productive than not having them.

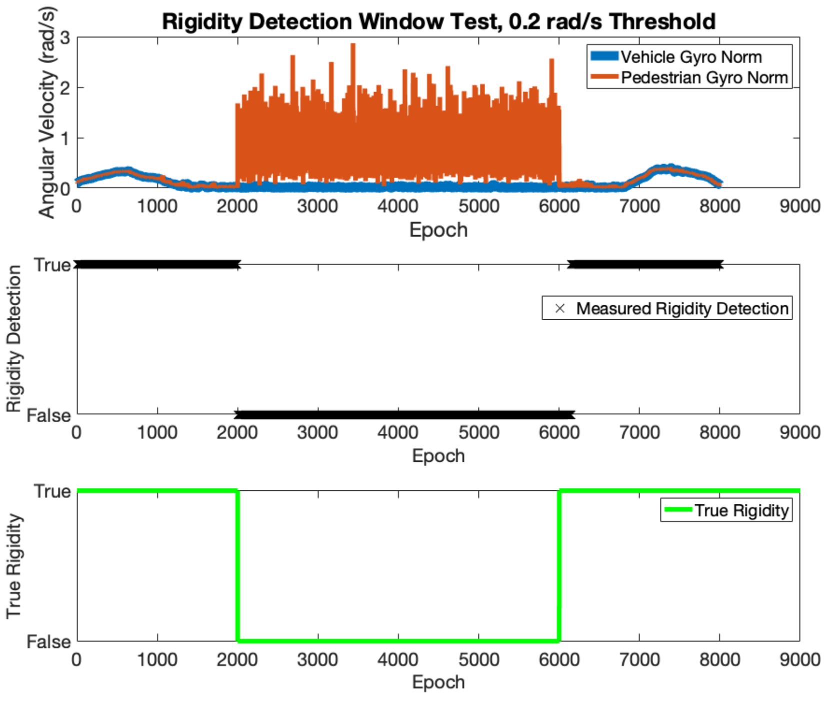

Figure 5.3: Rigidity Detector running with a threshold of 0.03 rad/s and a 150 epoch-long smoothing window.

To test the windowing approach, a 150 epoch-long window at a sample rate of 100 Hz was applied to both of the simulations from previous section. Running the window with the $0.2\,rad/s$ threshold allowed for nearly perfect differentiation between the rigid and non-rigid states, as shown in Figure 5.3.

5.2 Testing and Evaluation

5.2.1 Simulation

With the detector's operating principle proven, further simulation was performed to test the effects of varying both th_{rigid} and w. Representative normalized angular velocity data was created by pairing 100 seconds of rigidity followed by 100 seconds of relative motion. The relative motion periods were represented with Gaussian random noise with a predetermined standard deviation, σ_{test}, while zeros were used to represent the rigid periods.

This pairing was iterated a total 500 times per different value of σ_{test}, which were incremented from $0.01\,rad/s$ to $0.19\,rad/s$ in increments of $0.01\,rad/s$. The difference threshold, th_{rigid}, was permanently set to be in the middle of the range, at $0.1\,rad/s$. A larger value of σ_{test} implies that the pedestrian's IMU is moving more rapidly while inside the vehicle. The incrementation of σ_{test} was repeated at different values of w, ranging from $w = 0.01\,sec \times f$ to $w = 0.14\,sec \times f$ in increments of $0.01\,sec \times f$ where f represents the sampling frequency, which was set to 300 Hz for this simulation.

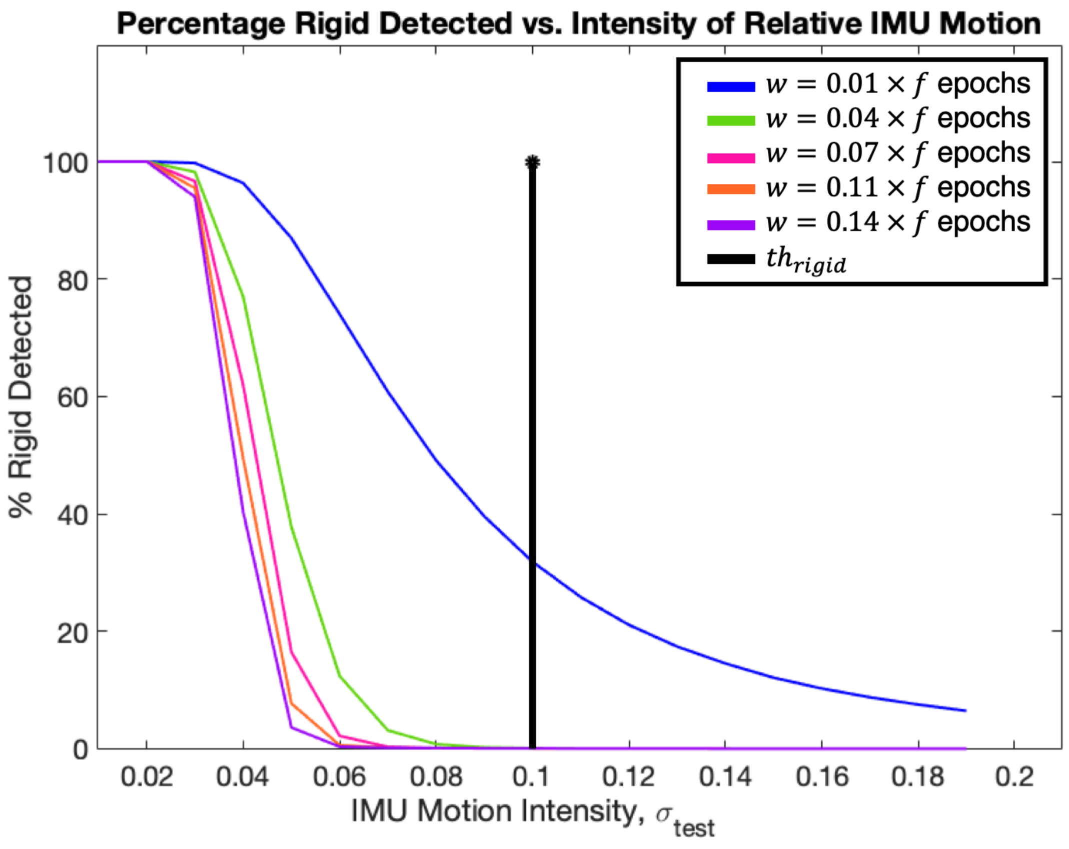

Figure 5.4: The effect of adjusting w and σ_{test} on the detection of rigid periods.

The results in Figure 5.4 show that a certain level of σ_{test}, in this case, $0.02\,rad/s$, does not return any false values. This is because the Gaussian random noise at that level does not contain even a single value larger than th_{rigid}. As the pedestrian IMU starts to move more rapidly, represented by higher and higher values of σ_{test}, the percentage of values that go undetected decreases since less of the angular velocity values fall under th_{rigid}.

Another effect shown by the results validates the reasoning behind using windowed thresholding. This effect is seen in the sharper rigidity/non-rigidity cutoff for larger values of w. Intuitively, in the presence of sensor noise, vehicle vibration, and in-vehicle pedestrian IMU motion, the bigger w is, the more likely it is to contain a value that is larger than th_{rigid} and return a false flag. However, there is a cost associated with increasing w. As the data window size increases, the length of rigid period detected by the rigidity detector decreases, since the data window introduces a detection delay. For this simulation, the detector was able to detect over 90% of each 100-second rigid period, so the cost for this scenario was minimal.

However, the delay effect caused by increasing w should be considered when tuning for real-world use, since the rigid periods may be significantly shorter. If a given rigid period is shorter than w, then it will not be detected at all. The size of th_{rigid} must also be considered prior to real-world application. If the sensors used are noisier than the ones simulated, or if the vehicle's vibration is above the sensors' noise floor, then th_{rigid} should be increased. However, care should be taken not to increase th_{rigid} above the expected dynamics, since doing so would result in an ineffectual detector.

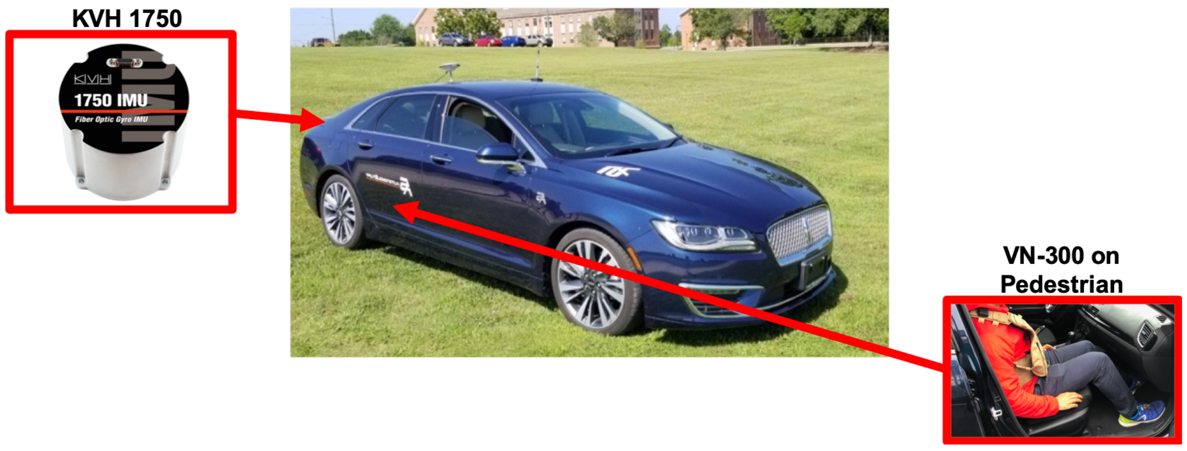

Figure 5.5: The setup used to collect data for the real-world rigidity detector test.

5.2.2 Driving Test

The final test for the rigidity detector was a real-world driving test in a Lincoln MKZ. The MKZ was instrumented with a KVH 1750 [93] in the trunk, while a passenger rode inside the vehicle wearing an Vectornav VN-300 [92] on their right foot. The passenger alternated between moving their foot and holding it still for 10, 20, and 30 second intervals. Each time the passenger started and stopped moving their foot, they pushed a button. The efficacy of the detector and the chosen tuning parameters was evaluated by comparing the times of rigidity and relative motion reported by the detector against the times between the button pushes. The chosen tuning parameters were $th_{rigid} = 0.5\,rad/s$ and $w = 0.5\,sec \times 100hz$. As shown in Table 5.1, the detector was able to detect over 95% of the rigid and relative motion periods.

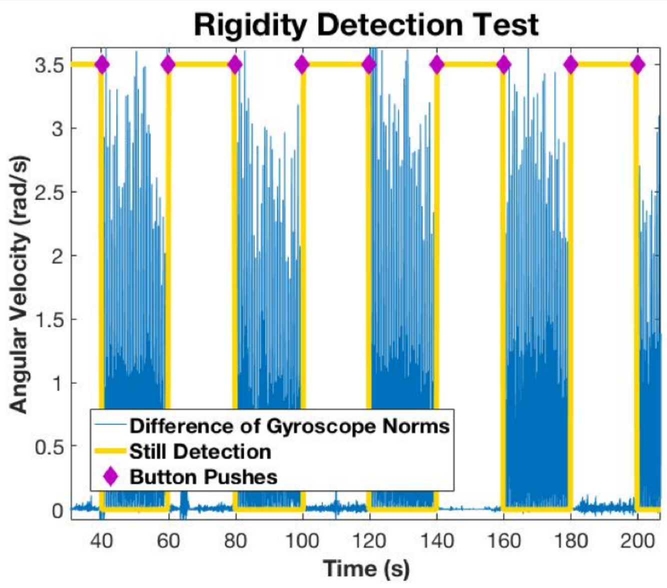

Figure 5.6: Rigidity detector being used on data from drive test.

Table 5.1: Percent rigidity detected during drive test.

Still-Movement Interval (sec)	Percent Rigid Detected	Percent Movement Detected
30	97%	99%
20	98%	99%
10	95%	97%

Chapter 6

Implementation and Performance Evaluation of the In-Vehicle PDR Initialization System

6.1 Outline of the Initialization System

The rigidity detection and transfer alignment algorithms outlined in the previous two chapters make up two parts of a larger combined system that provides initialization information to a pedestrian wearing a PDR system while inside a moving vehicle. The proposed initialization system operates as follows:

1. **Rigidity Detection**: While the pedestrian is riding inside the moving vehicle, detect the rigidity condition using the norm of the angular velocity measurements output from the vehicle and pedestrian IMUs, as described in Chapter 5.
2. **Offset Estimation**: If rigidity is detected, use the approaches discussed in Chapter 4 to estimate lever arm and misalignment.
3. **Break of Rigidity**: If rigidity is broken, store the estimated lever arm and misalignment, and return to the first step.
4. **Initialization and Dismount**: At the instant the pedestrian leaves/dismounts the vehicle, use the latest available lever arm and misalignment estimates to perform a one-shot transferal of the vehicle's PVA to the pedestrian, and use these values to initialize a PDR algorithm.

