Using the Global Positioning System for Relative Navigation of Mobile Robots

by

Benjamin Douglas Congram

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

University of Toronto Institute of Aerospace Studies
University of Toronto

Using the Global Positioning System for Relative Navigation of Mobile Robots

Benjamin Douglas Congram
Master of Applied Science

University of Toronto Institute of Aerospace Studies
University of Toronto
2021

Abstract

Mobile robots rely on many sensors including the Global Positioning System (GPS) for navigation. It is important to consider the particulars of the application when determining how best to incorporate GPS observables into a state estimation pipeline. In this thesis, we show how GPS can be fit into a relative navigation paradigm. We begin by developing a single-receiver GPS odometry pipeline that achieves a relative drift rate of 0.6%. We then combine GPS with visual localization and obtain few-centimetre-level path-tracking accuracy on a joint indoor-outdoor route with large appearance change. Finally, we implement GPS odometry as an alternative to visual odometry (VO) in Visual Teach & Repeat 3 providing robust navigation in the case of sensor failure. Our methods were validated through both offline evaluations on datasets we collected and live closed-loop robot experiments.

Acknowledgments

This thesis would not be what it is, nor would I be who I am, without the support of many.

First and foremost, I would like to thank my supervisor, Dr. Tim Barfoot. Tim's mentorship through many enjoyable conversations has provided me with both a thorough technical understanding of robotics and an intuition for problem-solving.

Thank you to all the members of the Autonomous Space Robotics Laboratory for sharing your expertise and creating a welcoming environment despite the challenges of these times. Special thanks to Yuchen, with whom I spent many hours developing and debugging VTR3, for making me a better roboticist along the way and to Mona for the support and introduction to field robotics on the Grizzly in my early days at UTIAS.

Finally, a huge thank you to my parents, sisters, and girlfriend. Your support on this journey and in the years leading up to it has been invaluable.

Contents

List of Acronyms

APC	antenna phase centre
BA	bundle adjustment
CPO	Carrier Phase Odometry
EBN	experience-based navigation
EKF	extended Kalman filter
ENU	East-North-Up
DCS	dynamic covariance scaling
DGPS	differential GPS
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
MEL	multi-experience localization
PRN	pseudorandom noise
PTAM	parallel tracking and mapping
RANSAC	random sample consensus
ROS	Robot Operating System
RTK	real-time kinematic
SBAS	satellite-based augmentation systems
SLAM	simultaneous localization and mapping
STEAM	simultaneous trajectory estimation and mapping
SURF	speeded-up robust features
TDCP	time-differenced carrier phase
UGV	unmanned ground vehicle
UTIAS	University of Toronto Institute for Aerospace Studies

UTM Universal Transverse Mercator

VIO visual-inertial odometry

VO visual odometry

VT&R Visual Teach and Repeat

WNOA white-noise-on-acceleration

Chapter 1

Introduction

1.1 Motivation

When the Global Positioning System (GPS) project was first proposed in the 1970s, it promised an alluring future: accurate, absolute positioning solutions almost anywhere on Earth with a relatively low-cost receiver [1]. There are extensive technical challenges involved in sending a signal from a power-constrained satellite, then accurately timing its arrival after it has propagated thousands of kilometres through a transient atmosphere at the speed of light. It is a wonder the system works at all, let alone that it achieves accuracy on the order of a metre using the standard positioning technique [2]. However, many applications, including those in robotics, require better performance. This gave rise to more advanced techniques addressing the varied sources of GPS measurement error. Differential GPS (DGPS), for instance, uses a second, stationary receiver to cancel out errors on the moving receiver.

In this thesis, we explore how variations of advanced GPS techniques can be used to improve robotic navigation systems. Specifically, we show why relaxing the constraint on absolute accuracy in favour of relative accuracy can be advantageous. The benefits pertain not only to the computational efficiency of our algorithms but also to the quality of our state estimates. They can be found both when GPS serves as the primary sensor, as in Chapter 3, and when used in combination with other sensors, as in Chapters 4 and 5.

As low-cost GPS receivers become standard, effectively utilizing the measurements from a single receiver becomes important even for mobile robots that primarily rely on rich sensors such as cameras or lidar. Visual path-following robots, for example, require an efficient localization method when visual localization fails due to outdoor appearance change. Robots unable to localize could drive longer distances via dead-reckoning provided they have good odometry estimates. A self-driving vehicle relying on camera images needs a method to safely pull to the side of the road should that camera be blocked by stray debris. All of these use cases can be addressed by considering GPS in the context of relative navigation.

1.2 Contributions

The primary novel contributions of this work are the following:

- A practical time-differenced carrier phase (TDCP) algorithm providing accurate state estimates with a single receiver for use on a nonholonomic robot.
- The first direct comparison of TDCP odometry with VO.
- A method to robustly fuse vision and GPS for localization in a path-following system that does not rely on an absolute coordinate frame, thereby retaining the advantages of a relative map.
- The addition of GPS odometry as an alternative to VO to improve the robustness of a highly successful relative navigation pipeline.

1.3 Thesis Overview

The rest of this work is organized as follows. Chapter 2 highlights relevant related work to put the chapters that follow in context and provides references for a more detailed treatment of the theoretical groundwork on which this thesis builds. In Chapter 3, we show how comparing GPS observables over a short timespan can mitigate errors and lead to accurate odometry when combined with other robotics techniques. Chapter 4 describes how GPS and visual localization may be smoothly fused to provide robust path-following in diverse environments. Then, in Chapter 5, we detail how the techniques from Chapter 5 have been extended for use in the latest version of Visual Teach and Repeat (VT&R). Finally, we summarize our findings and speculate on directions for future work in Chapter 6.

Chapter 2

Background

In this chapter, we summarize pertinent information to provide background for the following chapters. We also highlight relevant literature to put our contributions in context, and that the interested reader may look to for reference. There are no novel contributions in this chapter.

2.1 Primer on GPS

Single-frequency GPS receivers are now ubiquitous, coming standard in almost every smartphone. First operational in 1983 [1], GPS allows an absolute positioning solution to be calculated anywhere on Earth with a clear view of the sky. Since then, other Global Navigation Satellite System (GNSS) constellations such as GLONASS, Galileo, and BeiDou have come online and may be used independently or in combination with GPS. Note: we will primarily use the term GPS in this work as is common in other publications. However, the majority of the time our statements will generalize to other GNSS constellations.

The main component of the GPS system, the space segment, consists of 31 satellites each in approximately circular orbit around Earth. Each satellite sends a radio signal that may be measured by any receiver on Earth within its line of sight. By comparing the signal's time of arrival to its known time of transmission, the receiver can calculate a time of flight. Multiplying this duration by the speed of light provides a range. The range constrains the receiver position to a sphere centered around the satellite; with three of these constraints, the three-dimensional receiver position can be determined. In practice, the receiver clock bias significantly affects the measured time of flight so the clock bias must be estimated leading to a minimum of four satellites required. Because of this bias, the distance calculated based on the measured time of flight is referred to as a pseudorange and the technique is described as pseudorange positioning.

GPS positioning techniques assume knowledge of the satellite position at the transmission time. To achieve this, each satellite relays its ephemeris data (position and velocity as a function of time in a fixed, Earth-centered coordinate frame) in the form of parameters for an orbital model. The GPS signal itself is a radio wave transmitted at a fixed frequency (1575 MHz for GPS's L1 signal) called the carrier phase and modulated twice to embed information. The higher frequency modulation is done with a pseudorandom noise (PRN) sequence (also known as a Gold code) unique to each satellite to allow separation of signals from different satellites, while the lower frequency modulation embeds the digital navigation data itself.

