UNIVERSITY OF CALIFORNIA
RIVERSIDE

Realtime, Decimeter Accuracy Navigation Using Sliding Window Estimator and Autonomous Vehicle Trajectory Tracking Control

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Sheng Zhao

December 2014

The Dissertation of Sheng Zhao is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

First and foremost, I would like to express my thanks to Prof. Jay A. Farrell for his utmost guidance, support, encouragement, and patience during my journey towards a Ph.D. He let me know how important it is to be rigorous as an engineer which is going to have a life-long influence on me.

Secondly, I would like to thank Prof. Anastasios Mourikis for sharing his code base and provide thoughtful suggestions for my research. I would like to thank Prof. Matthew Barth for his support on experimental vehicles so we can have hassle-free demos. I would like to also express my gratitude to Prof. Wei Ren, Prof. Christian Shelton, and Prof. Amit K. Roy-Chowdhury for their efforts and insights in teaching the graduate courses, from which I have learned a lot.

In addition, I would like to thank Anning Chen, Arvind Ramanandan and Anh Vu for making a great starting platform for us to work on, and for their continuous suggestions and help throughout the projects. I would like to thank Yiming Chen, Dongfang Zheng, Haiyu Zhang, Mingyang Li and Suvarna Sarath for their support and insightful discussion in my projects, and for their help in many project demo events. I would also like to thank my fellow graduate students at the robotics and computer vision group, Tue-Cuong Dong-Si, Hongsheng Yu, Xing Zheng, Rathavut (Biggie) Vanitsthian and Akshay More for making the lab a wonderful and memorable place to learn and work.

Finally, thanks to my wife and my daughter, for their endless and unconditional love, encouragement and support. Their supports help me through many difficult times and finally finishing my dissertation. I dedicate this dissertation to them.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

Realtime, Decimeter Accuracy Navigation Using Sliding Window Estimator and Autonomous Vehicle Trajectory Tracking Control

by

Sheng Zhao

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2014
Prof. Jay A. Farrell , Chairperson

This dissertation focus on achieving high accuracy navigation using low-cost sensors and on high precision trajectory tracking control for Vertical Take-Off and Landing (VTOL) Unmanned Aerial Vehicles (UAVs).

For high accuracy navigation, this dissertation presents a real-time sliding-window estimator to tightly integrate Differential-GPS (DGPS) and inertial measurement unit (IMU) to achieve reliable, high precision navigation performance in GPS-challenged urban environments using a low-cost, single-frequency (L1) GPS receiver. The approach is novel in that it utilizes the phase measurements, without resolving the integer ambiguity, to improve the accuracy and the robustness of the estimation results. Experimental results demonstrate that the performance of the proposed navigation system is significantly better than the extended Kalman Filter (EKF) (improved by one order of magnitude) and the novel usage of phase measurements further improves the robustness of the estimator to the pseudorange multipath error, which could otherwise be several meters in urban environments.

Regarding precision trajectory tracking, this dissertation presents a new command filtered backstepping technique for under-actuated VTOL UAVs. Quaternions are used to represent the attitude of the vehicle to ensure the global attitude tracking without singularities. Since the quaternions have their own unique algebra, they cannot be filtered by a vector-based command filter; therefore, a second-order quaternion filter is developed to filter the quaternion and automatically compute its derivative, which determines the commanded angular rate vector. A quadrotor vehicle is used as an example to show the performance of the proposed controller.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation focuses on the navigation and the trajectory tracking control of autonomous vehicles. The navigation system provides the realtime, accurate estimate of the vehicle states. The trajectory tracking system controls the vehicle such that the vehicle follows the designed path precisely at high dynamics.

1.1 Navigation

High precision navigation is a core functionality for autonomous driving, autonomous mowing robot [10], unmanned air vehicles (UAVs) [52, 62]. In many such systems, GPS is the sensor responsible for measuring the global position of the vehicle. With a clear view of the sky, a well-designed GPS receiver typically can achieve 3-8 meter positioning accuracy with the the U.S. Global Positioning System (GPS) Standard Positioning Service (SPS) [44]. To achieve reliable higher precision positioning, differential GPS (DGPS) is a standard approach. The user can either set up a base station on their own [10] or utilize publicly available correction services, such as Continuously Operating Reference Station (CORS) [51] and Nationwide Differential Global Positioning

System (NDGPS) [45]. As the mobile communication networks (4G or WiFi) becomes ubiquitous, the DGPS technique can be used in most urban environments and provides 1 to 3 meter positioning accuracy [31] using GPS pseudorange measurements. Using carrier phase Real-Time Kinematic (RTK) techniques allows centimeter accuracy after resolving the integer ambiguities in realtime [18, 8]; however, integer resolution is challenging for single frequency receivers [1].

However, GPS has its limitations. First, GPS is a low bandwidth sensor able to estimate position and velocity, it cannot provide high bandwidth estimates of the full vehicle state vector. Second, especially in urban environments, the GPS signals can be blocked by trees and tall buildings and thus using GPS alone cannot reliably provide accurate global positioning solutions. As an example of typical GPS coverage issues in urban environments, the satellite availability along the campus test trajectory is shown in Fig. 5.6. Despite these challenges, many applications require high-precision, high-bandwidth, full-state estimates for use in urban environments, for example, advanced driver assistance systems.

Dual frequency RTK GPS receivers are typically too expensive for low-cost/consumer-grade robotics applications and projects. In addition, in recent years, the single frequency (L1-only) GPS receivers have become readily available in the market at a very low prices compared to dual frequency receivers. In addition, MEMS IMU's, which can be used to track the ego-motion, are getting much cheaper, so much so that they can be found now in many consumer-grade devices (e.g. mobile phones). Such readily available low-cost sensors are enabling new applications, e.g. [10]. More applications will also become feasible as the high precision navigation solution becomes more affordable. This thesis focuses on the design of high precision state estimation using low-cost single-frequency GPS receivers and MEMS IMU.

RTK GPS positioning is a well-known mature technique [15]. However, for low-cost, single frequency receivers, there still are many challenges. The most serious is the inability to use the multi-frequency combinations [1, 20, 21] that greatly reduce the size of the relevant integer searching space. Moreover, the number of measurements from a single frequency receiver is half of that from a dual frequency receiver. Also, in urban environments, the intermittent signal reception caused by signal blockage necessitates frequent integer search, because every time the receiver reacquires the satellite signal, the integer in the carrier phase measurement is different and must be estimated again. Without the correctly resolved integer, the phase measurement does not provide absolute range information and the position accuracy is limited to the pseudorange accuracy cited above.

Article [61] utilized a sliding-window estimator with pseudorange measurements only. In an open sky environment, it demonstrated performance superior to the EKF. To further improve the performance, this thesis proposes a novel sliding window Bayesian estimation method allowing use of the phase measurements without resolving the integers. The results are demonstrated in GPS-challenged urban environments such as that shown in Fig. 5.6.

1.2 Control

Recently, Vertical Take-Off and Landing (VTOL) Unmanned Aerial Vehicles (UAVs) have gained tremendous interest among researchers and practitioners. In many applications, VTOL UAVs have advantages over fixed wing UAVs due to their relatively smaller size, capability to operate in cluttered environments and their hover capability. Many aerial vehicles fall into the categories of VTOL UAVs: Quadrotor, Coaxial rotorcraft and Ducted-fan UAV.

However, the control of VTOL UAV is not straightforward because of its under-actuated dynamics. The under-actuated dynamics require that translational motion of VTOL UAV be determined in part by the attitude of the vehicle. Since the translational motion of the VTOL UAV is determined by the attitude of the vehicle, the performance of the attitude tracking loop directly affects the performance of the position tracking loop. Euler-angles are commonly used in attitude control loop to represent the vehicle's attitude [37, 25]. However, the singularity problem of the Euler-angles prevents the controller to have the global tracking capability. Quaternion can represent the attitude without any singularity and thus a global attitude tracking controller can be implemented. In contrast to the rotation matrix representation, quaternion use fewer parameters (4 as opposed to 9) and quaternion is commonly used in the state estimators to represent the attitude. Hence, this thesis uses quaternion to represent the attitude.

Regardless of how attitude is represented, the vehicle position trajectory tracking problem through the vehicle attitude involves strong nonlinearities. In many applications [40, 25] and commercial products [2], VTOL UAV control is implemented by linearizing the dynamics around a hover operating point and designing linear (e.g., PID-like) controllers to control the position and attitude of the vehicle. Such control design approaches have good performance near the linearized point (hover stage), but may perform poorly when the vehicle deviates away from the linearized point, which typically happens when tracking an user-specified trajectory. The trajectory commands considered in this thesis can take two forms: 1) position only trajectory or 2) position and the desired yaw trajectory.

To improve the global VTOL tracking performance, various model-based nonlinear control methodologies can be considered. Among these nonlinear control methodologies, backstepping based design is widely adopted due to its systematic design and physically intuitive approach. In the backstepping design, the derivatives of virtual control signals are required in each design step. When the number of steps is greater than three, like the case of VTOL UAV, the analytic derivation of the derivatives becomes prohibitively complicated. When quaternions are involved in the control design, their special algebra and dynamics also complicates the design procedure. The paper [48] designs a quaternion-based backstepping controller to track a position trajectory and the required derivatives are computed analytically. The procedure is already very cumbersome for position tracking only. If the trajectory contains the desired yaw angle, all the derivatives computed in [48] need to be recomputed with respect to (w.r.t.) the desired yaw angle which makes the already tedious procedure even worse. There are many reasons that prevent the wide adoption of nonlinear controller in real applications, and the complexity involved in the design process is one. The command filtered implementation of the backstepping approach [16] maintains the desirable aspects of the backstepping method, has the same provable convergence properties, and simplifies the implementation process.

A command filtered backstepping trajectory tracking control approach for VTOL aircraft is presented and analyzed herein. The quaternion attitude representation requires extension of the approach represented in [16]. By exploiting the special dynamics of quaternion, this thesis proposes a second-order quaternion filter to compute the commanded quaternion and its angular velocity, without differentiation. The proposed quaternion filter enables the command filtered backstepping design for the many types of vehicles utilizing quaternion-based attitude representations. Moreover, with the use of command filters, the flexibility of giving yaw commands in the trajectory is realized without further complicating the design process.

1.3 Main Contributions

1. The first literature report of a high performance sliding window estimator on tightly coupled DGPS/IMU using L1-only measurements in GPS-challenged urban environments.

2. The first literature report of how to incorporate the phase measurement in the sliding window estimator to achieve high precision navigation without resolving the integer ambiguities.

3. Derive a novel 2D LIDAR aiding measurement model to aid the INS when the vehicle is in urban environments. With LIDAR aiding, the position error in the lateral direction of the road can be corrected even when satellites are shadowed by the buildings.

4. Design quaternion-based command filter backstepping control for quadrotor to achieve high trajectory tracking accuracy. To utilize the idea of command filters, a second-order quaternion filter is proposed to automatically generate the derivative of quaternion while respecting the uniqueness of quaternion dynamics.

Chapter 2

Inertial Navigation System

2.1 Coordinates and Notations

The coordinates used in this thesis:

1. **ECEF**: It stands for Earth-Centered, Earth-Fixed Cartesian coordinate. The center of ECEF is at the center of mass of the Earth. The z-axis is pointing towards the north but it does not coincide exactly with the instantaneous earth rotational axis. The x-axis intersects the sphere of the Earth at 0° latitude and 0° longitude. This means that ECEF rotates with the earth and therefore, coordinates of a point fixed on the surface of the earth do not change.

2. **Tangent Plane**: It is a local Cartesian coordinate defined by fitting a tangent plane to the geodetic reference ellipse at a point of interest. It is also referred as north, east, down frame or the global frame ($\{G\}$ frame) in this thesis.

The notation conventions in this thesis:

1. ${}^{b}_{a}\mathbf{R}$: denotes the rotation matrix $\mathbf{R}$ from $\{a\}$-frame to $\{b\}$-frame.

2. ${}^{a}\mathbf{v}$: denotes a vector $\mathbf{v}$ represented in $\{a\}$-frame.

2.2 Attitude Representations

The attitude of a rigid body can be represented by rotation matrix, quaternion and Euler angles.

2.2.1 Rotation Matrix

The rotation matrix $\mathbf{R}$ evolves in the special orthogonal group of degree three:

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^\top \mathbf{R} = \mathbf{R}\mathbf{R}^\top = \mathbf{I}_3, det(\mathbf{R}) = 1\} \tag{2.1}$$

2.2.2 Quaternion

This section review the basics of quaternion that are useful for navigation and control. The details of the quaternion algebra can be found in [56]. The notation of quaternion used in this thesis follows the JPL convention, instead of the Hamilton convention. A good reference about the discussion of the two notations is [50].

2.2.2.1 Unit Quaternion

Unit quaternion ($\bar{\mathbf{q}}$) uses four parameters to represent a rotation matrix. Such representation is globally non-singular. Unit quaternion is evolving in the three-sphere $\mathbb{S}^3$, embedded in $\mathbb{R}^4$, $\mathbb{S}^3 = \{\bar{\mathbf{q}} \in \mathbb{R}^4 \,|\, \bar{\mathbf{q}}^\top \bar{\mathbf{q}} = 1\}$. The quaternion is a unit quaternion if it satisfies:

$$|\bar{\mathbf{q}}| = \sqrt{\bar{\mathbf{q}}^\top \bar{\mathbf{q}}} = \sqrt{|\mathbf{q}|^2 + q_4^2} = 1 \tag{2.2}$$

where $\mathbf{q} = [q_1, q_2, q_3]^\top$. Usually, the unit quaternion is written as:

$$\bar{\mathbf{q}} = \begin{bmatrix} \mathbf{q} \\ q_4 \end{bmatrix} \tag{2.3}$$

$$= \begin{bmatrix} \hat{\mathbf{k}} \sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix} \tag{2.4}$$

In this notation, the unit vector $\hat{\mathbf{k}}$ describes the rotation axis and θ is the angle of rotation. Note that, one attitude configuration has two quaternion representations: $\bar{\mathbf{q}}$ and $-\bar{\mathbf{q}}$. They are differed by the rotating directions around the rotation axis to reach the target configuration.

2.2.2.2 Quaternion Algebra

We define the skew-symmetric matrix $\lfloor \mathbf{x} \times \rfloor$ for any $\mathbf{x} \in \mathbb{R}^3$ as

$$\lfloor \mathbf{x} \times \rfloor = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{2.5}$$

For the convenience, the skew-symmetric matrix is also denoted as $S(\mathbf{x})$ in this thesis. So for any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$, the cross product can be denoted as $\mathbf{x} \times \mathbf{y} = \lfloor \mathbf{x} \times \rfloor \mathbf{y}$. Also we have $\lfloor \mathbf{x} \times \rfloor^\top = \lfloor -\mathbf{x} \times \rfloor$.

The corresponding rotation matrix for a given quaternion is

$$\mathbf{R}(\bar{\mathbf{q}}) = (2q_4^2 - 1)\mathbf{I}_{3\times 3} - 2q_4 \lfloor \mathbf{q} \times \rfloor + 2\mathbf{q}\mathbf{q}^\top \tag{2.6}$$

$$= \mathbf{I}_{3\times 3} - 2q_4 \lfloor \mathbf{q} \times \rfloor + 2 \lfloor \mathbf{q} \times \rfloor^2 \tag{2.7}$$

The multiplication of two quaternions is defined as

$$\bar{\mathbf{q}} \otimes \bar{\mathbf{p}} = \begin{bmatrix} q_4 \mathbf{p} + p_4 \mathbf{q} - \mathbf{q} \times \mathbf{p} \\ q_4 p_4 - \mathbf{q}^\top \mathbf{p} \end{bmatrix} \tag{2.8}$$

The quaternion multiplication is distributive and associative but not commutative. The multiplication of quaternions is analogous to the multiplication of rotation matrix, in the same order:

$$\mathbf{R}(\bar{\mathbf{q}})\mathbf{R}(\bar{\mathbf{p}}) = \mathbf{R}(\bar{\mathbf{q}} \otimes \bar{\mathbf{p}}) \tag{2.9}$$

The inverse of quaternion is defined as

$$\bar{\mathbf{q}}^{-1} = \begin{bmatrix} -\mathbf{q} \\ \\ q_4 \end{bmatrix} \tag{2.10}$$

We define the identity quaternion as $\bar{\mathbf{q}}_o = [\mathbf{0} \;\; 1]^\top$. For any given quaternion $\bar{\mathbf{q}}$, we have

$$\bar{\mathbf{q}} \otimes \bar{\mathbf{q}}^{-1} = \bar{\mathbf{q}}^{-1} \otimes \bar{\mathbf{q}} = \begin{bmatrix} \mathbf{0} \\ \\ 1 \end{bmatrix} \tag{2.11}$$

From a given vector $\mathbf{p}$, we define its quaternion form as

$$\bar{\mathbf{q}}_{\mathbf{p}} = \begin{bmatrix} \mathbf{p} \\ \\ 0 \end{bmatrix} \tag{2.12}$$

The transformation of a vector from frame a to frame b using quaternion is given by:

$$^{b}\bar{\mathbf{q}}_{\mathbf{p}} = {}^{b}_{a}\bar{\mathbf{q}} \otimes {}^{a}\bar{\mathbf{q}}_{\mathbf{p}} \otimes {}^{b}_{a}\bar{\mathbf{q}}^{-1} \tag{2.13}$$

$$= \begin{bmatrix} {}^{b}_{a}\mathbf{R}\,{}^{a}\mathbf{p} \\ \\ 0 \end{bmatrix} \tag{2.14}$$

The corresponding formula for skew-symmetric matrix $\lfloor \mathbf{x} \times \rfloor$ is

$$\lfloor {}^{b}\mathbf{x} \times \rfloor = {}^{b}_{a}\mathbf{R} \lfloor {}^{a}\mathbf{x} \times \rfloor {}^{b}_{a}\mathbf{R}^\top \tag{2.15}$$

2.2.2.3 Quaternion Kinematics

Here we use $\{G\}$ to represent inertial frame and $\{B\}$ to represent body frame. For the angular velocity, we use $\boldsymbol{\omega}_{GB}^{B}$ to denote the angular velocity of body frame w.r.t. inertial frame expressed in body frame.

The kinematic equation for rotation ${}^{B}_{G}\mathbf{R}$ is:

$$
\begin{aligned}
{}^{B}_{G}\dot{\mathbf{R}} &= -\lfloor \boldsymbol{\omega}^{B}_{GB} \times \rfloor {}^{B}_{G}\mathbf{R} && (2.16) \\
{}^{B}_{G}\dot{\bar{\mathbf{q}}} &= \frac{1}{2}\bar{\mathbf{q}}_{\boldsymbol{\omega}^{B}_{GB}} \otimes {}^{B}_{G}\bar{\mathbf{q}} && (2.17) \\
&= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}^{B}_{GB}){}^{I}_{G}\bar{\mathbf{q}} && (2.18) \\
&= \frac{1}{2}\begin{bmatrix} \lfloor \mathbf{q} \times \rfloor + q_4 \mathbf{I} \\ -\mathbf{q}^{\top} \end{bmatrix} \boldsymbol{\omega}^{B}_{GB} && (2.19)
\end{aligned}
$$

where

$$
\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor \boldsymbol{\omega} \times \rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^{\top} & \mathbf{0} \end{bmatrix}
$$

The kinematic equation for rotation ${}^{G}_{B}\mathbf{R}$ is:

$$
\begin{aligned}
{}^{G}_{B}\dot{\mathbf{R}} &= {}^{G}_{B}\mathbf{R}\lfloor \boldsymbol{\omega}^{B}_{GB} \times \rfloor && (2.20) \\
{}^{G}_{B}\dot{\bar{\mathbf{q}}} &= \frac{1}{2}{}^{G}_{B}\bar{\mathbf{q}} \otimes \bar{\mathbf{q}}_{-\boldsymbol{\omega}^{B}_{GB}} && (2.21) \\
&= \frac{1}{2}\begin{bmatrix} \lfloor \mathbf{q} \times \rfloor - q_4 \mathbf{I} \\ \mathbf{q}^{\top} \end{bmatrix} \boldsymbol{\omega}^{B}_{GB} && (2.22)
\end{aligned}
$$

The derivation can refer to Section 2.6.1 of [19] for the dynamics of rotation matrix and Section 2.4 in page 16 of [56] for the dynamics of quaternion.

Let $\boldsymbol{\Phi} = \begin{bmatrix} \lfloor \mathbf{q} \times \rfloor + q_4 \mathbf{I} \\ -\mathbf{q}^{\top} \end{bmatrix}$, then we have $\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi} = \mathbf{I}_{3\times 3}$. The proof will make use of this property: $\lfloor \mathbf{x} \times \rfloor^2 = \mathbf{x}\mathbf{x}^{\top} - \|\mathbf{x}\|^2 \mathbf{I}$ (see eqn. (55) in [56]).

The attitude error is expressed in *error quaternion* form:

$$\delta\bar{\mathbf{q}} = \bar{\mathbf{q}} \otimes \hat{\bar{\mathbf{q}}}^{-1} \tag{2.23}$$

$$= \begin{bmatrix} \hat{\mathbf{k}}\sin(\delta\theta/2) \\ \cos(\delta\theta/2) \end{bmatrix} \tag{2.24}$$

$$\simeq \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix} \tag{2.25}$$

The corresponding rotation matrix is

$$\mathbf{R}(\delta\bar{\mathbf{q}}) \simeq \mathbf{I}_{3\times 3} - \lfloor \delta\boldsymbol{\theta} \times \rfloor \tag{2.26}$$

2.2.3 Euler Angles

In air force convention, Euler angles parameterize the rotation matrix ${}^{I}_{G}\mathbf{R}$ by three rotation angles: roll(ϕ), pitch(θ) and yaw(ψ). However, such minimum representation of rotation suffers from the singularity problem. The Euler angles represent the rotation ${}^{I}_{G}\mathbf{R}$ that rotates the global frame to the IMU frame and are followed by the right hand rule.

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.27}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.28}$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.29}$$

Hence we have:

$${}^{I}_{G}\mathbf{R} = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi) \tag{2.30}$$

2.3 Inertial Measurement Unit (IMU)

2.3.1 IMU Measurements

The IMU provides measurements of rotational velocity ($\boldsymbol{\omega}_m(t)$) and specific force ($\mathbf{a}_m(t)$), both are expressed in IMU frame:

$$\boldsymbol{\omega}_m(t) = {}^{I}\boldsymbol{\omega}_{GI}(t) + {}^{I}\mathbf{b}_g(t) + \mathbf{n}_g(t) \tag{2.31}$$

$$\mathbf{a}_m(t) = {}^{I}_{G}\mathbf{R}({}^{G}\mathbf{a}_{GI}(t) - {}^{G}\mathbf{g}) + {}^{I}\mathbf{b}_a(t) + \mathbf{n}_a(t) \tag{2.32}$$

where $\boldsymbol{\omega}_{GI}$ and $\mathbf{a}_{GI}$ are the angular rate and the acceleration vectors of the $\{I\}$-frame with respect to the $\{G\}$-frame, ${}^{G}\mathbf{g}$ is the gravity vector in $\{G\}$-frame, ${}^{I}\mathbf{b}_g(t)$ and ${}^{I}\mathbf{b}_a(t)$ are time correlated sensor errors that we will refer to as biases, and $\mathbf{n}_g(t)$ and $\mathbf{n}_a(t)$

represent white Gaussian measurement noise processes with power spectral densities (PSD) $\mathbf{Q}_a$ ($\frac{m}{s\sqrt{s}}$) and $\mathbf{Q}_g$ ($\frac{rad}{\sqrt{s}}$), respectively. The IMU sampling time is denoted as Δt.

2.3.2 IMU Biases Modeling

There are many ways of modeling the IMU biases $\mathbf{b}_g$ and $\mathbf{b}_a$ [47]. One way is to model the bias as a random walk process:

$$\dot{x} = n_w \tag{2.33}$$

where n_w is a Gaussian white noise. This thesis follows this modeling as it is simple. But the drawback of this modeling is that the uncertainty of biases will grow unbounded over time which is not true. Therefore, there is a better modeling which models the bias as a first-order Gaussian Markov process:

$$\dot{x} = -\lambda x + w \tag{2.34}$$

where $\lambda > 0$ is the time constant, $\beta = \frac{1}{\lambda}$ is the correlation time and w is a Gaussian white noise whose PSD is $Q_w = \sigma_w^2$. Denote the covariance of x to be P_x. Then the differential equation of P_x can be obtained by using Riccati differential equation:

$$\dot{P}_x = -2\lambda P_x + Q_w \tag{2.35}$$

The above equation can also be derived from the state transition equation of the state x. If the PSD of the noise is constant (Q_w is constant), then the covariance P_x has a steady state which is denoted as $\bar{\mathbf{p}}_x$. The $\bar{\mathbf{p}}_x$ is

$$\bar{\mathbf{p}}_x = \frac{Q_w}{2\lambda} \tag{2.36}$$

In contrast to the random walk model, this model results in a bounded uncertainty of the IMU biases.

2.3.3 IMU Noise Calibration

The common approaches used to calibrate the IMU noise parameters are listed below [47]:

1. Allen Variance (AV): uses band filter and perform in the time domain.
2. PSD: similar to AV but perform in the frequency domain.
3. Autocorrelation: first-order Gauss-Markov model.
4. Wavelet De-Noising: removes high frequency components, typically used together with autocorrelation to analyze low frequency signal components.

2.4 Inertial Navigation System (INS)

2.4.1 State Propagation

Let $\mathbf{x}(t) \in \mathbb{R}^n$ denote the rover state vector at time t:

$$\hat{\mathbf{x}} = \begin{bmatrix} {}^I_G\bar{\mathbf{q}}^\top & {}^G\mathbf{p}_I^\top & {}^G\mathbf{v}_I^\top & {}^I\mathbf{b}_g^\top & {}^I\mathbf{b}_a^\top \end{bmatrix}^\top \tag{2.37}$$

where ${}^G\mathbf{p}_I \in \mathbb{R}^3$ and ${}^G\mathbf{v}_I \in \mathbb{R}^3$ are the rover position and velocity represented in the global frame, ${}^I_G\bar{\mathbf{q}} \in \mathbb{R}^4$ is the quaternion that represents the rotation from the global frame to the IMU body frame, and ${}^I\mathbf{b}_g \in \mathbb{R}^3$ and ${}^I\mathbf{b}_a \in \mathbb{R}^3$ are the IMU gyro and accelerometer biases represented in the IMU frame.

The dynamics of rover state are modeled as

$$\dot{\mathbf{x}}(t) = \boldsymbol{f}(\mathbf{x}(t), \tilde{\mathbf{u}}(t), \mathbf{n}(t)), \tag{2.38}$$

where $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$, $\tilde{\mathbf{u}} \in \mathbb{R}^m$ is the vector of measured accelerations and angular rates and $\mathbf{n} \in \mathbb{R}^{12}$ is the vector of noises. The function $\boldsymbol{f}$ is

$$ {}^{G}\dot{\mathbf{p}}_I = {}^{G}\mathbf{v}_I, \tag{2.39} $$

$$ {}^{G}\dot{\mathbf{v}}_I = {}^{G}\mathbf{a}_{GI} \tag{2.40} $$

$$ {}^{I}_{G}\dot{\bar{\mathbf{q}}} = \frac{1}{2}\boldsymbol{\Omega}({}^{I}\boldsymbol{\omega}_{GI})\,{}^{I}_{G}\bar{\mathbf{q}} \tag{2.41} $$

$$ {}^{I}\dot{\mathbf{b}}_g = \mathbf{n}_{wg}, \tag{2.42} $$

$$ {}^{I}\dot{\mathbf{b}}_a = \mathbf{n}_{wa} \tag{2.43} $$

where ${}^{G}\mathbf{a}_{GI}$ and ${}^{I}\boldsymbol{\omega}_{GI}$ are computed from the IMU measurements as

$$ {}^{G}\mathbf{a}_{GI} = {}^{G}_{I}\mathbf{R}\left(\mathbf{a}_m - {}^{I}\hat{\mathbf{b}}_a - \mathbf{n}_a\right) + {}^{G}\mathbf{g} \tag{2.44} $$

$$ {}^{I}\boldsymbol{\omega}_{GI} = \boldsymbol{\omega}_m - {}^{I}\hat{\mathbf{b}}_g - \mathbf{n}_g. \tag{2.45} $$

and $\mathbf{n} = [\mathbf{n}_a\ \mathbf{n}_g\ \mathbf{n}_{\omega a}\ \mathbf{n}_{\omega g}]^\top$ with PSD matrix $\mathbf{Q} = diag([\mathbf{Q}_a, \mathbf{Q}_g, \mathbf{Q}_{\omega a}, \mathbf{Q}_{\omega g}])$.

Given a distribution for the state vector initial condition $\mathbf{x}(0) \sim \mathcal{N}(\hat{\mathbf{x}}(0), \mathbf{P}(0))$, an Inertial Navigation System (INS) propagates an estimate of the vehicle state between aiding measurement times as a solution of

$$ \dot{\hat{\mathbf{x}}}(t) = \boldsymbol{f}(\hat{\mathbf{x}}(t), \tilde{\mathbf{u}}(t), \mathbf{0}), \tag{2.46} $$

where $\hat{\mathbf{x}}(t)$ denotes the estimate of $\mathbf{x}(t)$. In this thesis, the 4th-order Runge-Kutta method is used for the numerical integration.