The remainder of this chapter covers a specific implementation of this initialization system and an evaluation of its performance.

6.2 Implementing the Initialization System

6.2.1 Data Collection Equipment

The data needed to implement the initialization system was collected using an instrumented Lincoln MKZ, previously shown in Figure 5.5. The data itself was collected from two sensor packages, the KVH 1750 IMU [93] and the Vectornav VN-300 dual antenna GPS/INS [92]. The KVH was designated the reference IMU and rigidly mounted inside the MKZ's trunk while the VN-300 was designated the aligning/pedestrian IMU and mounted on a passenger's body, as shown in Figure 5.5.

Three sets of information output by these devices were used to implement the initialization system and then evaluate its performance. The first set was the inertial sensor outputs from both IMUs, which were used to perform transfer alignment and perform PDR once the pedestrian left the vehicle. The second set was the KVH's attitude output. This attitude was computed using a proprietary on-board algorithm, so its exact makeup is unknown. However, it is known to be derived from the KVH's angular velocity measurements, and for the purposes of this thesis, was treated as a source of attitude truth due to its relatively low error. The third and final piece of relevant information was the GPS position measurements made by the VN-300. This information was used as positioning truth after the pedestrian left the vehicle.

6.2.2 Rigidity Detector and PDR

For rigidity detection, the angular velocity-based detection method developed and tested in Chapter 5 was used to detect when the pedestrian and vehicle IMUs formed a rigid body, with a threshold of $th_{rigid} = 0.3\ rad/s$ and $w = 1\ sec$. For PDR, the 15-state IEZ algorithm presented in Chapter 3 was chosen to propagate the pedestrian's position after they exited the vehicle. This choice of PDR algorithm necessitated mounting the aligning IMU (VN-300) to the pedestrian's right foot, as previously shown in Figure 3.5.

6.2.3 Estimating Misalignment and Lever Arm

As discussed in Chapter 4, in order to transfer PVA information from a reference system to an aligning IMU, the misalignment and lever arm between the two systems must be accounted for. If these quantities are unknown, then it is possible to estimate them using the approaches described in Chapter 4. In the context of in-vehicle PDR initialization, estimating the misalignment C_b^R is a must, since knowledge of this quantity allows the post-dismount PDR trajectory to be mapped onto the correct coordinate frame. This is essential, since the post-dismount PDR trajectory would have large errors were it not mapped into the correct coordinate frame. To this end, the SVD approach was used to estimate the misalignment between the reference and aligning IMUs. Additionally, to see if there were any gains in performance from estimating the aligning IMU's gyroscope bias, the Gauss-Newton approach was also used for misalignment estimation.

Estimating the lever arm, however, is less important. Generally, incorporating an accurate lever arm will improve the accuracy of the aligning IMU's PVA and the PDR initialization. However, with the mounting and seating constraints imposed by the Lincoln MKZ, the lever arm, and therefore the error in PDR initialization, can never exceed 2.5 meters. The post-dismount positioning error that would have resulted from the 2.5 meter error would not by itself be discernible using the VN-300's GPS position output, as this error is within the system's margin of positioning error [4]. Therefore, the lever arm was assumed to be $r_{Rb}^R = [000]$ for the implementation in this chapter.

6.2.4 Transferring PVA and Initializing PDR

With the lever arm assumed to be zero, the initial position r_{nb}^n and velocity v_{nb}^n values transferred to the aligning IMU (and their respective covariances) were set to equal those of the reference IMU at the moment of dismount. The initial attitude C_b^n was computed by multiplying the reference IMU's attitude with the misalignment, given by,

$$C_b^n = C_R^n C_b^R. \tag{6.1}$$

where C_R^n is the reference IMU's attitude at the moment of dismount. The initialization of the attitude covariance P_b^n was done by summing the reference IMU's attitude covariance P_R^n and the misalignment covariance P_b^R rotated into the navigation frame. This computation is,

$$P_b^n = P_R^n + C_R^n P_b^R C_R^{nT}. \tag{6.2}$$

For the specific reference IMU chosen earlier in this chapter, the attitude uncertainty, P_R^n, is extremely small due to its quality. Since this uncertainty is small, it was assumed to be zero. Incorporating this assumption results in the initial attitude covariance simply being,

$$P_b^n = C_R^n P_b^R C_R^{nT}. \tag{6.3}$$

The form of the attitude angle uncertainty output by the SVD algorithm, P_{SVD} corresponds to the form of P_b^R, which means it can be directly used as P_b^R. If the Gauss-Newton method is used instead, then,

$$P_b^n = P_{GN}(1{:}3, 1{:}3). \tag{6.4}$$

Whichever way the misalignment is determined, the initial attitude covariance can be used to initialize part of IEZ's initial covaraince matrix by,

$$P_{IEZ,i}(7{:}9, 7{:}9) = P_b^n. \tag{6.5}$$

If the Gauss-Newton method is used to estimate the misalignment, estimates of the aligning IMU's gyroscope biases are also available. The bias covariances computed by the Gauss-Newton method can be used to initialize the gyroscope bias covariance in IEZ's initial co-varaince matrix by,

$$P_{IEZ,i}(12{:}15, 12{:}15) = P_{GN}(4{:}6, 4{:}6). \tag{6.6}$$

Starting from these initial PVA values and covariances, the IEZ method propagated the PVA of the aligning/pedestrian IMU as the pedestrian walked away from the vehicle.

6.3 A Best-Case Implementation

Ideally, the initialization system would be able to accurately transfer PVA from the vehicle's navigation system to that of the pedestrian's. In the context of the implementation outlined in the previous section, this requires an accurate estimate of misalignment. The simplest means to evaluate this quantity would be to compare the estimated misalignment values against their true values. However, determining the true misalignment between an IMU mounted on a person's foot and one in a vehicle's trunk is extremely difficult, and so was not attempted here. Instead, the quality of the misalignment was judged through indirect means.

One indicator of an accurate misalignment estimate would be if the heading of initial IEZ trajectory matched the pedestrian's GPS trajectory post-dismount. This indicator is essential, since having an accurate IEZ trajectory is the goal of developing the initialization system. As important as the quality of the initial IEZ trajectory is however, it is a flawed indicator. The biggest flaw lies in the tendency of an IEZ-computed PDR trajectory to drift soon after the initialization. Drift in a PDR trajectory makes it difficult to judge the quality of its initialization. Therefore, another method of evaluation should also be used.

A second indicator of an accurate misalignment estimate can be seen if IEZ is initialized inside the vehicle during the drive, i.e. well before the dismount. If the misalignment estimate is accurate, then the aligning IMU's attitude with respect to the navigation frame, should closely follow that of the reference IMU. Additionally, any errors in the aligning IMU's attitude should fall within the error bounds established by the attitude covariance. Though tracking the vehicle's attitude is not the ultimate goal of the initialization system, the ability to track its attitude provides an indicator of performance that is less prone to drift than an IEZ-computed PDR trajectory would be.

The performance of the initialization system was evaluated by comparing the performance of these two indicators against a set of baselines created by setting up and running a best-case scenario. This scenario represents an ideal situation that gives the best possible chance for IEZ to i) accurately track the pedestrian's position post-dismount, and ii) track the reference IMU's attitude during driving.

6.3.1 Tracking Position Post-Dismount

The IEZ algorithm must be provided with correct initialization values to give it the best possible chance of accurately tracking a pedestrian's position post-dismount. To establish a baseline against which to evaluate the performance of the initialization system, the best-case scenario initialized the IEZ algorithm using the standstill initialization procedure described in Chapter 3. Specifically, initial position was given by GNSS, initial heading was computed using GNSS course from the first leg of the walk, and initial pitch and roll were calculated using measurements of gravitational acceleration during a minute-long standstill prior to the walk. Once initialized, the pedestrian walked along the 750 foot-long rectangular path shown in Figure 6.6 a total of six times, with their trajectory tracked by the IEZ algorithm each time. This path was chosen since it was located in a large open parking lot, far away from any possible GPS obstructions. Error bounding for IEZ was provided by ZARUs during the standstill period, and by ZUPTs during the walk, as previously described in Chapter 3.

6.3.1.1 Results and Analysis

The six IEZ-computed PDR trajectories initialized from standstill are shown in Figure 6.1, along with their respective positioning errors. These trajectories show some drift, which is expected for a PDR trajectory whose error growth is being checked only by ZUPTs and ZARUs. The most notable characteristic of these trajectories is their initial heading error, shown in Figure 6.2. The heading errors, save for that of run 6, are all fairly small, indicating that the GPS course used initialize the heading was generally accurate. Run 6's large initial heading error is attributed to the fact that its IEZ-computed trajectory drifted significantly during the segment used to measure heading error. An implementation of the initialization system creating Initial PDR heading errors larger than the trend shown in Figure 6.2 could indicate errors in the in-vehicle misalignment estimation process.

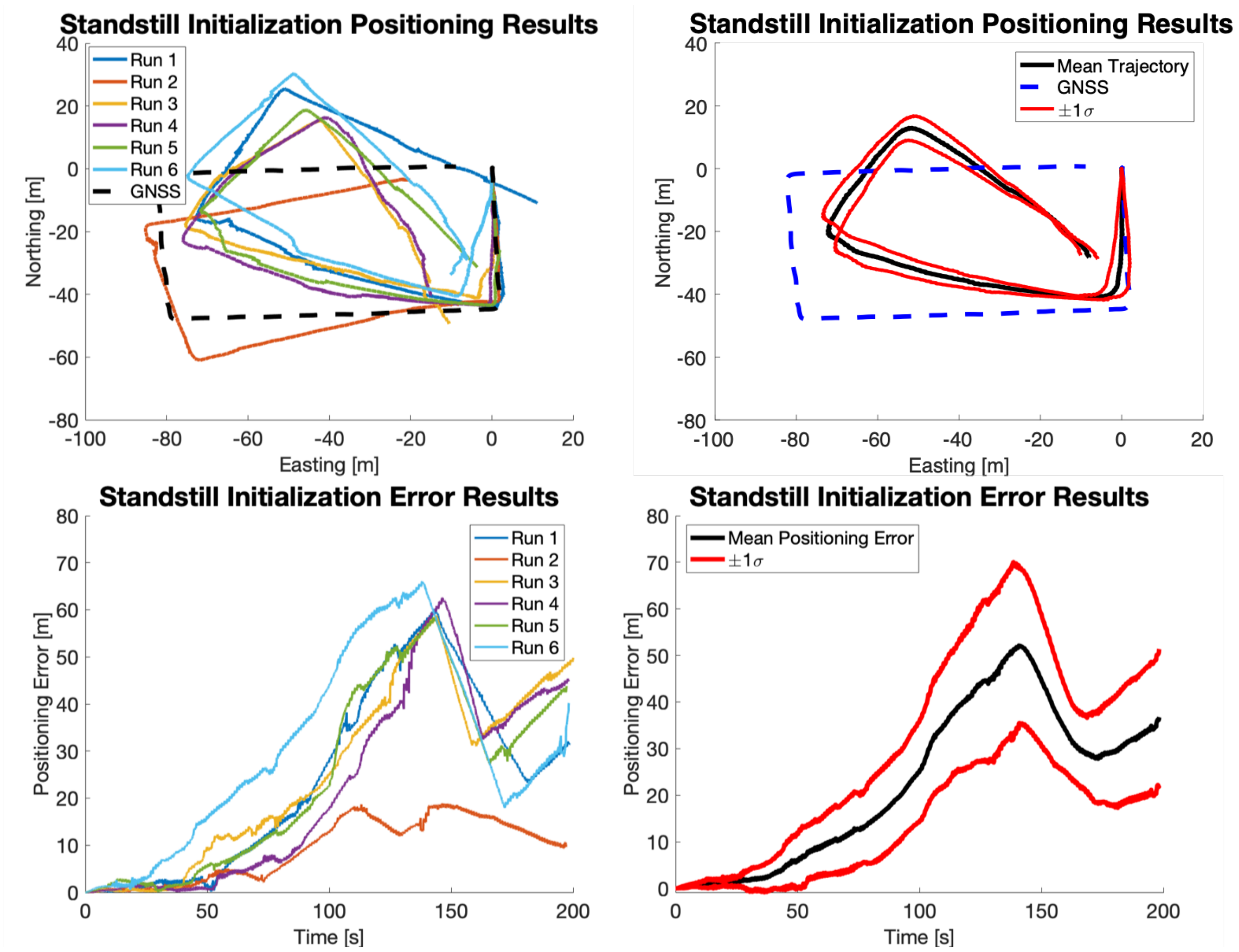

Figure 6.1: The six IEZ-computed PDR trajectories initialized from standstill.