Several sources of error affect the GPS signal on its way to the receiver, which limits the accuracy of the standard pseudorange positioning technique. Some of these are touched on in more detail in Section 3.2.2. More advanced techniques have been developed to use additional information to increase positioning accuracy. Satellite-based augmentation systems (SBAS) use separate satellites in geostationary orbit to relay information from dedicated ground stations. This data includes corrections that may be applied to the measurements as well as details on the integrity of the navigation satellites. DGPS involves configuring a second GPS receiver at a nearby base station with known coordinates. Because errors affecting the GPS signal are spatially and temporally correlated, an improved absolute position of the user's receiver can be gained by subtracting off the position error calculated by the stationary receiver. Real-time kinematic (RTK) positioning takes this strategy a step further by applying corrections at the raw observable level, utilizing both the pseudorange and the carrier phase measurements. While the phase of the GPS signal waveform can be measured quite accurately and precisely by the receiver, an unknown number of full wavelengths, known as the integer ambiguity, prevents directly converting that phase to a range. By comparing measurements from the two receivers over time, RTK can resolve this ambiguity for each observed satellite.

Time-differenced carrier phase (TDCP) is a similar advanced technique, though using only a single receiver. The idea of comparing carrier phase measurements from the same receiver at different times was first proposed by Ulmer et al. [3] but has received comparatively little attention in the robotics community. It was first developed for static geomatic surveying [3], [4], [5] but can be extended to full trajectories. When a receiver is in phase lock with a satellite, the ambiguity affecting carrier phase measurements is time-invariant. Within this period, differencing two phase measurements will cancel the ambiguity and avoid the need to resolve it. Therefore, better accuracy can be achieved in estimating the relative receiver displacement between the two times, though the absolute positioning error remains high [4]. TDCP has been used in applications as wide-ranging as vehicle convoying [6], [7] and bird-flight trajectory reconstruction [8]. Success has been shown in combining TDCP with inertial navigation systems (INS) [9], [10].

However, it has not previously been used on vision-based robots or integrated with other key robotics techniques such as motion models or robust cost functions to form a complete odometry solution. In Chapter 3, we show how TDCP can be used to generate highly accurate estimates of relative robot motion. For further details on GPS operation please see the main references used in this section: Kaplan and Hegarty [2] and Seeber [1].

2.2 Visual-GPS State Estimation

Visual odometry is the problem of estimating camera motion from a sequence of images in real-time. The first implementation was developed by Moravec [11] for a Mars rover. Since then it has evolved to become a standard component of mobile robotic navigation serving, for example, as an essential component of visual simultaneous localization and mapping (SLAM). It may be used alone for dead-reckoning or fused with data from other sensors such as lidar, inertial navigation systems, or wheel odometry. The basic pipeline involves detecting features in an image and tracking them in the image sequence before using the geometry of those features to estimate viewpoint motion [12]. There are many variations with algorithms available using both monocular and stereo cameras and using both sparse features and dense correspondences [13]. Recently, deep-learning approaches [14], [15] to VO have gained interest, though feature-based methods still remain relevant. Visual localization is a similar problem but, instead of calculating the robot's pose change from recently captured images, the estimator must determine the robot's position in a map. Similar techniques may be used but the problem can be more difficult due to factors such as environmental appearance change since the map images were taken and the lack of a motion-model prior.

Vision and GPS act as complementary sensors in many applications providing robustness via their independent failure modes. Yu et al. [16] use visual-inertial odometry (VIO) to estimate local pose changes then use GPS and nonlinear optimization to bound the estimated drift. They test on an unmanned surface vehicle with an omnidirectional stereo camera and show the benefit of added GPS information over a short path. Other works [17], [18] develop similar methods combining VO and GPS via an extended Kalman filter (EKF). Qin et al. [19] offer a general algorithm to fuse a local sensor such as VO with a global sensor such as GPS to estimate global poses. The local sensor provides high-rate local estimates of the path while pose-graph optimization is run with the global measurements to give low-rate estimates of the transformation to the absolute frame.

Fewer works attempt localization against a map using both vision and GPS. Choi et al. [20] use a threshold on GPS dilution of precision to switch between sensing modes. Several works [21], [22], [23] use GPS signals only as a prior to simplify the image retrieval task. Some then attempt metric localization using the retrieved images and the current camera frame.

Shi et al. [24] use GPS observations offline to improve the global accuracy of visual SLAM. Chen et al. [25] fuse vision and GPS information by first estimating the frame transformation then solving a series of bundle adjustment (BA) problems to generate globally consistent pose estimates. Their work is most similar to ours. However, they rely on estimating in an absolute frame, do not account for prolonged sensor dropout, and only test on a dataset with simulated GPS. Our work in Chapter 4 does not require a privileged frame or any post-processing of the map. It handles transitions between sensing modalities smoothly and is robust to prolonged sections of sensor dropout.

2.3 Visual Teach & Repeat

Autonomously driving the same route through an environment multiple times is a common task for mobile robots with applications to mining, warehouse robots, and guided tours. Visual Teach and Repeat [26] has shown this task can be achieved using only a single stereo camera and one training example in extremely non-planar environments to few-centimetre-level accuracy and across reasonable appearance change [27]. One of the keys to VT&R's success is its relative pose graph map structure. Vehicle transformations and landmark positions are calculated with respect to neighbouring poses, not an absolute frame. Global consistency in the map is not required. As a result, VT&R is computationally inexpensive and can handle large networks of paths. No post-processing of the map is needed, meaning the robot can re-drive the path immediately after it is taught. In the remainder of this section, we briefly describe some key references for relative navigation before summarizing extensions and details of VT&R relevant to this work.

Bundle adjustment [28] or batch SLAM, in which both vehicle poses and landmark positions are estimated from landmark observations, is the classic approach to large-scale localization and mapping. However, computation time for the naive implementation scales quadratically with the number of landmarks [29], necessitating more efficient formulations. Olson et al. [30] showed the benefit of using a relative-pose state space as opposed to global states in improving optimization performance. Submapping [31], [32], [33] has been used as a way to decouple computational complexity of the problem from map size. Sibley et al. [34] built on this with a completely relative formulation, doing away with any one privileged frame. This allowed constant-time map updates even during loop closures.

VT&R extended the idea of mapping on manifolds that are only required to be locally Euclidean to create a visual path-tracking system achieving high autonomy rates over many kilometres of highly non-planar terrain. In VT&R, a path is manually driven once as a single training example. Then, the robot is able to autonomously repeat the path using only a single stereo camera, taking advantage of the deliberately consistent camera viewpoints. The efficiency of the relative formulation allows the use of significantly larger factor graphs. This led to experience-based navigation (EBN) [35], [36] as a method to increase robustness to appearance change. The related multi-experience localization (MEL) [37] was added in VT&R with the ability to use landmarks from multiple experiences in the same metric localization problem and avoid drift from the original teach path over time. Fast triaging of visual experiences [38] is used to recall relevant landmarks in real-time. As a result, the robot can autonomously drive through environments with high appearance change with respect to the teach run provided enough repeats have been captured in between as bridging experiences.

Colour-constant image transformations have also been added within VT&R to increase robustness to changing lighting conditions [39]. VT&R has been shown to generalize well to other sensors such as lidar [40] and monocular cameras [41], as well as to other robot types such as unmanned aerial vehicles [42]. In Chapters 4 and 5, we demonstrate a method to add a global sensor, i.e. GPS, to VT&R while preserving the relative navigation formulation that is the key to its success.

The VO pipeline used in VT&R (and that we use on its own in

Chapter 3) is based on parallel tracking and mapping (PTAM) [43]. Motion estimates are computed at framerate while landmark positions are optimized in a windowed bundle adjustment after each keyframe. It is fast and reliable with the parameters pre-tuned on data separate from that used in this thesis. While testing on nearly 10km of driving over 30 hours, MacTavish et al. [44] found a 1.5% translational drift rate during daytime conditions and a 2.4% rate at nighttime via the use of headlights for this algorithm.

Chapter 3

Time-Differenced Carrier Phase Odometry

Odometry is an important component of almost any mobile robotic navigation strategy. It takes many forms including visual, visual-inertial, lidar, and wheel odometry. All of these use different sensors to accomplish the common goal of estimating the vehicle's path or trajectory. In mapping, odometry allows local reconstruction of the environment. In localization, it can provide a prior and is critical to the success of autonomous navigation systems such as VT&R [26]. EBN [35] and MEL [37] use odometry in short sections (i.e., less than 50m) where localization fails due to factors such as appearance change. If odometry drift becomes too large, the robot may not be able to navigate safely. This in turn causes a missed opportunity to improve the map. Better odometry allows a robot to dead-reckon for longer sections and therefore drive further successfully. Visual odometry is a common solution for obtaining robust and consistent relative motion estimates of the vehicle frame.