2.4.2 Error Dynamics

Due to initial condition errors, system calibration errors, and measurement noise, a state estimation error $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$ develops over time. The error state vector $\tilde{\mathbf{x}}$ is defined as

$$\tilde{\mathbf{x}} = \begin{bmatrix} {}^{G}\delta\boldsymbol{\theta}^\top & {}^{G}\tilde{\mathbf{p}}_I^\top & {}^{G}\tilde{\mathbf{v}}_I^\top & {}^{I}\tilde{\mathbf{b}}_g{}^\top & {}^{I}\tilde{\mathbf{b}}_a{}^\top \end{bmatrix}^\top \tag{2.47}$$

where the error angle vector ${}^{G}\delta\boldsymbol{\theta}$ represents the small angle rotation from the actual global frame $\{G\}$ to the estimated global frame $\{\hat{G}\}$ (${}^{\hat{G}}_{G}\mathbf{R} = \mathbf{I} - \lfloor {}^{G}\delta\boldsymbol{\theta}\times\rfloor$). Hence we have

$${}^{G}_{I}\hat{\mathbf{R}} = (\mathbf{I} - \lfloor {}^{G}\delta\boldsymbol{\theta}\times\rfloor){}^{G}_{I}\mathbf{R} \tag{2.48}$$

$${}^{I}_{G}\mathbf{R} = {}^{I}_{G}\hat{\mathbf{R}}(\mathbf{I} - \lfloor {}^{G}\delta\boldsymbol{\theta}\times\rfloor) \tag{2.49}$$

$${}^{G}_{I}\mathbf{R} = (\mathbf{I} + \lfloor {}^{G}\delta\boldsymbol{\theta}\times\rfloor){}^{G}_{I}\hat{\mathbf{R}}. \tag{2.50}$$

2.4.2.1 Deriving from Continuous Domain

To be used in the estimator, the error dynamics in the discrete form is desired. There are different ways of obtaining the discrete error dynamics. One way is starting from eqn. (2.38). Firstly, we linearize eqn. (2.38) at the estimated state $\hat{\mathbf{x}}$ to obtain the linearized differential equation for the INS error state in the continuous domain:

$$\dot{\tilde{\mathbf{x}}} = \mathbf{F}_c\tilde{\mathbf{x}} + \mathbf{G}_c\mathbf{n} \tag{2.51}$$

The linearization process and the matrices $\mathbf{F}_c$ and $\mathbf{G}_c$ are described in Section 11.4 in [19]. Then, the error state dynamic in the discrete domain can be derived from the above equation as:

$$\tilde{\mathbf{x}}_{k+1} = \boldsymbol{\Phi}_k\tilde{\mathbf{x}}_k + \mathbf{n}_k^d \tag{2.52}$$

where $\mathbf{\Phi}_k$ is the discrete-time state transition matrix and $\mathbf{Q}_k^d$ is the process noise covariance matrix computed from $\mathbf{F}_c$, $\mathbf{G}_c$, and the PSD matrix $\mathbf{Q}$:

$$\mathbf{\Phi}_k \triangleq \mathbf{\Phi}(t_{k+1}, t_k) = exp\left(\int_{t_k}^{t_{k+1}} \mathbf{F}_c(\tau)\mathrm{d}\tau\right) \tag{2.53}$$

$$\mathbf{Q}_k^d = \int_{t_k}^{t_{k+1}} \mathbf{\Phi}(t_{k+1}, \tau)\mathbf{G}_c\mathbf{Q}\mathbf{G}_c^\top\mathbf{\Phi}^\top(t_{k+1}, \tau)\mathrm{d}\tau. \tag{2.54}$$

Details can be found in Section 7.2.5.2 in [19].

The error covariance is propagated through time according to

$$\mathbf{P}_{k+1} = \mathbf{\Phi}_k\mathbf{P}_k\mathbf{\Phi}_k^\top + \mathbf{Q}_k^d \tag{2.55}$$

2.4.2.2 Deriving from Discrete Domain

Another way to derive eqn. (2.51) is directly from discrete rover state dynamic functions:

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}_k^d) \tag{2.56}$$

Linearizing the above equation can directly give us the error state dynamic in the discrete domain. The details about this approach can be found in [34]. In this thesis, we follow this approach. The $\mathbf{\Phi}_k$ can be obtained in a closed form as:

$$\mathbf{\Phi}_k = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{\Phi}_{\mathbf{q}\mathbf{b}_g} & \mathbf{0}_3 \\ \mathbf{\Phi}_{\mathbf{p}\mathbf{q}} & \mathbf{I}_3 & \Delta t\mathbf{I}_3 & \mathbf{\Phi}_{\mathbf{p}\mathbf{b}_g} & \mathbf{\Phi}_{\mathbf{p}\mathbf{b}_a} \\ \mathbf{\Phi}_{\mathbf{v}\mathbf{q}} & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{\Phi}_{\mathbf{v}\mathbf{b}_g} & \mathbf{\Phi}_{\mathbf{v}\mathbf{b}_a} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{2.57}$$

where

$$\boldsymbol{\Phi}_{\mathbf{pq}} = -\lfloor ({}^G\hat{\mathbf{p}}_{k+1} - {}^G\hat{\mathbf{p}}_k - {}^G\hat{\mathbf{v}}_k \Delta t - \frac{1}{2}\mathbf{g}\Delta t^2)\times\rfloor \tag{2.58}$$

$$\boldsymbol{\Phi}_{\mathbf{vq}} = -\lfloor ({}^G\hat{\mathbf{v}}_{k+1} - {}^G\hat{\mathbf{v}}_k - \mathbf{g}\Delta t)\times\rfloor \tag{2.59}$$

$$\boldsymbol{\Phi}_{\mathbf{qb}_g} = -\int_{t_k}^{t_{k+1}} {}^G_{I_\tau}\hat{\mathbf{R}}\, d\tau \tag{2.60}$$

$$\boldsymbol{\Phi}_{\mathbf{pb}_g} = \int_{t_k}^{t_{k+1}} \int_{t_k}^{w} \lfloor ({}^G\dot{\hat{\mathbf{v}}}_\tau - \mathbf{g})\times\rfloor \int_{t_k}^{\tau} {}^G_{I_s}\hat{\mathbf{R}}\, ds d\tau dw \tag{2.61}$$

$$\boldsymbol{\Phi}_{\mathbf{pb}_a} = -\int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau} {}^G_{I_s}\hat{\mathbf{R}}\, ds d\tau \tag{2.62}$$

$$\boldsymbol{\Phi}_{\mathbf{vb}_g} = \int_{t_k}^{t_{k+1}} \lfloor ({}^G\dot{\hat{\mathbf{v}}}_\tau - \mathbf{g})\times\rfloor \int_{t_k}^{\tau} {}^G_{I_s}\hat{\mathbf{R}}\, ds d\tau \tag{2.63}$$

$$\boldsymbol{\Phi}_{\mathbf{vb}_a} = -\int_{t_k}^{t_{k+1}} {}^G_{I_\tau}\hat{\mathbf{R}}\, d\tau \tag{2.64}$$

Chapter 3

Navigation Aiding Sensors

3.1 Differential-GPS

Two types of measurements are provided by GPS receivers: pseudorange (code) and carrier phase measurements. DGPS has advantages over stand-alone GPS in that the common-mode errors (e.g., ionosphere, troposphere, satellite clock and ephemeris errors) can be essentially removed by differencing measurements between the rover receiver and the GPS base station receiver. This step is referred as the single difference step. It is assumed in this paper that the (single difference) DGPS approach completely removes all common-mode errors. Receiver clock errors are not removed by DGPS. To avoid the modeling of the receiver clock error, which would involve two additional filter states, double differenced GPS measurements are considered.

The notation used follows several conventions:

- The overhead bar is used to denote the variable obtained from the single differencing step. For example, the single differenced pseudorange and phase measurements are denoted by $\bar{\rho}$ and $\bar{\phi}$, respectively.

- The superscript c on a variable is used to denote the common satellite chosen in the double differencing method.

3.1.0.3 Pseudorange Measurement

The double differenced pseudorange measurements for the i-th satellite vehicle (SV) at time t can be modeled as

$$\rho^{ic}(t) = \gamma^{ic}(\mathbf{x}(t)) + n_{\rho}^{ic}(t), \tag{3.1}$$

where $\gamma^{ic} = \bar{\gamma}^{i} - \bar{\gamma}^{c}$, $\bar{\gamma}^{i} = \|\mathbf{p}_r - \mathbf{p}_{sv}^{i}\|_2$ is the geometric distance between the vehicle antenna position $\mathbf{p}_r \in \mathbb{R}^3$ and the i-th SV antenna position $\mathbf{p}_{sv}^{i} \in \mathbb{R}^3$, $\rho^{ic} = \bar{\rho}^{i} - \bar{\rho}^{c}$, $\bar{\rho}^{i}$ is the single differenced pseudorange measurement of the i-th SV, $n_{\rho}^{ic} = \bar{n}_{\rho}^{i} - \bar{n}_{\rho}^{c}$, and $\bar{n}_{\rho}^{i}$ is the single differenced measurement noise. The double differenced noise standard deviation is typically $\sigma_{\rho} \in \sqrt{2} \cdot [0.1, 2.0]$ meter [41]. The noise $\bar{n}_{\rho}^{i}$ includes the multi-path error which can be several meters [53]. In this paper, the time correlation of the multi-path is ignored. It could be accommodated by augmenting additional states for each satellite. We leave that as a topic for future work.

Thus, the residual function of the double differenced code measurement can be formed from eqn. (3.1) as:

$$\mathbf{e}_{\rho}^{i}(\mathbf{x}(t)) = \rho^{ic}(t) - \gamma^{ic}(\mathbf{x}(t)). \tag{3.2}$$

The uncertainty of $\mathbf{e}_{\rho}^{i}(\mathbf{x}(t))$ is assumed to be Gaussian with the variance denoted by σ_{ρ}^{2}.

3.1.0.4 Carrier Phase Measurement

The double differenced phase measurement model for the i-th SV is:

$$\lambda\phi^{ic}(t) = \gamma^{ic}(\mathbf{x}(t)) + \lambda N^{ic}(t) + n_{\phi}^{ic}(t), \tag{3.3}$$

where $\phi^{ic} = \bar{\phi}^i - \bar{\phi}^c$ and $n_\phi^{ic} = \bar{n}_\phi^i - \bar{n}_\phi^c$, $\bar{\phi}^i$ is the single differenced phase measurement of i-th SV, N^{ic} is an unknown integer of the phase cycle, λ is the wavelength of the signal (19.05 cm for L1 signal), and $\bar{n}_\phi^i$ is the single differenced measurement noise. The phase noise n_ϕ^{ic} includes the multi-path error which has the standard deviation typically in the centimeter range [6]. Its time correlation is not modeled in this paper.

The integer N^{ic} is constant over time intervals when the receiver has phase lock for both SV's i and c. The receiver indicates this lock with a flag and lock time counter. When the unknown integer N^{ic} is resolved, the double differenced phase observable measures the range with centimeter accuracy ($\sigma_\phi \approx \sqrt{2} \cdot 0.02$ meter). Thus, the residual of the double differenced phase measurement after resolving the integer can be formed as:

$$\mathbf{e}_\phi^i(\mathbf{x}(t)) = \lambda(\phi^{ic} - N^{ic}) - \gamma^{ic}(\mathbf{x}(t)). \tag{3.4}$$

The uncertainty of $\mathbf{e}_\phi^i(\mathbf{x}(t))$ is assumed to be Gaussian with the variance denoted by σ_ϕ^2.

3.1.1 Integer-free Phase Measurement

3.1.1.1 Motivation

For a single-frequency receiver, the integer ambiguity is difficult to resolve reliably in realtime. Nonetheless, there are at least two reasons why we still want to use the phase measurements, even when the integers cannot be resolved:

1. Multi-path introduces only a few centimeters of error in the phase measurement while the code measurement can be affected by a few meters.

2. Phase measurements over a time window provide strong local kinematic constraints on the trajectory at the centimeter accuracy even when the correct integer cannot be resolved.

When the correct integer cannot be resolved, typically, there are two ways of utilizing the phase measurements:

1. The triple difference technique [11] creates an integer-free measurement: $\tilde{\boldsymbol{\phi}}_k^{ic}$ at t_k for i-th SV defined as:

$$\lambda\tilde{\boldsymbol{\phi}}_k^{ic} = \gamma^{ic}(\mathbf{x}(t_k)) - \gamma_{k-1}^{ic}(\mathbf{x}(t_{k-1})) + \tilde{n}_{\phi}^{ic}(t_k) \tag{3.5}$$

   where $\tilde{\boldsymbol{\phi}}_k^{ic} = \phi^{ic}(t_k) - \phi^{ic}(t_{k-1})$ and $\tilde{n}_{\phi}^{ic}(t_k) = n_{\phi}^{ic}(t_k) - n_{\phi}^{ic}(t_{k-1})$. This equation is derived by subtracting eqn. (3.3) at t_{k-1} from that at t_k. The benefit of this approach is that the resulting quantity $\tilde{\boldsymbol{\phi}}_k^{ic}$ is independent of the integer and thus it does not require integer estimation. However, the triple difference only considers two consecutive measurements, losing the strong time correlation between all the phase measurements (sharing a constant integer) over a time window. Note also that even when $n_{\phi}^{ic}(t)$ is white, $\tilde{n}_{\phi}^{ic}(t)$ is time correlated and that the measurement modeled in eqn. (3.5) depends on the state at two distinct times, which defies the standard EKF model.

2. Alternatively, the integers can be treated as real variables and estimated together with other vehicle states [3]. The major drawback of this approach is that the integer constraint is not respected in the estimation process, resulting in information loss. These approaches require 20 minutes for the real "integer" estimates to converge to single integer accuracy. Moreover, this approach requires adding all the integers into the estimator which increases the state vector size and thus increases the computational complexity.

To fully utilize the phase measurements when the correct integers cannot be resolved, this paper proposed a new way of using phase measurements over a CRT window. In the proposed framework, an integer-free phase measurement is constructed that is independent of the integer and that correctly captures the time correlation of phase measurements over a time window.

3.1.1.2 Phase Track

Let $\mathcal{C}_k = \{t_{k-M+1}, \cdots, t_k\}$ be the set of times in the present CRT window. Within $\mathcal{C}_k$ each satellite may have and lose phase lock several times. Let $\mathbb{T}^i_j \subseteq \mathcal{C}_k$ be the j-th set of times overwhich the i-th satellite is available and has phase lock. Fig. 3.1 illustrates an example of a phase measurement availability pattern within the CRT window. Let $\mathbb{T}^{ic}_j \subseteq \mathcal{C}_k$ be the j-th set of times overwhich the satellite pair i-c both have continuous phase lock. Note that $\mathbb{T}^{ic}_j \cap \mathbb{T}^{ic}_{j+1}$ is always an empty set. For all $t \in \mathbb{T}^{ic}_j$,

$$\lambda\phi^{ic}(t) = \gamma^{ic}(\mathbf{x}(t)) + \lambda N^{ic}_j + n^{ic}_\phi(t), \tag{3.6}$$

where N^{ic}_j is a constant integer. Define

$$\Xi^{ic}_j = \{\phi^{ic}(t) \mid t \in \mathbb{T}^{ic}_j\} \tag{3.7}$$

which will be referred to as a *phase track*. The length of the phase track Ξ^{ic}_j is the number of times in $\mathbb{T}^{ic}_j$. Fig. 3.2 illustrates the set of times $\mathbb{T}^{ic}_j$ in the phase track pattern Ξ^{ic}_j resulting from choosing the first SV as the common SV throughout the CRT window.

3.1.1.3 Integer-free Measurement

To fully utilize the phase measurement without resolving the integer vector, an integer-free measurement is constructed by adapting the technique proposed in [42].

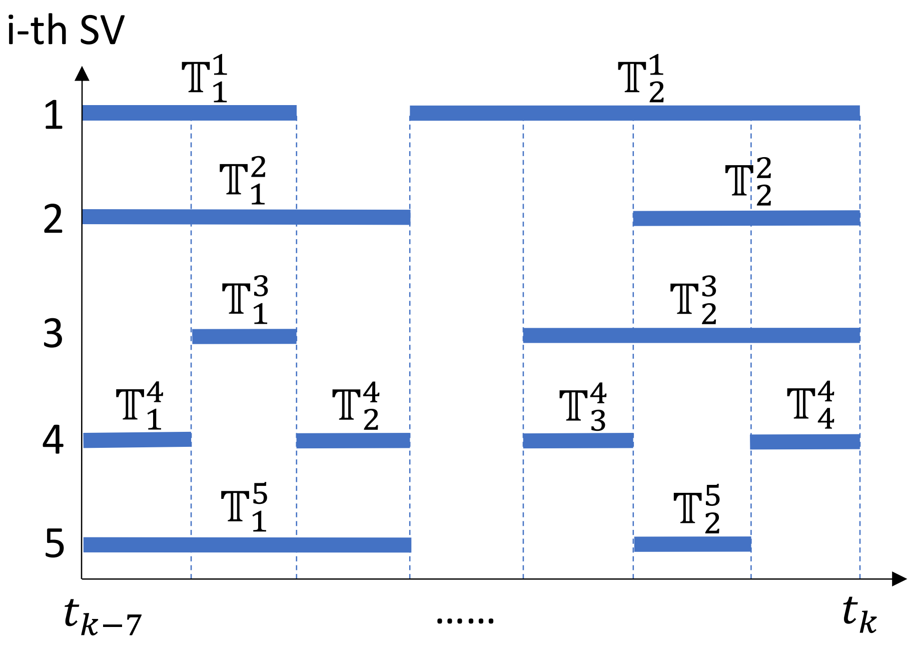


Figure 3.1: The illustration of the phase measurement availability pattern in a CRT window with $M = 8$. The y-axis is the SV ID number. The thick blue lines represents the set $\mathbb{T}_j^i$.

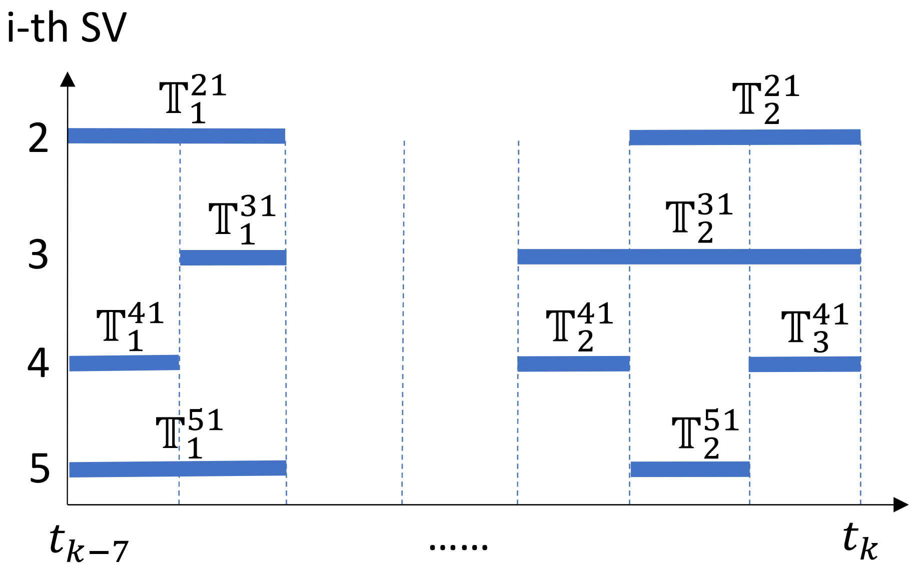


Figure 3.2: The illustration of the phase tracks obtained by choosing the first SV as the common SV for all time steps in the CRT window shown in Fig. 3.1. The thick blue lines represents the set $\mathbb{T}_j^{ic}$.

Starting from eqn. (3.6), for any phase track Ξ_j^{ic} that has length greater than one, stack all the double difference measurements from the phase track Ξ_j^{ic} to define a vector:

$$\lambda \boldsymbol{\phi}_j^{ic} = \mathbf{h}(\mathbf{X}_{\mathbb{T}_j^{ic}}) + \lambda \mathbf{G}_j^{ic} N_j^{ic} + \mathbf{n}_{\phi_j}^{ic} \tag{3.8}$$

where $\mathbf{X}_{\mathbb{T}_j^{ic}} = \{\mathbf{x}(t) \,|\, t \in \mathbb{T}_j^{ic}\}$ and

$$\boldsymbol{\phi}_j^{ic} = \begin{bmatrix} \phi^{ic}(\bar{t}_1) \\ \phi^{ic}(\bar{t}_2) \\ \vdots \\ \phi^{ic}(\bar{t}_l) \end{bmatrix}, \qquad \mathbf{h}(\mathbf{X}_{\mathbb{T}_j^{ic}}) = \begin{bmatrix} \gamma^{ic}(\mathbf{x}(\bar{t}_1)) \\ \gamma^{ic}(\mathbf{x}(\bar{t}_2)) \\ \vdots \\ \gamma^{ic}(\mathbf{x}(\bar{t}_l)) \end{bmatrix}, \tag{3.9}$$

$$\mathbf{G}_j^{ic} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \qquad \mathbf{n}_{\phi_j}^{ic} = \begin{bmatrix} n_\phi^{ic}(\bar{t}_1) \\ n_\phi^{ic}(\bar{t}_2) \\ \vdots \\ n_\phi^{ic}(\bar{t}_l) \end{bmatrix}. \tag{3.10}$$

The covariance of $\mathbf{n}_{\phi_j}^{ic}$ is a diagonal matrix $\mathbf{R}_\phi^{ic}$. In this paper, we ignore the correlation between satellites, so that $\mathbf{n}_\phi^{ic}$ is assumed to be uncorrelated with $\mathbf{n}_\phi^{dc}$ for $i \neq d$ to reduce the computational complexity (see Remark 1).

The integer can be eliminated from the equation by the following procedure. Select a unitary matrix $\mathbf{A}^i = [\mathbf{A}_1^i,\ \mathbf{A}_2^i]$ such that the columns of $\mathbf{A}_2^i$ form the basis of the left nullspace of $\mathbf{G}_j^{ic}$ (i.e., ${\mathbf{A}_2^i}^\top \mathbf{G}_j^{ic} = \mathbf{0}$). Multiplying ${\mathbf{A}_2^i}^\top$ on both sides of (3.8) gives:

$$\lambda \check{\boldsymbol{\phi}}_j^{ic} = \check{\mathbf{h}}(\mathbf{X}_{\mathbb{T}_j^{ic}}) + \check{\mathbf{n}}_{\phi_j}^{ic} \tag{3.11}$$

where $\check{\boldsymbol{\phi}}_j^{ic} = {\mathbf{A}_2^i}^\top \boldsymbol{\phi}_j^{ic}$, $\check{\mathbf{h}} = {\mathbf{A}_2^i}^\top \mathbf{h}$ and $\check{\mathbf{n}}_{\phi_j}^{ic} = {\mathbf{A}_2^i}^\top \mathbf{n}_{\phi_j}^{ic}$. Thus, the integer-free phase measurement induced residual equation is

$$\mathbf{e}^i_{\check{\boldsymbol{\phi}}_j}(\mathbf{X}) = \lambda \check{\boldsymbol{\phi}_j}^{ic} - \check{\mathbf{h}}(\mathbf{X}_{\mathbb{T}_j^{ic}}) \tag{3.12}$$

This integer-free phase residual function is used in eqn. (4.31) to obtain improved accuracy.

Note that eqns. (3.11–3.12) are independent of the integer. Eqn. (3.12) expresses the relative kinematic constraints between vehicle poses along the trajectory. The constraint is strong because the noise $\check{\mathbf{n}}_\phi^{ic} \sim \mathcal{N}(\mathbf{0}, {\mathbf{A}_2^i}^\top \mathbf{R}_\phi^{ic} \mathbf{A}_2^i)$ has component level standard deviations at the centimeter level with very strong cross-correlations that are known and correctly modeled. Therefore, the optimization will correctly accommodate the relative kinematic constraints between all the vehicle poses at the times in Ξ_j^{ic}. In contrast, the triple difference technique only captures the pairwise kinematic constraint between two consecutive vehicle poses, but neglects time correlation between subsequent measurements.

Remark 1 *Due to the double differencing procedure defined in eqn.* (3.3)*, the double difference noise terms* $n_\phi^{ic}(t)$ *and* $n_\phi^{dc}(t)$ *are correlated with each other:* $E(n_\phi^{ic}, n_\phi^{dc}) = \sigma_\phi^2$*. Therefore, the vectors* $\boldsymbol{n}_\phi^{ic}$ *and* $\boldsymbol{n}_\phi^{dc}$ *concatenating these quantities are also correlated. For computational reasons, in this paper, we neglect the correlation between* $\boldsymbol{n}_\phi^{ic}$ *and* $\boldsymbol{n}_\phi^{dc}$*.*

Remark 2 *It is easy to see that the triple difference measurement can be obtained from eqn.* (3.11) *by setting*

$$\boldsymbol{A}_2^\top = \begin{bmatrix} -1 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix} \tag{3.13}$$

and removing the off-diagonal elements in the covariance matrix of $\check{\boldsymbol{n}}_\phi^{ic}$ *(effectively ignoring the time correlation of the phase measurements). Ignoring of time correlation in the triple difference procedure yields a set of independent pairwise constraints instead of allowing window length constraints.*

3.1.1.4 Phase Tracks Construction

Let $\mathbb{S}_\Xi^c$ denotes the union of Ξ_j^{ic} over all i-s and j-s in the CRT window. Let $\mathbb{V}_k$ denotes the sequence of choices of common satellite vehicles for each time step in $\mathcal{C}_k$. The choice of $\mathbb{V}_k$ affects the structure of $\mathbb{S}_\Xi^c$. The problem of choosing the optimal set $\mathbb{V}_k$ can be formulated as an optimization problem [26]:

$$\mathbb{V}_k^* = \underset{\mathbb{V}_k}{\operatorname{argmax}} \det \left(\sum_{(i,j)\in\mathbb{S}_\Xi^c} \boldsymbol{\Lambda}_{ij}(\mathbf{X}) \right) \tag{3.14}$$

where $\boldsymbol{\Lambda}_{ij}$ is the information matrix formed by the phase track Ξ_j^{ic} using eqn. (4.21). In practice, it could be very difficult to solve the optimization problem in realtime to determine $\mathbb{V}_k$. Therefore, this section proposes an incremental approach to choose the common SV in the CRT window.

The incremental approach is described as follows: At time t_k, if $\mathbb{V}_k$ is available, then the common SV at t_k will remain the same, as long as that the SV is available.

When it becomes unavailable, the SV with the highest elevation is selected as the new common SV.

3.1.1.5 Marginalization of Integer-free Measurements

The marginalization approach of Section 4.3 can be used for the integer-free measurements. However, direct application of the marginalization described in Section 4.3 leads to a dense prior information matrix $\mathbf{\Lambda}_\alpha$ for the remaining trajectory $\mathbf{X}_r$. This leads to complications during optimization of eqn. (4.31) wherein the Jacobean cannot be relinearized for $\mathbf{x}(t_{k-M+2} \cdots \mathbf{x}(t_k)$, see [13]. It would also eliminate the ability to extend a phase track from $\mathcal{C}_k$ to $\mathcal{C}_{k+1}$.

Therefore, after obtaining all the phase tracks in the CRT window, they are divided into two sets: $\mathbb{S}_A$ and $\mathbb{S}_B$. The set $\mathbb{S}_A$ contains all the tracks that involve the rover states $\mathbf{x}(t_{k-M+1})$ that is going to be marginalized out at time t_k. The set $\mathbb{S}_B$ contains all the remaining tracks. For the phase tracks in $\mathbb{S}_B$, the integer-free residual function (3.12) is used.

For the phase tracks in $\mathbb{S}_A$, the corresponding integers are augmented into the vector $\mathbf{X}$ and estimated as real variables. Hence, the new CRT estimator state vector is defined as:

$$\mathbf{X} = \left[\mathbf{x}(t_{k-M+1})^\top \cdots \mathbf{x}(t_k)^\top \mid N_1 \cdots N_h\right]^\top \tag{3.15}$$

where N_s for $s = 1, \cdots, h$ are the integers corresponding to the phase tracks in $\mathbb{S}_A$. The residual function for phase tracks in $\mathbb{S}_A$ becomes:

$$\mathbf{e}^i_{\phi N}(\mathbf{x}(t), N^{ic}) = \lambda(\phi^{ic} - N^{ic}) - \gamma^{ic}(\mathbf{x}(t)). \tag{3.16}$$

In this method the marginalization can be carried out in the same manner shown in Section 4.3. The prior information matrix $\mathbf{\Lambda}_\alpha$ remains sparse and the full information matrix can be maintained properly even if the phase track extends to the next CRT window. Note that, if the length of a phase track becomes 0 after removing one state from the CRT window, the corresponding integer will also be removed from the vector **X** in the marginalization process.

3.1.2 Literature Review

3.1.2.1 GPS

It is well known that GPS carrier phase measurement can yield position accuracy at the centimeter level if the integer ambiguity can be resolved [20]. Integer ambiguity resolution is a well researched topic and there are many working solutions available, particularly for dual-frequency GPS receivers [54, 7]. When the integer vector is successfully resolved, an EKF is one estimation approach that could be implemented to obtain a centimeter accuracy INS solution [18]. The drawback of such systems is the high cost of the two-frequency receiver.

For a single-frequency GPS receiver, resolving the integer in realtime is much harder. One alternative is triple difference techniques in which the phase measurement is differenced between two consecutive times to eliminate the unknown integer. In [11] using triple differencing, the authors report achieving achieve submeter accuracy after 500 seconds of stationary operation. In [10], the authors proposed an integer search and validation method based on the technique developed in [11] for single frequency receivers. The time required to reliably resolve the integers was not reported. In [22], the authors utilize a modified triple difference technique in an EKF to track the relative positions of the receivers, when starting from a perfectly known initial configuration. The global positions are not estimated in this approach. Due to the nature of the tracking (perfect initial knowledge), it does not need a long time to converge to submeter accuracy. The reported accuracy of position tracking is in the decimeter level.

Most existing realtime algorithms that attempt to resolve the integers use a single epoch of GPS measurements. An alternative approach is to combine measurements from multiple measurement epochs [8] to increase the measurement redundancy. The existing multiple epoch approaches do not constrain the state vector estimates across the multiple times. Here we present a novel algorithm wherein the IMU measurements and system kinematics provide strong constraints on the state vector estimates across multiple measurement times, the integers are not resolved, yet decimeter accuracy is achieved.

The method proposed in this paper is related to, but distinct form, the triple difference technique. Instead of differencing two consecutive measurements to eliminate the integer and implementing an EKF, we propose a systematic method to eliminate the integer for a time window of measurements. Our approach utilizes both GPS carrier phase and inertial kinematic constraints between all the vehicle poses in the window. The proposed method is optimal in that it preserves all the information from the measurements and correctly captures the time correlation of the resulting kinematic constraints.

3.1.2.2 GPS/IMU Integration

GPS and IMU integration has been extensively studied [18]. For sensor aided inertial navigation system (INS), the most commonly used estimator is the extended Kalman filter (EKF). Fusion techniques can be roughly divided into two categories: loosely coupled and tightly coupled.

Loosely coupled approaches use the position and velocity information computed within the GPS receiver as measurements to the EKF to compute corrections for the INS state vector. As the position and velocity measurements from the GPS receiver are computed using only GPS measurements, at least four satellites are necessary. When the number of satellites falls below four, the GPS receiver cannot provide position and velocity measurements; therefore, the INS error cannot be corrected. Hence this approach will perform poorly in environments where satellite reception is poor. Moreover, the loosely coupled approach ignores correlation between the elements of the GPS estimated position and velocity measurement vectors, yielding sub-optimal INS corrections.