6.3.2 Tracking Attitude Pre-Dismount

An accurate estimate of misalignment must be used to give the IEZ algorithm the best possible chance of tracking the reference IMU's attitude during driving, To ensure this, and establish a baseline against which to compare the performance of the initialization system, a best-case scenario was devised. This scenario involved driving the MKZ on a short, 125-second drive, along the path shown in Figure 6.3. This best-case scenario contained two features. The first feature was the removal of relative IMU motion. Relative IMU motion violates the requirement of rigidity, and will corrupt any estimates of the misalignment. The possibility of relative motion was removed by taking the VN-300 off the pedestrian's foot and rigidly mounting it to the MKZ's roof above the front row of seats, as shown previously in Figure 4.9.

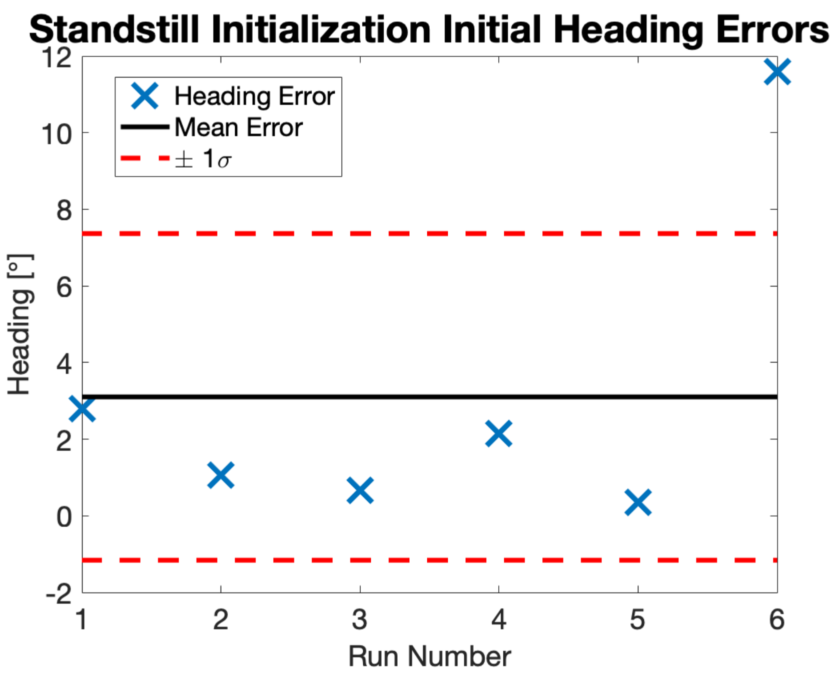

Figure 6.2: Absolute initial heading error for each run of the standstill initialization. The mean heading error was $3.1°$ with a standard deviation of $4.3°$

The second feature was the presence of maneuvers in the drive that are known to be observable. These maneuvers are the four turns that the path takes in between the green and red X's.

To estimate the misalignment between the two IMUs, the SVD and Gauss-Newton methods were used. During each turn, the misalignment was estimated twice, once with the SVD method and once with the Gauss-Newton method, using 10 seconds of angular velocity data [from each of the four turns]. This gave a total of eight misalignment estimates. Each estimate of misalignment was used to rotate the aligning IMU into the frame of the vehicle's. After each rotation, the attitude and attitude covariance of the aligning IMU was initialized at the green X and tracked over the whole 125-second duration of the drive eight different times, using the PVA transferal and PDR initialization procedure described in the previous section.

6.3.2.1 Results and Analysis

The eight estimates of misalignment are summarized in Table 6.1. To get a sense of the accuracy of these estimates, the resulting tracked attitudes were differenced from the vehicle IMU's attitude over the entire 125-second run. An average of these eight errors was taken to form a mean attitude error, and is shown in Figures 6.4 and 6.5. The standard deviation of the

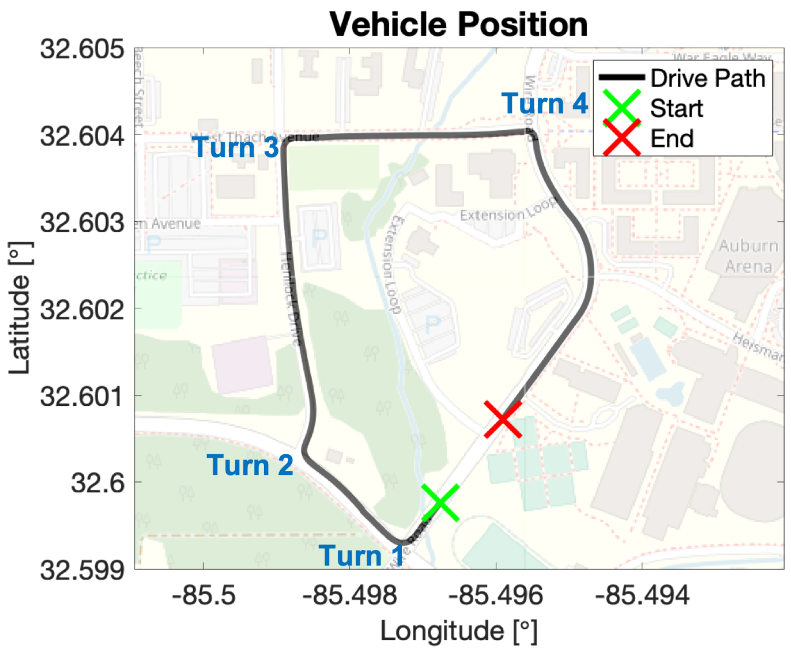

Figure 6.3: The drive path used to test in-vehicle attitude tracking. Angular velocity data from each turn was used separately to estimate the misalignment.

Table 6.1: Estimates of misalignment, once for each turn and estimation method

	Turn 1		Turn 2		Turn 3		Turn 4	
	SVD	GN	SVD	GN	SVD	GN	SVD	GN
Heading	−3.8°	−4.15°	−4.23°	−4.23°	−4.77°	−4.93°	−4.11°	−4.15°
Pitch	3.48°	3.58°	3.63°	3.63°	3.48°	3.59°	3.47°	3.51°
Roll	179.79°	179.99°	179.97°	179.99°	179.68°	179.89°	179.61°	179.83°

error is not shown since they are extremely small, i.e. all of the tracked attitudes were highly similar. The low mean error and the fact the errors fall almost entirely within the 1σ bounds (which come from the covariance matrix) shows that the attitude aligning IMU is generally able to track that of the reference IMU as expected. This finding that is further bolstered by the consistency of the misalignment estimates shown in Table 6.1. An implementation of the initialization system resulting in in-vehicle with errors trending differently than shown in Figure 6.4 could indicate errors in the in-vehicle misalignment estimation process.

Another takeaway from these results has to do with the relatively small effect that the incorporation of gyroscope bias has on the attitude tracking. As seen in Table 6.1, there was little difference between the the SVD and Gauss-Newton derived misalignment angles. As a result,

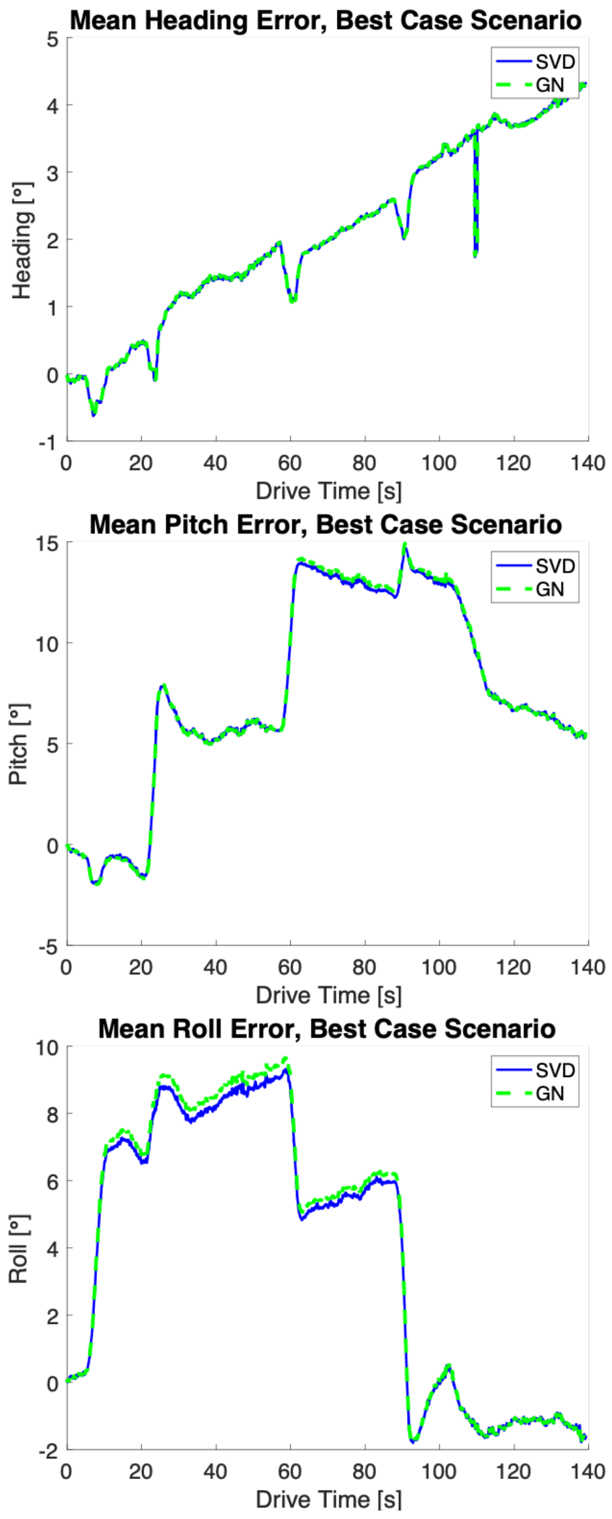

Figure 6.4: The mean attitude errors from performing attitude tracking inside the vehicle.

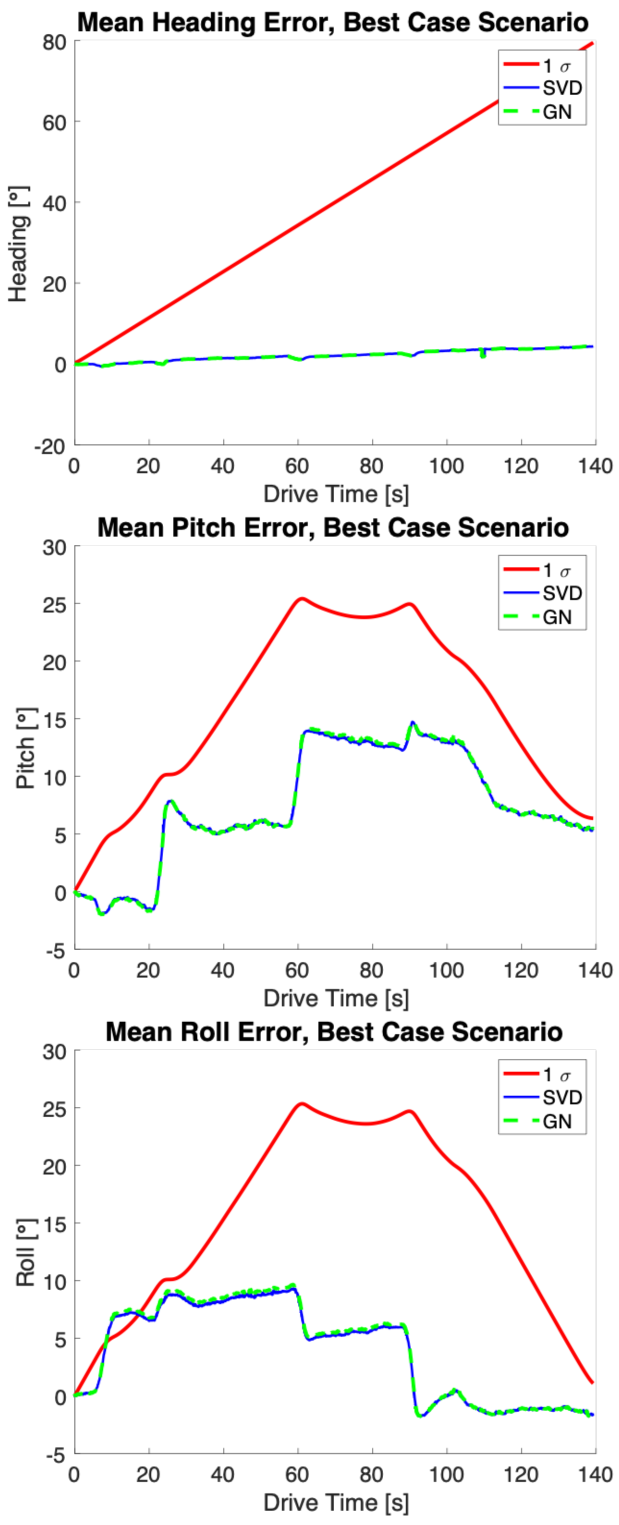

Figure 6.5: The mean attitude errors from performing attitude tracking inside the vehicle, plotted against the expected standard deviation of the errors, as reported by IEZ's covariance matrix.

the maximum mean difference in the tracked attitudes between SVD and Gauss-Newton was less than $0.35°$. This result is likely due to the relatively short length of the drive (approximatley 125 seconds). In such a short drive, the actual gyroscope biases likely did not have enough time to cause the aligning IMU's attitude to drift significantly, which meant that incorporating them provided little benefit.