Contrarily, GPS measurements are typically used for absolute positioning and localization. However, when the constraint on absolute accuracy is relaxed, carrier phase measurements can be used to find accurate relative position estimates with one single-frequency GPS receiver. This suggests practitioners may want to consider GPS odometry as an alternative or complement to VO or inertial solutions.

In this chapter, we describe a robust method for single-receiver GPS odometry on an unmanned ground vehicle (UGV). We first demonstrate the accuracy of our method over long sections of data collected by a real robot. We then present an experimental comparison of the performance of our single-receiver GPS odometry and VO on the same test trajectories. After 1.8km of testing, the results show our GPS odometry method has a 75% lower drift rate than a proven stereo VO method while maintaining a smooth error signal despite varying satellite availability. In Section 3.4, we show that combining both sensors into the same odometry pipeline can improve robustness when sensor dropout is a potential issue. The results of this chapter were previously reported in Congram and Barfoot [45]. To the best of our knowledge, this is the first direct comparison of stereo VO and TDCP and the first investigation of combining the two sensors in the literature.

3.1 Single-Receiver GPS Estimation

While more involved methods such as RTK provide higher quality GPS positioning, there remain many advantages to using only a single GPS receiver: fewer receivers mean a lower cost, no communications link needs to be established to a base station, the robot is not confined to the local area of the base station, and there is less setup and maintenance required for the operator. We would therefore like to use single-receiver GPS to improve robot odometry. However, standard pseudorange GPS positioning does not have the accuracy required to bound vehicle travel within the envelope required for visual localization, which typically de-

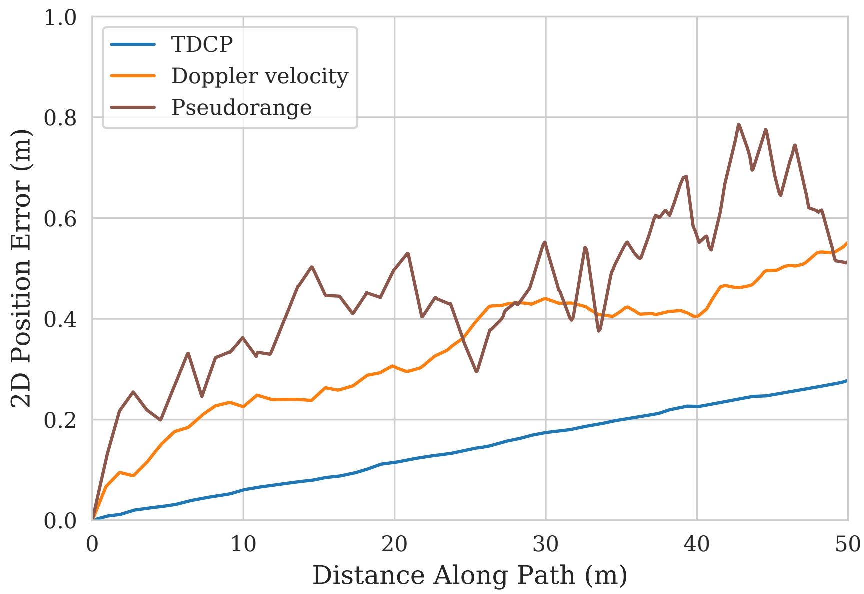

Figure 3.1: Comparison of the mean relative position error for three different techniques using a single GPS receiver across 12 independent paths. The TDCP method is both more accurate and much smoother than the pseudorange positioning while also outperforming the integrated Doppler velocity.

generates with decimetre-level lateral errors [26]. Utilizing other GPS observables over short windows of time can improve relative positioning. Fig. 3.1 illustrates the relative accuracy of three different single-receiver odometry strategies over a set of short trajectories. Pseudorange positioning does not take advantage of the more precise yet difficult to utilize carrier phase observable. It also does not explicitly consider the temporal correlation of measurements. Doppler velocity estimates derive from the frequency shift of the carrier wave. They can be integrated to provide a trajectory but are noisier than TDCP positioning. TDCP provides estimates with approximately half the drift rate compared to the other two methods. These preliminary results indicate TDCP is a prime candidate to be the basis for our single-receiver GPS odometry pipeline.

3.2 Methodology

We briefly summarize the coordinate frames relevant to this chapter before getting into the details of our estimator. The global East-North-Up (ENU) frame, $\underrightarrow{\mathcal{F}}_g$, is a stationary frame tangential to the Earth at the vehicle start position. All other frames are transient. The vehicle frame, $\underrightarrow{\mathcal{F}}_v$, is located at the center of the vehicle at axle height. All estimation is computed in $\underrightarrow{\mathcal{F}}_v$ before being transformed to the onboard GPS receiver frame, $\underrightarrow{\mathcal{F}}_r$, for comparison with ground truth positions. The origin of the orbiting satellite frame, $\underrightarrow{\mathcal{F}}_s$, is defined at the antenna phase centre (APC) for calculating ranges. Finally, the camera frame, $\underrightarrow{\mathcal{F}}_c$, is located at the left camera of the stereo module. The VO algorithm is also configured to output estimates in the vehicle frame.

3.2.1 Carrier Phase Error Equation

Whereas RTK positioning makes use of carrier phase measurements from two receivers separated in space, TDCP positioning makes use of carrier phase measurements from a single receiver separated by both time and space. The carrier phase range equation to a single satellite at time a is given by

$$\Phi_a = \rho_a + N + c\delta_a^R - c\delta_a^S + E_a + T_a - I_a + m_a + \epsilon_a, \tag{3.1}$$

where Φ_a is the measured phase in radians multiplied by the known wavelength so that all values have units of metres. GNSS receivers can measure the incoming phase quite accurately meaning the white noise affecting the measurement, ϵ, is typically less than 2mm [2]. However, the signal is affected by several sources of systematic error as it propagates from satellite to receiver causing the measured range, Φ, to differ from the true range to the satellite, ρ. These include receiver and satellite clock errors (δ^R and δ^S), satellite ephemeris error (E), tropospheric delay (T), ionospheric effects (I), and multipath (m).

N is the unknown wavelength ambiguity; if the receiver stays in phase lock with the satellite it is time-invariant. We can therefore eliminate it by differencing (3.1) taken at two times, a and b:

$$\Phi_b - \Phi_a = \rho_{ba} + c\delta^R_{ba} - c\delta^S_{ba} + E_{ba} + T_{ba} - I_{ba} + m_{ba} + \epsilon_{ba}. \tag{3.2}$$

The subscript ba denotes the difference between a quantity at time b and time a. The receiver clock error is typically large so it must be dealt with explicitly, either by estimating it or differencing the equation again for two different satellites to eliminate it. The latter gives us our measurement model:

$$\Phi^{\mathit{21}}_{ba} = \rho^{\mathit{21}}_{ba} - c\delta^{S,\mathit{21}}_{ba} + E^{\mathit{21}}_{ba} + T^{\mathit{21}}_{ba} - I^{\mathit{21}}_{ba} + m^{\mathit{21}}_{ba} + \epsilon^{\mathit{21}}_{ba}. \tag{3.3}$$

The term ρ^{21}_{ba}, for example, denotes the double difference $\left(\rho^{\mathit{2}}_b - \rho^{\mathit{2}}_a\right) - \left(\rho^{\mathit{1}}_b - \rho^{\mathit{1}}_a\right)$ for a pair of satellites *1* and *2* at times a and b. The ranges making up ρ^{21}_{ba} are calculated using

$$\rho_a = \left\|\mathbf{r}^{sr}_g(t_a)\right\| = \left\|\mathbf{r}^{sg}_g(t_a) - \mathbf{r}^{rg}_g(t_a)\right\|, \tag{3.4}$$

where $\mathbf{r}^{sg}_g$ is the known satellite ephemeris and $\mathbf{r}^{rg}_g$ is our state. The notation $\|\mathbf{r}\|$ denotes the Euclidean norm of the vector $\mathbf{r}$. It is important to recalculate the ephemeris at each measurement time because the satellites travel at 3.9km/s. From (3.3), we can write our error term for one pair of satellites seen at one pair of positions as

$$e^{\mathit{21}}_{ba} = \Phi^{\mathit{21}}_{ba} - \rho^{\mathit{21}}_{ba}. \tag{3.5}$$

Given n commonly seen satellites between t_a and t_b, our weighted least squares factor is

$$J_{ba} = \sum_{k=2}^{n} w_k \left(e_{ba}^{k1}\right)^2, \tag{3.6}$$

where w_k is a scalar variance parameter, which we set as a constant in our implementation though it could be tuned if more information on the measurement quality from each satellite was known. J_{ba} is symbolized as a blue dot in Figure 3.3.