Tightly coupled approaches use the GPS measurements (pseudorange and carrier phase) directly to correct the INS error. As few as one satellite can be used to correct a subspace of the INS state if the clock is estimated (at least two satellites are required if the clock is not estimated). The tightly coupled approach is more reliable than the loosely coupled approach, especially when GPS signals may be partially blocked by the surrounding environment. As the tightly coupled approach utilizes the GPS measurements directly, there is no information lost in sensor fusion and thus the navigation performance is typically better than the loosely coupled approach. The approach described in this paper follows the tightly coupled approach.

3.1.2.3 Estimator

To account for the strong nonlinearities present in the navigation system, this paper utilizes a realtime sliding window estimator to achieve better performance. Related algorithms have received significant attention in the SLAM community [12, 29, 9, 13, 27]. However, none of these papers report the performance for tightly coupled

DGPS/IMU systems. The most closely related approach is [27]. However, this paper use a simplified GPS model that is integrated in a loosely coupled way. Therefore, they reported similar performance between their approach and an EKF. The approach herein will demonstrate a significant performance improvement compared to the EKF, especially in GPS-challenged urban environments. The navigation system designed in this paper is similar in concept to the one proposed in [9, 13]. However, [9, 13] are focused on the Vision/IMU integration and they have not reported any tightly coupled GPS/IMU results. To the best of our knowledge this is the first report of high performance of a realtime sliding window tightly coupled DGPS/IMU estimation system.

3.2 LIDAR

3.2.1 LIDAR Sensor

A LIDAR sensor is an optical sensing device that uses one or more laser beams to determine the range and azimuth angle to objects in the environment. The $k-th$ scan of the LIDAR at time t_k returns a set of measurements where is the angle measurement and R is the range measurement. This type of ranging sensor is based on time-of-flight principles. The range sensors not only provide high accuracy distance measurements, they also achieve a high angular resolution due to the very small divergence of the emitted laser pulse. LIDAR sensors are commonly utilized as part of vehicle navigation systems for detecting surrounding vehicles, obstacles, and roadway infrastructure such as curbs. It can also be used as part of a vehicle localization solution, either as a single sensor or be combined with other sensors such as GPS and Inertial Measurement Units (IMU).

There are different kinds of LIDAR in the market. The most common and cheapest one is 2D LIDAR which only scan in a single plane, providing a high number of measurements across a large field-of-view (e.g., 180 degrees). As an example, one of the most popular 2D LIDAR sensors is the SICK LMS series. A SICK LIDAR (LMS200) operates at distance up to 80m with an angular resolution of 0.5° and a range measurement accuracy of typically $\pm 5cm$ (1σ value). The distance between the sensor and an object is calculated by measuring the time interval between an emitted laser pulse and reception of the reflected pulse. The amplitude of the received signal is used to determine the reflectivity of the object surface. Fig. 6.2 (a) and (b) illustrate the SICK LMS200 LIDAR and its operating principle.

As another example of the 2D LIDAR, the HOKUYO UXM-30LN LIDAR is a single planar range sensor designed for intelligent robots and vehicles. Its detection range is up to 60m, and the horizontal field of view is 190°. The range accuracy is 30mm when the ranging distance is less than 10m, and 50mm when the ranging distance is between 10 to 30m. The angular resolution is 0.25°. This device is shown in Fig. 6.2 (c). As an extension of 2D LIDAR, the multi-planar LIDAR splits the laser beam into different vertical planes in order to detect more objects. As an example, the ALASCA XT LIDAR has aperture angle to be 3.2°. Its distance range is up to 200 meters, and the horizontal field-of-view is 240° . Fig. 6.2 (d) shows the IBEO sensor.

The most expensive LIDAR is the 3D LIDAR which can scan a full 360 degree environment. The most popular 3D LIDAR is the Velodyne HDL-64E LIDAR and it was specifically designed for autonomous vehicle navigation. With horizontal by vertical field of view, 0.09 degree angular resolution, and 5-20Hz rotation rate, the Velodyne provides surrounding 3-D environmental information with high accuracy ($< 2cm$ resolution). The detection range is up to 100 meter, and the latency is less than 0.05 milliseconds. Figure

Table 3.1: Different LIDAR Comparison

	2D LIDAR	Multi-planar LIDAR	3D LIDAR
Sense range:	Single plane	Multi-plane	360 degree
Detectable feature:	2D feature	2D feature	3D feature (plane)
Cost:	Low	Medium	High
Size:	Small	Small-Medium	Big

6.2 (e) and (f) demonstrate the Velodyne HDL-64E LIDAR and its 3-D range data. All these LIDAR mentioned above are capable of detecting features in the environment hence all of them can potentially be used to aid the vehicle positioning. However, among them, 2D LIDAR is small, lightweight and low-cost. Hence, the 2D LIDAR is selected as the aiding sensor among these different kinds of LIDAR sensors. The comparison among these LIDAR sensors is given in Table 3.1.

3.2.2 LIDAR Aided INS

Accurate vehicle positioning is an essential requirement for next generation intelligent transportation systems. GNSS aided INS is a standard technique that has been widely adopted to provide accurate vehicle position [15, 17, 14]. The drawback of the GNSS based positioning is that when GNSS signals are shadowed (as in urban environments), the positioning accuracy degrades at a rate determined by the IMU quality. This rate can be rapid for MEM's devices. Without accurate positioning, the performance of other position-dependent vehicle applications, like lane-departure warning and route planning are also be affected. Therefore, there is interest in sensors other than GNSS to aid INS. Camera, LIDAR and RADAR are all potential sensors that can improve INS performance. This paper addresses LIDAR aiding of INS to improve vehicle positioning accuracy. In the commercial market, there are 2D planar and 3D LIDAR's. The 3D LIDAR is able to obtain the 3D point cloud of the surrounding

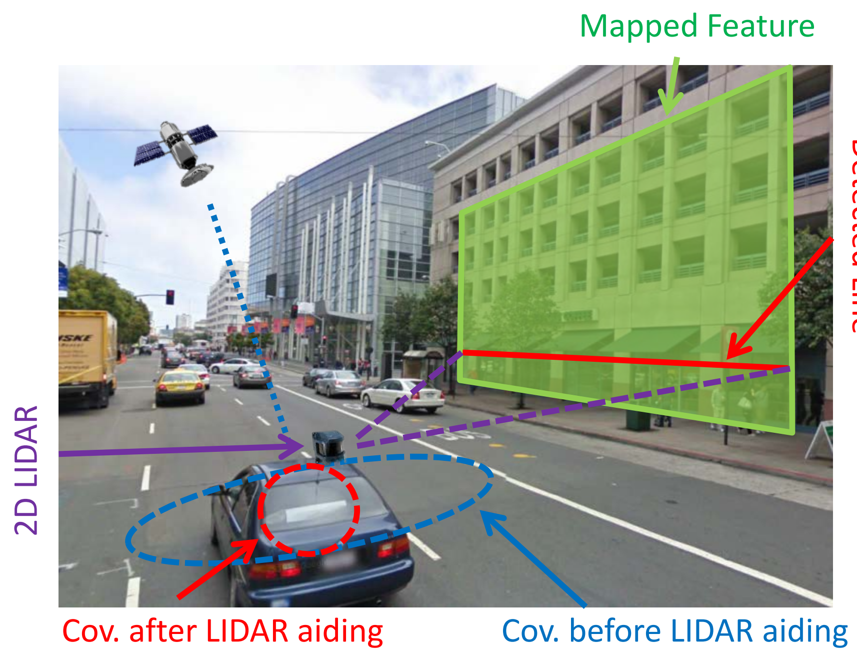


Figure 3.3: The typical use of 2D LIDAR aiding for vehicle positioning in urban environments. The uncertainty in the longitude direction can be corrected by GPS, while the LIDAR measurements can provide corrections in the lateral direction where the GPS signals are blocked. Together, LIDAR and GPS can reduce estimated position uncertainty to lane-level accuracy. Note, in practice, the detected line is usually broken into multiple line segments due to occluding objects (e.g., light poles, trees).

environment, from which the 3D features can be extracted. However, the high cost of 3D LIDAR prohibits the mass deployment in personal vehicles. In contrast, the 2D LIDAR is small and lightweight with rapidly declining cost. Hence, 2D LIDAR aiding is the focus of this paper. The concept of LIDAR aiding, as discussed in this paper, is illustrated in Fig. 3.3.

The use of 2D LIDAR in robot localization has been extensively investigated by various authors [55, 4]. In some existing approaches, the robot pose is tracked either by matching sequential LIDAR measurements or by matching the most recent LIDAR measurement with a 2D map. However, these approaches assume that the robot only moves in a plane. Unfortunately, the planar motion assumption does not apply to the vehicles driving on roads. Due to bumps, road inclination, and the loaded suspension of the vehicle, the vehicle and sensor motion define 3D trajectories. Motion estimation in 3D is challenging, requiring estimation of the position, velocity, attitude and sensor calibration vectors in three dimensions.

The aiding approach taken in this thesis falls into the category of *feature-based navigation* wherein the raw LIDAR data is processed to detect certain 2D shapes (e.g., lines) that correspond to 3D features (e.g., planes). We assume that a map of the features is known a priori. The detected shapes are then associated with the mapped features to form the measurement residuals. The measurement residuals are then fed into an Extended Kalman Filter (EKF) to estimate the state of the vehicle. Since the motion of vehicles is 3D, associating a 2D shape to a 3D feature is difficult, as it requires the prediction of the nature of the curve of intersection between the mapped feature and the LIDAR x-y detection plane.

The features that are commonly used in *feature-based navigation* are created by the intersection of the LIDAR x-y plane with common shapes: points (e.g. corners), arcs of ellipses (e.g. cylinder created by trees or poles) and lines (e.g. planes sides of buildings). Since buildings are common in the urban environment, large, and unmoving, they can be easily and reliably detected from 2D LIDAR data, this paper focuses on planar features for LIDAR aiding.

In summary, this thesis develops a complete solution that enables the use of 2D LIDAR in aiding the 3D state estimate of a vehicle driving in the urban environment. Specifically, for the purposes of feature association, residual formation and GUI display, this paper derives an exact closed-form prediction of the location of the line-of-intersection measurement in the LIDAR frame. Due to the simplicity of this formula, feature association and residual formation can be easily solved. Moreover, being able to display the predicted line-of-intersections in the LIDAR's frame makes it easy to visualize the residual and debug the code.

3.2.3 Related Work

Feature-based navigation is a very active research area [59, 36]. Cameras are popular due to their low cost and capability to capture 3D features. For aiding purposes, cameras are effectively treated as angle sensors. The performance of camera image processing is not robust to the lighting conditions and is computationally intense.

Compared to cameras, LIDARs are active sensors that are significantly more robust to lighting conditions. Each measurement returns angle, range, and reflection intensity. Utilization of 2D LIDAR in localization has a long history in the robotics community [43, 55]. There are two dominant ways to use LIDAR data: raw point measurements processing [46] and high-level features processing [4]. Often, such work restricts the motion to 2D. Recently, there is a paper using 2D LIDAR aiding IMU to estimate the 3D state of an aggressively flying UAV for an indoor environment [5]. The approach in [5] uses point measurements directly and relies on a 3D occupancy map of the operating environment. This approach would be difficult to extend to outdoor applications because the size of the 3D occupancy map grows cubically in the outdoor environment.

The most closely related work to this paper is [23] which uses 2D LIDAR IMU aiding to estimate the 3D position of a person walking inside a building. The present paper differs from [23] in the formation of residuals, and residual error model equations. In [23], they use geometric constraints to form the residuals, resulting in residual equations that are nonlinear to the line measurement (ϕ and ρ, see Section 3.2.4) noise components. Herein, the measurement residuals are formed directly resulting in equations that are linear in the line measurement noise components. In addition, the residual formulation herein is more natural – the error between the measured line and the predicted line. Hence, the measurements, predicted measurements and residuals can be easily visualized for a GUI and for debugging.

3.2.4 LIDAR Measurement Processing

The vehicle is assumed to be operating in a known environment where certain plane features have been previously mapped. The planes may represent, for example, signs or sides of buildings. The map database is known *a priori* and stored onboard the vehicle. For the i-th plane feature, we store two parameters: ${}^G\boldsymbol{\pi}_i \in \Re^3$ and ${}^G d_i \in \Re^+$. Hence, we have a library of mapped plane features, following the notation in [23] (see Fig. 3.4):

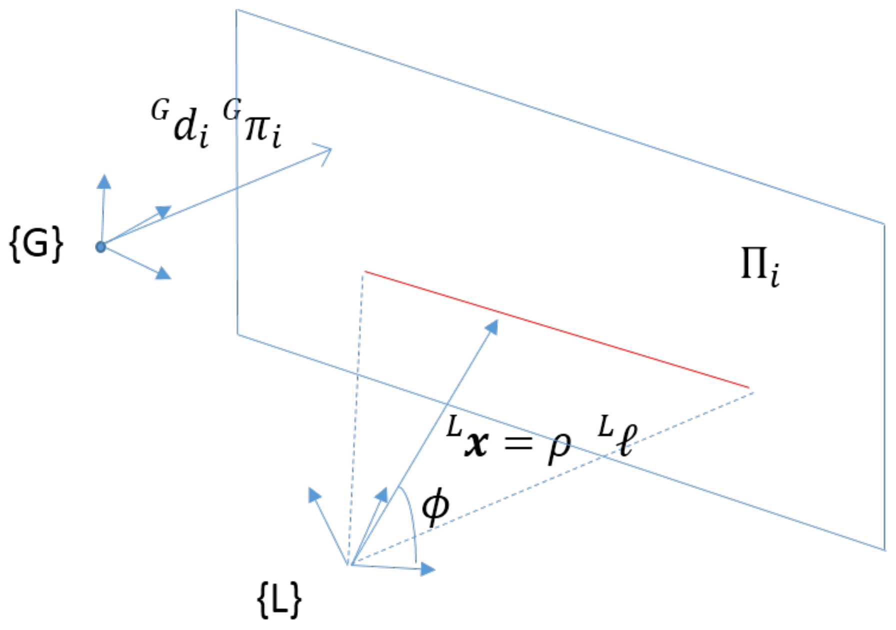
$$ {}^G\Pi = \{ {}^G\boldsymbol{\pi}_i, {}^G d_i \}_{i=1,\ldots,N_\pi}. $$

The set

$$ \Pi_i = \{ {}^G\mathbf{x} \in \Re^3 | {}^G\boldsymbol{\pi}_i \cdot {}^G\mathbf{x} = {}^G d_i \} $$

represents the i-th 2D-plane feature in the $\{G\}$-frame where ${}^G\boldsymbol{\pi}_i$ is the unit normal vector to the plane and ${}^G d_i$ is the shortest distance to the plane from the $\{G\}$-frame origin.

The intersection, if it exists, between the LIDAR measurement plane (i.e., x-y) and any other plane is a line (see Fig. 3.4). Such lines must be detected and tested for association with planes in the feature library. When a detected line is associated with a



${}^{G}d_i$ ${}^{G}\pi_i$
{G}
Π_i
${}^{L}\mathbf{x} = \rho\ {}^{L}\boldsymbol{\ell}$
ϕ
{L}

Figure 3.4: The plane feature and LIDAR line measurement representations.

plane in the feature library, then 2D LIDAR based state correction is possible using an appropriately formed residual between the two lines.

The k-th scan of the LIDAR at time t_k returns a set of data $\mathcal{D}_k = \{(\theta_i, R_i)\}_{i=1}^{N_l}$ where for the Hokuyo LIDAR, $N_l = 760$, $\theta_i \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and R_i is the range measurement. This section discusses the processing of each LIDAR data scan $\mathcal{D}_k$ to extract lines, to associate extracted lines with plane features to form line measurements, to compute the covariance of line measurements, to form the residual measurement between each predicted and extracted line, and to aid IMU in an EKF framework.

Any line in the LIDAR x-y plane that does not pass through the origin is uniquely defined by the shortest vector from the origin of $\{L\}$-frame to the line. The shortest vector is represented as ${}^{L}\mathbf{x} = \rho\ {}^{L}\boldsymbol{\ell}$, where ${}^{L}\boldsymbol{\ell} = [\cos\phi \quad \sin\phi \quad 0]^\top$ is a unit vector and ρ is the magnitude of ${}^{L}\mathbf{x}$. Because the line-of-intersection cannot pass through the LIDAR origin, it can be represented by two parameters: ϕ and ρ. Let $\boldsymbol{\chi} = [\phi\ \rho]^\top$.

In this paper, we assumes all the points in one LIDAR scan are taken simultaneously. This assumption is reasonable at low speeds. At higher speeds, the method can be extended, using the IMU data, to compensate for vehicle motion.

3.2.4.1 Measurement Prediction

For various purposes (e.g., association of detected lines with mapped planar features, measurement residual formation, and graphical display), it is useful to have formulas to compute $\hat{\boldsymbol{\chi}}_i = [\hat{\phi}_i,\ \hat{\rho}_i]^\top$, when Π_i, ${}^G\hat{\mathbf{p}}_I$, ${}^G_I\hat{\mathbf{R}}$ and the LIDAR extrinsic calibration parameters ${}^I_L\mathbf{R}$, ${}^I\mathbf{p}_L$, are given.

The problem can be solved in an optimization framework with two constraints. The first constraint, that the vector ${}^L\mathbf{x}_i$ must be in the LIDAR x-y plane, is

$$ {}^L\mathbf{z}_L \cdot {}^L\mathbf{x}_i = 0 \tag{3.17} $$

where ${}^L\mathbf{z}_L = [0\ 0\ 1]^\top$. The second constraint is that the end of the vector ${}^L\mathbf{x}_i$ must be on the plane Π_i:

$$ ({}^L_G\mathbf{R}\,{}^G\boldsymbol{\pi}_i) \cdot {}^L\mathbf{x}_i = {}^L\hat{d}_i \tag{3.18} $$

where ${}^L\hat{d}_i = {}^Gd_i - {}^G\boldsymbol{\pi}_i \cdot ({}^G\hat{\mathbf{p}}_I + {}^G_I\hat{\mathbf{R}}\ {}^I\mathbf{p}_L)$. The problem is to find the shortest vector ${}^L\hat{\mathbf{x}}_i$ satisfying constraints (3.17) and (3.18):

$$ {}^L\hat{\mathbf{x}}_i = \arg\min_{{}^L\mathbf{x}_i}(\|{}^L\mathbf{x}_i\|^2) \tag{3.19} $$

$$ s.t.\ \text{eqn. (3.17) and (3.18).} \tag{3.20} $$

The closed form solution is (see the remark)

$$ \hat{\phi}_i = \arctan\left(\operatorname{sgn}({}^L\hat{d}_i)\frac{a_2}{a_1}\right), \quad \hat{\rho}_i = \frac{|{}^L\hat{d}_i|}{\sqrt{a_1^2 + a_2^2}} \tag{3.21} $$

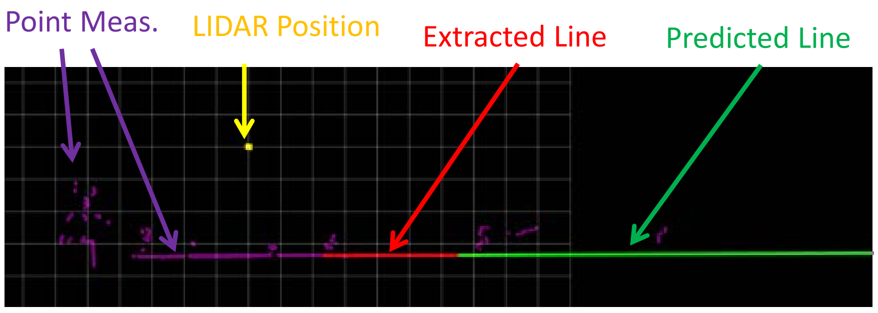


Figure 3.5: LIDAR's GUI showing raw measurements (purple dots), the extracted line (red) and the predicted line (green).

where ${}^{L}\mathbf{a}_i = [a_1\ a_2\ a_3]^\top$ and ${}^{L}\mathbf{a}_i = {}^{L}_{G}\mathbf{R}^{G}\boldsymbol{\pi}_i$. The computed variable $\hat{\boldsymbol{\chi}}_i = [\hat{\phi}_i\ \hat{\rho}_i]^\top$ allows prediction of the line-of-intersection in the LIDAR frame, which is required both for aiding and for a LIDAR frame GUI. The LIDAR GUI is shown in Fig. 3.5.

Remark 3 *This section presents a derivation of eqns. (3.21) which are a closed-form solution to the optimization problem of eqn. (3.19).*

Let $\boldsymbol{a} = {}^{L}_{G}\boldsymbol{R}^{G}\boldsymbol{\pi}_i = [a_1\ \ a_2\ \ a_3]^\top$ *which satisfies* $\|\boldsymbol{a}\|^2 = a_1^2 + a_2^2 + a_3^2 = 1$. *The solution procedes as follows:*

$$(\boldsymbol{A}\boldsymbol{A}^\top)^{-1} = \frac{\begin{bmatrix} 1 & -a_3 \\ -a_3 & 1 \end{bmatrix}}{1 - a_3^2}. \tag{3.22}$$

Then, we have

$${}^{L}\boldsymbol{p}_L = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{b} \tag{3.23}$$

$$= \begin{bmatrix} a_1 \\ a_2 \\ 0 \end{bmatrix} \frac{\hat{d}_L}{a_1^2 + a_2^2}. \tag{3.24}$$

Therefore, we have

$$\hat{\rho} = \frac{|\hat{d}_L|}{\sqrt{a_1^2 + a_2^2}} \tag{3.25}$$

$$\hat{\phi} = atan\left(sgn({}^L\hat{d})\frac{a_2}{a_1}\right). \tag{3.26}$$

3.2.4.2 Line Fitting

This section discusses the method to obtain line measurements, represented by $\boldsymbol{\chi}$, from $\mathcal{D}_k$ and how to associate mapped features to the line measurements.

The starting point is a set of 2D points ${}^L\mathbf{p}_i = [x_i \;\; y_i]^\top, \;\; i = 1 \ldots n$ extracted from the k-th LIDAR data scan $\mathcal{D}_k$ and represented in $\{L\}$-frame that are assumed to be associated with a line. This set of points could be found, for example, by the Hough transform or the *Split-and-Merge* algorithm [43]. The LIDAR raw data R_i and θ_i for the i-th point are related to the $\{L\}$-frame rectangular coordinates by ${}^L\mathbf{p}_i = R_i[\cos\theta_i \;\; \sin\theta_i]^\top$. We define the range and angle measurement noise to be $\bar{\mathbf{n}} = [n_R \; n_\theta]^\top$, and assume n_R and n_θ are uncorrelated zero mean Gaussian noises with standard deviation σ_R and σ_θ respectively. The covariance of $\bar{\mathbf{n}}$ is

$$\mathbf{P}_{\bar{\mathbf{n}}} = \begin{bmatrix} \sigma_R^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}. \tag{3.27}$$

If the set of points was exactly on the line $\boldsymbol{\chi}$, then they would each satisfy the equation

$$0 = {}^L\mathbf{N} \cdot {}^L\mathbf{p}_i - \rho \tag{3.28}$$

which is equivalent to $\bar{\alpha}x + \bar{\beta}y = \rho$. For our set of points $\{{}^L\mathbf{p}_i\}_{i=1}^n$ the distance ρ and the unit vector ${}^L\mathbf{N}^\top = [\bar{\alpha} \; \bar{\beta}] = [\cos\phi \;\; \sin\phi]$ are unknown. In addition, the point locations, are computed from the noise corrupted raw LIDAR data.

For any hypothesized $\boldsymbol{\chi}$, due to the measurement noise $\bar{\mathbf{n}}$, the distance of each point ${}^{L}\mathbf{p}_i$ from the line

$$r_{l_i} = {}^{L}\mathbf{N} \cdot {}^{L}\mathbf{p}_i - \rho \tag{3.29}$$

is a random variable. The (linearized) covariance of r_{l_i} is $P_{r_{l_i}} = {}^{L}\bar{\mathbf{M}}_i \mathbf{P}_{\bar{\mathbf{n}}} \left({}^{L}\bar{\mathbf{M}}_i\right)^{\top}$, where

$${}^{L}\bar{\mathbf{M}}_i = {}^{L}\mathbf{N}^{\top} \begin{bmatrix} \cos\theta_i & -R_i \sin\theta_i \\ \sin\theta_i & R_i \cos\theta_i \end{bmatrix}. \tag{3.30}$$

The linearized covariance is accurate when n_θ is small, which is typically the case for LIDAR applications. Under this assumption, the distribution of $\bar{\mathbf{M}}_i$ is accurately approximated as Gaussian. For the optimal line, the sequence r_{l_i} is zero mean and white.

With the above discussions, the goal of the line fitting algorithm is to find the line parameters $\boldsymbol{\chi}$ that maximizes the likelihood function

$$\mathcal{L}(\{r_{l_i}\}_{i=1}^{n} \mid \boldsymbol{\chi}) = \mathcal{L}(r_{l_1} \mid \boldsymbol{\chi}) \ldots \mathcal{L}(r_{l_n} \mid \boldsymbol{\chi}) \tag{3.31}$$

$$= exp\left(-\frac{1}{2}\sum_{i=1}^{n} r_{l_i}^{\top} P_{r_{l_i}}^{-1} r_{l_i}\right) \tag{3.32}$$

where we have used the fact that r_{l_i} is independent of r_{l_j} for $i \neq j$ and dropped the constant of normalization. Because $\mathcal{L}(\{r_{l_i}\}_{i=1}^{n} \mid \boldsymbol{\chi})$ is a nonlinear function of $\boldsymbol{\chi}$, there is no closed-form solution to the above problem. However, the minimum of the cost function can be found rapidly by an iterative algorithm.

Algorithm Setup: To fit a line, we reparameterize it using $[\alpha, \beta] = [\bar{\alpha}, \bar{\beta}]/\rho$ as

$$\alpha x + \beta y - 1 = 0. \tag{3.33}$$

This representation is appropriate for LIDAR applications because any detected line cannot pass through the origin of the LIDAR frame (i.e., $\rho \neq 0$).

Define $\boldsymbol{\eta} = [\alpha\ \beta]^\top$, then, from eqn. (3.29), we can form a linear estimation problem as

$$\mathbf{A}\boldsymbol{\eta} = \mathbf{b} + \frac{1}{\rho}\mathbf{r}_l \tag{3.34}$$

where $\mathbf{A} = [{}^L\mathbf{p}_1, \ldots, {}^L\mathbf{p}_n]^\top$, $\mathbf{b} = [1, \ldots, 1]^\top$ and $\mathbf{r}_l = [r_{l_1},\ \ldots,\ r_{l_n}]^\top$. The symbol $\mathbf{r}_l \in \Re^n$ represents error in units of meters with $cov(\mathbf{r}_l) = \mathbf{P}_{\mathbf{r}_l} = diag(P_{r_{l_1}}, \ldots, P_{r_{l_n}})$ as discussed in eqn. (3.30), while $\mathbf{n}_L \triangleq \frac{1}{\rho}\mathbf{r}_l$ is a dimensionless quantity with $cov(\mathbf{n}_L) = \mathbf{P}_{\mathbf{n}_L} = \frac{1}{\rho^2}\mathbf{P}_{\mathbf{r}_l}$.

We also define a function $\boldsymbol{\chi} = h(\boldsymbol{\eta})$ to extract the line parameter $\boldsymbol{\chi}$ from $\boldsymbol{\eta}$. The function $h(\cdot)$ is defined as

$$\phi = \arctan\left(\frac{\beta}{\alpha}\right), \quad \rho = \frac{1}{\|\boldsymbol{\eta}\|}. \tag{3.35}$$

Initialization: At the initialization, because $\boldsymbol{\chi}$ is not yet available, $\mathbf{P}_{\mathbf{n}_L}$ and $\mathbf{P}_{\mathbf{n}_l}$ cannot be computed; therefore, minimization of eqn. (3.32) is not possible. Instead, we approximate $\mathbf{P}_{\mathbf{n}_l} = \mathbf{I}$ and minimize $\sum_{i=1}^n r_{l_i}^\top r_{l_i}$ using $\boldsymbol{\eta}^0$ that is the solution of $\left(\mathbf{A}^\top\mathbf{A}\right)\boldsymbol{\eta}^0 = \mathbf{A}^\top\mathbf{b}$. Then we have $\boldsymbol{\chi}^0 = h(\boldsymbol{\eta}^0)$.

Step 1: Let the superscript k denote the k-th iteration. Use eqn. (3.30) with $\boldsymbol{\chi}^{k-1}$ to compute $\mathbf{P}_{\mathbf{n}_L}$. Then we re-solve (3.34) as $(\mathbf{A}^\top\mathbf{P}_{\mathbf{n}_L}^{-1}\mathbf{A})\boldsymbol{\eta}^k = \mathbf{A}^\top\mathbf{P}_{\mathbf{n}_L}^{-1}\mathbf{b}$ for $\boldsymbol{\eta}^k$.

Step 2: Compute $\boldsymbol{\chi}^k$ using $\boldsymbol{\chi}^k = h(\boldsymbol{\eta}^k)$.

Step 3: If $\left\|\boldsymbol{\chi}^k - \boldsymbol{\chi}^{k-1}\right\| > \delta_c$, then return to **Step1**; otherwise the algorithm ends, having computed $\boldsymbol{\chi}^k$. The parameter δ_c is chosen to trade off the accuracy and speed.

At the conclusion of the iteration, the covariance of $\boldsymbol{\eta}^k$ is $\mathbf{P}_{\boldsymbol{\eta}} = (\mathbf{A}^\top\mathbf{P}_{\mathbf{n}_L}^{-1}\mathbf{A})^{-1}$. The covariance of $\boldsymbol{\chi}^k$ can be computed by linearizing $h(\cdot)$ around $\boldsymbol{\eta}^k$ to get $\tilde{\boldsymbol{\chi}} = \mathbf{H}\tilde{\boldsymbol{\eta}}$, where

$$\mathbf{H} = \begin{bmatrix} -\frac{\beta}{\alpha^2+\beta^2} & \frac{\alpha}{\alpha^2+\beta^2} \\ -\frac{\alpha}{\|\boldsymbol{\eta}\|^3} & -\frac{\beta}{\|\boldsymbol{\eta}\|^3} \end{bmatrix}. \tag{3.36}$$

Hence the linearized covariance of $\boldsymbol{\chi}^k$ is

$$\mathbf{P}_{\boldsymbol{\chi}} = \mathbf{H}\mathbf{P}_{\boldsymbol{\eta}}\mathbf{H}^\top. \tag{3.37}$$

Note: In practice, this algorithm converges rapidly. With δ_c chosen to be 10^{-5}, the algorithm converges in 2 or 3 iterations.