6.4 A Realistic Implementation

With a performance baseline having been established, it then became possible to implement the initialization system more realistically. To make this implementation as realistic as possible, all of the constraints imposed by the best-case scenario were removed. This meant i) removing the rigidity guarantee by placing the aligning IMU (VN-300) back on the pedestrian's foot, and ii) initializing the IEZ algorithm using misalignment values estimated inside the vehicle. The removal of these constraints introduced the possibility of added initialization error into the system.

Whenever possible, the realistic scenario was designed to be as similar to the best-case scenario as possible. To that end, the walking path that was used in the best case scenario (Figure 6.1) was also used for the realistic scenario, as shown on the right side of Figure 6.6. This similarity between the two scenarios made it possible to compare PDR results from the two scenarios. As stated earlier, this path was chosen since it presented few opportunities for GPS obstructions. However, choosing this path made it impossible to use the same drive path that was used in the best-case scenario (Figure 6.3), since that drive path was neither near nor overlapped with the chosen walking path. Therefore, a new drive path, closer to the chosen walking path than the earlier one, was chosen. This path is shown on the left side of Figure 6.6. Though the new drive path was different from the earlier one, it was not expected to make a significant difference from the standpoint of misalignment estimation. This was due to the fact that the path included several turns, most of which were performed at speeds similar to those of the earlier drive path. The presence of the turns satisfied the linear independence requirement for observing misalignment, while the similarity in driving speed made sure that the observability problem for the new drive path was as well-conditioned as that of the earlier drive path. Therefore, the use of this new path was not expected to make a significant difference from a PDR initialization standpoint.

6.4.1 Data Collection

To test the initialization system, the MKZ with the pedestrian inside of it was taken on the short drive shown in Figure 6.6. The MKZ started the drive at the red X on the north side of the figure, followed the black arrows in a clockwise direction, and finished the loop at the red X. During the drive, the pedestrian sitting the back row created rigid and non-rigid periods by randomly holding their foot still and then moving it. During each rigid period, the SVD method was used to estimate the misalignment between the aligning and reference IMUs.The Gauss-Newton method was not used this time, since the results from the best-case scenario showed that incorporating gyroscope biases into the estimation process did not significantly benefit the initialization problem.

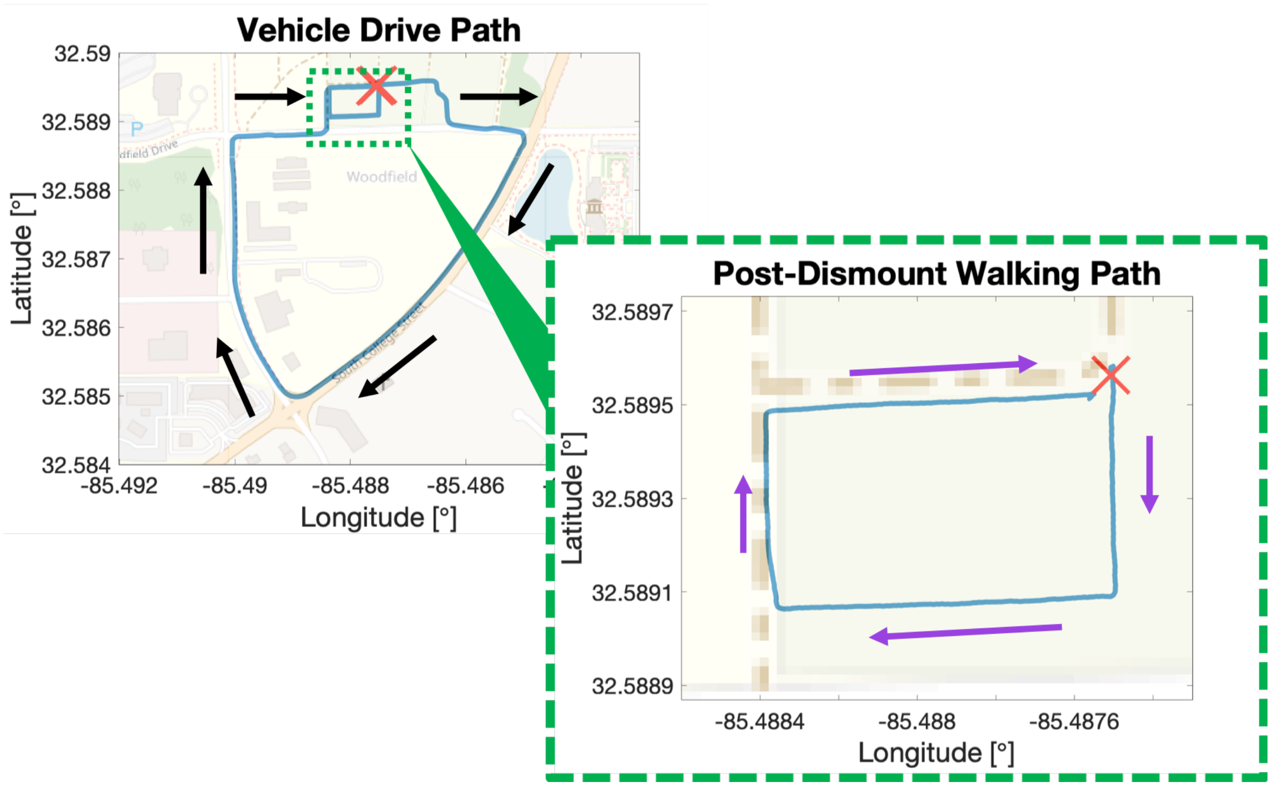

Figure 6.6: The drive and walking path used to collect data for the realistic test of the in-vehicle PDR initialization system.

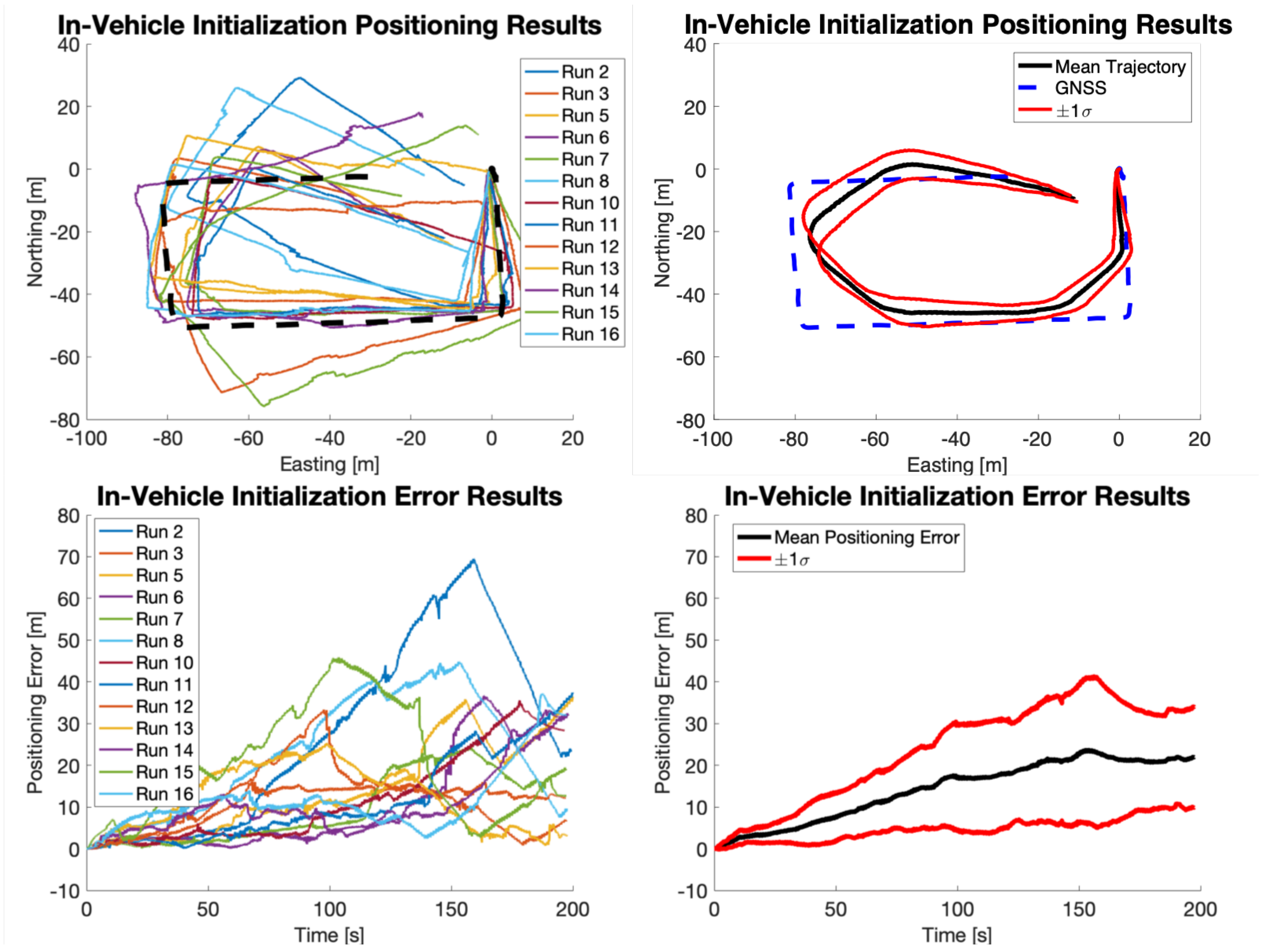

Figure 6.7: The 13 resulting trajectories resulting from in-vehicle SVD-derived initialization.

Once the car returned to the red X, the latest available misalignment was used to initialize the IEZ algorithm as per the procedures described earlier in the chapter. The pedestrian then exited the vehicle and walked along the rectangular path in the direction indicated by the purple arrows, also shown in Figure 6.6. This was done $N = 16$ times to analyze the initialization system's repeatability. Note that runs 1, 4, and 9 were excluded due to the occurrence of technical errors in the data collection hardware.

6.4.2 Tracking Position Post-Dismount: Results and Analysis

The thirteen in-vehicle initialized IEZ-computed PDR trajectories are shown in Figure 6.7, along with their positioning errors. Similar to the standstill initialization, the trajectories show some drift, which is a result of the fact that the error is checked only by ZUPTs and ZARUs.