For optimization, a linearized error term is needed. We derive this by noting that $\mathbf{r}_g^{sr}(t_a)$ and $\mathbf{r}_g^{sr}(t_b)$, the vectors from the receiver to a particular satellite at t_a and t_b, are approximately parallel for small t_{ba} since the distance between receiver and satellite is much larger than the distance either travels in this timespan. As illustrated in Figure 3.2, the range to the satellite can change due to both the receiver's movement and the satellite's movement between measurement times. For succinctness we define the unit vector from the receiver to the satellite as $\mathbf{u} = \frac{\mathbf{r}_g^{sr}(t_a)}{\left\|\mathbf{r}_g^{sr}(t_a)\right\|}$. From Figure 3.2 we see that the range difference due to the satellite's cross-track movement is equal to the satellite displacement vector projected onto this unit vector. Likewise, the change due to receiver movement (i.e., the robot driving) corresponds to the negative of the receiver displacement vector projected onto $\mathbf{u}$. Combining these gives:

$$\rho_{ba} = -\hat{\mathbf{u}}^T \left(\mathbf{r}_g^{rg}(t_b) - \mathbf{r}_g^{rg}(t_a)\right) + \hat{\mathbf{u}}^T \left(\mathbf{r}_g^{sg}(t_b) - \mathbf{r}_g^{sg}(t_a)\right), \tag{3.7}$$

where the second half of the right-hand side (the satellite movement term) is independent of the state. After substituting (3.7) into our error equation, (3.5), we can calculate the Jacobian required to perform Gauss-Newton optimization.

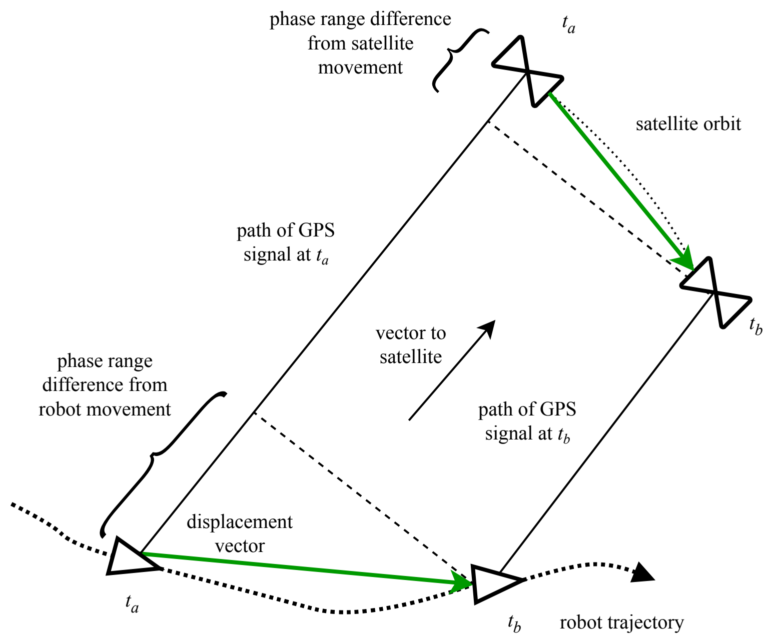

Figure 3.2: In linearizing the error term, we make the assumption that the unit vectors from receiver to satellite at times a and b are parallel. As a result, the difference in measured range due to receiver movement has the same magnitude as the scalar projection of the receiver displacement vector onto the satellite vector.

3.2.2 Carrier Phase Noise Properties

Our error equation, (3.5), constrains the transformation between the receiver pose at two times. Given a set of carrier phase measurements collected at a fixed rate (e.g. 1Hz), we have a choice of how to pair these measurements to form error terms. Figure 3.3(a) illustrates three potential options. If our measurements at each timestamp were primarily affected by Gaussian noise, then the "Dense" strategy would be best. More factors in our factor graph would average out the noise and improve our estimates. However, looking at our measurement model, (3.1), we see that the majority of error sources are systematic in nature. That is, if the receiver could take two measurements of the carrier phase at the same instance, they would be almost exactly the same save for the very small measurement error, ϵ. The other error sources vary smoothly. Neglecting ϵ, we would find that

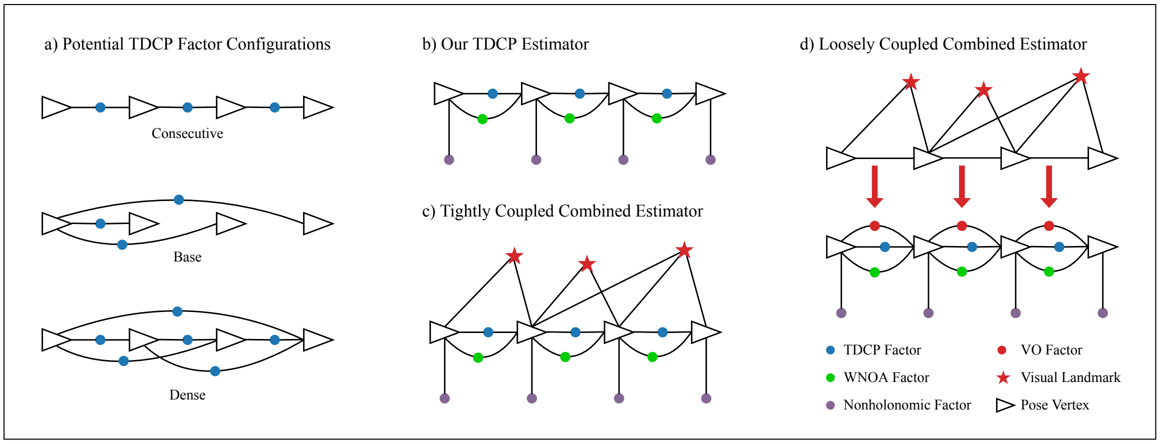

Figure 3.3: a) Potential ways TDCP factors can be added. Due to the error characteristics of the phase range, they all give very similar position estimates. The "Consecutive" configuration was chosen for our estimator. b) Factor graph for our TDCP algorithm. c) Factor graph that comes from combing VO and TDCP in a tightly-coupled fashion. d) Factor graph for the loosely-coupled estimator used in Section 3.4 for ease of comparison. VO is first run to estimate pose changes then those estimates are added as factors.

e_{ca}, the error term on the poses at times a and c is a simple linear combination of e_{cb} and e_{ba}. Therefore, the additional error terms in the "Dense" configuration compared to the other configurations add very little to the optimization problem besides computational burden. The "Base" and "Consecutive" strategies are very similar but, the "Consecutive" method has subtle advantages when the set of satellites available is time-varying. In "Base", neighbouring pose estimates with respect to the base vertex may be calculated with different satellites so the transformation between these vertices is liable to be less smooth. The advantages of the "Consecutive" strategy are analogous to the advantages of the relative framework of VT&R and the relative localization strategy described later in Chapter 4.