Note: In practice, QR decomposition is used to compute $\boldsymbol{\eta}^k$ to improve the computational efficiency.

Note: As the number of points n associated with the line increases, the variance of the estimated line parameter $\boldsymbol{\eta}$ decreases. This can be seen from

$$\mathbf{P}_{\boldsymbol{\eta}} = (\mathbf{A}^\top \mathbf{P}_{\mathbf{n}_L}^{-1} \mathbf{A})^{-1} \tag{3.38}$$

$$= ((\mathbf{P}_{\mathbf{n}_L}^{-\top/2}\mathbf{A})^\top (\mathbf{P}_{\mathbf{n}_L}^{-\top/2}\mathbf{A}))^{-1} \tag{3.39}$$

$$= (\bar{\mathbf{A}}^\top \bar{\mathbf{A}})^{-1} \tag{3.40}$$

$$= ({}^L\bar{\mathbf{p}}_1 {}^L\bar{\mathbf{p}}_1^\top + \ldots + {}^L\bar{\mathbf{p}}_n {}^L\bar{\mathbf{p}}_n^\top)^{-1}. \tag{3.41}$$

The diagonal elements of each ${}^L\bar{\mathbf{p}}_i {}^L\bar{\mathbf{p}}_i^\top$ are positive. Hence, the more points used in the line fitting, the smaller the diagonal elements of $\mathbf{P}_{\boldsymbol{\eta}}$ will be.

3.2.4.3 Line Merging

Several lines may be extracted from $\mathcal{D}_k$. The i-th and j-th lines can be merged if their *Mahalanobis Distance* passes the chi-squared test:

$$\|\boldsymbol{\chi}_i - \boldsymbol{\chi}_j\|^2_{(\mathbf{P}_{\boldsymbol{\chi}_i} + \mathbf{P}_{\boldsymbol{\chi}_j})} < \delta_d \tag{3.42}$$

where the threshold δ_d is computed from the chi-squared distribution.

After we decide to merge the i-th and j-th line, we group the set of points together from two lines and then go back to the line fitting step to obtain the merged line parameters.

3.2.4.4 Feature Association

After the line merging step, the j-th line measurement will be associated to no more than one of the mapped features to form its residual. Feature association uses the *Mahalanobis Distance*: find the k-th mapped feature that minimizes $\|\boldsymbol{\chi}_j - \hat{\boldsymbol{\chi}}_k\|_{\mathbf{P}_{\boldsymbol{\chi}_j}}$, where $\hat{\boldsymbol{\chi}}_k$ is the predicted line-of-intersection for the k-th mapped feature computed from eqn. (3.21). If the k-th mapped feature satisfies $\|\boldsymbol{\chi}_j - \hat{\boldsymbol{\chi}}_k\|_{\mathbf{P}_{\boldsymbol{\chi}_j}} < \delta_a$, then we associate the k-th mapped feature to the j-th line masurement. Otherwise no measurement is associated to this feature. The threshold δ_a is obtained from the chi-squared distribution.

3.2.4.5 Measurement Residual Formation

To use measurements in the EKF, we define the measurement equation and the measurement prediction as:

$$\mathbf{y} = h(\mathbf{x}, \mathbf{n}), \quad \hat{\mathbf{y}} = h(\hat{\mathbf{x}}, \mathbf{0}) \tag{3.43}$$

where $h(\cdot,\cdot)$ is a nonlinear function of the current state $\mathbf{x}$ and measurement noise $\mathbf{n}$, $\mathbf{y}$ is the measurement vector and $\hat{\mathbf{y}}$ is the predicted measurement. Then the residual is defined as $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$. The measurement equation $h(\mathbf{x}, \mathbf{n})$ linearized around the estimated state $\hat{\mathbf{x}}$ is

$$\mathbf{r} = \mathbf{H}\tilde{\mathbf{x}} + \boldsymbol{\Gamma}\mathbf{n} \tag{3.44}$$

where $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$.

In LIDAR aiding, we use two measurements: ϕ and ρ. Here we define ϕ and ρ to be the measurements obtained from the line extraction step. The measurement noise is defined as $\mathbf{n} = [\tilde{\phi} \quad \tilde{\rho}]^\top$, and we assume $\mathbf{n}$ to be zero-mean, white Gaussian with covariance matrix $\mathbf{R} \triangleq \mathbf{P}_{\boldsymbol{\chi}}$, where $\mathbf{P}_{\boldsymbol{\chi}}$ is given in eqn. (3.37). The measurement equations of ϕ and ρ are given in eqn. (3.21), and are rewritten here for clarity:

$$\phi = h_1(\mathbf{x}, \mathbf{n}) = \arctan\left(\operatorname{sgn}({}^L d_i)\frac{a_2}{a_1}\right) + \tilde{\phi}, \tag{3.45}$$

$$\rho = h_2(\mathbf{x}, \mathbf{n}) = \frac{|{}^L d_i|}{\sqrt{a_1^2 + a_2^2}} + \tilde{\rho} \tag{3.46}$$

where ${}^L d_i = {}^G d_i - {}^G\boldsymbol{\pi}_i \cdot ({}^G\mathbf{p}_I + {}^G_I\mathbf{R}\,{}^I\mathbf{p}_L)$, ${}^L\mathbf{a}_i = [a_1 \ a_2 \ a_3]^\top$ and ${}^L\mathbf{a}_i = {}^L_G\mathbf{R}\,{}^G\boldsymbol{\pi}_i$. The prediction $\hat{\phi}$ and $\hat{\rho}$ are computed by $\hat{\phi} = h_1(\hat{\mathbf{x}}, \mathbf{0})$ and $\hat{\rho} = h_2(\hat{\mathbf{x}}, \mathbf{0})$. Hence we can define two residuals r_1 and r_2 to be $r_1 = \phi - \hat{\phi}$ and $r_2 = \rho - \hat{\rho}$. Since r_1 is an angle residual, it is normalized into $[-\pi, \pi]$ in practice. In the following we will form the residual model equations in the form of eqn. (3.44) for ϕ and ρ, respectively.

For ϕ, we have $\frac{\partial h_1(\cdot)}{\partial {}^L\mathbf{a}_i} = \frac{1}{\mu}\boldsymbol{\lambda}^\top$, where $\mu = a_1^2 + a_2^2$ and $\boldsymbol{\lambda}^\top = \operatorname{sgn}({}^L\hat{d}_i)\begin{bmatrix} -a_2 & a_1 & 0 \end{bmatrix}$. To compute $\frac{\partial {}^L\mathbf{a}_i}{\partial(\delta\boldsymbol{\theta})}$, we linearize ${}^L\mathbf{a}_i$ using the estimated state to obtain

$$\begin{aligned}
{}^L\mathbf{a}_i &= {}^L_G\mathbf{R}\,{}^G\boldsymbol{\pi}_i &(3.47)\\
&= {}^L_I\mathbf{R}\,{}^I_G\mathbf{R}\,{}^G\boldsymbol{\pi}_i &(3.48)\\
&= {}^L_I\mathbf{R}\,{}^I_G\hat{\mathbf{R}}(\mathbf{I} - \lfloor\delta\boldsymbol{\theta}\times\rfloor)\,{}^G\boldsymbol{\pi}_i &(3.49)\\
&= {}^L\hat{\mathbf{a}}_i + {}^L_I\mathbf{R}\,{}^I_G\hat{\mathbf{R}}\lfloor{}^G\boldsymbol{\pi}_i\times\rfloor\delta\boldsymbol{\theta}. &(3.50)
\end{aligned}$$

The above equation yields

$$\frac{\partial {}^L\mathbf{a}_i}{\partial(\delta\boldsymbol{\theta})} = {}^L_I\mathbf{R}\,{}^I_G\hat{\mathbf{R}}\lfloor{}^G\boldsymbol{\pi}_i\times\rfloor. \tag{3.51}$$

Thus we can write

$$\frac{\partial h_1(\cdot)}{\partial(\delta\boldsymbol{\theta})} = \frac{\partial h_1(\cdot)}{\partial {}^L\mathbf{a}_i} \cdot \frac{\partial {}^L\mathbf{a}_i}{\partial(\delta\boldsymbol{\theta})} \tag{3.52}$$

$$= \frac{1}{\mu}\boldsymbol{\lambda}^\top {}^L_I\mathbf{R}^I_G\hat{\mathbf{R}}\lfloor {}^G\boldsymbol{\pi}_i \times\rfloor. \tag{3.53}$$

Hence the linearized residual model for ϕ is

$$r_1 = \mathbf{h}_1^\top \tilde{\mathbf{x}} + \tilde{\phi} \tag{3.54}$$

where $\mathbf{h}_1^\top = \begin{bmatrix} \mathbf{0}_{1\times 6} & \frac{\partial h_1(\cdot)}{\partial(\delta\boldsymbol{\theta})} & \mathbf{0}_{1\times 6} \end{bmatrix}$.

For ρ, we have

$$\frac{\partial h_2(\cdot)}{\partial {}^G\mathbf{p}_I} = \frac{1}{\sqrt{\mu}}\mathrm{sgn}({}^L\hat{d}_i)\frac{\partial {}^L d_i}{\partial {}^G\mathbf{p}_I} + {}^L d_i\left(\boldsymbol{\kappa}^\top \frac{\partial {}^L\mathbf{a}_i}{\partial {}^G\mathbf{p}_I}\right) \tag{3.55}$$

$$= \frac{1}{\sqrt{\mu}}\mathrm{sgn}({}^L\hat{d}_i)(-{}^G\boldsymbol{\pi}_i^\top) + {}^L d_i(\boldsymbol{\kappa}^\top \cdot \mathbf{0}) \tag{3.56}$$

$$= -\mathrm{sgn}({}^L\hat{d}_i)\frac{{}^G\boldsymbol{\pi}_i^\top}{\sqrt{\mu}} \tag{3.57}$$

where $\boldsymbol{\kappa}^\top \triangleq \frac{\partial(1/\sqrt{\mu})}{\partial {}^L\mathbf{a}_i} = -\mu^{-\frac{3}{2}}\begin{bmatrix} a_1 & a_2 & 0 \end{bmatrix}$. In addition, we have

$$\frac{\partial h_2(\cdot)}{\partial(\delta\boldsymbol{\theta})} = \frac{1}{\sqrt{\mu}}\mathrm{sgn}({}^L\hat{d}_i)\frac{\partial {}^L d_i}{\partial(\delta\boldsymbol{\theta})} + {}^L d_i\left(\boldsymbol{\kappa}^\top \frac{\partial {}^L\mathbf{a}_i}{\partial(\delta\boldsymbol{\theta})}\right). \tag{3.58}$$

The partial $\frac{\partial {}^L\mathbf{a}_i}{\partial(\delta\boldsymbol{\theta})}$ is given in eqn. (3.51). To compute $\frac{\partial {}^L d_i}{\partial(\delta\boldsymbol{\theta})}$, we have

$${}^L d_i = {}^G d_i - {}^G\boldsymbol{\pi}_i \cdot ({}^G\hat{\mathbf{p}}_I + {}^G_I\mathbf{R}^I\mathbf{p}_L) \tag{3.59}$$

$$= {}^G d_i - {}^G\boldsymbol{\pi}_i \cdot {}^G\hat{\mathbf{p}}_I - {}^G\boldsymbol{\pi}_i \cdot (\mathbf{I} + \lfloor\delta\boldsymbol{\theta}\times\rfloor){}^G_I\hat{\mathbf{R}}^I\mathbf{p}_L \tag{3.60}$$

$$= {}^L\hat{d}_i + {}^G\boldsymbol{\pi}_i \cdot \lfloor {}^G_I\hat{\mathbf{R}}^I\mathbf{p}_L \times\rfloor\delta\boldsymbol{\theta}. \tag{3.61}$$

So

$$\frac{\partial {}^L d_i}{\partial(\delta\boldsymbol{\theta})} = {}^G\boldsymbol{\pi}_i^\top \lfloor {}^G_I\hat{\mathbf{R}}^I\mathbf{p}_L \times\rfloor. \tag{3.62}$$

Substituting eqn. (3.51) and (3.62) into eqn. (3.58) yields

$$\frac{\partial h_2(\cdot)}{\partial(\delta\boldsymbol{\theta})} = \mathrm{sgn}({}^L\hat{d}_i)\frac{{}^G\boldsymbol{\pi}_i^\top}{\sqrt{\mu}}\lfloor {}^G_I\hat{\mathbf{R}}^I\mathbf{p}_L \times\rfloor + {}^L d_i(\boldsymbol{\kappa}^\top {}^L_I\mathbf{R}^I_G\hat{\mathbf{R}}\lfloor {}^G\boldsymbol{\pi}_i \times\rfloor). \tag{3.63}$$

Hence the linearized residual model of ρ is

$$r_2 = \mathbf{h}_2^\top \tilde{\mathbf{x}} + \tilde{\rho} \tag{3.64}$$

where $\mathbf{h}_2^\top = \begin{bmatrix} \frac{\partial h_2(\cdot)}{\partial {}^G\mathbf{p}_I} & \mathbf{0}_{1\times 3} & \frac{\partial h_2(\cdot)}{\partial(\delta\boldsymbol{\theta})} & \mathbf{0}_{1\times 6} \end{bmatrix}$.

Stacking (3.54) and (3.64) together, we obtain the residual dynamics in the form shown in (3.44), with

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 \end{bmatrix}^\top, \quad \mathbf{\Gamma} = \mathbf{I} \tag{3.65}$$

and the residual vector $\mathbf{r} = \begin{bmatrix} r_1 & r_2 \end{bmatrix}^\top$.

Chapter 4

Estimator Design

4.1 System Model

The estimator design in this section is based on the following two equations:

$$\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u}, \mathbf{n}) \tag{4.1}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) + \mathbf{n_y} \tag{4.2}$$

where eqn. (4.1) is the state transition model and eqn. (4.2) is the measurement model, both $\boldsymbol{f}$ and $\mathbf{h}$ are nonlinear functions, $\mathbf{x}$ is the state vector, $\mathbf{u}$ is the control input vector, $\mathbf{n}$ is the noise vector in the state transition model, $\mathbf{y}$ is the measurement vector and $\mathbf{n_y}$ is the measurement noise vector. The covariance matrix of the noise $\mathbf{n_y}$ is $\mathbf{R}$.

To implement the estimator in a computer, the linearized discrete time error state models are required:

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{\Phi}_k \tilde{\mathbf{x}}_k + \mathbf{n}_k^d \tag{4.3}$$

$$\mathbf{r}_k = \mathbf{H}_k \tilde{\mathbf{x}}_k + \mathbf{n_y} \tag{4.4}$$

where $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$, $\mathbf{r}_k = \mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}_k)$ is the residual vector, $\mathbf{\Phi}_k$ is the error state transition matrix and $\mathbf{H}_k$ is the Jacobian matrix of measurement function. The covariance matrix of $\mathbf{n}_k^d$ is $\mathbf{Q}_k^d$.

Although this section focuses on the measurement model specified in eqn. (4.2), the other two kinds of measurement models that are commonly encountered can also be applied throughout this section:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{x}_l) + \mathbf{n}_\mathbf{y} \tag{4.5}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{X}) + \mathbf{n}_\mathbf{y} \tag{4.6}$$

4.2 EKF

In EKF, there is two steps: Prediction and Correction. In prediction step, we predict the state at the next time step $\hat{\mathbf{x}}_{k+1}^-$ by integrating $\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u}, \mathbf{0})$ and compute the expected state covariance using:

$$\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k \mathbf{P}_k \mathbf{\Phi}_k^\top + \mathbf{Q}_k^d. \tag{4.7}$$

In the correction step, the following equations are computed:

$$\mathbf{r}_{k+1} = \mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}_{k+1}^-, \mathbf{0}) \tag{4.8}$$

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1} \mathbf{P}_{k+1}^- \mathbf{H}_{k+1}^\top + \mathbf{R} \tag{4.9}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^- \mathbf{H}_{k+1}^\top \mathbf{S}_{k+1}^{-1} \tag{4.10}$$

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_{k+1}^- + \mathbf{K}_{k+1} \mathbf{r}_{k+1} \tag{4.11}$$

$$\mathbf{P}_{k+1} = \mathbf{P}_{k+1}^- - \mathbf{K}_{k+1} \mathbf{S}_{k+1} \mathbf{K}_{k+1}^\top. \tag{4.12}$$

4.3 Sliding-window Estimator

A sliding-window estimator, referred to as a Contemplative RealTime (CRT) estimator in [61], is used to combine the GPS and IMU data. The CRT approach has both real-time and contemplative aspects. The real-time state estimate is required for control and planning purposes, without latency. The contemplative aspects, inspired by recent research in the field robotics literature [9, 12, 29, 34, 13], are intended to enhance accuracy and robustness to faulty measurements.

4.3.1 Full MAP Estimation

Let $\mathbf{X}$ denotes the vehicle trajectory over a time window

$$\mathbf{X} = \left[\mathbf{x}(t_0)^\top \cdots \mathbf{x}(t_k)^\top\right]^\top \tag{4.13}$$

where $\mathbf{x}(t)$ denotes the vehicle state at time t.

Assume there is a prior for the first state: $\mathbf{x}(t_0) \sim \mathcal{N}(\mathbf{x}_0, \mathbf{P}_0)$. The corresponding residual function is

$$\mathbf{e}_0(\mathbf{x}(t_0)) = \mathbf{x}(t_0) - \mathbf{x}_0. \tag{4.14}$$

The IMU induced constraint $\mathbf{e}^i_\Delta$ between state $\mathbf{x}(t_{i+1})$ and $\mathbf{x}(t_i)$ can be defined as:

$$\mathbf{e}^i_\Delta(\mathbf{x}(t_{i+1}), \mathbf{x}(t_i)) = \mathbf{x}(t_{i+1}) - \mathbf{F}(\mathbf{x}(t_i), \mathbf{U}_i). \tag{4.15}$$

The uncertainty of $\mathbf{e}^i_\Delta$ is assumed to be Gaussian with the covariance matrix denoted by $\mathbf{Q}^d_i$.

Assume there are n_s aiding sensors where the j-th sensor provides a measurement $\mathbf{y}_{ij} \triangleq \mathbf{y}_j(t_i) \in \mathbb{R}^{m_j}$. Each sensor has a model:

$$\mathbf{y}_{ij} = \mathbf{h}_j(\mathbf{x}(t_i)) + \mathbf{n}_{\mathbf{y}_j}(t_i). \tag{4.16}$$

The covariance of $\mathbf{n}_{\mathbf{y}_j}(t_i)$ is $\mathbf{R}_{ij}$. The corresponding residual equation is

$$\mathbf{e}^j_{\mathbf{y}_i} = \mathbf{y}_{ij} - \mathbf{h}(\mathbf{x}(t_i)). \tag{4.17}$$

Estimation of the vehicle trajectory $\bar{\mathbf{X}}$ can be formulated as a MAP problem:

$$\hat{\mathbf{X}} = \underset{\mathbf{X}}{\operatorname{argmax}} \{p(\mathbf{X}, \mathbf{U}, \mathbf{Y})\} \tag{4.18}$$

where $\mathbf{U} = \{\tilde{\mathbf{u}}(t) \,|\, t \in [t_0, t_k]\}$ is the IMU data with a high sampling rate, $\mathbf{Y} = \{\mathbf{y}_{ij} \mid i = 0, \cdots, k, \text{and } j = 1, \cdots, n_s\}$. The appendix shows that $p(\mathbf{X}, \mathbf{U}, \mathbf{Y})$ can be decomposed as

$$\begin{aligned} &p(\bar{\mathbf{X}}, \mathbf{U}, \mathbf{Y}) = \\ &p(\mathbf{x}(t_0)) \prod_{i=0}^{k-1} p(\mathbf{x}(t_{i+1}) | \mathbf{x}(t_i), \mathbf{U}_i) \prod_{(i,j)\in\mathbf{Y}} p(\mathbf{y}_{ij} | \mathbf{x}(t_i)), \end{aligned} \tag{4.19}$$

where $p(\mathbf{x}(t_{i+1})|\mathbf{x}(t_i), \mathbf{U}_i)$ is the distribution of the kinematic constraint that as in eqn. (4.15), and $p(\mathbf{y}_{ij}|\mathbf{x}(t_i))$ is the distribution of the measurement constraint as defined in eqn. (4.16).

Assuming that the noise terms have Gaussian distributions, using log-likelihood, the MAP estimation problem can be converted to an equivalent nonlinear least squares problem:

$$\hat{\mathbf{X}} = \underset{\mathbf{X}}{\operatorname{argmin}} \left\{ \sum_{s\in\mathbb{S}} \|\mathbf{e}_s(\mathbf{X})\|^2_{\mathbf{R}_s} \right\} \tag{4.20}$$

where $\mathbf{e}_s(\mathbf{X})$ is the vector residual function defined for all the information available (denoted by the set $\mathbb{S}$). The set $\mathbb{S}$ contains $\{\mathbf{e}^i_\Delta|_{i=0\ldots k-1}\}$ defined in eqn. (4.15), $\{\mathbf{e}^j_{\mathbf{y}_i}|_{(i,j)\in\mathbf{Y}}\}$ defined in (4.17) and the prior $\mathbf{e}_0$ defined in (4.14). The matrix $\mathbf{R}_s$ is the covariance matrix and $\|\cdot\|^2_{\mathbf{R}_s}$ denotes the squared Mahalanobis norm: $\|\mathbf{x}\|^2_{\mathbf{A}} = \mathbf{x}^\top\mathbf{A}^{-1}\mathbf{x}$ for positive definite $\mathbf{A}$.

This nonlinear optimization problem can be solved iteratively (e.g., by Gauss-Newton method). At each iteration, the residual function $\mathbf{e}_s(\mathbf{X})$ is linearized at the current trajectory estimate $\hat{\mathbf{X}}$. Let

$$\mathbf{J}_s = \left.\frac{\partial \mathbf{e}_s}{\partial \mathbf{X}}\right|_{\hat{\mathbf{X}}} \tag{4.21}$$

$$\boldsymbol{\Lambda}_s = \mathbf{J}_s^\top \mathbf{R}_s^{-1} \mathbf{J}_s \tag{4.22}$$

$$\boldsymbol{\eta}_s = \mathbf{J}_s^\top \mathbf{R}_s^{-1} \mathbf{e}_s(\hat{\mathbf{X}}). \tag{4.23}$$

The linearized normal equation is:

$$\boldsymbol{\Lambda} \delta \mathbf{X} = \boldsymbol{\eta} \tag{4.24}$$

where $\boldsymbol{\Lambda} = \sum_{s \in \mathbb{S}} \boldsymbol{\Lambda}_s$ is the trajectory information matrix, $\boldsymbol{\eta} = \sum_{s \in \mathbb{S}} \boldsymbol{\eta}_s$ is the trajectory information vector. Eqn. (4.24) can be solved efficiently for $\delta \mathbf{X}$ by the Cholesky factorization as discussed in [12]. For a long vehicle trajectory, the sparsity of $\boldsymbol{\Lambda}$ can be exploited to speed up the Cholesky factorization. After solving $\delta \mathbf{X}$, the state is corrected by $\mathbf{X} = \mathbf{X} + \delta \mathbf{X}$. The iteration repeats starting from eqn. (4.21) by re-computing $\boldsymbol{\Lambda}$ and $\boldsymbol{\eta}$ using the new linearization point $\mathbf{X}$. The iterative process is terminated when $\|\boldsymbol{\eta}\|_2 < \epsilon$, where $\epsilon > 0$ is a user defined termination threshold.

Solving the full MAP problem results in the optimal estimate of the entire vehicle trajectory. When the structure of $\boldsymbol{\Lambda}$ is sparse, the computation is near linear with the length of trajectory. Otherwise if the structure of $\boldsymbol{\Lambda}$ is dense, the computation is cubic with the length of trajectory. The techniques from iSAM [29] can be used to speed up the computation by fixing the linearization point for a short period of time and re-evaluating $\boldsymbol{\Lambda}$ periodically. However, iSAM requires the storage of all the measurements over the entire trajectory and the periodic re-linearization of the entire trajectory is still computationally intense. The iSAM2 [28] avoids the periodic re-linearization of the entire trajectory by performing fluid relinearization.

However, solving a full MAP problem at each sensor measurement time to generate the realtime navigation solution is difficult due to the linear growth with time of the computation and memory requirements. Therefore, in this paper, we use a sliding window estimator, with computational complexity determined by the size of the window. Throughout this paper, we name the sliding-window the CRT window and denote the size of CRT window to be M.

4.3.2 Marginalization

At time t_{M-1}, after solving the MAP estimation problem, the oldest state $\mathbf{x}(t_0)$ needs to be removed, to keep the CRT window size constant when the next state $\mathbf{x}(t_M)$ is added. To correctly account for the information lost with the removed state, a marginalization step needs to be performed.

Let $\mathbf{X} = [\mathbf{X}_m^\top, \mathbf{X}_r^\top]^\top$ where $\mathbf{X}_m$ are the states that are going to be removed and $\mathbf{X}_r$ are the states that will be retained. This marginalization process can be performed by the following procedure [13]. Let $\mathbb{S}_m \subset \mathbb{S}$ such that for any $s \in \mathbb{S}_m$ the residual function $\mathbf{e}_s$ depends on $\mathbf{X}_m$ and for any $s \notin \mathbb{S}_m$, $\mathbf{e}_s$ is not a function of $\mathbf{X}_m$. Define $\bar{\boldsymbol{\Lambda}} = \sum_{s\in\mathbb{S}_m} \boldsymbol{\Lambda}_s$ and $\bar{\boldsymbol{\eta}} = \sum_{s\in\mathbb{S}_m} \boldsymbol{\eta}_s$. Thus, the information matrix $\boldsymbol{\Lambda}$ and information vector $\boldsymbol{\eta}$ can be partitioned accordingly:

$$\bar{\boldsymbol{\Lambda}} = \begin{bmatrix} \boldsymbol{\Lambda}_{mm} & \boldsymbol{\Lambda}_{mr} \\ \boldsymbol{\Lambda}_{mr}^\top & \boldsymbol{\Lambda}_{rr} \end{bmatrix} \tag{4.25}$$

$$\bar{\boldsymbol{\eta}} = \begin{bmatrix} \boldsymbol{\eta}_m \\ \boldsymbol{\eta}_r \end{bmatrix} \tag{4.26}$$

where $\boldsymbol{\Lambda}_{mm}$ and $\boldsymbol{\eta}_m$ correspond to the state that will be marginalized out.

The Schur complement is employed to marginalize out $\mathbf{X}_m$:

$$\mathbf{\Lambda}_\alpha = \mathbf{\Lambda}_{rr} - \mathbf{\Lambda}_{mr}^\top \mathbf{\Lambda}_{mm}^{-1} \mathbf{\Lambda}_{mr} \tag{4.27}$$

$$\boldsymbol{\eta}_\alpha = \boldsymbol{\eta}_r - \mathbf{\Lambda}_{mr}^\top \mathbf{\Lambda}_{mm}^{-1} \boldsymbol{\eta}_m \tag{4.28}$$

where $\mathbf{\Lambda}_\alpha$ and $\boldsymbol{\eta}_\alpha$ are the trajectory information matrix and vector for $\mathbf{X}_r$ after marginalization of $\mathbf{X}_m$.

After solving the MAP problem by iterations of eqns. (4.21)–(4.24), we have an estimate of the trajectory $\hat{\mathbf{X}} = [\hat{\mathbf{X}}_m^\top, \hat{\mathbf{X}}_r^\top]^\top$. The estimate $\hat{\mathbf{X}}_r$ can be used as the prior for $\mathbf{X}_r$ for the next CRT window after sliding. Define $\hat{\mathbf{X}}_\alpha \triangleq \hat{\mathbf{X}}_r$. The residue function for this prior is

$$\mathbf{e}_\alpha(\mathbf{X}_r) = \mathbf{X}_r - \hat{\mathbf{X}}_\alpha. \tag{4.29}$$

The uncertainty of $\mathbf{e}_\alpha(\mathbf{X}_r)$ is specified by the information matrix $\mathbf{\Lambda}_\alpha$. The measurement residual projected on the state space is stored in $\boldsymbol{\eta}_\alpha$.

4.3.3 MAP Estimation in a CRT Window

For each CRT window, the vehicle trajectory $\mathbf{X}$ is

$$\mathbf{X} = \left[\mathbf{x}(t_{k-M+1})^\top \cdots \mathbf{x}(t_k)^\top\right]^\top. \tag{4.30}$$

The MAP estimation problem in the CRT window can be reformulated as [13]:

$$\hat{\mathbf{X}} = \underset{\mathbf{X}}{\operatorname{argmin}} \left\{ \sum_{s \in \mathbb{S}} \|\mathbf{e}_s(\mathbf{X})\|^2_{\mathbf{R}_s} - 2\boldsymbol{\eta}_\alpha^\top \mathbf{e}_\alpha(\mathbf{X}) \right\} \tag{4.31}$$

where the $\mathbf{e}_s(\bar{\mathbf{X}})$ is the residual function defined for all the information available in the CRT window (denoted by $\mathbb{S}$). The set $\mathbb{S}$ contains $\{\mathbf{e}^i_\Delta|_{i=k-M+1\ldots k}\}$ defined in eqn. (4.15), $\{\mathbf{e}^j_{\mathbf{y}_i}|_{(i,j)\in\mathbb{S}_\mathbf{y}}\}$ defined in (4.17) where $\mathbb{S}_\mathbf{y}$ is the set of measurements available in the window, and the prior $\mathbf{e}_\alpha$ is defined in (4.29).

To analyze the performance of the CRT estimator, it is convenient to define several variables. The cost C is defined as:

$$C(\mathbf{X}) = \sum_{s \in \mathbb{S}} \|\mathbf{e}_s(\mathbf{X})\|^2_{\mathbf{R}_s} - 2\boldsymbol{\eta}_\alpha^\top \mathbf{e}_\alpha(\mathbf{X}). \tag{4.32}$$

Then the cost reduction ΔC^i at each iteration can be defined as $\Delta C^i = C^0 - C^i$ where C^i for $i = 0, \cdots, \zeta$ denotes the cost after the i-th iteration and ζ is the total number of iterations.