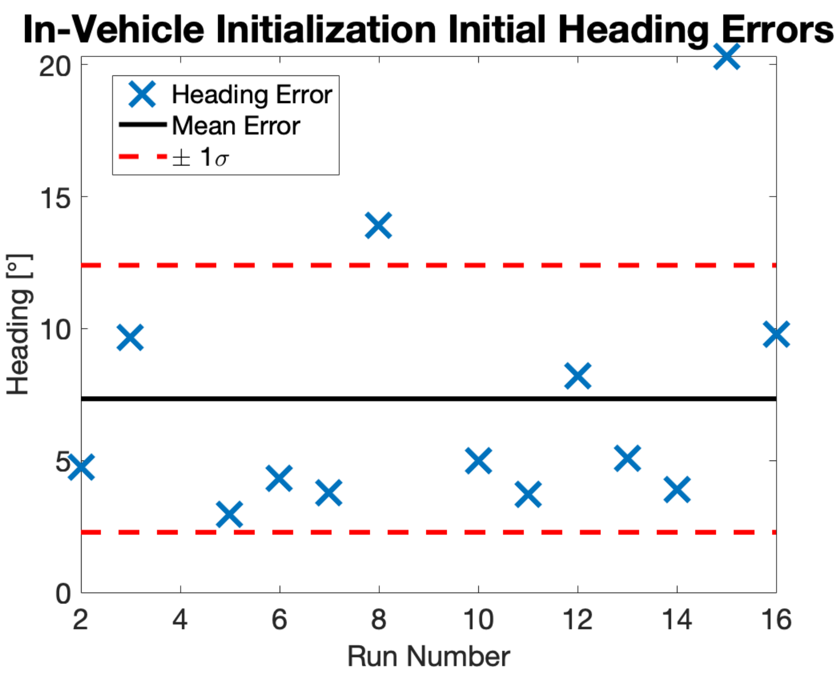

Figure 6.8: The 13 resulting trajectories resulting from in-vehicle SVD-derived initialization. The mean heading error was $7.3°$ with a standard deviation of $5.06°$

At first glance, the initialization system appears to be working. It was able to initialize IEZ in the right position and in the correct general direction, without having the pedestrian go through the inconvenience of the standstill initialization procedure outlined earlier.

However, the results also point to imperfections within the initialization. Though the system was able to initialize in the correct general direction, there was, unlike with the standstill-initialized trajectories shown earlier, significant initial heading error, as shown in Figure 6.8. The mean initial heading error over the standstill initialized IEZ trajectories increased by over 100%. This large increase in error over the best case scenario suggests that there may have been errors in the estimation of misalignment during the drive. However, the initial heading error cannot by itself be taken as an indicator of error in the estimate of misalignment. This is because, as discussed earlier in the chapter, it is difficult to separate the effects of initialization quality on an IEZ-computed trajectory from the effects of drift. Therefore, the other performance indicator, the tracking of the aligning IMU's attitude inside the vehicle, must be used.

6.4.3 Tracking Attitude Pre-Dismount: Results and Analysis

For the last rigid period in each run, the attitude of the aligning IMU was tracked using the estimated misalignment and the procedure outlined earlier in the chapter. These tracked heading, pitch, and roll angles were then compared against those of the vehicle's attitude during each run's last rigid period. The mean and standard deviation of the attitude errors across all of the runs are shown in Figure 6.10. Note that the length of the pre-dismount rigid period was different for each run. Therefore, to be able to compare the attitude tracking errors across runs, the attitude tracking was truncated to the length of the shortest rigid period, which was approximately 25 seconds.

The results in Figure 6.10 show, on average, that there is a significant divergence between the attitudes of the aligning IMU and that of the vehicle, particularly in the heading axis. This occurance of heading divergence is not new. A similar trend was seen for the best-case scenario in Figure 6.4. However, in that scenario, the aligning IMU's heading closely followed that of the vehicle's, an example of which is shown on the left side of Figure 6.9. This effect was captured by the relatively small heading divergence seen in Figure 6.4 ($4°$ over 125 seconds for the best-case scenario). However, for the more realistic scenario, the heading error grew at a much faster rate, as exemplified on the right side of Figure 6.9. As seen in Figure 6.10, this quantity grew $6°$ over 25 seconds. The increase in the size and rate of growth of the heading error for the realistic implementation of in-vehicle PDR initialization strongly suggests the presence of

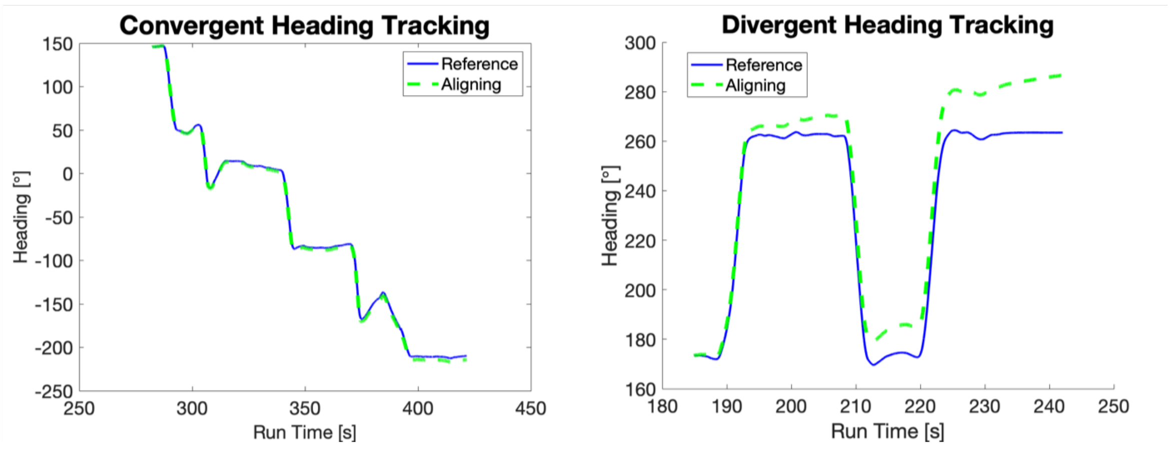

Figure 6.9: An example of good (left) and bad (right) heading tracking. The left plot is from the best case scenario, while the right is from run 5 of the realistic scenario.

estimation error in the misalignment estimates that were much larger than what was present in the best-case scenario, which in turn explains the initial heading shown in Figure 6.8.

6.5 Comparing Residuals

The introduction of large misalignment errors after switching from the best-case to the more realistic scenario can be attributed to the one key difference between the two setups: in the latter, the aligning IMU had freedom to move independently of the vehicle. This freedom came from the fact that the IMU was mounted the pedestrian's foot, which was itself unconstrained. This freedom of movement opened up the possibility for the aligning IMU to experience extraneous motion and consequently violate the rigidity that is needed to successfully estimate the misalignment.

Evidence (or the lack thereof) of extraneous IMU movement can be seen in the angular velocity residual, which is computed by,

$$\omega_{resid} = \omega_{nR_k}^{R} - C_b^R \omega_{nb_k}^{b}. \tag{6.7}$$

Assuming the existence of a true rigid period and an observable maneuver, this quantity would appear as zero mean, normally distributed noise. An example of this can be seen in Figure 6.12, which shows one of the angular velocity residuals for the best-case scenario, when the IMU was mounted rigidly on the roof.

However, these properties did not necessarily apply to the angular velocity residuals computed for the more realistic, IMU on foot, scenario, with two examples shown in the top half of Figure 6.12. The residuals shown in these two plots contain several uncharacteristically large spikes. As the bottom half of Figure 6.12 shows, these spikes correspond to moments of high angular motion. This correspondence suggests that centripetal force during driving maneuvers caused the foot with the aligning IMU attached to actually experience motion independent of the vehicle. This extraneous motion caused the assumptions made by the transfer alignment algorithms, namely that the two IMUs be rigidly attached, to be untrue, and led to erroneous

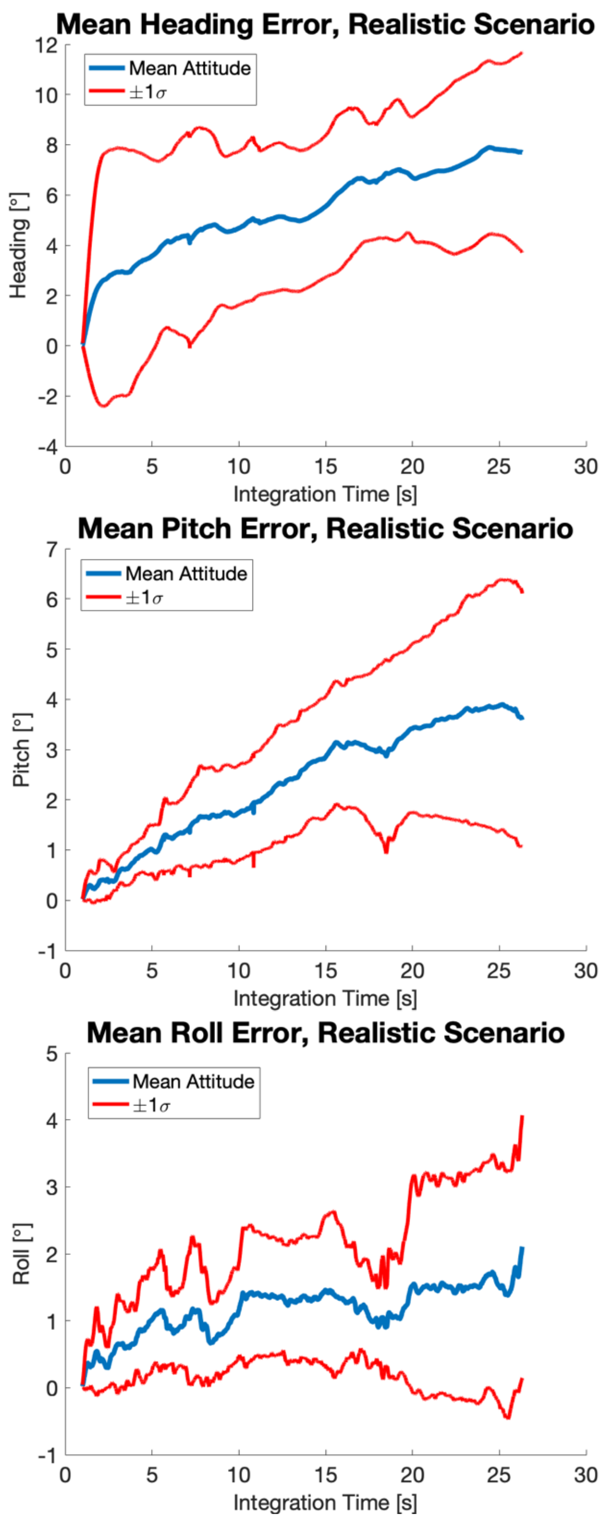

Figure 6.10: The 13 resulting trajectories resulting from in-vehicle SVD-derived initialization. The mean heading error was 7.3° with a standard deviation of 5.06°

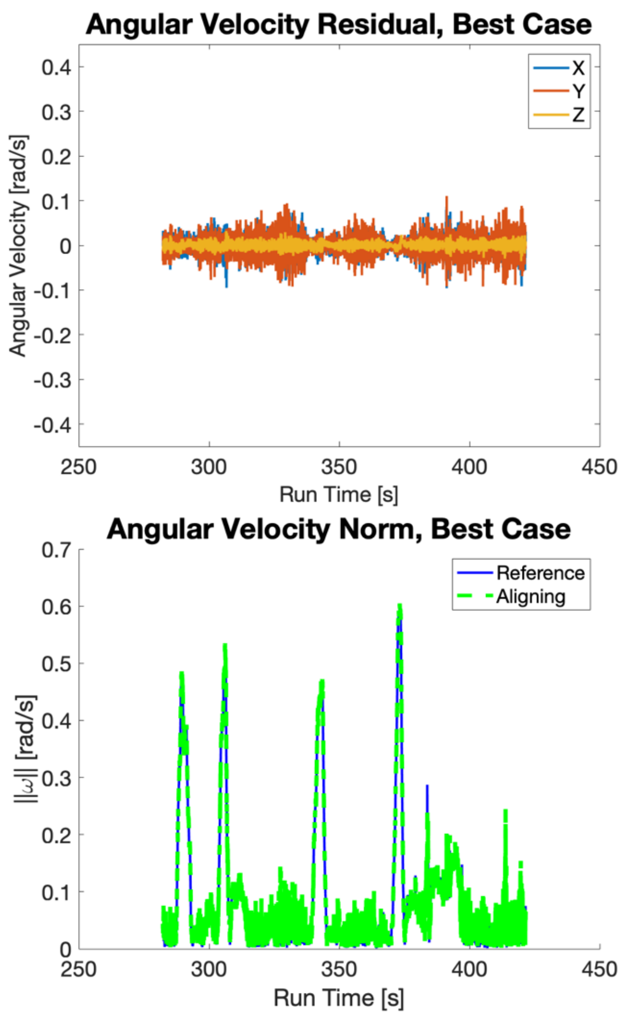

Figure 6.11: A comparison between the angular velocity residual and magnitude for the best case scenario. The residual is a performance indicator for misalignment estimation, while the magnitude is an indicator of motion. Only one of the eight instances of the best case scenario is shown since they all look similar.

estimates of misalignment. Consequently, these erroneous estimates led to increased initialization error of the IEZ algorithm that ultimately caused the errors shown in Figures 6.7, 6.8, and 6.10. The rigidity detector could have been tuned to exclude these moments of non-rigidity. However, doing so would have resulted in excluding the high-dynamic angular velocity data created by maneuvers that was necessary for estimation of the misalignment. In other words, it was not possible to seperate the extraneous angular motion from the maneuver. In summary, taking the aligning IMU's motion constraints away created a situation that made it easier for it to behave in manner that was counter-productive to performing successful transfer alignment.