Some of the errors in (3.1), the phase range equation, can be mitigated through modelling. It is typical to use the Klobuchar model [46] to partially correct for ionospheric effects, the parameters of which are available in the GPS navigation message. The Niell mapping function [47] with the UNB3 model parameters [48] can be used to estimate the tropospheric delay. Both models are a function of atmospheric conditions and satellite elevation. Because atmospheric conditions change slowly and the errors are differenced in (3.7), their impact is lessened compared to the effect on a single phase measurement. However, the effect of satellite elevation change over the time difference can be significant for satellites close to the horizon. In our experiments, we model the tropospheric delay but omit the ionospheric correction because the applicable messages were not logged for all runs. We find the difference in performance without the ionospheric correction to be negligible. An analysis of the importance of the tropospheric correction is given in Section 5.2.2.

3.2.3 Time-Differenced Carrier Phase Optimization

The first step in our trajectory estimation pipeline is to parse the raw phase (logged as binary RTCM1004 messages) and calculate coarse pseudorange positions for initializing our state. Preprocessing was done using the C library RTKLIB [49]. TDCP cost terms were only added between consecutive vertices in the factor graph (defined once per second) using commonly seen satellites that maintained phase lock. This is the “Consecutive” factor graph configuration seen in Figure 3.3 and discussed in Section 3.2.2. If the receiver loses phase lock with a satellite, that particular satellite is simply excluded from the set used to construct TDCP factors until phase lock is regained.

Given enough satellites, TDCP will provide a positioning solution. But, to be practical for vehicle odometry, and as a fair comparison for VO, we require full $SE(3)$ pose estimates in the vehicle frame. Our algorithm is designed and tested for a nonholonomic robot so constraints that penalize lateral velocity of the vehicle frame are added. We also use a white-noise-on-acceleration (WNOA) motion prior [50] to encourage smoothness. Together, these allow us to resolve the vehicle orientation. Yaw and pitch are determined by these factors implicitly aligning the longitudinal axis of the robot with the direction of the vehicle frame velocity while roll is determined by the shape of the trajectory. To fit our use case, we have assumed a ground vehicle on a primarily planar surface. The algorithm has not been tested on robots that do not meet these assumptions (e.g. an unmanned aerial vehicle).

Unlike other TDCP algorithms, the use of a motion model such as this allows the robot to make use of carrier phase information and still calculate a state estimate when less than four phase-locked satellites are available. The factor graph can be seen in Figure 3.3(b). The optimization is run as a filter (forward-pass only) to simulate online odometry calculations. It is solved with the simultaneous trajectory estimation and mapping (STEAM) [50] implementation of the dogleg Gauss-Newton algorithm [51] and the motion model applied over a 10-second sliding window. Carrier-phase measurements are subject to outliers so a robust cost function, dynamic covariance scaling (DCS) [52], is used on the TDCP factors.

3.3 Comparison to Visual Odometry

Using raw GNSS observables for state estimation has been given relatively little attention in the robotics community thus far. We found no existing datasets of raw observables collected aboard a robot. For that reason, we decided to collect our own. While sometimes specified for specific datasets such as KITTI [53], no standard benchmark exists for odometry on an arbitrary path. The accuracy of odometry is dependent on dataset-specific factors such as terrain, vehicle speed, path curvature, and others. Therefore, when evaluating our algorithm we not only test the absolute accuracy of its estimates but compare the results to a commonly used odometry source in robotics – stereo VO. Our collected dataset contains several runs with both the stereo image stream and raw GPS observables logged.

In our experiments, stereo VO pose estimates are computed via the same algorithm used in VT&R. The odometry pipeline follows a similar strategy as [43] in which one module estimates camera pose with respect to the previous keyframe at framerate while another performs a local windowed bundle adjustment on map landmarks after each keyframe. Sparse speeded-up robust features (SURF) [54] are used with random sample consensus (RANSAC) [55] to detect outliers. The same WNOA motion prior is used as in 3.2.3. The stereo error terms also have a DCS robust cost function applied to them. Relative pose estimates are computed by solving the Gauss-Newton optimization problem with the STEAM solver.

3.3.1 Experimental Setup

All data was collected aboard the Clearpath Grizzly UGV pictured in Figure 3.4. The vehicle maintained an average velocity of 1m/s across terrain that included pavement and snow-covered grass at the University of Toronto Institute for Aerospace Studies (UTIAS) campus. Stereo images were captured by a front-facing Point Grey Research Bumblebee XB3 stereo camera, which has a 24cm baseline, a 66° horizontal field of view and captures 512x384 pixel images at a 16Hz framerate. GPS measurements were recorded by a NovAtel SMART6-L receiver mounted near the front of the vehicle. Carrier phase measurements were logged at 1Hz while RTK ground truth was logged separately at 4Hz. The RTK positioning is expected to have an RMS error

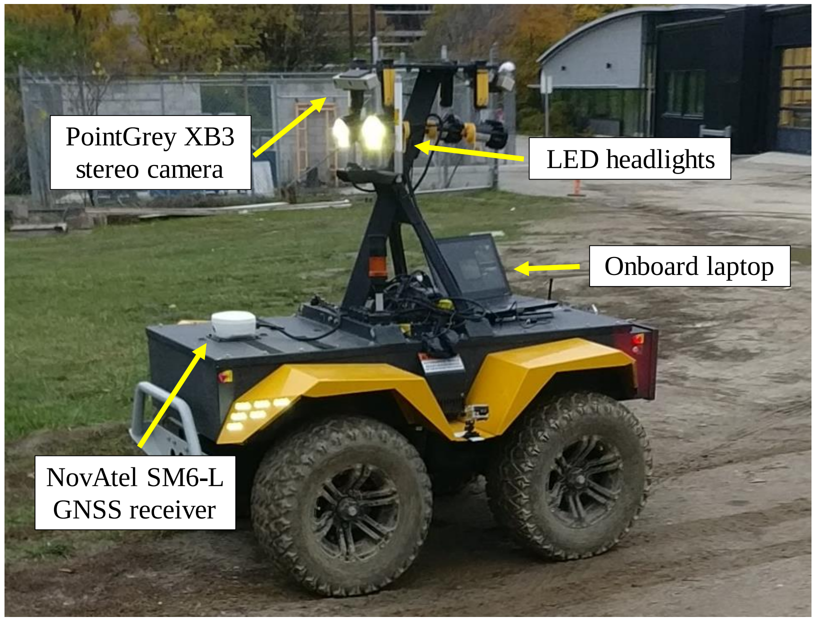

Figure 3.4: The Clearpath Grizzly Robotic Utility Vehicle used for data collection in Chapter 3 and live experiments in Chapters 4 and 5.

of 1cm + 1ppm under nominal conditions. RTK does not provide us with ground truth orientation; therefore we can only compare positioning results. We focus on only the two-dimensional planar estimates (x and y) as these are the most important for navigation on a ground vehicle.

Two independent experiments were conducted to analyze our TDCP-odometry method. The first, to study the absolute accuracy of our algorithm, involved manually driving the robot on four separate runs over two data collection days during which only GPS data was logged. These results are presented in Figure 3.5. To facilitate the second experiment, a study comparing TDCP-odometry to VO, five independent runs were driven on a third day, each spanning several minutes, during which stereo images were also logged. These runs were then split into 15 independent 50m sections, approximately equally spaced, for evaluation. We chose 50m as an evaluation distance as we do not anticipate driving a robot on dead reckoning farther than this and it was sufficient for measuring odometry drift rate. As VO does not estimate orientation in the global ENU frame, the 10m of trajectory preceding the test section was used for alignment of the VO estimates. The continuous-time trajectories computed by STEAM are used to interpolate the VO estimates to the ground truth GPS timestamps (as they are asynchronous to the VO keyframe timestamps). Evaluation is considered based on the amount of drift (absolute translation error) after 25m and 50m.

3.3.2 Results

Satellite availability for the GPS-only experiment varied throughout the runs as buildings and even the vehicle sensor mast itself caused partial occlusions of the sky. Despite this, the receiver kept enough satellites in phase lock throughout the runs for a consistent position estimate at all times. The median number of satellites seen was seven with a minimum of four and a maximum of nine.

Each 250m trajectory presented in Figure 3.5 encompasses nearly five minutes of driving. This provides enough data to characterize the error growth properties while dead-reckoning. We find that the total horizontal translational error after 250m is less than 1m for all runs and the mean error at this point is 0.78m. The errors grow smoothly and approximately linearly. The drift in both the x (East) and y (North) directions is reasonably consistent as we might expect considering the systematic errors affecting the phase measurements in Eq. (3.1).