4.3.4 Covariance Recovery

For outlier detection, the covariance matrix of the latest state $\mathbf{x}(t_k)$ is required and can be efficiently recovered using the methods in [29, 30].

Let the Cholesky decomposition of $\boldsymbol{\Lambda}$ be $\boldsymbol{\Lambda} = \mathbf{R}^\top\mathbf{R}$ where $\mathbf{R}$ is an upper triangular matrix. Let $\mathbf{P} = \boldsymbol{\Lambda}^{-1}$ be the covariance matrix of the trajectory and

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{(d-d_x)\times d_x} \\ \mathbf{I}_{d_x \times d_x} \end{bmatrix} \tag{4.33}$$

where $d = d_x M$ is the size of $\boldsymbol{\Lambda}$ and d_x is defined in eqn. (2.47). Hence by solving

$$(\mathbf{R}^\top\mathbf{R})\mathbf{P}_{ck} = \mathbf{B} \tag{4.34}$$

where $\mathbf{P}_{ck} = [\mathbf{P}_{1k}^\top, \cdots, \mathbf{P}_{kk}^\top]^\top$ contains the last fifteen columns of $\mathbf{P}$, the covariance of the latest vehicle state ($\mathbf{P}_{kk}$) can be obtained. The solution employs the forward and backward substitution:

$$\mathbf{R}^\top\mathbf{K} = \mathbf{B} \tag{4.35}$$

$$\mathbf{R}\mathbf{P} = \mathbf{K} \tag{4.36}$$

where $\mathbf{K}$ can be easily obtained by solving eqn. (4.35):

$$\mathbf{K} = \begin{bmatrix} \mathbf{0} \\ \\ \mathbf{R}_{kk}^{-\top} \end{bmatrix} \tag{4.37}$$

where $\mathbf{R}_{kk}$ is the right bottom portion of $\mathbf{R}$ that corresponds to the latest vehicle state. From $\mathbf{K}$, eqn. (4.36) can be solved for $\mathbf{P}_{kk}$.

The elements in arbitrary positions of the covariance matrix $\mathbf{P}$ also can be recovered by the method proposed in [30] at an extra computational cost.

4.3.5 Application to GPS/INS

For GPS/INS, the measurement equation (4.16) is redefined for pseudorange and phase measurements in eqn. (3.1) and (3.3), and the integer-free phase measurements that is going to be defined in (3.12). A typical measurement scenario is depicted in Fig. 4.1. The red dots on the time-line represent the state at the GPS measurement times t_k. The state transition between these times is constrained by the kinematic model of eqn. (2.46) and the IMU data $\mathbf{U}$. Additional constraints are imposed by the initial condition $(\mathbf{x}_0, \mathbf{P}_0)$, and GPS measurements $\mathbf{Y}$. The initial condition constraint is shown above the time-line. The GPS measurement constraints $(\mathbf{y})$ are depicted below the time-line. The GPS measurements are not synchronized with the IMU measurement time. The unaligned measurements can be addressed by interpolation, and unknown latencies could be calibrated by the methods in [35].

A typical CRT process starts when $t = t_k$, using all IMU and GPS measurements that are available for the time interval $[t_{k-M+1},\ t_k]$. Starting with the prior defined in (4.29) and integrating forward through time using the IMU data and kinematic model provides an initial trajectory across the CRT window. Starting from this

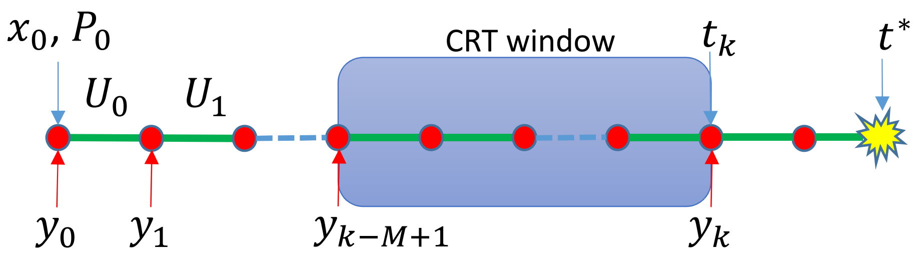


Figure 4.1: An illustration of measurement time line and CRT window. The red dots represent the vehicle states at the GPS measurement times. The green lines represent the IMU constraints between two consecutive states. At t_k the CRT window is formed and the optimization problem begins its computation. The optimization solution is not availble until t^* and after the solution is obtained, we propagate the state from t_k to $t \geq t^*$ using all the IMU measurements.

initial trajectory, the CRT algorithm will consider all the available information to reliably and accurately compute the state trajectory over the CRT window using the optimization process discussed above and fault detection methods (which are left for future work). This CRT process ends at a time $t^* > t_k$, ideally providing an optimal trajectory estimate from which the effects of sensor faults have been removed. Within the computation time interval $t \in [t_k, t^*]$, the real-time state estimate $\hat{\mathbf{x}}(t)$ is maintained by the INS using the IMU data and starting from the initial estimate of $\mathbf{x}(t_k)$. At $t = t^*$, $\hat{\mathbf{x}}(t_k)$ is corrected to the result of the CRT contemplative process and propagated through time using the IMU data and eqn. (2.46) to provide an improved estimate of $\mathbf{x}(t)$ at the present time. The computation time $(t^* - t_k)$ is typically a fraction of a second, small enough that the INS does not accumulated significant error. At some time $t \geq t^*$, the CRT window can be redefined and the process can repeat indefinitely. For the GPS/INS case, the CRT process happens at each GPS epoch.

4.3.6 Reliable Removal of Faulty Data

Receiver Autonomous Integrity Monitoring (RAIM, [24]) is a set of techniques to detect, identify and remove GNSS receiver outlier measurements. Traditionally, in the navigation community, RAIM is designed assuming only one outlier occurs and that there is enough measurement redundancy to detect and identify the source. The proposed CRT approach, which enhances the redundancy by incorporating a window of IMU and GPS data, can be expected to enable multiple outlier detection, identification, and removal. This outlier rejection scheme, which enhances the robustness of vehicle GPS-INS significantly, could make critical contributions as necessary for life-safety applications. The key technique in standard RAIM is *outlier detection and identification.* This section considers the detection and identification, removal procedures within the CRT framework. Interested readers may find more details in [24].

Suppose that the MAP optimization in eqn. (4.31) finally converges to a optimal estimate $\mathbf{X}^*$. The generalized *a-posteriori* variance factor test evaluated as

$$\hat{\sigma}_0^2 = \frac{\|\boldsymbol{r}(\mathbf{X}^*)\|_2^2}{M-N}, \tag{4.38}$$

can be used to detect outlier. In this expression, $M = n(1+K) + \sum_{j=1}^{K}(m_j - 1)$ is the total number of residuals (constraints) and $N = n(K+1)$ is the dimension of $\mathbf{X}$. Note that $\sum_{j=1}^{K}(m_j - 1)$ is the total number of double-differenced GPS measurements. The degrees-of-freedom $(M - N)$ can be considered as the index of the measurement redundancy. For conventional GNSS-only RAIM which uses one epoch measurements, the redundancy is $(m_j - 4)$, which requires at least five satellites to be available. For the proposed CRT framework, the measurement redundancy is $M - N = \sum_{j=1}^{K}(m_j - 1)$, which indicates that it has enhanced detectability against faulty data.

The final step of outlier detection is to test the above variance factor against the two-tailed Chi-square test limits with respect to a significance level α,

$$\frac{\chi^2_{1-\alpha/2,M-N}}{M-N} \leq \hat{\sigma}_0^2 \leq \frac{\chi^2_{\alpha/2,M-N}}{M-N}. \tag{4.39}$$

If the test succeeds, $\mathbf{X}^*$ is finalized as the smoothing result and the real-time part will use it to reinitialize. If not, outlier identification executes by testing each residual with the w-test. Once the source of the outlier is identified, the corresponding measurement will be removed and the procedure continues until no additional outliers are identified.

When outlier identification completes, the outliers are removed and the MAP optimization is formed with the cleaned-up measurement set, then the detect-identify-remove procedure repeats.

4.4 Software Architecture Design

To meet the realtime performance requirements and facilitate the software development, the navigation system is designed in a multi-threading framework as shown in Fig. 4.2. Each thread is assigned a priority to ensure the high priority threads (e.g. INS propagation thread) use the CPU first and the low priority threads (e.g. data logging) use the CPU only when the system is idle. The system runs on Ubuntu 10.04 with a realtime kernel.

There are 3 running modes in the system: *Realtime*, *Rerun_TrueTime* and *Rerun_FastTime*. In all three modes, the system solves the trajectory estimation problem over a window of length M while maintaining an estimate of the current vehicle state. The three modes differ in where they source the data from and whether they operate at or faster than realtime.

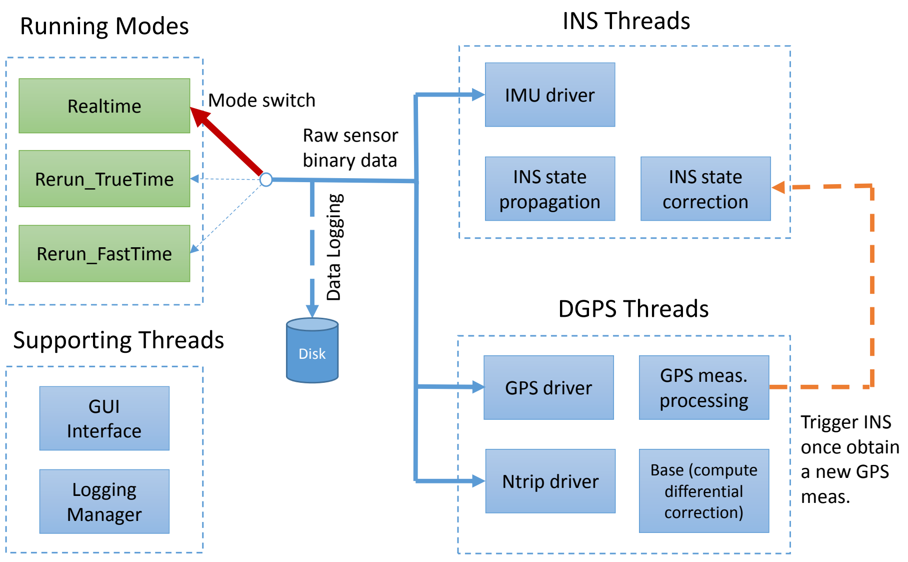


Figure 4.2: Software Architecture. The GPS base measurements are sent via internet using the NTRIP protocol in the realtime mode. All the data logging is managed by the logging manager thread which accesses the hard disk at the lowest priority. The important run time information are sent to a separate GUI program through the GUI interface thread at 1Hz.

- *Realtime* mode: In this mode, the system obtains the sensor data from the physical sensors (the GPS base measurements are received via internet).
- *Rerun* modes: To enable fast algorithm verification and realtime performance tuning and debugging, two rerun modes are designed. Both rerun modes retrieve the sensor data from log files.
  - *Rerun_TrueTime* mode: The logged sensor data is fed into the system at the exact same time (within the accuracy of the computer clock) relative to the start of the program as it was in the realtime mode. This mode is ideal for exactly reproducing the realtime running situation and/or debugging for realtime errors.

  - *Rerun_FastTime* mode: The logged sensor data is fed into the system immediately after all the system processing is complete for the previous sensor data. This mode is ideal for testing the algorithm on multiple datasets to generate statistics and/or debugging as it processes each data set quickly. For example, processing a 500sec trajectory using $M = 10$ takes around 40sec in this mode.

Therefore, the same code can be re-used for different purposes with only the change of desired running modes.

To facilitate the comparison of various estimators, the INS correction thread is designed as a general estimator interface and can be easily replaced with any estimator, such as the EKF or sliding-window estimator. Moreover, the multi-threaded system architecture can also be extended to include multiple aiding sensors such as vision [59] and Lidar [63].

4.5 Comparison of CRT with EKF

The tradeoffs of the CRT relative to the EKF are:

1. CRT has the ability to change the linearization point of the trajectory within the CRT window. For the EKF the linearization point is the prior. Errors in the prior if large relative to the higher order terms of the linearization can cause the EKF to diverge.

2. In the CRT approach multiple iterations, each with relinearization, are possible to fully address the nonlinearities in the MAP optimization. The standard EKF performs a single iteration per measurement.

3. With a window of sensor data, the CRT redundancy (prior, INS state transitions, measurements for all M measurement times) is extensive allowing detailed outlier detection. Also, since all raw data over the window is retained, fault decisions at one time can be reconsidered at future times, as long as the data is still in the window. The effects of a previous incorrect decision can be fully removed by the MAP optimization process. With the standard EKF, an incorrect fault decision can be catastrophic and the redundancy available (prior and a measurements at a single time) is often insufficient to make confident fault decisions.

4. The computation effort of the CRT is larger than EKF, increasing with window size and iterations.

5. The numerical stability of EKF is better than CRT because EKF inverses $\mathbf{S}$ which has a small condition number while EKF inverses $\mathbf{\Lambda}$ which has a large condition number.

Chapter 5

Experimental Results for Navigation

5.1 Pseudorange Only

This section uses 200Hz MEMS IMU and 1Hz Differential GPS data collected on a vehicle. The navigation system is implemented in C++ with multi-threading and reports the vehicle state in realtime at the IMU sampling rate. The CRT window is chosen to be 10 sec. In this experiment, the vehicle stays stationary while pointing north at the beginning. After about 20 seconds the vehicle accelerated north. The trajectory of vehicle is illustrated in Fig. 5.1.

This section compares the following real-time estimators:

1. CRT using double-differenced code measurements;
2. EKF using double-differenced code measurements;
3. EKF using integer-resolved double-differenced carrier phase measurements.

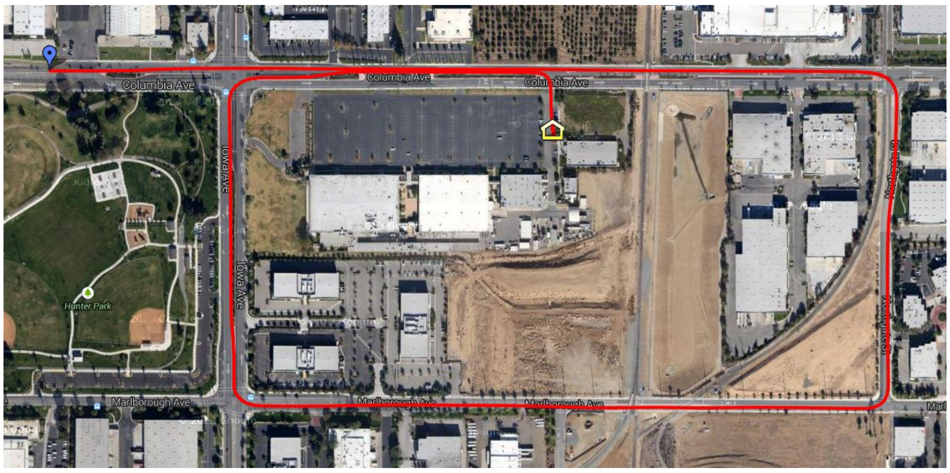


Figure 5.1: Vehicle trajectory (red). The yellow and blue markers show the start and end points, respectively.

The definition of the state vector is the same for all cases. In addition, we post-process the data through an off-line batch smoother for the entire trajectory using integer-resolved double-differenced carrier phase and code measurements. This post-processed trajectory will be considered as the ground truth to which the other three estimators are compared to determine error statistics.

Firstly, we demonstrate the capability of the proposed estimator to initialize yaw, without a compass. For a stationary vehicle, yaw is unobservable from GPS measurements. When the vehicle accelerates, yaw becomes observable. For the EKF, the yaw needs to be initialized close to the true value to satisfy the EKF small error assumption; otherwise the EKF may diverge. For EKF implementations, yaw may be initialized via magnetometer, or other means. The CRT estimator optimally initialize the yaw using the data in the CRT window. Everything is done naturally within the smoothing framework and it is also optimal with respect to the pre-defined noise model. To demonstrate this, the initial yaw is set to have an error of 180 degrees. Fig. 5.2 shows

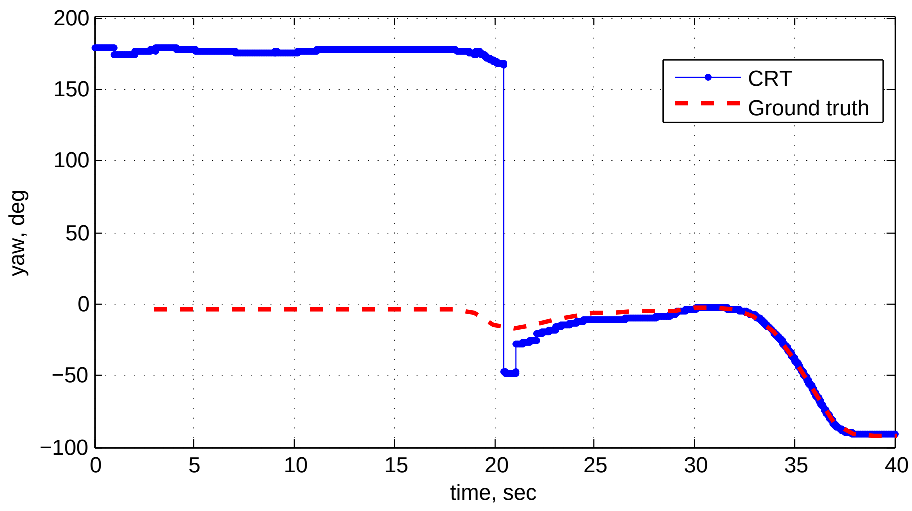


Figure 5.2: Yaw estimated by the CRT at the beginning of the trajectory. The yaw is initialized naturally as the vehicle starts to accelerate at around 20 sec.

that, when the vehicle accelerates (near $t = 20s$), the yaw rapidly converges toward the correct value.

Secondly, the proposed estimator significantly improves the state estimate accuracy. Fig. 5.3 and Fig. 5.4 compare the estimated trajectory error from the three estimators mentioned above. Fig. 5.3 shows a segment of the estimated trajectories. The trajectory of the CRT approach using double-differenced code is very near the trajectory of the EKF using double-differenced integer-resolved phase. The EKF using double-differenced code has significantly larger errors. The distribution of the norm of the position error in the horizontal plane (north-east) is shown in Fig. 5.4. The Fig. 5.4 use the state estimate data at 1 Hz (324 data points in total for this trajectory). The error statistics clearly show that the proposed estimator has the position estimate error in the decimeter level, while the EKF using the same code measurements has the error in the meter level. This large accuracy improvement is gain by leveraging a window of measurements. The performance of the proposed estimator is already very close to that of the EKF using phase measurement (centimeter level). However, to achieve the centimeter accuracy, the integer ambiguity needs to be resolved in realtime.

The performance of the CRT and EKF approaches have also been evaluated on the following two testing trajectories:

- Test2: 600 sec driving in mostly open sky environment.
- Test3: 500 sec driving in an area where a portion of the trajectory has partial GPS signal blockage due to trees and buildings along the road.

The statistical comparison of the position error in the north and east directions over the entire trajectory for three testing trajectories is given in Table. 5.1. The Test1 trajectory is the one shown in Fig. 5.1. It is clear to see from the table that the CRT approach consistently outperforms the EKF approach by keeping the position error at the decimeter level while the EKF tends to have position errors in meters.

The proposed CRT method also improves the estimate of the other variables in the 6DOF state vector (eqn. (2.37)). Fig. 5.5 shows the estimated accelerometer bias from the EKF and CRT method for the Test2 trajectory. It is obvious that the accelerometer bias estimate converges significantly faster in the CRT approach. The fast convergence of the IMU bias estimate is the key to maintaining high precision navigation performance.

5.2 Pseudorange and Integer-free Carrier Phase

This section presents analysis of data accumulated using a test loop around the campus of University of California, Riverside, see Fig. 5.6. Along the test path there are

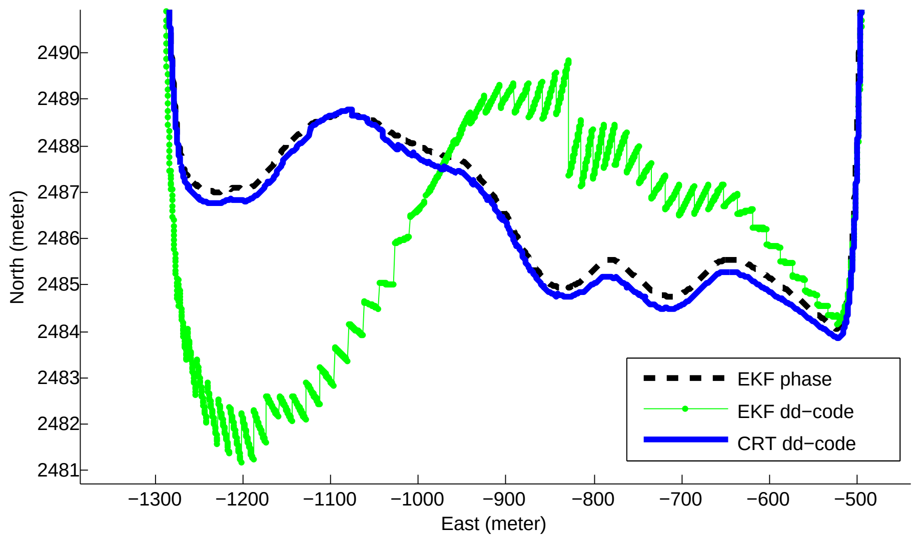


Figure 5.3: EKF and CRT position accuracy comparison for one segment of the trajectory.

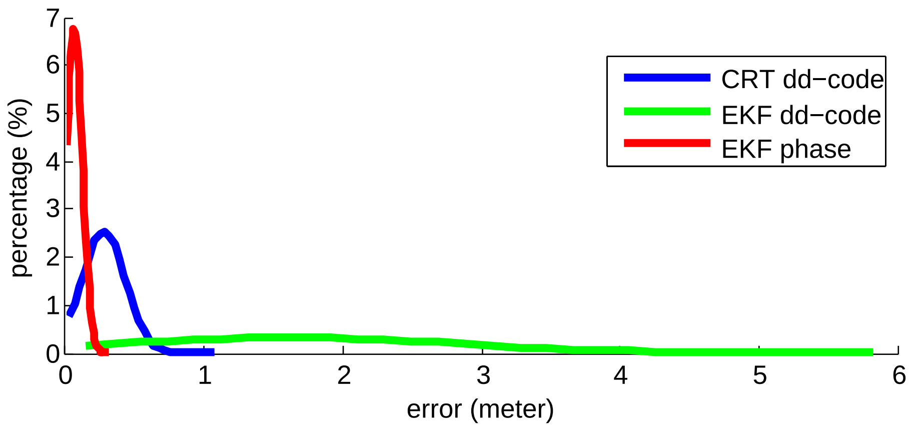


Figure 5.4: Distribution of horizontal position error.

Table 5.1: Comparison of position error statistics. Mean and standard deviation are denoted as μ and σ (unit is in meter). Double differenced code measurements are used in both estimators.

		Test1	Test2	Test3
North:	CRT:	$\mu = -0.016,\ \sigma = 0.18$	$\mu = -0.174,\ \sigma = 0.27$	$\mu = 0.003,\ \sigma = 0.31$
	EKF:	$\mu = 0.162,\ \sigma = 1.69$	$\mu = -0.097,\ \sigma = 1.08$	$\mu = 0.040,\ \sigma = 1.10$
East:	CRT:	$\mu = 0.164,\ \sigma = 0.21$	$\mu = -0.081,\ \sigma = 0.19$	$\mu = 0.220,\ \sigma = 0.41$
	EKF:	$\mu = 0.326,\ \sigma = 1.08$	$\mu = 0.034,\ \sigma = 0.67$	$\mu = 0.629,\ \sigma = 1.39$

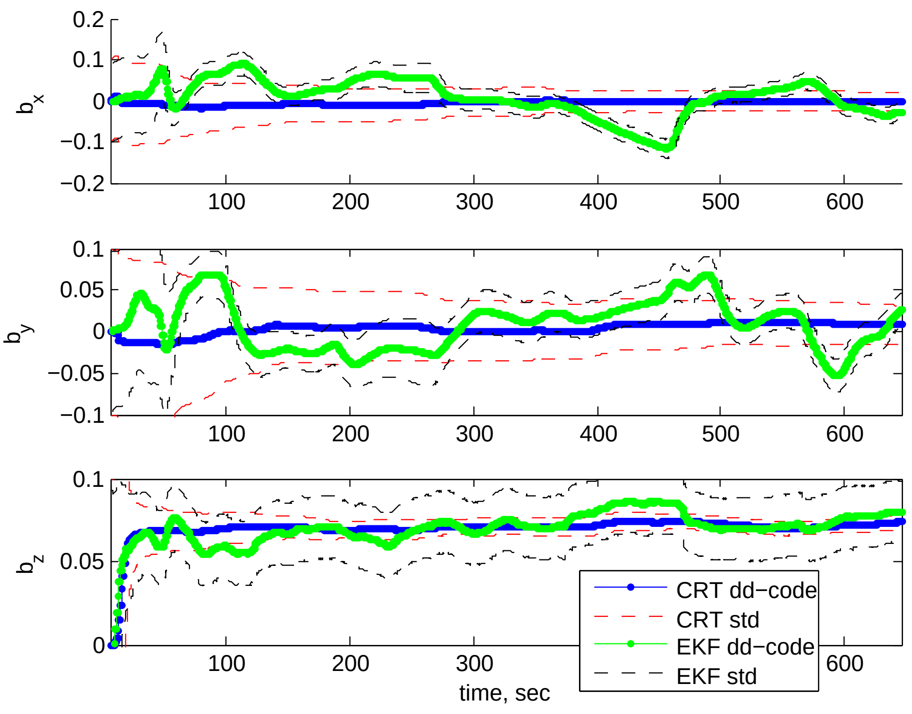


Figure 5.5: The comparison of the estimated accelerometer bias with the $\pm 1\sigma$ bound. The unit of bias is m/s^2.

many trees and buildings as is representative of a typical urban environment. The test path is challenging for GPS based navigation. Snapshots along the path can be found in [58]. Fig. 5.6 uses color coding along the trajectory to indicate the number of satellite signals received as a function of position. There are many places where only a few satellites are available. The effect of tree cover and multipath is more difficult to discern. Any of these effects (e.g., small number of satellites, tree cover, or multipath) can make it extremely difficult to maintain integer lock and to resolve the correct integers.

In the experiment, the vehicle is equipped with a dual-frequency GPS receiver and a MEMs IMU, but no form of compass. The IMU provides measurements at 200 Hz. GPS measurements are taken at 1 Hz. All GPS measurements are used in a differential

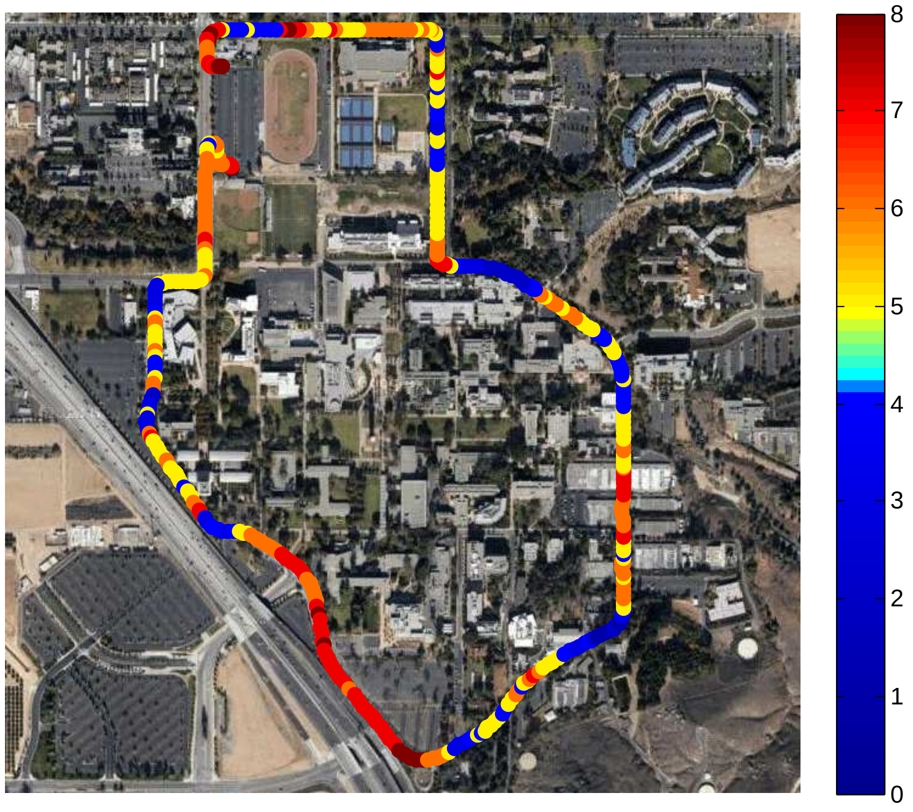


Figure 5.6: UCR test trajectory marked with colors to represent the number of satellites available to the receiver along the trajectory. This is a challenging route for the GPS based navigation, due to signal blockage by buildings and trees. There are many places (highlighted in blue) where less than 4 satellites are available for extended durations.

mode. This set of instruments allows a post-processing algorithm to determine the *ground truth* trajectory with centimeter accuracy [58].

To initialize the CRT estimator (obtaining $\mathbf{x}_0$), the vehicle is assumed to be stationary for a few seconds at the start of the experiment. The initial roll and pitch estimate are obtained by extracting the gravity vector. The initial position estimate is obtained from GPS-only double differenced pseudorange measurements. Yaw is completely unknown. To emphasize the ability to correct errors that are large relative to the curvature of the nonlinearities, we intentionally initialize the yaw with the worst case error of 180 degrees.

The CRT estimator uses the exact same data set as the ground truth post-processing algorithm, so that we can compare the CRT estimation results with the ground truth trajectory. The CRT estimator only uses the L1 GPS data, discarding the L2 data. For the CRT estimator, the window size is $M = 10$ and the termination threshold is $\epsilon = 10^{-3}$.