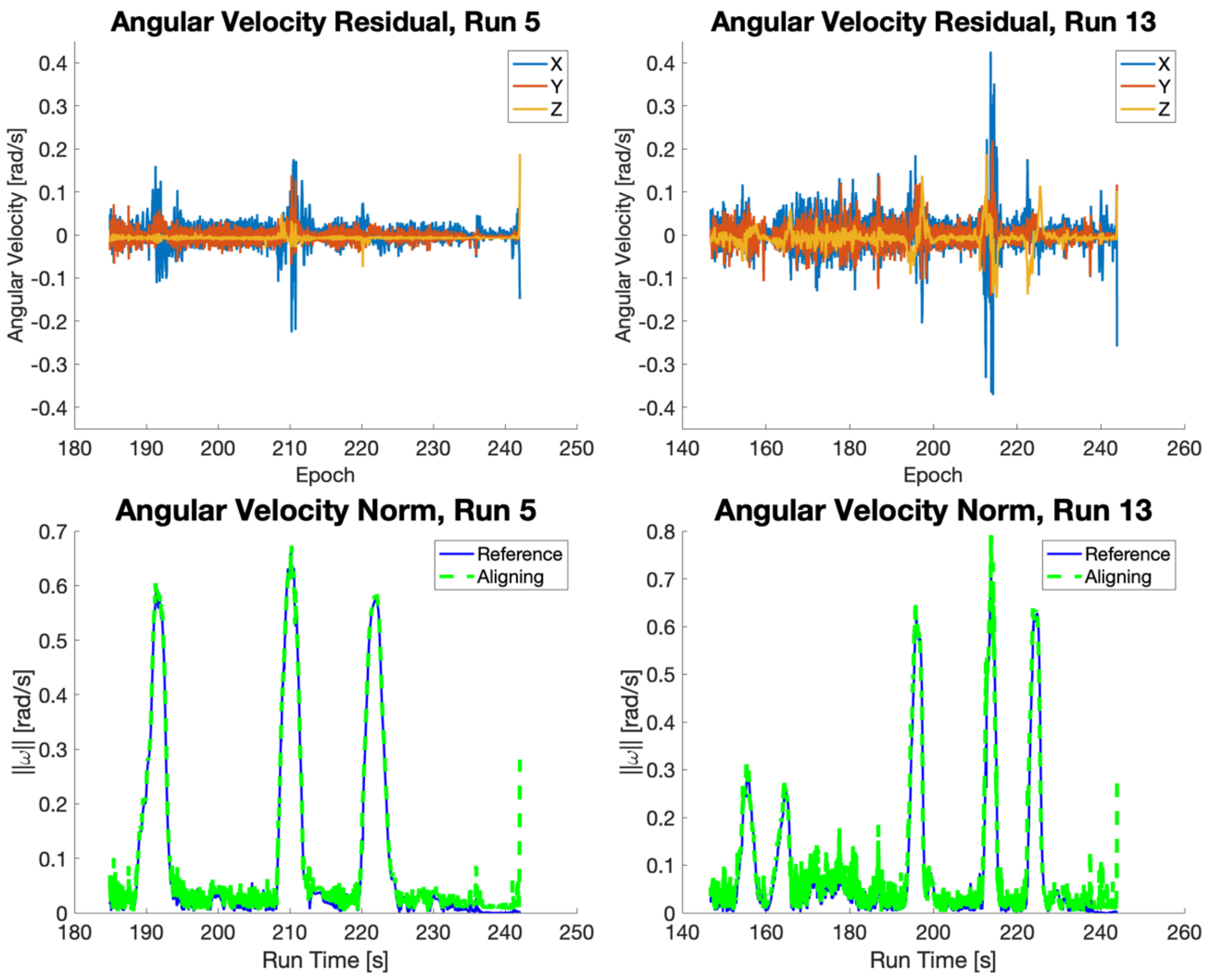

Figure 6.12: A comparison between the angular velocity residual and magnitude for two of the runs of the realistic scenario. The residual is a performance indicator for misalignment estimation, while the magnitude is an indicator of motion. Note the large residual spikes that correspond instances of high angular motion.

The potential for this sort of behavior means that, as well as the initialization system may work *sometimes*, it cannot be guaranteed to work *every time*.

Chapter 7

Summary, Conclusions, and Future Work

7.1 Summary and Conclusion

A challenge faced by users of PDR is the acquisition of initialization information. Currently, such information comes from GNSS for initial position, calibrated magnetic field readings for initial heading, and a standstill period to measure gravity for initial pitch and roll. These methods are permissible for research purposes. However, they are not always practical for real world users who may have to operate in GNSS-scare or magnetically dirty environments, or for users who do not have the time to stand still. Recent developments in vehicle navigation systems have resulted in vehicles being equipped with their own navigation navigation sensors. Assuming these sensors are of a high enough quality, this creates an opportunity for a pedestrian riding inside one of these vehicles to use the vehicle's navigation system to initialize and calibrate their PDR system using transfer alignment techniques.

Applying transfer alignment to initialize a PDR system from a vehicle's navigation system requires that the misalignment and lever arm between the aligning and reference systems be estimated. The lever arm can be estimated in a linear estimation framework, while the misalignment must be estimated in way that accounts for the non-linear nature of the relationship and preserves its orthonormality. For this reason, it was determined that the misalignment must be estimated using either a Wahba's problem solution or a non-linear least squares approach, rather than the existing ESFK approaches.

To successfully estimate lever arm, misalignment, and the aligning IMU's biases it was determined through observability analysis and simulation that the vehicle carrying the pedestrian must undergo linearly independent, non-constant, angular motion in at least two orthogonal axes. Practically, these maneuvers can be any that involve turning the steering wheel, such as lane changing or turning, provided enough motion to be able to successfully estimate lever arm and misalignment. A driving test was used estimate the misalignment and lever arm to within a degree and 0.4 meters of truth, respectively.

The successful application of transfer alignment required that a state of rigidity exist between the vehicle and pedestrian IMUs. To this end, a rigidity detector was developed that used angular velocity outputs from the two IMUs. Both simulation and real world testing showed that the detector was able to flag over 90 % of the non-rigid periods.

Using transfer alignment and rigidity detection techniques, an in-vehicle PDR initialization system was devised. The system operated as follows: while driving, the rigidity detector ascertained when the IMU worn by the pedestrian is being held still relative to the vehicle. During each rigid period, the misalignment between the vehicle and pedestrian IMUs was estimated. Lever arm estimation was omitted since it was determined that the benefits were limited. Once the vehicle came to a stop, the latest available misalignment was used in conjunction with the vehicle IMU's attitude, which was assumed to be known, to initialize a PDR algorithm. A realistic implementation of this system was able to accurately initialize the position of the pedestrian, and orient the pedestrian-mounted IMU in the correct general direction, with a mean initial heading error of $7.3°$. Errors in the initialization process were attributed to the pedestrian IMU becoming momentarily non-rigid during high dynamic maneuvers such as turns. These small but significant movements during what was nominally considered to be a rigid period, momentarily violated the requirement of rigidity, which introduced additional error into the estimates of misalignment. These moments of non-rigidity were due to the fact that the pedestrian IMU's motion was not constrained in any way. In general, the PDR initialization system could detect most occurrences of non-rigidity, but could not not account for *all* of such instances.

7.2 Future Work

There are several options for advancing the work presented in this thesis. These options include:

1. **Rigid Transfer Alignment**: The small, extraneous movements experienced by the pedestrian-mounted IMU during periods of nominal rigidity was determined to cause error in the estimation of misalignment due to the pedestrian-mounted IMU being unconstrained. This problem could be eliminated by constraining the IMU's motion. A motion constraint could be implemented by taking the IMU off the pedestrian's foot, rigidly mounting it to the vehicle's interior, and estimating the relative misalignment from the rigidly mounted position. When it comes time for the pedestrian to dismount, the estimated misalignment can be used to initialize PDR on the mounting point, and track the IMU's motion as the pedestrian attaches it to their foot and walks away from the vehicle.

2. **Tracking the pedestrian IMU during non-rigid periods**: An assumption made was made in this thesis that the misalignment remained constant during periods of non-rigidity, which is technically incorrect since the pedestrian-mounted IMU does move around. Integrating the pedestrian IMU's output during non-rigid periods could be a means to bridge the informational gap during the non-rigid periods. A starting point would be to implement a solution based on the approach discussed in [90].

3. **A centralized non-linear filtering framework**: This thesis implemented the transfer of alignment information from the vehicle to the pedestrian IMU as a one-shot solution. This was because the existing centralized framework, TA-INS-ESKF, cannot be used to estimate large misalignment angles, due to the framework's assumptions of linearity. The whole estimation process could be streamlined if misalignment angles could be estimated as states alongside PVA. This would require a framework that could handle non-linearities such as those that come with large misalignment angles. A starting point would be to turn the ESKF presented in [62] into a particle filter or Unscented Kalman filter.

4. **Using other sources of information to initialize PDR from a vehicle**: To get around the problems inherent to using information from an unconstrained IMU to perform transfer alignment, other sources of information may be used to initialize a PDR algorithm while the pedestrian is inside or even walking away from the vehicle. Cameras and ultra-wideband ranging radios could be used for this purpose.

Further investigation of these and other methods will improve the accuracy and robustness of PDR initialization, which will further enable the practical use of pedestrian dead reckoning systems.

References

[1] K. R. Howe, Vaka moana: Voyages of the ancestors: The discovery and settlement of the Pacific. University of Hawaii Press, 2006.

[2] Groves, P., 2013, Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition.

[3] Groves, P., 2013, Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, First Edition.

[4] Misra, P., and Enge, P., 2006, "Global Positioning System: Signals, Measurements and Performance Second Edition," Global Positioning System: Signals, Measurements And Performance Second Editions.

[5] Groves, P., Pulford, G., Aaron Littlefield, C., L J Nash, D., and J Mather, C., 2007, Inertial Navigation Versus Pedestrian Dead Reckoning: Optimizing the Integration.

[6] Harle, R., 2013, "A Survey of Indoor Inertial Positioning Systems for Pedestrians," IEEE Communications Surveys & Tutorials, 15(3), pp. 1281–1293.

[7] Jimenez, A. R., Seco, F., Prieto, J. C., and Guevara, J., 2010, "Indoor Pedestrian Navigation Using an INS/EKF Framework for Yaw Drift Reduction and a Foot-Mounted IMU," IEEE, pp. 135–143.

[8] Mainetti, L., Patrono, L., and Sergi, I., 2014, "A Survey on Indoor Positioning Systems," 2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 111–120.

[9] Woodman, O., 2010, “Pedestrian Localisation for Indoor Environments.”

[10] Woodman, O., and Harle, R., 2008, “Pedestrian Localisation for Indoor Environments,” Proceedings of the 10th International Conference on Ubiquitous Computing, ACM, pp. 114–123.

[11] Gabaglio, V., 2003, GPS/INS Integration for Pedestrian Navigation, EPFL (Lausanne).

[12] Kim, J. W., Jang, H. J., Hwang, D.-H., and Park, C., 2004, “A Step, Stride and Heading Determination for the Pedestrian Navigation System,” Positioning, 1(08), p. 0.

[13] Alvarez, D., González, R. C., López, A., and Alvarez, J. C., 2008, “Comparison of Step Length Estimators from Wearable Accelerometer Devices,” Encyclopedia of Healthcare Information Systems, IGI Global, pp. 244–250.

[14] Jimenez, A. R., Seco, F., Prieto, C., and Guevara, J., 2009, “A Comparison of Pedestrian Dead-Reckoning Algorithms Using a Low-Cost MEMS IMU,” 2009 IEEE International Symposium on Intelligent Signal Processing, IEEE, pp. 37–42.

[15] Pierce, D., 2016, “Incorporation of a Foot-Mounted IMU for Multi-Sensor Pedestrian Navigation.”

[16] Shin, S., Park, C., Kim, J., Hong, H., and Lee, J., 2007, “Adaptive Step Length Estimation Algorithm Using Low-Cost MEMS Inertial Sensors,” 2007 Ieee Sensors Applications Symposium, IEEE, pp. 1–5.