Figure 3.6 shows an overhead view of the estimates from both algorithms on three of the test trajectories representative of the larger test set in the second experiment. Even at this macroscopic scale, we can see the GPS odometry outperforms VO. Figure 3.7 depicts both the errors for the individual runs and an average horizontal position error for each algorithm. After 50m, the TDCP method has a smaller trans-

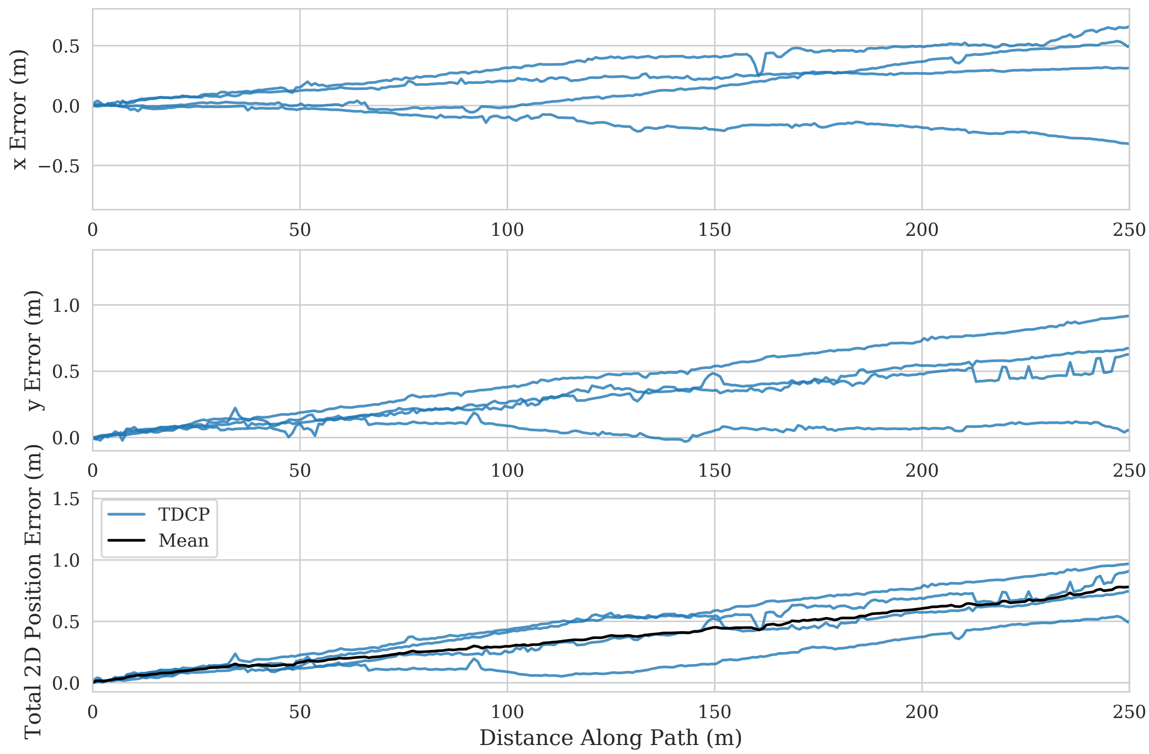

Figure 3.5: Plot of position errors from the TDCP algorithm over the first 250m of the four trajectories in the GPS-only experiment. The drift rate is low and the errors change approximately linearly.

lational error than VO on all but one of the 15 test trajectories. VO has a mean final translational error of 1.127m or 2.25% while TDCP does 75% better with a mean error of 0.281m or 0.56%. The results are similar after just 25m, with drift rates of 2.26% and 0.57%, respectively. The variance in drift rate between runs is also a lot higher for VO as can be seen in the spread of data in Figure 3.7. This implies the expected errors may be more predictable for TDCP.

A similar number of satellites were available for the comparison experiment as in the GPS-only experiment with the minimum five, the median seven, and the maximum nine. There is a negative correlation between errors and number of satellites as expected though the relationship is weak ($r = -0.10$). Other factors such as the particular

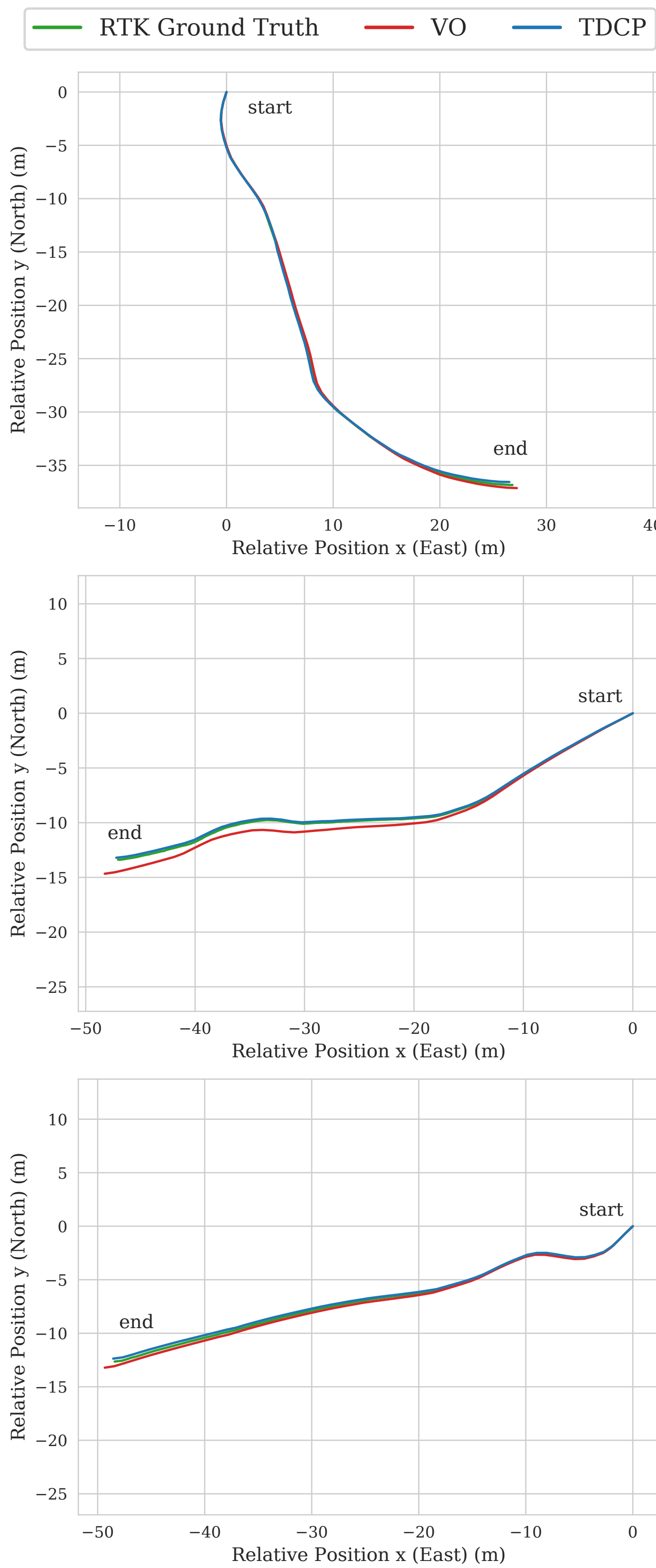

Figure 3.6: Overhead view of ground truth and estimates for three of the 15 test trajectories. VO drifts noticeably further from ground truth than the TDCP-based odometry.