After each CRT experiment is completed, performance is analyzed by subtracting the CRT state estimate at each time from the ground truth state estimate for the same time. Analysis of this error state will demonstrate that the CRT window based Bayesian estimation approach combining pseudorange and phase measurements rapidly calibrates the IMU biases and IMU attitude after vehicle motion makes them observable and mitigates the effects of code multipath. The fast, accurate, and reliable estimation of attitude and IMU biases estimates is the key to maintaining high precision navigation performance in GPS-challenged environments because the INS uses the corrected IMU data to integrate through GPS time-intervals when few (if any) satellite signals are received.

The results of the CRT estimator using code and integer-free phase are shown in Figs. 5.7-5.12 and 5.13. Fig. 5.7-5.12 show the INS errors (position, velocity, attitude and IMU biases) of the CRT estimator with the reported $\pm 3\sigma$ bounds. The estimator starts at $t = 5$sec. The vehicle is stationary at the beginning with forward acceleration starting at $t \approx 8$sec. Note that the yaw error is $< 1^\circ$ by $t = 10$sec. The results show that the position error for most of the time is within $\pm 0.5m$ in the horizontal plane (north and east direction), the velocity error is within $\pm 0.1m/s$ and the roll and pitch errors are within $\pm 0.2^\circ$ and the yaw errors are within $\pm 1^\circ$. The reported $\pm 3\sigma$ bounds also correctly reflect the uncertainty of errors. Fig. 5.13 shows the histogram of errors (position, velocity and attitude) of the CRT estimator along the trajectory. The first 10 seconds are excluded from the histogram to prevent the presentation being skewed by the errors in the initial state (e.g. yaw).

Fig. 5.14 displays the CRT position estimation errors in blue when only pseudorange is used and in red when both pseudorange and code are used. The estimated trajectory using the integer-free phase measurement is in general smoother than only using the code measurements. From this comparison, we can see that the integer-free phase measurements are able to improve the robustness of estimator to pseudorange multi-path error and noise. This robustness derives from the kinematic constraints imposed by the integer-free phase measurements, which prevent the large changes in the position estimates that could otherwise result from multi-path errors.

To gain insight into the status of the CRT estimator, several key variables are plotted in Fig. 5.15 versus the GPS epoch counter. The figure shows the total number of iterations ζ, the cost reduction ΔC^ζ, the final cost C^ζ, the final value of $\|\boldsymbol{\eta}\|_2$ in the optimization. Immediately after initialization, it takes more iterations per time step to converge from the initial $\mathbf{x}_0$ to an estimate $\hat{\mathbf{x}}$ such that $\delta\mathbf{x}$ lies within the unobservable space. Due to the initial inaccuracy of $\mathbf{x}_0$, the error within the unobservable space may still be large. For example, the yaw is still completely unknown, until the vehicle accelerates. At this time, when the bias and attitude errors become observable, the effect of nonlinearities can be very significant; this is demonstrated by the number of iterations again increasing. It is the fact that the unobservable subspace changes, that allows the attitude and bias vectors to be more accurately estimated. Note from Fig. 5.7-5.12 that yaw is accurately estimated, within 1 degree, quickly after the vehicle accelerates, without any form of magnetometer. After the trajectory estimate becomes accurate, the optimization needs fewer iterations to converge. The final value of $\|\boldsymbol{\eta}\|_2$ is always less than the termination threshold $\epsilon = 10^{-3}$. The vector $-\boldsymbol{\eta}$ is the gradient of the cost function $C(\mathbf{X})$; therefore, its final norm should be near zero when the trajectory estimate is near any local minimum of the cost function. The fact that final cost reduction ΔC^{ζ} is always positive indicates that the optimization improves accuracy and does not jump to a worse local minimum (if one exists). The final cost C^{ζ} is also given as a reference. In practice, the inconsistent GPS data, possibly caused by multi-path or overhead trees, could explain the large values of C^{ζ} as is shown near time 200 sec and 350 sec. For example, Fig. 5.16 shows the histogram of the CRT window phase residuals (eqn. (3.16)) after optimization converges at time 200 sec. The measurement that has the residual error around -0.025 m is suspicious and could be removed by advanced fault detection methods. Note that the cost in Fig.5.15 defined in eqn. (4.32) weights the residual from Fig. 5.16 by its inverse covariance, which accounts for the large difference in the scales of the two plots. The opportunity of removing suspicious measurements and recomputing the trajectory estimate is an advantage of the CRT estimator. Such methods are an interesting topic for future research.

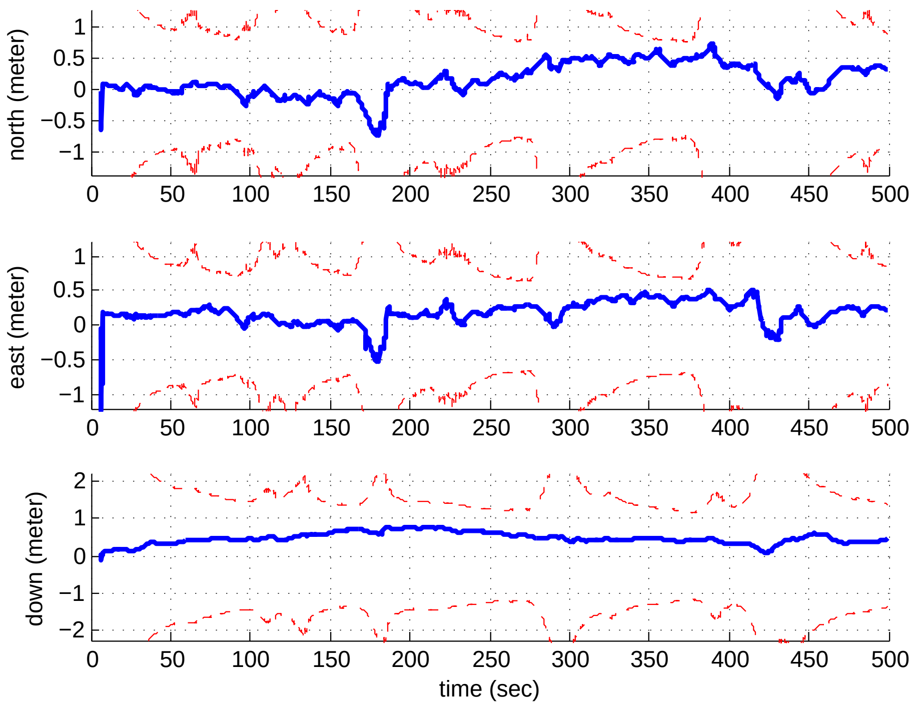


Figure 5.7: Position error.

The normal equation gives the solution to the relative error of about $\mathcal{O}(c\bar{\epsilon})$ where c is the condition number of $\mathbf{\Lambda}$ and $\bar{\epsilon}$ is the machine precision (10^{-16} for double precision floating point, and 10^{-7} for single precision floating point). The desired relative accuracy of the solution is about 10^{-6} [57]. Using double precision floating point, the condition number c is safe for $c \leq 10^{10}$. The condition number of $\mathbf{\Lambda}$ over time is given in Fig. 5.17. For most of the time, we can obtain a solution with the relative accuracy of 10^{-6}.

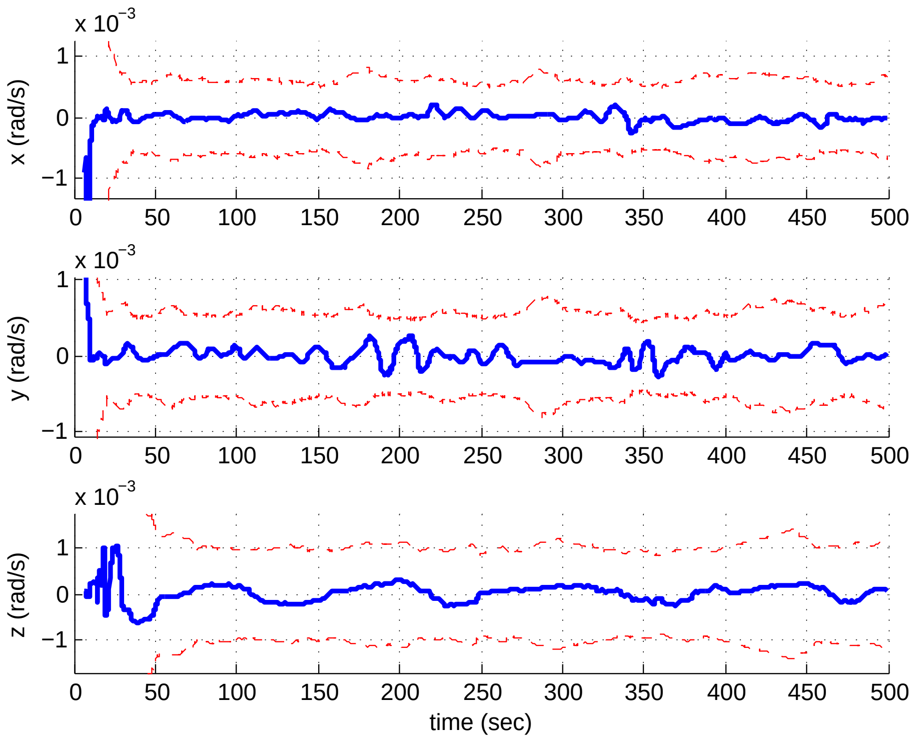


Figure 5.8: Velocity error.

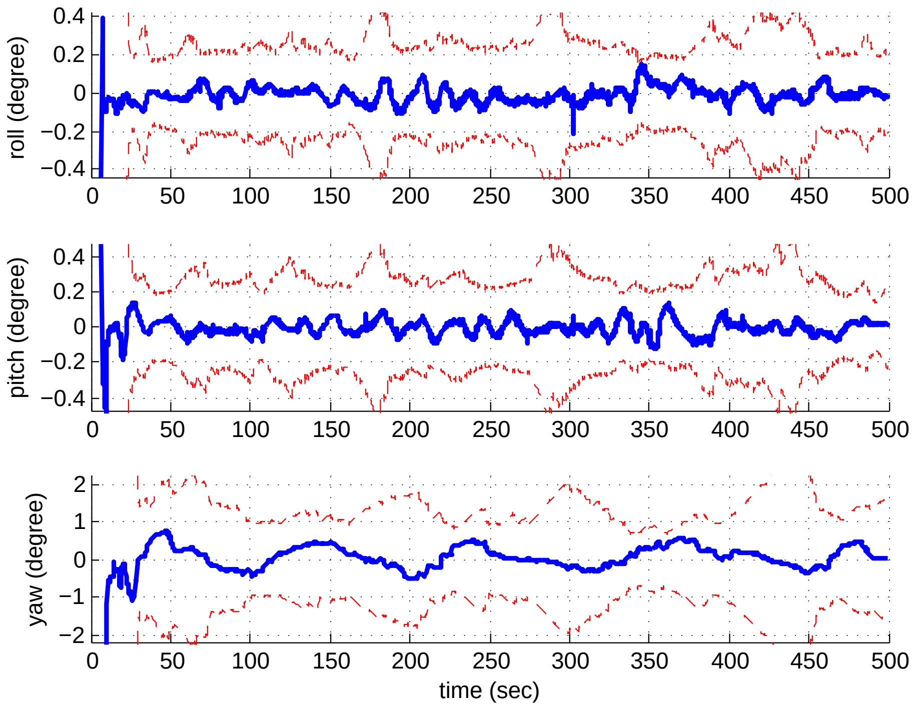


Figure 5.9: Attitude error.

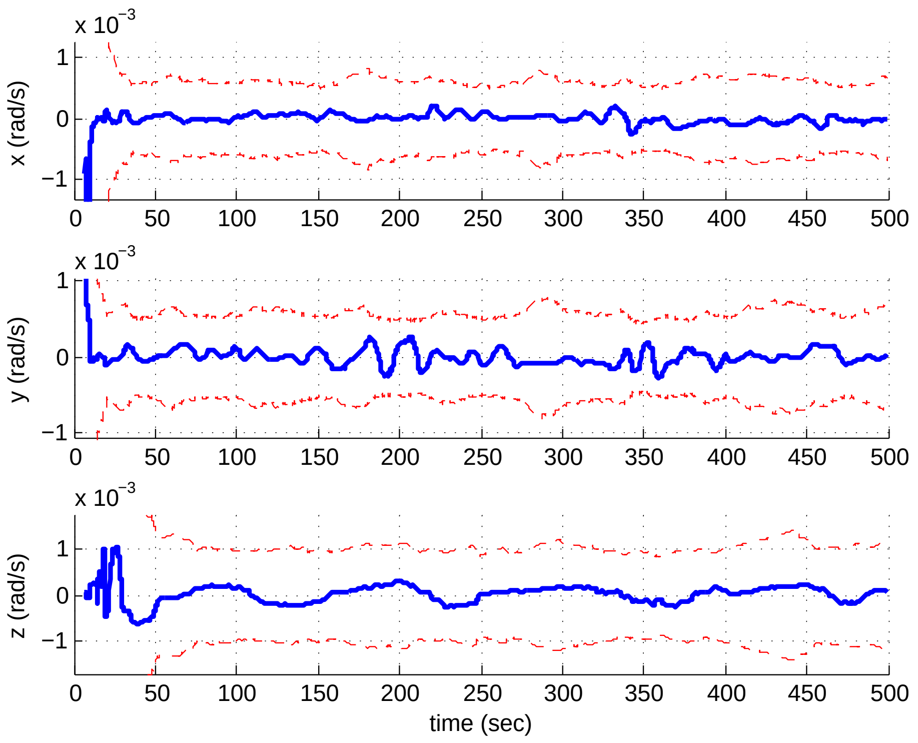


Figure 5.10: Gyro bias error.

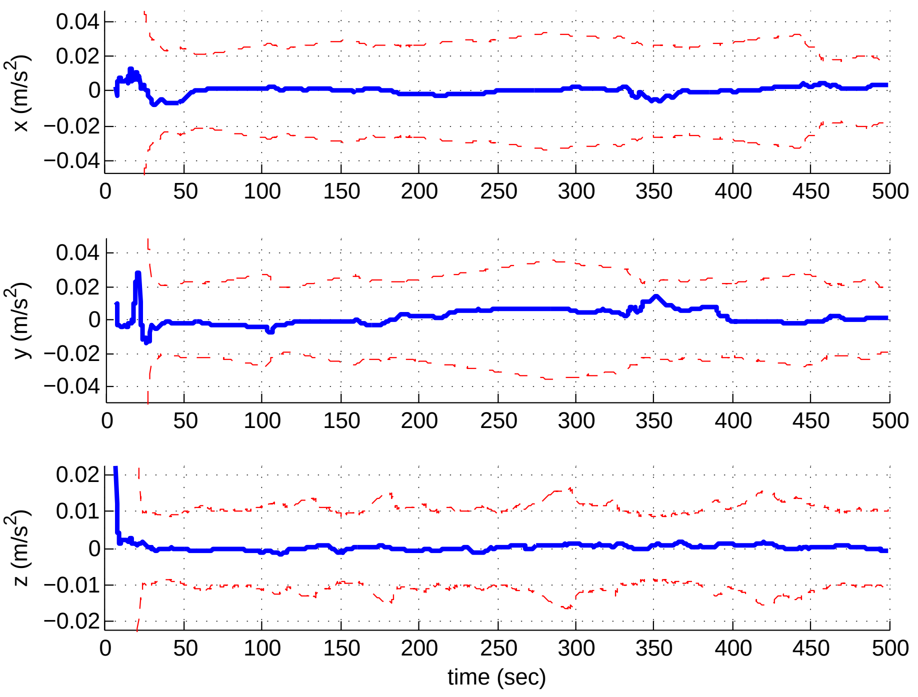


Figure 5.11: Accelerometer bias error.

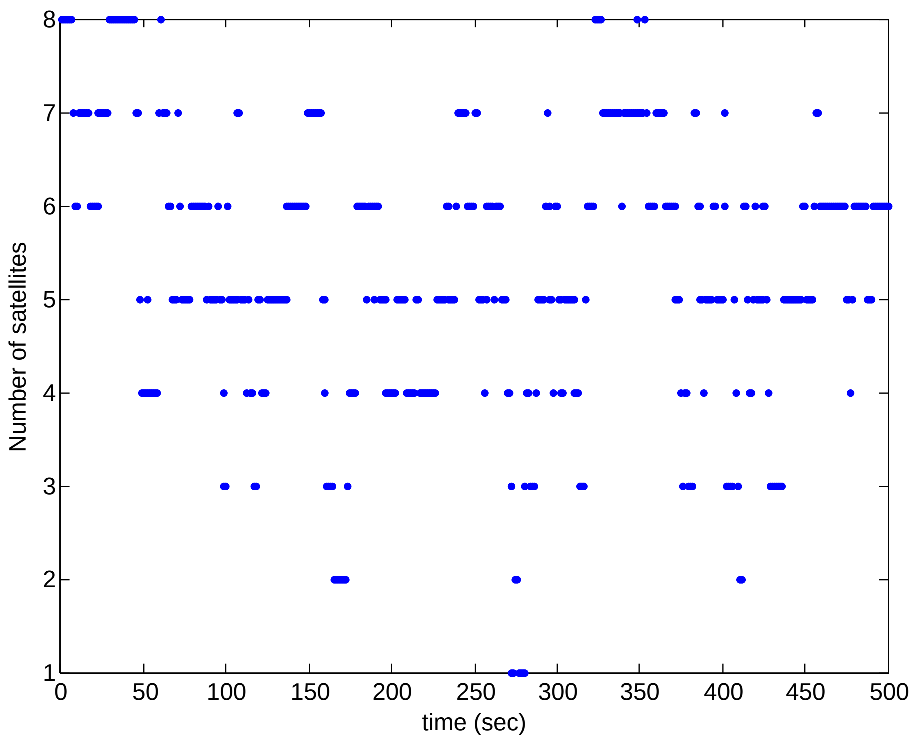
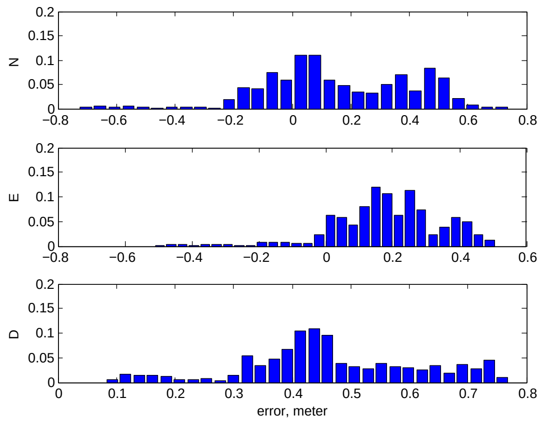
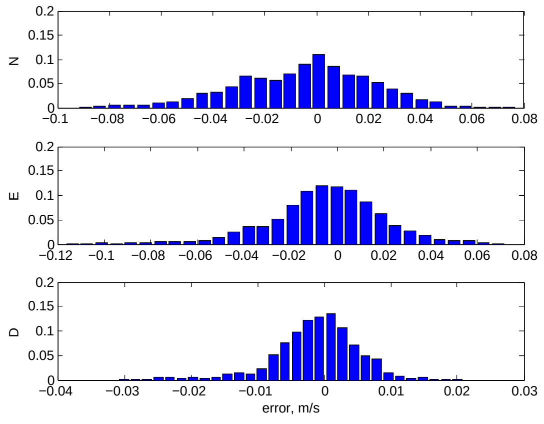


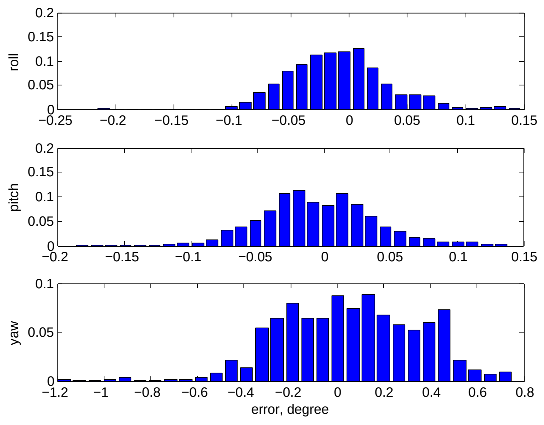
Figure 5.12: The number of satellites.



(a) The histogram of position errors.



(b) The histogram of velocity errors.



(c) The histogram of attitude errors.

Figure 5.13: Error histograms (position, velocity and attitude) for the entire trajectory (10sec-500sec). The CRT estimator uses pseudorange and integer-free phase measurements. The y-axis is the percentage.

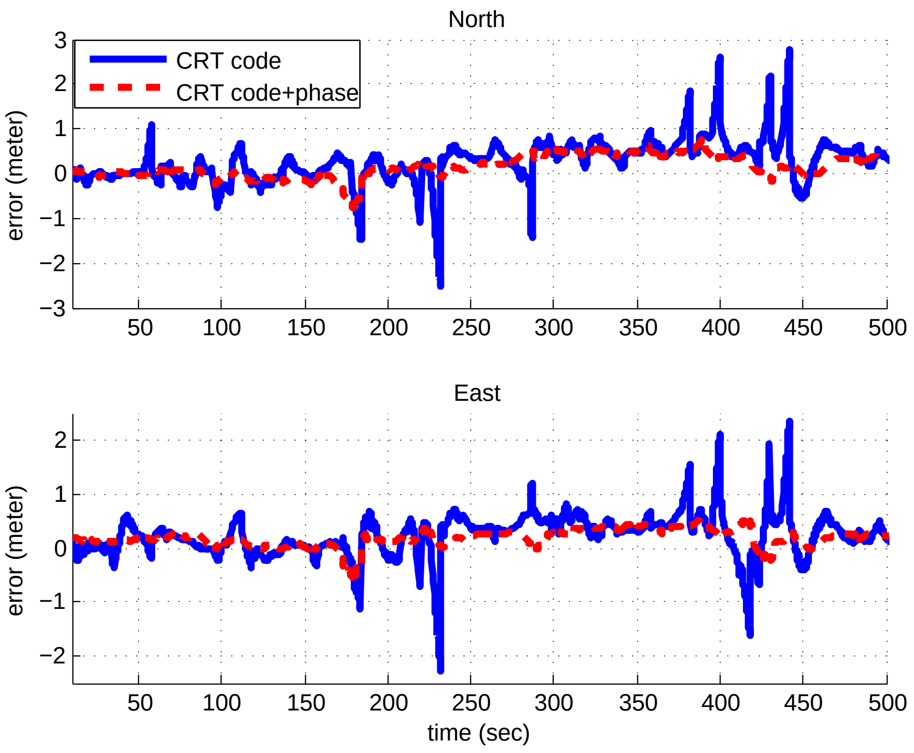


Figure 5.14: North and East position error comparison for CRT estimator with and without phase measurements.

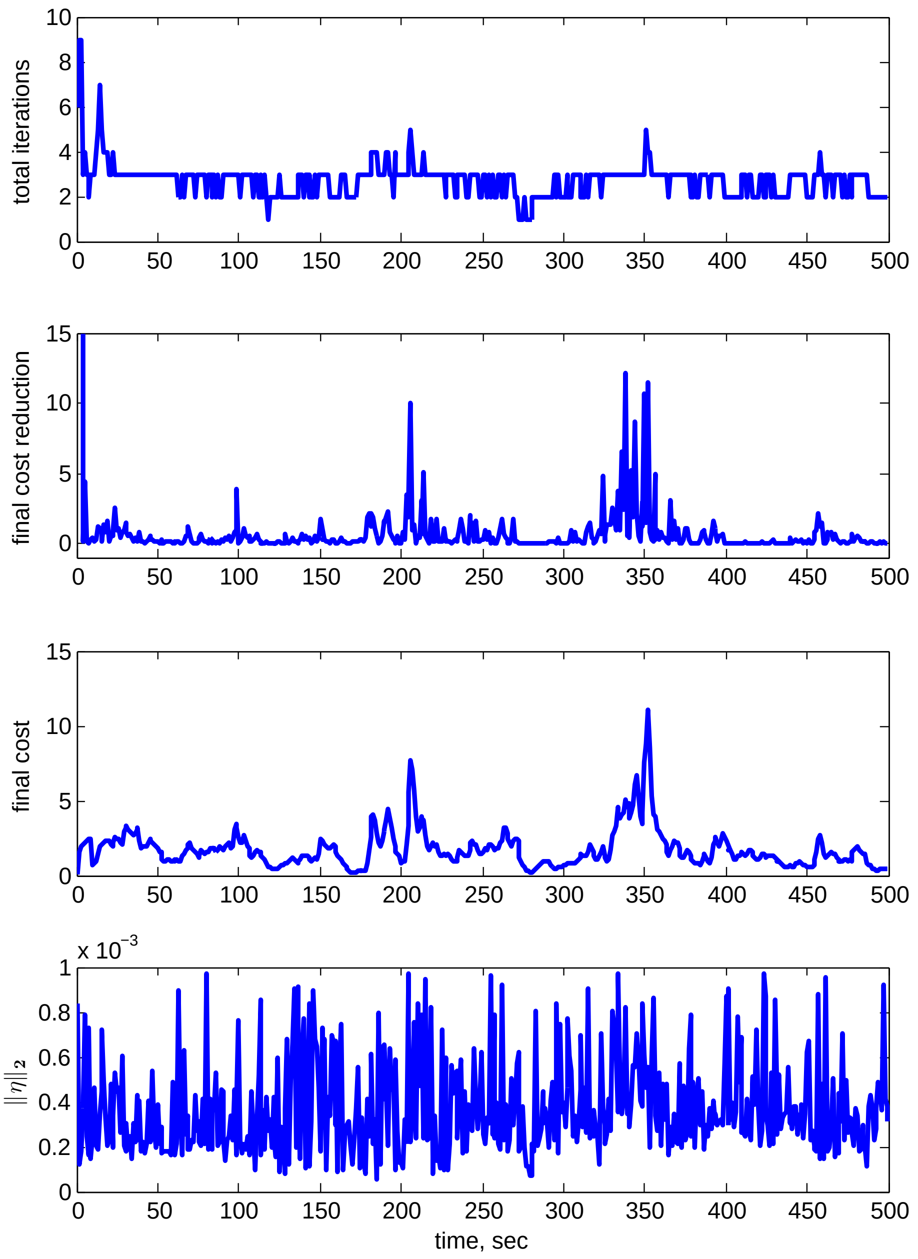


Figure 5.15: The total number of iterations ζ, the final cost reduction ΔC^{ζ}, the final cost C^{ζ}, and the final value of $\|\boldsymbol{\eta}\|_2$ in the optimization.

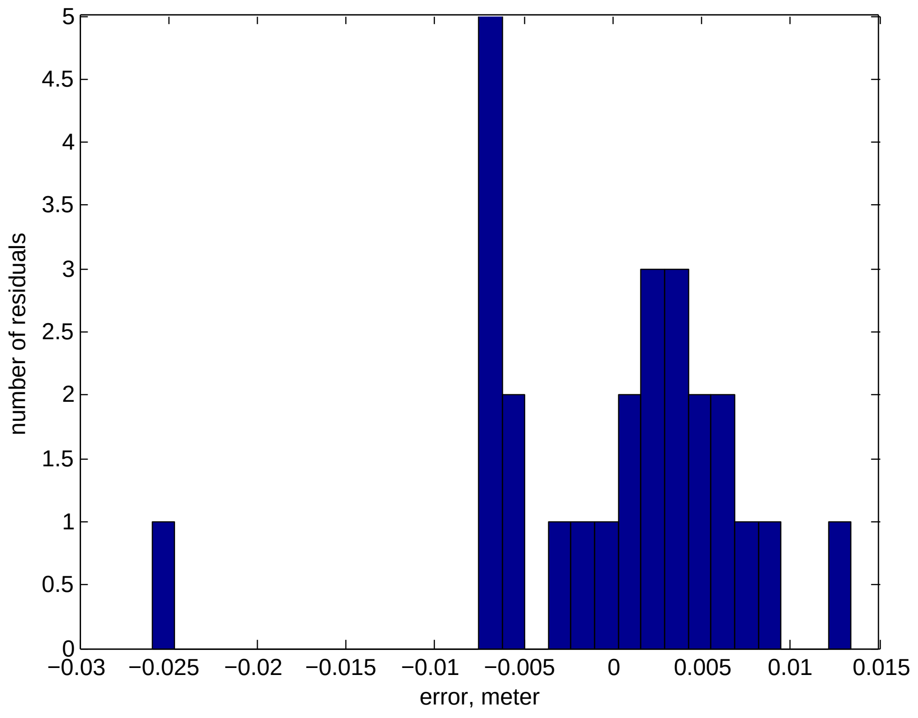


Figure 5.16: The histogram of the posterior CRT window phase residuals (eqn. (3.16)) at time 200sec.

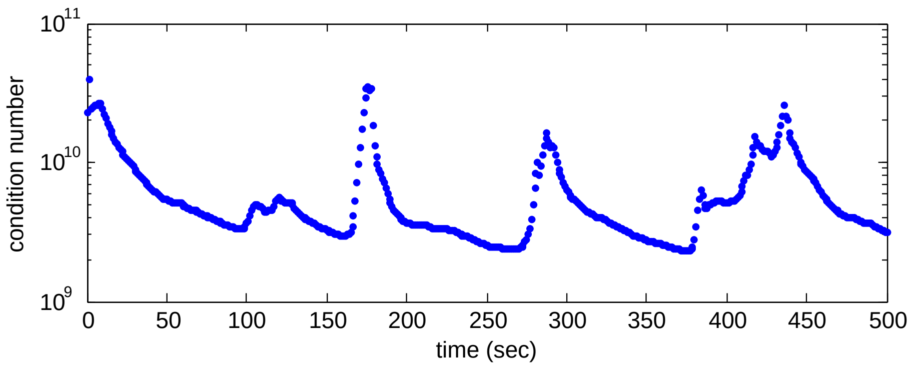


Figure 5.17: The condition number of $\mathbf{\Lambda}$ over time.

Chapter 6

Trajectory Tracking Control

6.1 System Dynamics

The rigid body dynamics of a VTOL UAV are:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{6.1}$$

$$\dot{\mathbf{v}} = \frac{F}{m}{}^{G}_{B}\mathbf{R}\mathbf{e}_3 - g\mathbf{e}_3 \tag{6.2}$$

$${}^{B}_{G}\dot{\bar{\mathbf{q}}} = \frac{1}{2}\bar{\mathbf{q}}_{\boldsymbol{\omega}^{B}_{GB}} \otimes {}^{B}_{G}\bar{\mathbf{q}} \tag{6.3}$$

$$\mathbf{J}\dot{\boldsymbol{\omega}}^{B}_{GB} = -\boldsymbol{\omega}^{B}_{GB} \times \mathbf{J}\boldsymbol{\omega}^{B}_{GB} + \boldsymbol{\tau} \tag{6.4}$$

where $\mathbf{e}_3 = [0\ 0\ 1]^\top$, F is the collective force, $\boldsymbol{\tau}$ is the torque w.r.t. the center of gravity of the vehicle, $\mathbf{J}$ is the body-referenced inertia matrix, m is the mass of the vehicle and g the gravity constant. The definition of $\{G\}$-frame and $\{B\}$-frame is illustrated by using the example of a quadrotor in Fig. 6.1.