[17] Skog, I., Handel, P., Nilsson, J.-O., and Rantakokko, J., 2010, “Zero-Velocity Detection—An Algorithm Evaluation,” IEEE transactions on biomedical engineering, 57(11), pp. 2657–2666.

[18] Stirling, R., Collin, J., Fyfe, K., and Lachapelle, G., 2003, “An Innovative Shoe-Mounted Pedestrian Navigation System,” Proceedings of European Navigation Conference GNSS.

[19] Luettel, T., Himmelsbach, M., and Wuensche, H.-J., 2012, “Autonomous Ground Vehicles—Concepts and a Path to the Future,” Proceedings of the IEEE, 100(Special Centennial Issue), pp. 1831–1839.

[20] Shaout, A., Colella, D., and Awad, S., 2011, “Advanced Driver Assistance Systems-Past, Present and Future,” 2011 Seventh International Computer Engineering Conference (ICENCO’2011), IEEE, pp. 72–82.

[21] Ginsberg, J., 2008, Engineering Dynamics, Cambridge University Press.

[22] Curtis, H. D., 2013, Orbital Mechanics for Engineering Students, Butterworth-Heinemann.

[23] Choukroun, D., Weiss, H., Bar-Itzhack, I. Y., and Oshman, Y., 2006, “Kalman Filtering for Matrix Estimation,” IEEE Transactions on Aerospace and Electronic Systems, 42(1), pp. 147–159.

[24] Ray, T., 2019, “Pedestrian Navigation Using Particle Filtering and a Priori Building Maps.”

[25] Foxlin, E., 2005, “Pedestrian Tracking with Shoe-Mounted Inertial Sensors,” IEEE Computer Graphics and Applications, 25(6), pp. 38–46.

[26] D. Levine, J. Richards, and M. W. Whittle, Whittle’s gait analysis. Elsevier Health Sciences, 2012.

[27] Feliz Alonso, R., Zalama Casanova, E., and Gómez García-Bermejo, J., 2009, “Pedestrian Tracking Using Inertial Sensors.”

[28] Strozzi, N., Parisi, F., and Ferrari, G., 2016, “On Single Sensor-Based Inertial Navigation,” 2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN), IEEE, pp. 300–305.

[29] Ojeda, L., and Borenstein, J., 2006, "Non-GPS Navigation for Emergency Responders," 2006 International Joint Topical Meeting: Sharing Solutions for Emergencies and Hazardous Environments, pp. 12–15.

[30] Ladetto, Q., 2000, "On Foot Navigation: Continuous Step Calibration Using Both Complementary Recursive Prediction and Adaptive Kalman Filtering," Proceedings of ION GPS.

[31] Beauregard, S., 2006, "A Helmet-Mounted Pedestrian Dead Reckoning System," 3rd International Forum on Applied Wearable Computing 2006, VDE, pp. 1–11.

[32] Beauregard, S., and Haas, H., 2006, "Pedestrian Dead Reckoning: A Basis for Personal Positioning," Proceedings of the 3rd Workshop on Positioning, Navigation and Communication (WPNC'06).

[33] Godha, S., Lachapelle, G., and Cannon, M. E., 2006, "Integrated GPS/INS System for Pedestrian Navigation in a Signal Degraded Environment," Proceedings of the 19th International Technical Meetings of the Satellite Division of the Institute of Navigation, Fort Worth, TX.

[34] Leppakoski, H., Kappi, J., Syrjarinne, J., and Takala, J., 2002, "Error Analysis of Step Length Estimation in Pedestrian Dead Reckoning," ION GPS 2002: 15 Th International Technical Meeting of the Satellite Division of The Institute of Navigation.

[35] Zhao, N., 2010, "Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer," Analog Dialogue, 44(06), pp. 1–5.

[36] Randell, C., Djiallis, C., and Muller, H., 2003, "Personal Position Measurement Using Dead Reckoning," Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings., IEEE, pp. 166–173.

[37] Cho, S. Y., Lee, K. W., Park, C. G., and Lee, J. G., 2003, "A Personal Navigation System Using Low-Cost MEMS/GPS/Fluxgate."

[38] Weinberg, H., 2002, “Using the ADXL202 in Pedometer and Personal Navigation Applications.”

[39] Bylemans, I., Weyn, M., and Klepal, M., 2009, “Mobile Phone-Based Displacement Estimation for Opportunistic Localisation Systems,” 2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, IEEE, pp. 113–118.

[40] Davidson, P., and Piché, R., 2016, “A Survey of Selected Indoor Positioning Methods for Smartphones,” IEEE Communications Surveys and Tutorials, 19(2), pp. 1347–1370.

[41] Zijlstra, W., and Hof, A. L., 1997, “Displacement of the Pelvis during Human Walking: Experimental Data and Model Predictions,” Gait and posture, 6(3), pp. 249–262.

[42] J. Jahn, U. Batzer, J. Seitz, L. Patino-Studencka, and J. Gutiérrez Boronat, 2010, “Comparison and Evaluation of Acceleration Based Step Length Estimators for Handheld Devices,” 2010 International Conference on Indoor Positioning and Indoor Navigation, pp. 1–6.

[43] Yang, S., and Li, Q., 2010, “Ambulatory Walking Speed Estimation under Different Step Lengths and Frequencies,” 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, IEEE, pp. 658–663.

[44] Bertram, J. E., and Ruina, A., 2001, “Multiple Walking Speed–Frequency Relations Are Predicted by Constrained Optimization,” Journal of theoretical Biology, 209(4), pp. 445–453.

[45] D. Gusenbauer, C. Isert, and J. Krösche, 2010, “Self-Contained Indoor Positioning on off-the-Shelf Mobile Devices,” 2010 International Conference on Indoor Positioning and Indoor Navigation, pp. 1–9.

[46] Gebre-Egziabher, D., Elkaim, G. H., David Powell, J., and Parkinson, B. W., 2006, “Calibration of Strapdown Magnetometers in Magnetic Field Domain,” Journal of Aerospace Engineering, 19(2), pp. 87–102.

[47] Madgwick, S., 2010, “An Efficient Orientation Filter for Inertial and Inertial/Magnetic Sensor Arrays,” Report x-io and University of Bristol (UK), 25, pp. 113–118.

[48] Marins, J. L., Yun, X., Bachmann, E. R., McGhee, R. B., and Zyda, M. J., 2001, “An Extended Kalman Filter for Quaternion-Based Orientation Estimation Using MARG Sensors,” Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180), IEEE, pp. 2003–2011.

[49] Sabatini, A. M., 2006, “Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic Sensing,” IEEE transactions on Biomedical Engineering, 53(7), pp. 1346–1356.

[50] Suh, Y. S., 2010, “Orientation Estimation Using a Quaternion-Based Indirect Kalman Filter with Adaptive Estimation of External Acceleration,” IEEE Transactions on Instrumentation and Measurement, 59(12), pp. 3296–3305.

[51] SPALDING, K., 1992, “An Efficient Rapid Transfer Alignment Filter,” Astrodynamics Conference, p. 4598.

[52] Yüksel, Y., 2005, “Design and Analysis of Transfer Alignment Algorithms,” Middle East Technical University, pp. 35–38.

[53] Kain, J., and Cloutier, J., 1989, “Rapid Transfer Alignment for Tactical Weapon Applications,” Guidance, Navigation and Control Conference, p. 3581.

[54] Sutherland Jr, A. A., and Gelb, A., 1968, The Kalman Filter in Transfer Alignment of Airborne Inertial Guidance Systems, ANALYTIC SCIENCES CORP READING MA.

[55] ROGERS, R., 1991, “Velocity-plus-Rate Matching for Improved Tactical Weapon Rapid Transfer Alignment,” Navigation and Control Conference, p. 2783.

[56] Si, F., Zhao, Y., Lin, Y., and Zhang, X., 2018, “A New Transfer Alignment of Airborne Weapons Based on Relative Navigation,” Measurement, 122, pp. 27–39.

[57] Sinpyo Hong, Man Hyung Lee, Sun Hong Kwon, and Ho Hwan Chun, 2004, “A Car Test for the Estimation of GPS/INS Alignment Errors,” IEEE Transactions on Intelligent Transportation Systems, 5(3), pp. 208–218.

[58] Hong, S., Chang, Y.-S., Ha, S.-K., and Lee, M.-H., 2002, “Estimation of Alignment Errors in GPS/INS Integration,” Proceedings of the ION GPS, Portland, OR, USA, pp. 24–27.

[59] Sinpyo Hong, Man Hyung Lee, Ho-Hwan Chun, Sun-Hong Kwon, and Speyer, J. L., 2006, “Experimental Study on the Estimation of Lever Arm in GPS/INS,” IEEE Transactions on Vehicular Technology, 55(2), pp. 431–448.

[60] Montalbano, N., and Humphreys, T., 2018, “A Comparison of Methods for Online Lever Arm Estimation in GPS/INS Integration,” 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS), IEEE, pp. 680–687.

[61] Tang, Y., Wu, Y., Wu, M., Wu, W., Hu, X., and Shen, L., 2008, “INS/GPS Integration: Global Observability Analysis,” IEEE Transactions on Vehicular Technology, 58(3), pp. 1129–1142.

[62] Lee, M. K., Hong, S., Lee, M. H., Kwon, S. H., and Chun, H.-H., 2005, “Observability Analysis of Alignment Errors in GPS/INS,” Journal of Mechanical Science and Technology, 19(6), pp. 1253–1267.

[63] Hong, S., Lee, M. H., Rios, J. A., and Speyer, J. L., 2002, “Observability Analysis of Ins with a GPS Multi-Antenna System,” KSME International Journal, 16(11), pp. 1367–1378.

[64] Rhee, I., Abdel-Hafez, M. F., and Speyer, J. L., 2004, “Observability of an Integrated GPS/INS during Maneuvers,” IEEE Transactions on Aerospace and Electronic systems, 40(2), pp. 526–535.

[65] Sinpyo Hong, Man Hyung Lee, Ho-Hwan Chun, Sun-Hong Kwon, and Speyer, J. L., 2005, “Observability of Error States in GPS/INS Integration,” IEEE Transactions on Vehicular Technology, 54(2), pp. 731–743.

[66] Groves, P. D., Wilson, G. G., and Mather, C. J., 2001, “Robust Rapid Transfer Alignment with an INS/GPS Reference,” Proceedings of the 2002 National Technical Meeting of The Institute of Navigation, pp. 301–311.

[67] G. Wahba, “A Least Squares Estimate of Satellite Attitude,” SIAM Rev., vol. 7, no. 3, pp. 409–409, Jul. 1965.

[68] Chardonnens, J., Favre, J., and Aminian, K., 2012, “An Effortless Procedure to Align the Local Frame of an Inertial Measurement Unit to the Local Frame of Another Motion Capture System,” Journal of biomechanics, 45(13), pp. 2297–2300.

[69] Vries, W. H. K. de, Veeger, H. E. J., Baten, C. T. M., and Helm, F. C. T. van der, 2009, “Magnetic Distortion in Motion Labs, Implications for Validating Inertial Magnetic Sensors,” Gait and Posture, 29(4), pp. 535–541.

[70] Lu, Y., and Cheng, X., 2014, “Random Misalignment and Lever Arm Vector Online Estimation in Shipborne Aircraft Transfer Alignment,” Measurement, 47, pp. 756–764.

[71] Markley, Landis, “Attitude Determination Using Vector Observations and the Singular Value Decomposition,” Journal of the Astronautical Sciences, Nov. 1987.

[72] Markley, Landis, “Attitude Determination from Vector Observations: A Fast Optimal Matrix Algorithm,” Journal of the Astronautical Sciences, vol. 2, no. 41, pp. 261–280, Jun. 1993.

[73] Markley, Landis, “30 Years of Wahba’s Problem,” presented at the Flight Mechanics, Greenbelt, MD; United States, 1999.
[**?**] Mortari, D, 1997, “ESOQ: A Closed-Form Solution to the Wahba Problem,” JOURNAL OF THE ASTRONAUTICAL SCIENCES, 2(45), pp. 195–204.

[74] Keat, J, “Analysis of Least Sqaures Attitude Determination Routine,” Computer Sciences Corporation, Feb. 1977.

[75] Shuster, M.D, Oh, S.D., “Three-axis Attitude Determination from Vector Observations,” Journal of Guidance and Control, pp. 70–77, Feb. 1981.

[76] Madinehi, N., 2013, “Rigid Body Attitude Estimation: An Overview and Comparative Study.”

[77] Schneider, A. M., 1983, “Kalman Filter Formulations for Transfer Alignment of Strapdown Inertial Units,” Navigation, 30(1), pp. 72–89.