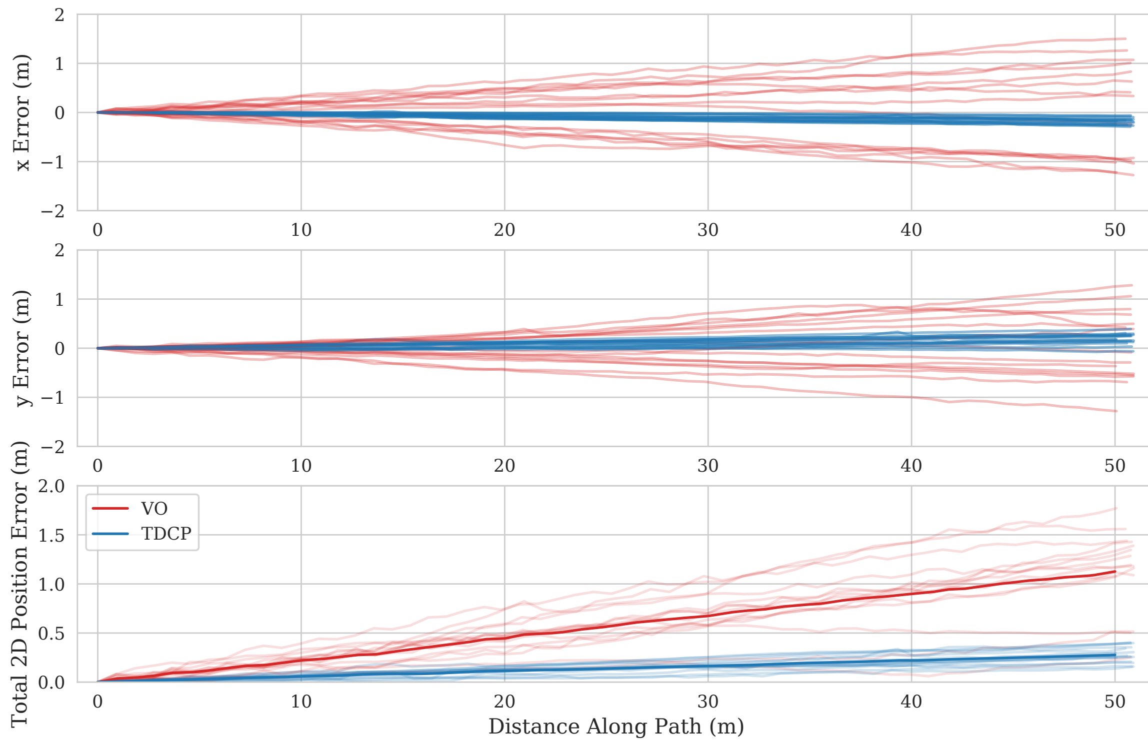

Figure 3.7: Comparison of VO and TDCP-based single-receiver GPS odometry position drift. The fainter lines represent individual trajectories while the darker line plots the average error for the algorithm.

geometry of the satellites, the atmospheric conditions, and the shape of the trajectory may have more influence. We note that the Grizzly's GPS receiver was not configured for use with a power-hungry stereo camera nearby so was somewhat affected by electromagnetic interference. It is possible the satellite availability could have been improved with proper shielding.

Looking more closely at the VO results, we see the number of feature matches varies somewhat between the two major types of terrain seen – dry pavement and snow, but is enough for a reasonable motion estimate throughout. There were no VO failures (i.e., there were always enough landmark matches to produce a consistent estimate). We notice the VO tends to slightly overestimate or underestimate distances within a run. As a result, the total translational error over a full loop trajectory is smaller than the drift rate for shorter sections though still significant. Further tuning might be able to improve VO performance slightly, but it is unlikely to reach the level of the GPS odometry. Finally, we note the TDCP method also has computational advantages as it only requires one error term per satellite pair compared to the potentially hundreds of stereo landmark terms involved in VO. In our head-to-head comparison, GPS odometry was clearly superior in a two-dimensional planar setting.

3.4 Combining with Visual Odometry

As our GPS odometry algorithm has been set up as a factor graph, it is amendable to adding factors from other sensors. A natural choice would be to combine the visual and GPS odometry estimators as shown in Figure 3.3 (c) and (d). The results in this section are from a loosely coupled estimator for ease of comparison. We find under good conditions the addition of vision does not significantly improve accuracy because the estimates from VO are of worse quality. If the uncertainties are improperly set, the inclusion can actually degrade performance. But, using both sensors does improve robustness when the quality or availability of one or both sensors cannot be guaranteed. To show this, we simulate both full (zero satellites available) and partial (two satellites available) temporary GPS dropouts and observe the effect on our odometry with and without the inclusion of VO. The 15-second dropouts occur near the beginning of the approximately one-minute long trajectories.

The results from these experiments are summarized in Table 3.1. We get similar estimates with and without VO when sufficient satellites are available throughout, as seen in Figure 3.8. In this scenario, the

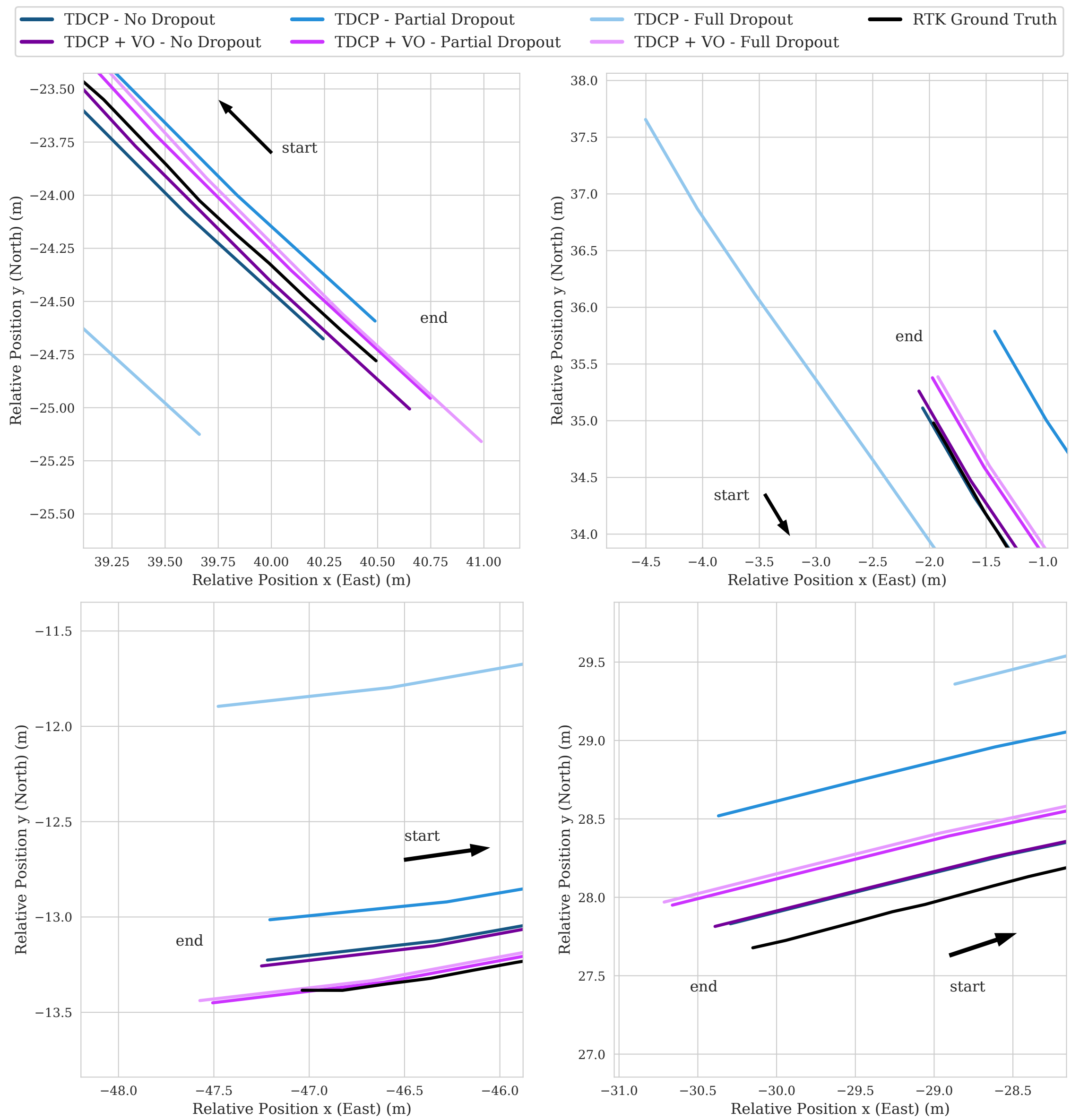

Figure 3.8: Close-up plot of the final positional errors for four 50m paths illustrating the effect of GPS dropout on our TDCP algorithm. The black line shows the end of the trajectory from ground truth while the coloured lines show the results from different experimental conditions. The beginning of each trajectory is not shown but its location is indicated via an arrow. We find the addition of lower quality VO measurements to our algorithm makes little difference when enough satellites are available. However, the TDCP+VO algorithm is much more robust when a short satellite dropout is simulated in the trajectory.