In this paper, the standard backstepping controller is derived first, including the attitude and force command extraction steps. Then command filters are designed to generate the required virtual control signals and their derivatives required at each step for implementation. This is done in a manner that maintains the desirable stability

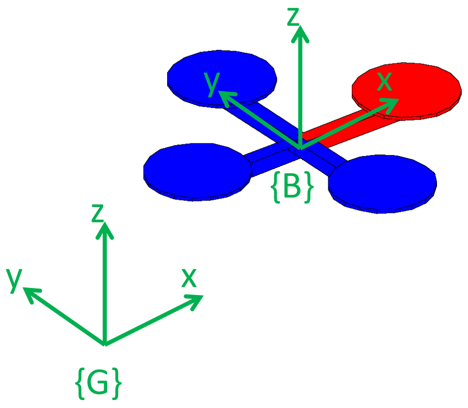


Figure 6.1: The definitions of global and body frames

properties of the command filter implementation, relative to the standard backstepping approach, as discussed in [16].

6.2 Backstepping Control

This section derives the quaternion-based backstepping control laws for the position trajectory tracking task. This derivation is similar to the control laws presented in [48]. The subscript c denotes the virtual control variable. The gains $\mathbf{K}_i \in \Re^{3\times 3}$, $i = 1...4$ are positive definite matrices.

Step 1 (Position Control)

Assume that the continuous signals $\mathbf{p}_d(t)$ and $\dot{\mathbf{p}}_d(t)$ are given. The goal in this step is to define a signal $\mathbf{v}_c(t)$ that causes $\mathbf{p}(t)$ to track the desired position $\mathbf{p}_d(t)$. The position tracking error dynamics are

$$\dot{\tilde{\mathbf{p}}} = \mathbf{v}_c + \tilde{\mathbf{v}} - \dot{\mathbf{p}}_d \tag{6.5}$$

where $\tilde{\mathbf{v}} = \mathbf{v} - \mathbf{v}_c$ and $\tilde{\mathbf{p}} = \mathbf{p} - \mathbf{p}_d$. Let the Lyapunov function be

$$V_1 = \frac{1}{2}\tilde{\mathbf{p}}^\top \tilde{\mathbf{p}}. \tag{6.6}$$

The time derivative of V_1 along solutions of eqn. (6.5) is

$$\dot{V}_1 = \tilde{\mathbf{p}}^\top(\mathbf{v}_c + \tilde{\mathbf{v}} - \dot{\mathbf{p}}_d). \tag{6.7}$$

Choosing $\mathbf{v}_c = -\mathbf{K}_1\tilde{\mathbf{p}} + \dot{\mathbf{p}}_d$, yields $\dot{V}_1 = -\tilde{\mathbf{p}}^\top\mathbf{K}_1\tilde{\mathbf{p}} + \tilde{\mathbf{p}}^\top\tilde{\mathbf{v}}$.

Step 2 (Velocity Control)

Assume that the continuous signals $\mathbf{v}_c(t)$ and $\dot{\mathbf{v}}_c(t)$ are given. The goal in this step is to define a signal $\boldsymbol{\mu}_c(t)$ that causes $\mathbf{v}(t)$ to track the desired velocity $\mathbf{v}_c(t)$. The velocity tracking error dynamic is

$$\dot{\tilde{\mathbf{v}}} = \frac{F}{m}\,{}^G_B\mathbf{R}\,\mathbf{e}_3 - g\mathbf{e}_3 - \dot{\mathbf{v}}_c = \boldsymbol{\mu}_c + \tilde{\boldsymbol{\mu}} - g\mathbf{e}_3 - \dot{\mathbf{v}}_c$$

where $\boldsymbol{\mu} = \frac{F}{m}\,{}^G_B\mathbf{R}\,\mathbf{e}_3$, $\boldsymbol{\mu}_c$ is the desired value of $\boldsymbol{\mu}$, and $\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu} - \boldsymbol{\mu}_c$. The desired force F_c and the desired attitude ${}^G_{B^c}\bar{\mathbf{q}}$ will be determined from $\boldsymbol{\mu}_c$ at Step 3. Let the Lyapunov function be

$$V_2 = V_1 + \frac{1}{2}\tilde{\mathbf{v}}^\top\tilde{\mathbf{v}}. \tag{6.8}$$

Then the time derivative of V_2 is

$$\dot{V}_2 = \dot{V}_1 + \tilde{\mathbf{v}}^\top(\boldsymbol{\mu}_c + \tilde{\boldsymbol{\mu}} - g\mathbf{e}_3 - \dot{\mathbf{v}}_c). \tag{6.9}$$

Choosing

$$\boldsymbol{\mu}_c = -\mathbf{K}_2\tilde{\mathbf{v}} + g\mathbf{e}_3 + \dot{\mathbf{v}}_c - \tilde{\mathbf{p}}, \tag{6.10}$$

yields $\dot{V}_2 = -\tilde{\mathbf{p}}^\top\mathbf{K}_1\tilde{\mathbf{p}} - \tilde{\mathbf{v}}^\top\mathbf{K}_2\tilde{\mathbf{v}} + \tilde{\mathbf{v}}^\top\tilde{\boldsymbol{\mu}}$. In this paper, we assume $\|\boldsymbol{\mu}_c\| > 0$. This is typically the case, because if $\boldsymbol{\mu}_c = \boldsymbol{\mu} = \mathbf{0}$, then the system is in freefall.

Step 3 (Attitude & Force Extraction)

The total applied thrust vector $\boldsymbol{\mu}$ is determined by the force F and the attitude ${}^B_G\bar{\mathbf{q}}$: $\boldsymbol{\mu}(F, {}^B_G\bar{\mathbf{q}}) = \frac{F}{m}\,{}^G_B\mathbf{R}\,\mathbf{e}_3$. The desired value $\boldsymbol{\mu}_c$ defined in eqn. (6.10) is implemented through selection of a desired force F_c and desired attitude ${}^B_G\bar{\mathbf{q}}_c$ such that $\boldsymbol{\mu}(F_c, {}^B_G\bar{\mathbf{q}}_c) = \boldsymbol{\mu}_c$. The desired force is $F_c = m \|\boldsymbol{\mu}_c\|$. Let $\bar{\boldsymbol{\mu}}_c = \boldsymbol{\mu}_c / \|\boldsymbol{\mu}_c\|$.

Attitude extraction is discussed below separately depending on whether the yaw angle trajectory is or is not specified.

Case 1

When the yaw trajectory is specified, let the unit vector $\mathbf{p}_y$ be the desired yaw direction at t and assume that $\mathbf{p}_y$ is not parallel with $\boldsymbol{\mu}_c$. The desired direction cosine matrix is ${}^G_B\mathbf{R}_c = [\mathbf{y} \times \bar{\boldsymbol{\mu}}_c,\ \mathbf{y},\ \bar{\boldsymbol{\mu}}_c]$, where $\mathbf{y} = \bar{\boldsymbol{\mu}}_c \times \mathbf{p}_y$ [32]. Therefore, ${}^B_G\mathbf{R}_c = {}^G_B\mathbf{R}_c^\top$, and ${}^B_G\bar{\mathbf{q}}_c$ can be retrieved from ${}^B_G\mathbf{R}_c$ by using eqn. (D.15) in page 504 of [14] (see the faq).

Case 2

When the yaw trajectory is not specified, the desired attitude is not unique, due to the freedom to choose the yaw angle. The attitude extraction in this case is denoted as ${}^B_G\bar{\mathbf{q}}_c = \Xi(\boldsymbol{\mu}_c, \bar{\mathbf{q}}_r)$ where we have introduced a reference $\{r\}$-frame and a rotation $\bar{\mathbf{q}}_r \triangleq {}^r_G\bar{\mathbf{q}}$ to accommodate this free variable. Define $\bar{\mathbf{q}}_m \triangleq {}^r_B\bar{\mathbf{q}}$, which rotates the $\{B\}$ frame to the $\{r\}$ frame. Lemma 4 is used to obtain $\bar{\mathbf{q}}_m$ by setting $u = \mathbf{e}_3$ and $v = \mathbf{R}(\bar{\mathbf{q}}_r)\bar{\boldsymbol{\mu}}_c$. Then the desired quaternion is computed by ${}^B_G\bar{\mathbf{q}}_c = \bar{\mathbf{q}}_m^{-1} \otimes \bar{\mathbf{q}}_r$.

Lemma 4 *[48] Given two unit vectors u and v with $u \neq -v$, the unit quaternion $\bar{\boldsymbol{q}}$ with minimal rotation angle θ that satisfies $\boldsymbol{R}(\bar{\boldsymbol{q}}_m)u = v$ is given by*

$$\boldsymbol{\epsilon} = \sqrt{\frac{1}{2(1+u^\top v)}} S(v)u, \quad \eta = \sqrt{\frac{1+u^\top v}{2}} \tag{6.11}$$

Remark 5 *The reference attitude ($\bar{\boldsymbol{q}}_r$) could be chosen as the global frame (${}^G_G\bar{\boldsymbol{q}}$), the current vehicle attitude (${}^B_G\bar{\boldsymbol{q}}$) or the current desired attitude (${}^B_G\bar{\boldsymbol{q}}_c$). If a design goal is to choose the desired attitude such that the yaw rotation is minimized, then choosing* *the global frame as the reference attitude could result in unnecessary yaw rotation at start-up. Through simulations, we find that using the current desired attitude ($^B_G\bar{\mathbf{q}}_c$) as the reference attitude results in smaller yaw rotation during the trajectory tracking than using the current vehicle attitude.*

Step 4 (Attitude Control)

Assume that the continuous signals F_c, $^B_G\bar{\mathbf{q}}_c$ and $\bar{\boldsymbol{\omega}}$ are available. To simplify notation, we define $\bar{\mathbf{q}} \triangleq {}^B_G\bar{\mathbf{q}}$ and $\bar{\mathbf{q}}_c \triangleq {}^B_G\bar{\mathbf{q}}_c$, and let $\bar{\boldsymbol{\omega}} \triangleq \bar{\boldsymbol{\omega}}^{B^c}_{GB^c}$ represent the rotational velocity vector of $\bar{\mathbf{q}}_c$ whose dynamic is defined in eqn. (2.16). Computation of $\bar{\boldsymbol{\omega}}$ is discussed in Section 6.4. The goal in this step is to choose $\boldsymbol{\omega}_c$ to ensure the attitude of vehicle $^B_G\bar{\mathbf{q}}(t)$ tracks the desired attitude $\bar{\mathbf{q}}_c(t)$.

The attitude tracking error $\tilde{q}$ is defined as

$$\tilde{q} = {}^B_G\bar{\mathbf{q}} \otimes \bar{\mathbf{q}}_c^{-1}. \tag{6.12}$$

Thus, using (2.9) we have

$$\tilde{q} = \bar{\mathbf{q}} \otimes \bar{\mathbf{q}}_c^{-1} \text{ and } \mathbf{R}(\tilde{q})\mathbf{R}(\bar{\mathbf{q}}_c) = \mathbf{R}(\bar{\mathbf{q}}). \tag{6.13}$$

With this definition of attitude tracking error, the dynamic of $\tilde{q}$ is

$$\dot{\tilde{q}} = \frac{1}{2}\Phi(\tilde{q})\left(\boldsymbol{\omega} - \mathbf{R}(\tilde{q})\bar{\boldsymbol{\omega}}\right), \tag{6.14}$$

$$= \frac{1}{2}\Phi(\tilde{q})(\boldsymbol{\omega}_c + \tilde{\boldsymbol{\omega}} - \mathbf{R}(\tilde{q})\bar{\boldsymbol{\omega}}), \tag{6.15}$$

where $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} - \boldsymbol{\omega}_c$ and $\boldsymbol{\omega} \triangleq \boldsymbol{\omega}^B_{GB}$ to simplify the notation. The derivation can be found in [60].

Let the Lyapunov function be

$$V_3 = V_2 + 2(1 - \tilde{h}\tilde{\eta}) \tag{6.16}$$

where $\tilde{\eta}$ is the scalar part of $\tilde{q}$, and $\tilde{h} \in \{-1, 1\}$ is a hybrid variable introduced in [39] that determines the convergence point of $\tilde{q}$ to either $[\mathbf{0}\ 1]^\top$ or $[\mathbf{0}\ -1]^\top$. The dynamics of $\tilde{h}$ contain both continuous and discontinuous parts:

$$\begin{cases} \dot{\tilde{h}} = 0, & \text{if} \quad \mathbf{x} \in C \\ \tilde{h}^+ = -\tilde{h}, & \text{if} \quad \mathbf{x} \in D \end{cases} \tag{6.17}$$

where $\tilde{h}^+$ denotes the discrete update of the variable, $\mathbf{x}$ is the system state vector, C is the flow set and D is the jump set [38]. The flow set and the jump set will be defined in the next control step.

The time derivative of V_3 is

$$\dot{V}_3 = \dot{V}_3^s + \tilde{\mathbf{v}}^\top \tilde{\boldsymbol{\mu}} + \tilde{h}\tilde{\boldsymbol{\epsilon}}^\top(\boldsymbol{\omega}_c + \tilde{\boldsymbol{\omega}} - \mathbf{R}(\tilde{q})\bar{\boldsymbol{\omega}}) \tag{6.18}$$

where $\dot{V}_3^s = -\tilde{\mathbf{p}}^\top \mathbf{K}_1 \tilde{\mathbf{p}} - \tilde{\mathbf{v}}^\top \mathbf{K}_2 \tilde{\mathbf{v}}$. To proceed, we rewrite $\tilde{\boldsymbol{\mu}}$ as $\tilde{\boldsymbol{\mu}} = \mathbf{W}(\bar{\mathbf{q}}_c, \tilde{q}, F_c)\tilde{\boldsymbol{\epsilon}}$ [60]. This yields

$$\dot{V}_3 = \dot{V}_3^s + \tilde{\mathbf{v}}^\top \mathbf{W}\tilde{\boldsymbol{\epsilon}} + \tilde{h}\tilde{\boldsymbol{\epsilon}}^\top(\boldsymbol{\omega}_c + \tilde{\boldsymbol{\omega}} - \mathbf{R}(\tilde{q})\bar{\boldsymbol{\omega}}). \tag{6.19}$$

Choosing

$$\boldsymbol{\omega}_c = -\mathbf{K}_3\tilde{h}\tilde{\boldsymbol{\epsilon}} - \mathbf{W}^\top \tilde{\mathbf{v}} + \mathbf{R}(\tilde{q})\bar{\boldsymbol{\omega}} \tag{6.20}$$

and noticing that $\tilde{h}^2 = 1$, we have

$$\dot{V}_3 = \dot{V}_3^s - \tilde{\boldsymbol{\epsilon}}^\top \mathbf{K}_3 \tilde{\boldsymbol{\epsilon}} + \tilde{h}\tilde{\boldsymbol{\epsilon}}^\top \tilde{\boldsymbol{\omega}}. \tag{6.21}$$

Step 5 (Angular Velocity Control)

In this step, the goal is choose $\boldsymbol{\tau}$ to cause the actual angular rate $\boldsymbol{\omega}(t)$ track the desired angular velocity $\boldsymbol{\omega}_c(t)$. We assume $\boldsymbol{\omega}_c$ and $\dot{\boldsymbol{\omega}}_c$ are available, continuous, and known.

The dynamic of the angular velocity tracking error $\tilde{\boldsymbol{\omega}}$ is

$$\mathbf{J}\dot{\tilde{\boldsymbol{\omega}}} = \boldsymbol{\Sigma}(\tilde{\boldsymbol{\omega}}, \boldsymbol{\omega}_c)\tilde{\boldsymbol{\omega}} - S(\boldsymbol{\omega}_c)\mathbf{J}\tilde{\boldsymbol{\omega}} + \boldsymbol{\tau}_f(\boldsymbol{\omega}_c, \dot{\boldsymbol{\omega}}_c) + \boldsymbol{\tau} \tag{6.22}$$

where

$$\boldsymbol{\Sigma}(\tilde{\boldsymbol{\omega}}, \boldsymbol{\omega}_c) = S(\mathbf{J}\tilde{\boldsymbol{\omega}}) + S(\mathbf{J}\boldsymbol{\omega}_c) \tag{6.23}$$

$$\boldsymbol{\tau}_f(\boldsymbol{\omega}_c, \dot{\boldsymbol{\omega}}_c) = S(\mathbf{J}\boldsymbol{\omega}_c)\boldsymbol{\omega}_c - \mathbf{J}\dot{\boldsymbol{\omega}}_c. \tag{6.24}$$

Note that $\boldsymbol{\Sigma}$ is a skew-symmetric matrix for any $\tilde{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}_c$.

Choosing the Lyapunov function as

$$V_4 = V_3 + \tilde{\boldsymbol{\omega}}^\top \mathbf{J}\tilde{\boldsymbol{\omega}}, \tag{6.25}$$

yields the time derivative of V_4 as

$$\dot{V}_4 = \dot{V}_4^s + \tilde{h}\tilde{\boldsymbol{\epsilon}}^\top \tilde{\boldsymbol{\omega}} + \tilde{\boldsymbol{\omega}}^\top(\boldsymbol{\Sigma}\tilde{\boldsymbol{\omega}} - S(\boldsymbol{\omega}_c)\mathbf{J}\tilde{\boldsymbol{\omega}} + \boldsymbol{\tau}_f + \boldsymbol{\tau}) \tag{6.26}$$

where $\dot{V}_4^s = \dot{V}_3^s - \tilde{\boldsymbol{\epsilon}}^\top \mathbf{K}_3 \tilde{\boldsymbol{\epsilon}}$. Choosing

$$\boldsymbol{\tau} = -\mathbf{K}_4\tilde{\boldsymbol{\omega}} - \tilde{h}\tilde{\boldsymbol{\epsilon}} + S(\boldsymbol{\omega}_c)\mathbf{J}\tilde{\boldsymbol{\omega}} - \boldsymbol{\tau}_f \tag{6.27}$$

and noticing that $\tilde{\boldsymbol{\omega}}^\top \boldsymbol{\Sigma}\tilde{\boldsymbol{\omega}} = 0$ yields

$$\dot{V}_4 = \dot{V}_4^s - \tilde{\boldsymbol{\omega}}^\top \mathbf{K}_4\tilde{\boldsymbol{\omega}} \tag{6.28}$$

which is a negative definite function of the error state. The change in V_4 following a jump in $\tilde{h}$ can be shown, using eqns. (6.16), (6.20) and (6.25) and the procedure on page 2526 in [38], to be

$$V_4(\tilde{h}^+) - V_4(\tilde{h}) = 4\tilde{h}(\tilde{\eta} - \tilde{\boldsymbol{\epsilon}}^\top \mathbf{K}_3^\top \mathbf{J}\mathbf{G}) \tag{6.29}$$

where $\mathbf{G} = \boldsymbol{\omega} + \mathbf{W}^\top \tilde{\mathbf{v}} - \mathbf{R}(\tilde{q})\bar{\boldsymbol{\omega}}$. Letting $\boldsymbol{\Lambda} = \tilde{\eta} - \tilde{\boldsymbol{\epsilon}}^\top \mathbf{K}_3^\top \mathbf{G}$, then the flow and jump sets can be defined as

$$C_1 = \{\tilde{h}\boldsymbol{\Lambda} \geq -\delta\} \tag{6.30}$$

$$D_1 = \{\tilde{h}\boldsymbol{\Lambda} \leq -\delta\} \tag{6.31}$$

where $\delta \in (0, 1)$ is a design variable that determines at what point to switch the convergence point. Therefore, by Corollary 7.7 in [49], the system is asymptotically stable. However, implementation of this control law requires computation of the various command signals and their derivatives.

6.3 Command Filtered Backstepping

The backstepping controller an signal requirements at each step are summarized below.

Step	Rqrd. sig.	Control Law
1	$\mathbf{p}_d$, $\dot{\mathbf{p}}_d$	$\mathbf{v}_c = -\mathbf{K}_1\tilde{\mathbf{p}} + \dot{\mathbf{p}}_d$
2	$\mathbf{v}_c$, $\dot{\mathbf{v}}_c$	$\boldsymbol{\mu}_c = -\mathbf{K}_2\tilde{\mathbf{v}} + g\mathbf{e}_3 + \dot{\mathbf{v}}_c - \tilde{\mathbf{p}}$
3	$\boldsymbol{\mu}_c$	$F = m\,\Vert\boldsymbol{\mu}_c\Vert$, $\bar{\mathbf{q}}_c = \Xi(\boldsymbol{\mu}_c, \bar{\mathbf{q}}_c)$
4	$\bar{\mathbf{q}}_c$, $\bar{\boldsymbol{\omega}}$	$\boldsymbol{\omega}_c = -\mathbf{K}_3\tilde{h}\tilde{\boldsymbol{\epsilon}} - \mathbf{W}^\top\tilde{\mathbf{v}} + \mathbf{R}(\tilde{q})\bar{\boldsymbol{\omega}}$
5	$\boldsymbol{\omega}_c$, $\dot{\boldsymbol{\omega}}_c$	$\boldsymbol{\tau} = -\mathbf{K}_4\tilde{\boldsymbol{\omega}} - \tilde{h}\tilde{\boldsymbol{\epsilon}} + S(\boldsymbol{\omega}_c)\mathbf{J}\tilde{\boldsymbol{\omega}} - \boldsymbol{\tau}_f$

In the standard backstepping approach, the variables $\dot{\mathbf{v}}_c$, $\bar{\boldsymbol{\omega}}$ and $\dot{\boldsymbol{\omega}}_c$ are derived analytically. This is cumbersome as shown in [48], which only considered position tracking. For the position and yaw tracking, the analytic derivation would become even more cumbersome as all derivatives in [48] must re-computed to include the commanded yaw direction. Command filters are employed herein to automate the computation.

The command filtered backstepping controller is summarized below:

1. $\mathbf{v}_c^o = -\mathbf{K}_1\tilde{\mathbf{p}} + \dot{\mathbf{p}}_d$; $\qquad [\mathbf{v}_c, \dot{\mathbf{v}}_c] = CF_1(\mathbf{v}_c^o)$.

2. $\boldsymbol{\mu}_c^o = -\mathbf{K}_2\tilde{\mathbf{v}} + g\mathbf{e}_3 + \dot{\mathbf{v}}_c - \tilde{\mathbf{p}}$.

3. $F_c = m\,\|\boldsymbol{\mu}_c^o\|$, $\bar{\mathbf{q}}_c^o = \Xi(\boldsymbol{\mu}_c^o, \bar{\mathbf{q}}_c)$; $\qquad [\bar{\mathbf{q}}_c, \bar{\boldsymbol{\omega}}] = QF(\bar{\mathbf{q}}_c^o)$.

4. $\boldsymbol{\omega}_c^o = -\mathbf{K}_3\tilde{h}\tilde{\boldsymbol{\epsilon}} - \mathbf{W}^\top\tilde{\mathbf{v}} + \mathbf{R}(\tilde{q})\bar{\boldsymbol{\omega}}$; $\qquad [\boldsymbol{\omega}_c, \dot{\boldsymbol{\omega}}_c] = CF_2(\boldsymbol{\omega}_c^o)$.

5. $\boldsymbol{\tau} = -\mathbf{K}_4\tilde{\boldsymbol{\omega}} - \tilde{h}\tilde{\boldsymbol{\epsilon}} + S(\boldsymbol{\omega}_c)\mathbf{J}\tilde{\boldsymbol{\omega}} - \boldsymbol{\tau}_f$.

where QF is the quaternion command filter discussed in Section 6.4, and CF_1 and CF_2 are command filters defined as

$$\dot{\mathbf{x}}_1 = \mathbf{x}_2 \tag{6.32}$$

$$\dot{\mathbf{x}}_2 = -\omega_n^2(\mathbf{x}_1 - \mathbf{u}) - 2\xi\omega_n\mathbf{x}_2 \tag{6.33}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are in $\Re^3$, ξ is the damping ratio, ω_n is the natural frequency, $\mathbf{u} = \mathbf{v}_c^o$ for CF_1 and $\mathbf{u} = \boldsymbol{\omega}_c^o$ for CF_2. The outputs of the filters are $\mathbf{x}_1$ and $\mathbf{x}_2$. For example, in the case of CF_1, $\mathbf{v}_c(t) = \mathbf{x}_1(t)$, $\dot{\mathbf{v}}_c(t) = \mathbf{x}_2(t)$.

The analysis in [16] ensures that as ω_n is increased, the tracking error performance of the command filtered implementation approaches the performance of the backstepping controller using analytically derived command derivatives. To complete the derivation for this application, we must present a quaternion command filter and show that it fits within the framework of [16].

6.4 Second-order Quaternion Filter

This section develops a second-order quaternion filter that takes the quaternion $\bar{\mathbf{q}}_c^o(t)$ as the input and produces the filtered quaternion $\bar{\mathbf{q}}_c(t)$ and the corresponding angular velocity $\bar{\boldsymbol{\omega}}(t)$ as outputs. The purpose of the filter is to ensure producing $\bar{\boldsymbol{\omega}}(t)$ without differentiation and to ensure that the error between $\bar{\mathbf{q}}_c^o(t)$ and $\bar{\mathbf{q}}_c(t)$, defined as

$$\tilde{\bar{\mathbf{q}}} = \bar{\mathbf{q}}_c \otimes (\bar{\mathbf{q}}_c^o)^{-1}, \tag{6.34}$$

is small. The symbol $\tilde{\boldsymbol{\epsilon}}$ represents the vector part of ${}^{\hat{B}}_{B}\tilde{\bar{\mathbf{q}}}$. Both $\bar{\mathbf{q}}_c^o(t)$ and $\bar{\mathbf{q}}_c(t)$ are known and available at every time instant; hence, $\tilde{\bar{\mathbf{q}}}$ and $\tilde{\boldsymbol{\epsilon}}$ can be computed at every time instant.

The proposed quaternion filter is

$$\dot{\bar{\mathbf{q}}}_c = \frac{1}{2}\Phi(\bar{\mathbf{q}}_c)\bar{\boldsymbol{\omega}} \tag{6.35}$$

$$\dot{\bar{\boldsymbol{\omega}}} = \alpha(-\tilde{k}\tilde{h}_f\tilde{\boldsymbol{\epsilon}} - \bar{\boldsymbol{\omega}}) \tag{6.36}$$

where $\tilde{h}_f$ is a hybrid variable as defined in (6.17), and α, $\tilde{k} \in \mathbb{R}_+$. The form of eqn. (6.35) is designed to maintain the unit norm property for $\bar{\mathbf{q}}_c$. The form of eqn. (6.36) is designed to cause $\bar{\mathbf{q}}_c(t)$ to track $\bar{\mathbf{q}}_c^o(t)$. In this filter, the parameter α determines how fast $\bar{\boldsymbol{\omega}}$ tracks $-\tilde{k}\tilde{h}_f\tilde{\boldsymbol{\epsilon}}$ and $\tilde{k}$ determines how fast $\bar{\mathbf{q}}_c$ tracks the input attitude $\bar{\mathbf{q}}_c^o$.

The dynamic of $\tilde{\bar{\mathbf{q}}}$ is [60]

$$\dot{\tilde{\bar{\mathbf{q}}}} = \frac{1}{2}\Phi(\tilde{\bar{\mathbf{q}}})\tilde{\boldsymbol{\omega}} \tag{6.37}$$

where $\tilde{\boldsymbol{\omega}} = \bar{\boldsymbol{\omega}} - \mathbf{R}(\tilde{\bar{\mathbf{q}}})\bar{\boldsymbol{\omega}}^o$, and $\bar{\boldsymbol{\omega}}^o$ is the angular rate of the $\bar{\mathbf{q}}_c^o$ which is not available (i.e., unknown).

The stability of the filter is shown by considering the zero input case. Define the Lyapunov function $V = 2\alpha\tilde{k}(1 - \tilde{h}_f\tilde{\eta}) + \tilde{\boldsymbol{\omega}}^\top\tilde{\boldsymbol{\omega}}$, where $\tilde{\eta}$ is the scalar part of $\tilde{\bar{\mathbf{q}}}$. Then the time derivative of V can be computed as

$$\dot{V} = \alpha\tilde{k}\tilde{h}_f\tilde{\boldsymbol{\epsilon}}^\top\tilde{\boldsymbol{\omega}} + \tilde{\boldsymbol{\omega}}^\top(-\alpha\tilde{k}\tilde{h}_f\tilde{\boldsymbol{\epsilon}} - \alpha\tilde{\boldsymbol{\omega}}) \tag{6.38}$$

$$= -\alpha\tilde{\boldsymbol{\omega}}^\top\tilde{\boldsymbol{\omega}} \leq 0. \tag{6.39}$$

The change in V following the jump of $\tilde{h}_f$ is $V(\tilde{h}_f^+) - V(\tilde{h}_f) = 4\alpha\tilde{k}\tilde{h}_f\tilde{\eta}$. By defining the flow and jump sets of the filter to be

$$C_2 = \{\tilde{h}_f\tilde{\eta} \geq -\delta\} \quad \text{and} \quad D_2 = \{\tilde{h}_f\tilde{\eta} \leq -\delta\}, \tag{6.40}$$

it follows that $V(\tilde{h}_f^+) - V(\tilde{h}_f) < 0$. Since $\dot{V} = 0$ if only if $\tilde{\boldsymbol{\omega}} = 0$ and $\{\mathbf{x} \in D_2 : V(\tilde{h}_f^+) - V(\tilde{h}_f) = 0\} = \emptyset$, by Theorem 4.7 in [49] the filter is global asymptotically stable.