[78] Reid, J., 1980, “An Extended Kalman Filter for the Estimation of Transfer Alignment Errors to an Airborne Vehicle,” Guidance and Control Conference, p. 1720.

[79] Markley, Landis, 1987, “Attitude Determination Using Vector Observations and the Singular Value Decomposition,” Journal of the Astronautical Sciences.

[80] BAR-ITZHACK, I. Y., and IDAN, M., 1987, “Recursive Attitude Determination from Vector Observations: Euler Angle Estimation,” Journal of Guidance, Control, and Dynamics, 10(2), pp. 152–157.

[81] Crabolu, M., Pani, D., Raffo, L., Conti, M., Crivelli, P., and Cereatti, A., 2017, “In Vivo Estimation of the Shoulder Joint Center of Rotation Using Magneto-Inertial Sensors: MRI-Based Accuracy and Repeatability Assessment,” Biomed Eng Online, 16(1), pp. 34–34.

[82] Stengel, R. F., 1994, Optimal Control and Estimation, Courier Corporation.

[83] Khan, M., Salman, N., Ali, A., Khan, A., and Kemp, A., 2015, “A Comparative Study of Target Tracking with Kalman Filter, Extended Kalman Filter and Particle Filter Using Received Signal Strength Measurements,” 2015 International Conference on Emerging Technologies (ICET), IEEE, pp. 1–6.

[84] Goshen-Meskin, D., and Bar-Itzhack, I. Y., 1992, “Observability Analysis of Piece-Wise Constant Systems. I. Theory,” IEEE Transactions on Aerospace and Electronic Systems, 28(4), pp. 1056–1067.

[85] Goshen-Meskin, D., and Bar-Itzhack, I. Y., 1992, "Observability Analysis of Piece-Wise Constant Systems. II. Application to Inertial Navigation in-Flight Alignment (Military Applications)," IEEE Transactions on Aerospace and Electronic Systems, 28(4), pp. 1068–1075.

[86] Zhilan Xiong, Feng Sun, and Qi Nie, 2006, "Observability Analysis of INS Rapid Transfer Alignment," 2006 6th World Congress on Intelligent Control and Automation, pp. 1664–1668.

[87] Bar-Itzhack, I. Y., and Porat, B., 1980, "Azimuth Observability Enhancement during Inertial Navigation System In-Flight Alignment," Journal of Guidance and Control, 3(4), pp. 337–344.

[88] Groves, P., Pulford, G., Aaron Littlefield, C., L J Nash, D., and J Mather, C., 2007, Inertial Navigation Versus Pedestrian Dead Reckoning: Optimizing the Integration.

[89] Welch, G., and Bishop, G., 2006, An Introduction to the Kalman Filter.

[90] Si, F., Zhao, Y., Lin, Y., and Zhang, X., 2018, "A New Transfer Alignment of Airborne Weapons Based on Relative Navigation," Measurement, 122, pp. 27–39.

[91] Bar-Itzhack, I. Y., and Reiner, J., 1984, "Recursive Attitude Determination from Vector Observations: Direction Cosine Matrix Identification," Journal of Guidance, Control, and Dynamics, 7(1), pp. 51–56.

[92] "VN-300 Specifications" [Online]. Available: https://www.vectornav.com/products/vn-300. [Accessed: 10-Apr-2020].

[93] "1750 IMU Fiber Optic Gyro Inertial Measurement Unit" [Online]. Available: https://kpp-public.s3.amazonaws.com/DATASHEET+-+1750+IMU. [Accessed: 10-Apr-2020].

Appendices

Appendix A

Kalman Filtering

Kalman filtering is a framework that forms the basis of many estimation algorithms used in navigation systems. Its uses include the maintenance of a satellite navigation solution, alignment and calibration of an INS, and integration of an INS with GNSS and other navigation sensors. It is primarily used to obtain an optimal navigation solution from the various measurements available to a navigation system. This chapter provides an overview of Kalman filter based off the discussions in [2] and [89].

A.1 The Linear Kalman Filter

The most basic Kalman filter is one that addresses the problem of trying to estimate the state x of a process that is governed by the linear stochastic difference equation

$$x_{k+1} = Fx_k + Gw_k \tag{A.1}$$

with an empirical measurement y such that

$$y_k = Hx_k + v_k. \tag{A.2}$$

x is an $n \times 1$ column vector of the system states. These states are propagated through time by the $n \times n$ state transition matrix F. w is an $p \times 1$ vector of the system's process noise sources. They are mapped onto the state domain through the $n \times p$ system noise matrix G. y is an $m \times 1$ vector of measurements. These measurements are related to the states by the

$m \times n$ measurement model H. Finally, v is a $m \times 1$ column vector of measurement noises. For notational simplicity, all of the aforementioned matrices, F, G, H, Q, and R, are depicted as constants. However, any of them may vary with time.

A critical assumption of the KF is that w and v are assumed to be uncorrelated with each other, white, and with Gaussian (normal) probability distributions

$$p(w) \sim \; N(0, Q) \text{ and } p(v) \sim \; N(0, R) \tag{A.3}$$

where Q and R are the process and measurement noise covariance matrices respectively. The KF tracks estimates of the states x by maintaining the first two moments of the state distribution. These are the mean

$$\hat{x} = E[x], \tag{A.4}$$

and the variance

$$P = E[(x - \hat{x})(x - \hat{x})^T], \tag{A.5}$$

where P is commonly known as the error covariance matrix. As such, the state estimates returned by the KF have the probability distributions

$$p(x|z) \sim \; N(E[x], E[(x - \hat{x})(x - \hat{x})^T]). \tag{A.6}$$

The ultimate goal of the KF is to compute the best-fit or optimal estimate of the state distribution, or one that minimizes

$$||y_k - Hx_k||^2 \tag{A.7}$$

given the process (Equation (A.1)) and a measurement of it (Equation (A.2)) while taking into account their respective probabilities, Q and R. The KF does this in two phases, the process update and the measurement update.

The process update propagates the state distribution using the process model. It does this by first propagating the state vector using the process model,

$$x_{k+1}^{-} = Fx_k, \tag{A.8}$$

and then using Q to propagate the error covariance matrix,

$$P_{k+1}^{-} = FP_{k+1}F^T + GQG^T. \tag{A.9}$$

Note the superscript minus in $\hat{x}_{k+1}^{-}$ and P_{k+1}^{-}. The minus stands for *a priori*. The state vector and covariance matrix output by the process update are known as the *a priori* state estimate and error covariance matrix. These represent the process model's contribution to the optimal estimation process. Generally, Equation (A.9) uses Q to increase the uncertainty in the state distribution represented by P_{k+1}^{-}. This growth, if left unchecked, results in state estimates with infinite uncertainty. It is incumbent upon the next phase of the KF to bound this growth.

The second phase of the KF is known as the measurement update. As a measurement y_{k+1} is taken, the KF applies the a priori error covariance P_{k+1}^{-} and the measurement covariance R to compute the Kalman gain K_{k+1}. This is performed with the equation

$$K_{k+1} = P_{k+1}^{-}H^T(HP_{k+1}^{-}H^T + R)^{-1}. \tag{A.10}$$

The Kalman gain is the quantification of the relative weight between the process and measurement. A near-zero value of K indicates that the KF is giving very little weight to the measurement information. On the other hand, a high value of K indicates that the KF is giving a great deal of weight to the measurement. The weighting and incorporation of the measurement into the state estimate occurs in

$$\hat{x}_{k+1} = \hat{x}_{k+1}^{-} + K_{k+1}(y_{k+1} - H\hat{x}_{k+1}^{-}) \tag{A.11}$$

where $y_{k+1} - H\hat{x}_{k+1}^{-}$ is known as the measurement innovation. The result of Equation (A.11) is the *a posteriori* state estimate $\hat{x}_{k+1}$. Unlike the a priori quantities, they lack the minus superscript. The a posteriori state estimates represent the mean of the optimal state distribution that results from optimally weighting the process model and measurements. The final step in the measurement update is the adjustment of the error covariance with the measurement covariance. This is done by incorporating the Kalman gain with the a priori error covariance matrix. This is given by

$$P_{k+1} = (I + K_{k+1}H)P_{k+1}^{-}, \tag{A.12}$$

which gives the a posteriori error covariance matrix. By incorporating the measurement covariance into the error covariance matrix through the Kalman gain, Equation (A.12) serves to bound the uncertainty growth in P caused by Q.

A.2 The Error State Kalman Filter

For certain applications, it may be convenient to estimate the errors of the state rather than the state itself. The relationship between the two quantities is given by

$$\tilde{x} = a(\delta x, x) \tag{A.13}$$

where $\tilde{x}$ and δx represent the true states and the error states respectively, and a represents the relationship between the two. Note that a may be non-linear. This implementation is known as an Error State Kalman Filter (ESKF). ESKFs are potentially useful in situations where the full states do not fit the assumptions made by the standard KF. These situations include the full states being non-vector or having a non-linear process model. If the error quantities do fit these assumptions, then it may be easier to estimate them using an ESKF than to apply a more advanced filtering technique.

As with the standard KF, first step in the ESKF algorithm is to propagate the full states forward in time. This is given by

$$\delta\hat{x}_{k+1}^{-} = f(\delta\hat{x}_k). \tag{A.14}$$

Note the use of lowercase f. This signifies the potential non-linearity of the full state's transition equations. The next step is propagation of the error states forward in time. This is given by

$$\delta\hat{x}_{k+1}^{-} = F\delta\hat{x}_k, \tag{A.15}$$

and then using Q to propagate the error covariance matrix,

$$P_{k+1}^{-} = FP_{k+1}F^T + GQG^T. \tag{A.16}$$

Note that the error state transition matrix F is still required to have a linear relationship with the error states. A significant difference feature of the ESKF that δx is set and remains at zero until a measurement becomes available.

The ESKF's measurement update starts off similarly to that of the KF's. As a measurement comes in, its covariance R to compute the Kalman gain K_{k+1}. This is performed with the equation

$$K_{k+1} = P_{k+1}^{-}H^T(HP_{k+1}^{-}H^T + R)^{-1}. \tag{A.17}$$

The ESKF's measurement model is slightly different from the KF's. It is given by

$$\delta y_{k+1} = H\delta x_{k+1} + v_k. \tag{A.18}$$

This measurement model requires that the measurement error δy_{k+1} be used for the measurement update rather than the measurement itself. The measurement error is given by

$$\delta y_{k+1} = y_{k+1} - \hat{y}_{k+1} = y_{k+1} - H\hat{x}_{k+1}^{-}. \tag{A.19}$$

The measurement error is incorporated into the a priori error states by

$$\delta\hat{x}_{k+1} = \delta\hat{x}_{k+1}^{-} + K_{k+1}(y_{k+1} - H\delta\hat{x}_{k+1}^{-}). \tag{A.20}$$

Since $\delta\hat{x}_{k+1}^{-}$ is zero, this step is simply the application of a weight to the measurement innovation without actually incorporating it into the full states. The next step in the measurement update is the adjustment of the error covariance using the measurement covariance. This is done by incorporating the Kalman gain with the a priori error covariance matrix, which is given by

$$P_{k+1} = (I + K_{k+1}H)P_{k+1}^{-}. \tag{A.21}$$

The final step in ESKF is the incorporation of the error state estimates into the full states. This is given by

$$\hat{x}_{k+1} = a(\hat{x}_{k+1}^{-}, \delta\hat{x}_{k+1}^{-}). \tag{A.22}$$

After this step, the error state $\delta\hat{x}_{k+1}$ is reset to zero and the cycle is repeated.

At its core, the ESKF is extremely similar to the KF. The covariance propagation in both approaches are identical. The ESKF's weighting of the measurement, though notationally different than the KF's, is ultimately the same. This can be seen if Equation (A.19) is substituted into Equation (A.20). Making this substitution and taking into account the fact that $\delta\hat{x}_{k+1}^{-}$ equals zero yields

$$K_{k+1}(y_{k+1} - H\hat{x}_{k+1}^{-}), \tag{A.23}$$

which is the same as the KF's measurement innovation. A key difference between the ESKF and KF arises in the incorporation of the above product into the full state. Whereas the KF simply adds Equation (A.23) into the a priori state estimate, the ESKF uses a, a function that allow for any mathematical operation, including multiplication, quaternion products, and exponentiation. Another key difference lies in the propagation of the full states. As shown in Equation (A.14), the full state propagation may be governed by a non-linear function f, as opposed to the linear assumption made in the KF. Such is the strength of the ESKF. Unlike the KF, which assumes linear state propagation and updates, the ESKF allows both of these operations to be non-linear, allowing for a wider range of processes to be estimated.