Remark 6 *The vector command filter of eqns. (6.32-6.33) could also be used to produce* $\bar{\boldsymbol{q}}_c(t)$ *and* $\bar{\boldsymbol{\omega}}(t)$ *from* $\bar{\boldsymbol{q}}_c^o(t)$. *However, that filter would not maintain the quaternion having unit norm and would not necessarily track the input quaternion through the path of minimal rotation. In contrast, the hybrid variable* $\tilde{h}_f$ *in the proposed quaternion filter ensures the input quaternion is tracked through the path of minimal rotation. Moreover, the state vector in the proposed quaternion filter has smaller size than the one in the vector-based filter (7 as opposed to 8).*

Remark 7 *Either a normalization step or the Lie group variational integrator, see e.g., [33], can be used to ensure the unit norm property of the quaternion in practice.*

6.5 Stability Analysis

The only difference between the proposed controller in this paper and the controller in [16] is the involvement of the quaternion filter. Therefore, this section shows that the quaternion filter can be written into the form of $\dot{\mathbf{z}} = \bar{\boldsymbol{\epsilon}} F(\cdot)$, which is required to apply the singular perturbation analysis as presented in [16]. Once this is shown, the stability of the proposed controller in this paper can be proved in an identical manner as was done in [16].

Eqn. (6.36) can be rewritten as

$$\dot{\bar{\boldsymbol{\omega}}} = -\alpha \tilde{k}(\tilde{h}_f \tilde{\boldsymbol{\epsilon}}) - \alpha \bar{\boldsymbol{\omega}}. \tag{6.41}$$

Comparing eqns. (6.33) and (6.41), we note that $\tilde{h}_f\tilde{\boldsymbol{\epsilon}}$ is equivalent to $(\mathbf{x}_1 - \mathbf{u})$ and $\bar{\boldsymbol{\omega}}$ is equivalent to $\mathbf{x}_2$. Hence we can let $\alpha\tilde{k} = \omega_n^2$ and $\alpha = 2\xi\omega_n$ to obtain

$$\omega_n = \sqrt{\alpha\tilde{k}}, \quad \xi = \frac{1}{2}\sqrt{\alpha/\tilde{k}}. \tag{6.42}$$

Thus, eqn. (6.41) can be written as

$$\dot{\bar{\boldsymbol{\omega}}} = -\omega_n^2(\tilde{h}_f\tilde{\boldsymbol{\epsilon}}) - 2\xi\omega_n\bar{\boldsymbol{\omega}}. \tag{6.43}$$

Letting $\mathbf{z}_1 = \bar{\mathbf{q}}$ and $\mathbf{z}_2 = \bar{\boldsymbol{\omega}}/\omega_n$, we can rewrite the quaternion filter (eq. (6.35) and (6.36)) into the form

$$\begin{bmatrix} \dot{\mathbf{z}}_1 \\ \\ \dot{\mathbf{z}}_2 \end{bmatrix} = \omega_n \begin{bmatrix} [r]\frac{1}{2}\Phi(\mathbf{z}_1)\mathbf{z}_2 \\ \\ -\tilde{h}_f\tilde{\boldsymbol{\epsilon}} - 2\xi\mathbf{z}_2 \end{bmatrix}$$

which has the desired form $\dot{\mathbf{z}} = \bar{\boldsymbol{\epsilon}}F(\mathbf{z}, \bar{\mathbf{q}}_c)$, where $\bar{\boldsymbol{\epsilon}} = \omega_n$. Then, the singular perturbation theorem can be applied as in [16] to prove the stability of the proposed controller.

6.6 Simulation Results

First, we demonstrate in Fig. 6.2 that the vector-based filter (red) may take a non-minimum angle path while tracking the input quaternion. The vector-based filter uses eqns. (6.32-6.33) with $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\Re^4$, and $\mathbf{u} = \bar{\mathbf{q}}_c^o(t)$ is the input quaternion. The angular velocity is $\boldsymbol{\omega} = 2\Phi(\mathbf{x}_1)^\top\mathbf{x}_2$ based on eqn. (2.16) using the fact that $\Phi^\top\Phi = \mathbf{I}$. The initial quaternion is set to $[\mathbf{0}\ 1]^\top$ and the initial angular velocity is set to $\mathbf{0}$ for both the vector-based filter and the quaternion filter. The input quaternion is a constant value created by letting $\hat{\mathbf{k}} = [0\ 0\ 1]^\top$ and $\theta = 240°$ in eqn. (2.4). Fig. 6.2 shows that both filters successfully track the input quaternion, but the vector-based filter (red) rotates counter-clockwise to reach the desired attitude while the proposed quaternion (blue) filter rotates clockwise which is a shorter route to reach the desired attitude.

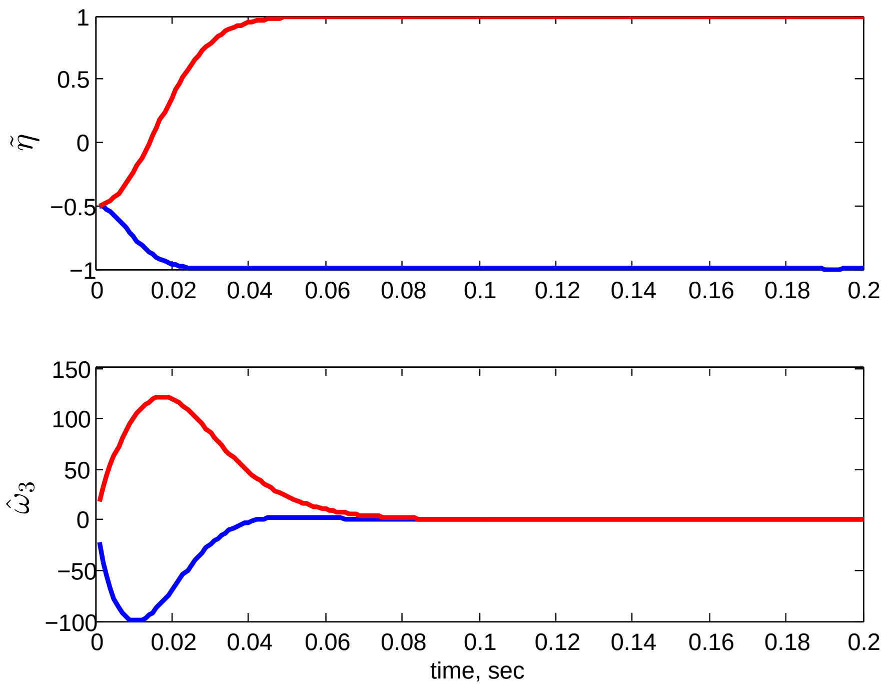


Figure 6.2: The quaternion tracking error ($\tilde{\eta}$) and the last component of the output angular velocity ($\hat{\omega}_3$) are shown for the vector-based filter (red) and the proposed quaternion filter (blue).

In the second simulation, we use quadrotor as an example of a VTOL UAV to demonstrate the performance of the proposed tracking controller. The desired trajectory in this simulation is a position trajectory which is shown in blue in Fig. 6.3 with the actual trajectory of the vehicle in red. The position tracking errors $\mathbf{p} - \mathbf{p}_d$ are plotted in Fig. 6.4. In this figure, we compare the performance of controller by setting a different α in the quaternion filter. We also compare the yaw rotation of the vehicle in the trajectory tracking by running the simulation separately with using the current desired attitude and the current vehicle attitude as the reference attitude. The comparison result given in Fig. 6.5 shows that using the current desired attitude results in lesser yaw rotation.

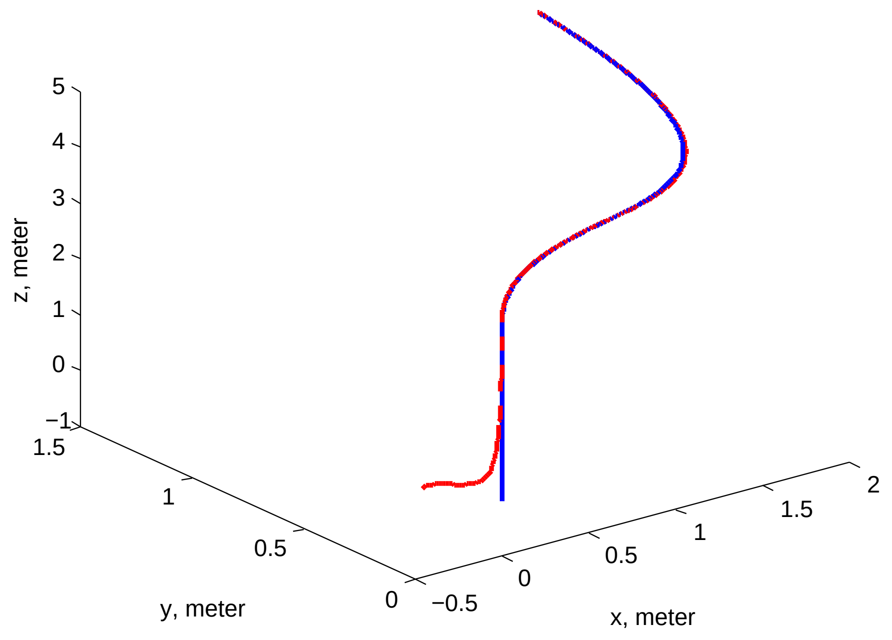


Figure 6.3: The desired trajectory (blue), and the actual trajectory of the vehicle (red). The initial position of the vehicle is $[-0.2, 0.2, 0]^\top$.

The parameters used in the simulation are: $m = 0.5$kg, $\mathbf{J} = diag([0.0820; 0.0845; 0.1377])$, $\mathbf{K}_1 = 2\mathbf{I}$, $\mathbf{K}_2 = \mathbf{I}$, $\mathbf{K}_3 = 8\mathbf{I}$, $\mathbf{K}_4 = \mathbf{I}$, $\alpha = 200$, $\tilde{k} = 50$, $\delta = 10^{-2}$, and $\xi = 1$, $\omega_n = 100$ for both CF_1 and CF_2.

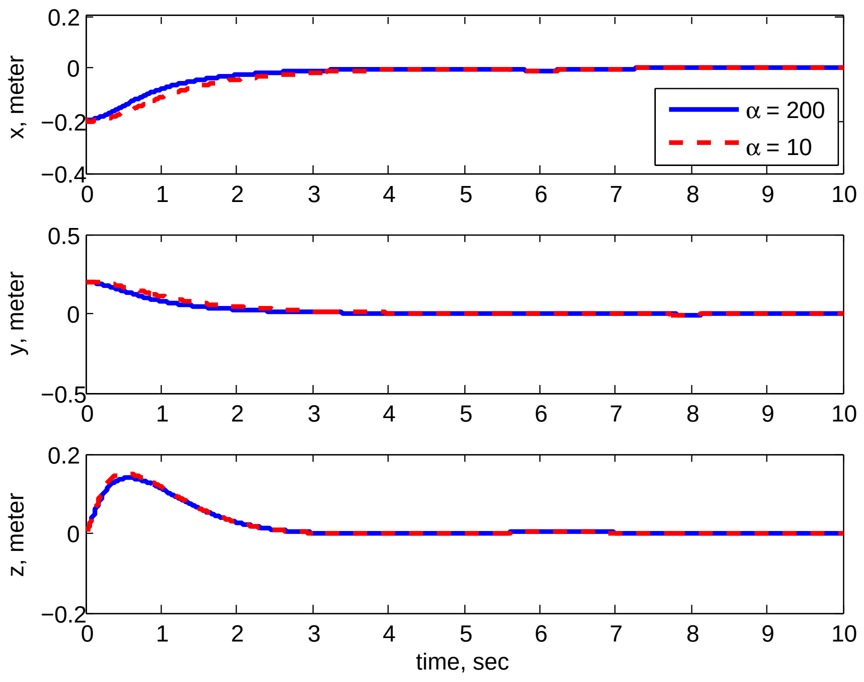


Figure 6.4: The position tracking errors $(\mathbf{p} - \mathbf{p}_d)$. The blue line is using $\alpha = 200$ and the red dash line is using $\alpha = 10$.

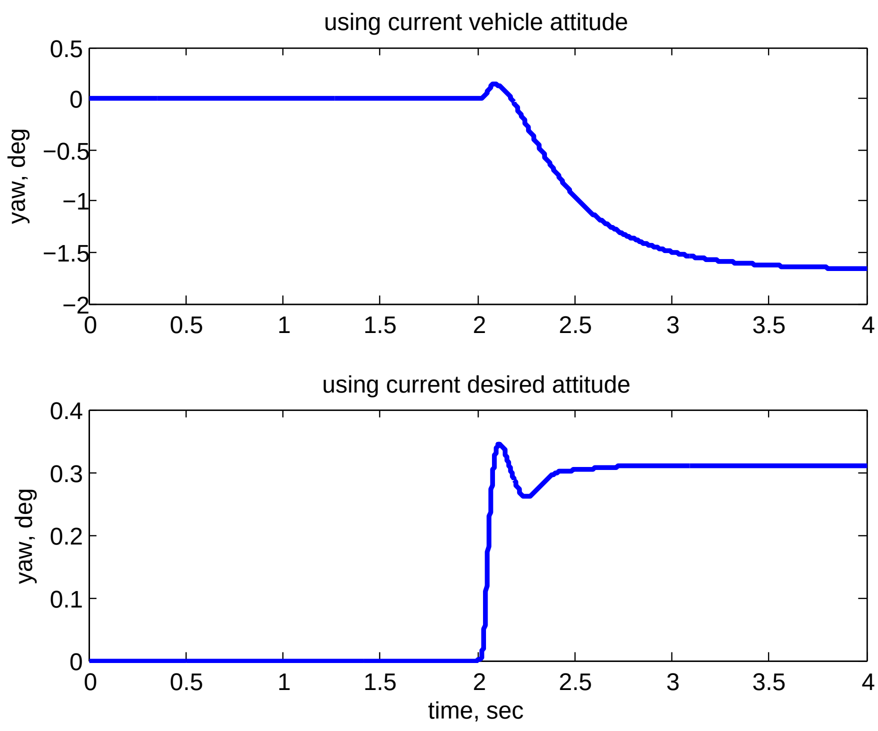


Figure 6.5: The comparison of yaw rotation by choosing the reference attitude to be the current desired attitude or the current vehicle attitude.

Chapter 7

Conclusions and Future Work

7.1 Navigation

This thesis has proposed a novel DGPS/IMU integration approach that significantly improves performance, compared to EKF solutions. Such performance improvement is especially needed in urban environments. The new algorithm performs optimization in realtime, for all IMU and GPS measurement within a time window, to provide a realtime state estimate at the current time. The approach leads to improved performance for a few reasons. First, optimization over a time window provides the capability to re-linearize the system kinematic and measurement models around the improved trajectory estimate. Multiple steps of the iterative optimization converges to the minimum of the nonlinear MAP problem. Once they become observable, this provides the ability to estimate attitude and biases, even yaw, accurately without a magnetometer. Second, the large set of measurement data provides sufficient redundancy to allow the effects of noise to be significantly reduced in the optimization. In addition, it allows the detection of anomalous measurements, by RAIM type techniques, so that they can be eliminated from the measurement set. Those methods are not presented herein, but will be developed and presented in future work. Third, the proposed integer-free phase measurement is able to provide accurate local kinematic constraints, without resolving the integers, which helps to improve the robustness to multipath errors and GPS noise, which are common in urban environments.

In addition to presenting the improved approach, this article has presented and analyzed data from the application of this method to a test trajectory using L1-only GPS measurements. The experimental data analysis demonstrated several points. The CRT navigation system is able to reliably achieve sub-meter (often decimeter) positioning accuracy in a GPS-challenged urban environment. The CRT approach calibrates the attitude and biases rapidly following initial acceleration, which then allows the approach to maintain state accuracy through periods with few satellites. Use of the integer-free phase measurements in the CRT approach yields a smoother trajectory than the pseudorange only solution.

There are a variety of directions for future work to improve the proposed system performance:

- Develop and demonstrate reliable methods to check the measurement residuals in order to reject outliers.
- Develop realtime algorithms that allow correct modeling of the correlation between satellites in the integer-free measurements. This was is ignored in this thesis. One approach is to add the receiver clock bias into the state vector to avoid the double-differencing, thus avoiding introducing the correlation between satellites.
- Develop CRT methods to reliably resolve the integer ambiguity to achieve centimeter positioning accuracy using single frequency receivers. The CRT framework has

the potential to increase the success rate of the integer ambiguity resolving in realtime.

- Construct the optimal integer-free phase tracks Ξ in the CRT window to maximize the information extracted from phase measurements.
- Test the algorithm using data from a low-cost, single-frequency receiver to evaluate robustness to the noisy signal reception from a low-cost GPS antenna.
- Find a convenient way to monitor the status of the CRT estimator in realtime.
- Augment of states to model GNSS time correlation measurement errors per satellite.

7.2 Control

This thesis presents trajectory tracking control for VTOL UAVs using the command filtered backstepping technique. The commanded trajectory may include just the position or position and desired yaw. Quaternions are used for attitude control, which ensures the global attitude tracking performance. The quaternion used in this thesis follows the modified definition, which is well recognized by the navigation and robotics communities, to facilitate the adoption of various quaternion-based nonlinear controllers by practitioners outside the control community in their applications. More importantly, in the backstepping control design, command filters are used to avoid the often prohibitively difficult analytic computation of the required command derivatives in each step. For the quaternion filtering, a standard vector-based command filter does not exploit the special dynamics of the quaternion, which could result in a longer route being taken by the filter to track the desired quaternion. To address this issue, a second-order quaternion filter is introduced that automatically computes the derivative of quaternion (namely the angular velocity) without differentiation, always follows the smallest angular path, and maintains the unit norm property of the quaternion. As a benefit of using command filters, the flexibility of giving yaw commands in the trajectory is realized without adding extra efforts in the design process. In the future, model error, actuator allocation, and an adaptive version of the controller can be considered. Moreover, it is a very interesting topic to tightly integrate the proposed CRT navigation system with the command filtered backstepping control in a real quadrotor and test the performance.

7.3 Publications Resulting from Ph.D. Study

1. S. Zhao, Y. Chen, and J. Farrell, High Precision Vehicle Navigation in Urban Environments using a MEMs IMU and Single-frequency GPS Receiver, in Intelligent Transportation Systems, IEEE Transactions on, 2014 (submitted)

2. Y. Chen, S. Zhao, and J. A. Farrell, Computationally Efficient Carrier Integer Ambiguity Resolution in GPS/INS: a Common-Position-Shift approach, in Control Systems Technology, IEEE Transactions on, 2014, (submitted).

3. S. Zhao, Y. Chen, and J. Farrell, High precision vehicle navigation in urban environments using a low-cost single-frequency GPS receiver, in IEEE IROS 6th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, 2014 (to appear)

4. Y. Chen, S. Zhao, D. Zheng, and J. Farrell, High Reliability Integer Ambiguity Resolution of 6DOF RTK GPS/INS, in IEEE Conference on Decision and Control (CDC), 2014. (to appear)

5. S. Zhao, Y. Chen, H. Zhang, and J. Farrell, Differential GPS aided inertial navigation: a contemplative realtime approach, in 19th IFAC World Congress (Finalist, Best Application Paper), pp. 8959–8964, 2014.

6. S. Zhao and J. Farrell, 2D LIDAR aided INS for vehicle positioning in urban environments, in IEEE Multi-Conference on Systems and Control (MSC), pp. 376–381, 2013.

7. S. Zhao and J. Farrell, Quaternion-based trajectory tracking control of VTOL-UAVs using command filtered backstepping, in American Control Conference (ACC), pp. 1018–1023, 2013.

8. S. Zhao and J. Farrell, Optimization-based road curve fitting, in 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), 2011, pp. 5293-5298.

Bibliography

[1] M Bahrami and M Ziebart. Doppler-aided positioning-improving single-frequency RTK in the urban enviornment. *GPS World*, pages 47–56, May 2011.

[2] P.J. Bristeau, F. Callou, D. Vissière, N. Petit, et al. The navigation and control technology inside the AR. Drone micro UAV. In *Preprints of the 18th IFAC World Congress*, pages 1477–1484, 2011.

[3] R Grover Brown and Patrick YC Hwang. A Kalman filter approach to precision GPS geodesy. *Navigation*, 30(4):338–349, 1983.

[4] E. Brunskill and N. Roy. SLAM using incremental probabilistic PCA and dimensionality reduction. In *ICRA*, pages 342–347. IEEE, 2005.

[5] A. Bry, A. Bachrach, and N. Roy. State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2012.

[6] Peter Cederholm. Statistical characteristics of L1 carrier phase observations from four low-cost GPS receivers. *Nordic journal of surveying and real estate research*, 7(1):58–75, 2010.

[7] Xiao-Wen Chang and Tianyang Zhou. MILES: MATLAB package for solving mixed integer least squares problems. *GPS Solutions*, 11(4):289–294, 2007.

[8] Anning Chen, Dongfang Zheng, Arvind Ramanandan, and Jay A Farrell. Near-real-time GPS integer ambiguity resolution. In *Decision and Control and European Control Conference (CDC-ECC), 50th IEEE Conference*, pages 7281–7286, 2011.

[9] Han-Pang Chiu, Stephen Williams, Frank Dellaert, Supun Samarasekera, and Rakesh Kumar. Robust vision-aided navigation using sliding-window factor graphs. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 46–53, 2013.

[10] Jean-Marie Codol, Michèle Poncelet, André Monin, and Michel Devy. Safety robotic lawnmower with precise and low-cost L1-only RTK-GPS positioning. In *Proceedings of IROS Workshop on Perception and Navigation for Autonomous Vehicles in Human Environment, San Francisco, California, USA*, pages 69–72, 2011.

[11] JM Codol and A Monin. Improved triple difference GPS carrier phase for RTK-GPS positioning. In *Statistical Signal Processing Workshop (SSP), IEEE*, pages 61–64, 2011.

[12] Frank Dellaert and Michael Kaess. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *Int. J. Rob. Res.*, 25(12):1181–1203, 2006.

[13] Tue-Cuong Dong-Si and Anastasios I Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *IEEE ICRA*, pages 5655–5662, 2011.

[14] J. A. Farrell. *Aided navigation: GPS with high rate sensors.* McGraw-Hill New York, NY, USA, 2008.

[15] J. A. Farrell, T. Givargis, and M. Barth. Real-time differential carrier phase GPS-aided INS. *IEEE Transactions on Control Systems Technology*, 8(4):709–721, 2000.

[16] J. A. Farrell, M. Polycarpou, M. Sharma, and W. Dong. Command filtered back-stepping. *IEEE Transactions on Automatic Control*, 54(6):1391–1395, 2009.

[17] J. A. Farrell, H.S. Tan, and Y. Yang. Carrier phase GPS-aided INS based vehicle lateral control. *ASME Journal of Dynamics Systems, Measurement, & Control*, 125(3):339–353, 2003.

[18] J.A. Farrell, T.D. Givargis, and M.J. Barth. Real-time Differential Carrier Phase GPS-aided INS. *IEEE Trans. CST*, 8(4):709–721, 2000.

[19] Jay A. Farrell. *Aided Navigation: GPS with High Rate Sensors.* McGraw Hill, 2008.

[20] Ron Hatch. The synergism of GPS code and carrier measurements. In *International geodetic symposium on satellite doppler positioning*, volume 1, pages 1213–1231, 1983.

[21] Ron Hatch. Instantaneous ambiguity resolution. In *Kinematic systems in geodesy, surveying, and remote sensing*, pages 299–308. Springer, 1991.

[22] Will Hedgecock, Miklos Maroti, Janos Sallai, Peter Volgyesi, and Akos Ledeczi. High-accuracy differential tracking of low-cost GPS receivers. In *ACM 11th International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*, Taipei, Taiwan, 2013. ACM, ACM.

[23] J.A. Hesch, F.M. Mirzaei, G.L. Mariottini, and S.I. Roumeliotis. A laser-aided inertial navigation system (L-INS) for human localization in unknown indoor environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5376–5382. IEEE, 2010.

[24] Steve Hewitson, Hung Kyu Lee, and Jinling Wang. Localizability Analysis for GPS/Galileo Receiver Autonomous Integrity Monitoring. *J. of Navigation*, 57:245–259, 2004.

[25] G.M. Hoffmann, S. Waslander, and C.J. Tomlin. Quadrotor helicopter trajectory tracking control. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[26] Viorela Ila, Josep M Porta, and Juan Andrade-Cetto. Information-based compact pose SLAM. *Robotics, IEEE Transactions on*, 26(1):78–93, 2010.

[27] Vadim Indelman, Stephen Williams, Michael Kaess, and Frank Dellaert. Factor graph based incremental smoothing in inertial navigation systems. In *Information Fusion (FUSION), 15th International Conference on*, pages 2154–2161. IEEE, 2012.

[28] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31(2):216–235, February 2012.

[29] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental Smoothing and Mapping. *IEEE Trans. Robotics,*, 24(6):1365–1378, 2008.

[30] Michael Kaess and Frank Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, 57(12):1198–1210, 2009.

[31] E.D. Kaplan and C.J. Hegarty. *Understanding GPS Principles and Applications, 2nd Ed.* Artech House, 2006.

[32] T. Lee, M. Leoky, and N.H. McClamroch. Geometric tracking control of a quadrotor UAV on SE (3). In *IEEE Conference on Decision and Control*, pages 5420–5425, 2010.

[33] Taeyoung Lee, N Harris McClamroch, and Melvin Leok. Optimal control of a rigid body using geometrically exact computations on SE (3). In *IEEE Conference on Decision and Control*, pages 2710–2715, 2006.

[34] Mingyang Li and Anastasios I Mourikis. High-precision, consistent EKF-based visual–inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[35] Mingyang Li and Anastasios I Mourikis. Online temporal calibration for camera–IMU systems: Theory and algorithms. *The International Journal of Robotics Research*, 33(7):947–964, 2014.

[36] Kai Lingemann, Andreas Nüchter, Joachim Hertzberg, and Hartmut Surmann. High-speed laser localization for mobile robots. *Robotics and Autonomous Systems*, 51(4):275–296, 2005.

[37] Tarek Madani and Abdelaziz Benallegue. Control of a quadrotor mini-helicopter via full state backstepping technique. In *Decision and Control, 45th IEEE Conference on*, pages 1515–1520. IEEE, 2006.

[38] Christopher G Mayhew, Ricardo G Sanfelice, and Andrew R Teel. Robust global asymptotic attitude stabilization of a rigid body by quaternion-based hybrid feedback. In *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference (CDC/CCC)*, pages 2522–2527, 2009.

[39] Christopher G Mayhew, Ricardo G Sanfelice, and Andrew R Teel. Quaternion-based hybrid control for robust global attitude tracking. *Automatic Control, IEEE Transactions on*, 56(11):2555–2566, 2011.

[40] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, pages 664–674, 2012.

[41] Pratap Misra and Per Enge. *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2006.

[42] A.I. Mourikis and S.I. Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In *IEEE ICRA*, pages 3565–3572, 2007.

[43] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In *Intelligent Robots and Systems, (IROS). IEEE/RSJ International Conference on*, pages 1929–1934, 2005.

[44] U.S. Department of Defense. Global positioning system standard positioning service performance standard. Technical report, U.S. Department of Defense, 2008.

[45] Gary P. and C. E. Fly. NDGPS assessement final report. Technical report, U.S. Department of Transportation, 2008.

[46] S.T. Pfister, K.L. Kriechbaum, S.I. Roumeliotis, and J.W. Burdick. Weighted range sensor matching algorithms for mobile robot displacement estimation. In *ICRA 2002*, volume 2, pages 1667–1674. IEEE.

[47] Alex G Quinchia, Gianluca Falco, Emanuela Falletti, Fabio Dovis, and Carles Ferrer. A comparison between different error modeling of mems applied to GPS/INS integrated systems. *Sensors*, 13(8):9549–9588, 2013.

[48] A. Roberts and A. Tayebi. Adaptive position tracking of VTOL UAVs. *IEEE Transactions on Robotics*, (99):1–14, 2011.

[49] Ricardo G Sanfelice, Rafal Goebel, and Andrew R Teel. Invariance principles for hybrid systems with connections to detectability and asymptotic stability. *IEEE Transactions on Automatic Control*, 52(12):2282–2297, 2007.

[50] M.D. Shuster. The nature of the quaternion. *Journal of the Astronautical Sciences*, 56(3):359, 2010.

[51] Richard A Snay and Tomás Soler. Continuously operating reference station (CORS): history, applications, and future enhancements. *Journal of Surveying Engineering*, 134(4):95–104, 2008.

[52] W Stempfhuber and M Buchholz. A precise, low-cost RTK GNSS system for UAV applications. In *Conference on Unmanned Aerial Vehicle in Geomatics, Zürich*, pages 289–293, 2011.

[53] Tomoji Takasu and Akio Yasuda. Evaluation of RTK-GPS performance with low-cost single-frequency GPS receivers. In *Proceedings of international symposium on GPS/GNSS*, pages 852–861, 2008.

[54] Tomoji Takasu and Akio Yasuda. Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB. In *International Symposium on GPS/GNSS, International Convention Center Jeju, Korea*, pages 4–6, 2009.

[55] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial intelligence*, 128(1):99–141, 2001.

[56] Nikolas Trawny and Stergios I. Roumeliotis. Indirect Kalman filter for 3D attitude estimation. Technical Report 2005-002, University of Minnesota, Dept. of Comp. Sci. & Eng., March 2005.

[57] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[58] Anh Vu, J Farrell, and Matthew Barth. Centimeter-accuracy smoothed vehicle trajectory estimation. *Intelligent Transportation Systems Magazine, IEEE*, 5(4):121–135, 2013.

[59] Anh Vu, Arvind Ramanandan, Anning Chen, Jay A Farrell, and Matthew Barth. Real-time computer vision/DGPS-aided inertial navigation system for lane-level vehicle navigation. *Intelligent Transportation Systems, IEEE Transactions on*, 13(2):899–913, 2012.

[60] S. Zhao and J. A. Farrell. Quaternion-based trajectory tracking control of VTOL-UAVs using command filtered backstepping. Technical report, Dept. of Electrical Engineering, Univ. of CA, Riverside, 2013. http://www.ee.ucr.edu/ farrell/?page=content/PubSupp.html.

[61] Sheng Zhao, Yiming Chen, Haiyu Zhang, and Jay A. Farrell. Differential GPS aided inertial navigation: a contemplative realtime approach. In *IFAC World Congress*, pages 8959–8964, 2014.

[62] Sheng Zhao, Wenjie Dong, and Jay A Farrell. Quaternion-based trajectory tracking control of VTOL-UAVs using command filtered backstepping. In *American Control Conference (ACC)*, pages 1018–1023. IEEE, 2013.

[63] Sheng Zhao and Jay A Farrell. 2D LIDAR aided INS for vehicle positioning in urban environments. In *Control Applications (CCA), 2013 IEEE International Conference on*, pages 376–381, 2013.