Condition	Method	Final Position Error [m]						
		Run 1	2	3	4	5	6	**Avg**
No GPS Dropout	GPS	0.269	0.163	0.196	0.261	0.241	0.209	**0.223**
	GPS + VO	0.276	0.309	0.241	0.348	0.248	0.276	**0.283**
Partial Dropout	GPS	0.187	0.972	0.266	0.314	0.407	0.869	**0.503**
	GPS + VO	0.311	0.399	0.49	0.439	0.474	0.579	**0.449**
Full Dropout	GPS	0.901	3.689	1.605	0.27	1.553	2.115	**1.689**
	GPS + VO	0.624	0.409	0.539	0.511	0.539	0.634	**0.543**

Table 3.1: Results from the experiments combining our TDCP odometry with VO. As the severity of GPS dropouts increases, so too does the benefit of using VO in a combined estimator.

results from the algorithm with only TDCP are slightly better on average, likely because the parameters of the combined algorithm were not thoroughly tuned. But, when dropouts occur, the GPS-only estimator is forced to rely heavily on its motion model and accuracy suffers. Local accuracy does recover once satellites are reacquired. In the partial dropout experiment, the receiver displacement is not fully constrained as only two satellites are available, but our algorithm can still make use of the carrier phase information to some degree and performance is much better than with zero satellites. However, in many applications, the added error would still be considered a failure. With the addition of VO, the performance loss from the dropout is reduced significantly. A combined approach provides the added accuracy of TDCP with the reliability of VO.

3.5 Summary

In this chapter, we described a method for highly accurate odometry using a single GPS receiver. While GPS users are typically concerned with the absolute accuracy of their positioning algorithms, we recognize that only relative accuracy is required for odometry. Relaxing this constraint meant we could cancel many of the temporally correlated error sources affecting GPS and get better displacement estimates. We compared the performance of our single-receiver GPS odometry with stereo VO on the same set of test trajectories. The novel contributions in this chapter are as follows:

(a) we detailed a practical TDCP odometry algorithm complete with a motion model for use on a UGV,

(b) we provided the first direct comparison of single-receiver GPS and VO estimation,

(c) we showed that combining visual and carrier-phase-based odometry works well when GPS dropout may occur.

We believe TDCP odometry is an effective navigation technique and is underutilized in robotics compared to other odometry methods. To show this, we simultaneously collected a large set of GPS data and stereo imagery from a ground robot driving outdoors. We evaluated our TDCP-based single-receiver, single-frequency GPS odometry algorithm against a proven stereo VO pipeline in the first known experiment of this kind. The results showed the GPS odometry produced far smaller positional errors with respect to the RTK ground truth. TDCP odometry is a good alternative to VO for outdoor navigation. VO is still preferred in areas where occlusions or other sources of GNSS signal interference are a frequent issue. For added robustness, or in applications such as indoor-outdoor navigation, the two sensors may be combined. Though we did not explore incorporating additional GNSS constellations in our algorithm, it is likely their use could improve positioning accuracy even further by increasing the number of satellites available. Improved odometry allows robots to build better maps and safely drive further when localization against a map is challenging. This would be beneficial for many robots and robotic systems. In Chapter 5 we show how our TDCP-odometry can be used to complement VT&R's localization pipeline leading to an improved autonomy rate.

Chapter 4

Relative Localization with Vision and GPS

4.1 Overview

In this chapter, we describe a method for performing GPS localization against a relative map. We show how to smoothly incorporate both vision and GPS measurements in the same map while avoiding any discontinuities when switching between sensor modalities. Our method retains the advantages of the relative formulation with no global optimization required and computational complexity decoupled from map size. We validate our method with a practical implementation in VT&R but the method may generalize to other relative navigation algorithms and absolute sensors other than GPS. The results of this chapter were previously published in Congram and Barfoot [56].

Appearance change is the Achilles' heel of visual localization algorithms operating over unstructured terrain. Natural environments vary visually on both diurnal (e.g., lighting change) and seasonal time scales. Unsuccessful localization due to natural appearance change is the primary failure mode of VT&R. GPS measurements do not suffer from appearance change. The independent failure modes of vision and GPS support their use as complementary sensors. Applications such as joint indoor-outdoor navigation motivate the fusion of vision and GPS into a common state estimation problem.

Absolute sensors such as GPS present a challenge for relative localization. Our goal is to determine the relationship between the current local vehicle pose with respect to the vehicle's pose during the teach path. But GPS provides observations in a global frame independent from the robot pose. An accurate estimation of the global-to-local transformation is required to make use of the measurements in the path-tracking problem. This is not guaranteed after a prolonged section of sensor dropout in which the robot relies on a form of dead-reckoning such as VO, which drifts.

One option is to instead estimate all poses in a single global frame. However, that would require a computationally expensive batch map optimization to avoid jumps in the relative poses used in navigation. Instead, we note that the key to the path-tracking problem is a good estimate of the path-tracking error — the difference between the current and map states; a good estimate of the states themselves (i.e., in a global frame) is not required. This was the primary motivation to use a relative map in VT&R.

Our "lazy mapping" approach delays sensor fusion until the error estimation step. During the teach phase, our solution simply logs GPS observations and associates them with VO keyframes. It makes no effort to reconcile them into a consistent metric map with the pose estimates from VO. On repeat, a local window of GPS measurements is used to estimate the orientation of the local frame with respect to the global frame and then rotate the error vector. Like the odometry problem of Chapter 3, the GPS measurements are sparse compared to their visual counterparts. Our method adds little computational overhead to the already lightweight VT&R algorithm. It handles the transition between sensing modalities smoothly and the regions of availability of each sensor do not need to be specified a priori. The following sections describe the method in more detail and summarize the experimental results.

4.2 Methodology

4.2.1 Sensor Fusion

When path-tracking, VT&R uses stereo VO as a form of prediction and visual localization to a local map as a form of correction. Our approach is to add GPS as a second independent form of correction. Each sensor can be used independently or fused when both are available as shown in Figure 4.1. Other forms of prediction could be used in place of VO including the TDCP-based odometry discussed in Chapter 3. However, we wish to independently evaluate the use of GPS on the localization side so we use the same VO pipeline of VTR2 as discussed in Section 2.3. During the teach phase, the robot is manually driven and a pose graph is built in the same way as normal using VO. GPS measurements are also logged and associated with each keyframe but no effort is made to make the poses from VO consistent with these observations. In the repeat phase, a local window of GPS observations from the current section of the map is recalled. Together with a local window of the live repeat GPS measurements, these are used to estimate a path-tracking error in the vehicle frame. The details of this are described in Section 4.2.2. The GPS error is then used to define the cost function:

$$J_{\mathrm{GPS}} = \frac{1}{2}\mathbf{e}_{\mathrm{GPS}}^T \mathbf{\Sigma}_{\mathrm{GPS}}^{-1} \mathbf{e}_{\mathrm{GPS}}. \tag{4.1}$$

The GPS cost is combined with the VT&R vision cost function, J_{vision}, consisting of stereo landmark cost terms to form a nonlinear least squares optimization problem:

$$J = J_{\text{vision}} + J_{\text{GPS}}. \tag{4.2}$$

Like the odometry problem in Chapter 3, it is solved with the STEAM implementation of the dogleg Gauss-Newton algorithm [51]. The problem is well-defined as long as at least one of vision or GPS is available. If neither is available at a given time, localization is not attempted and the robot relies on VO and a motion prior.

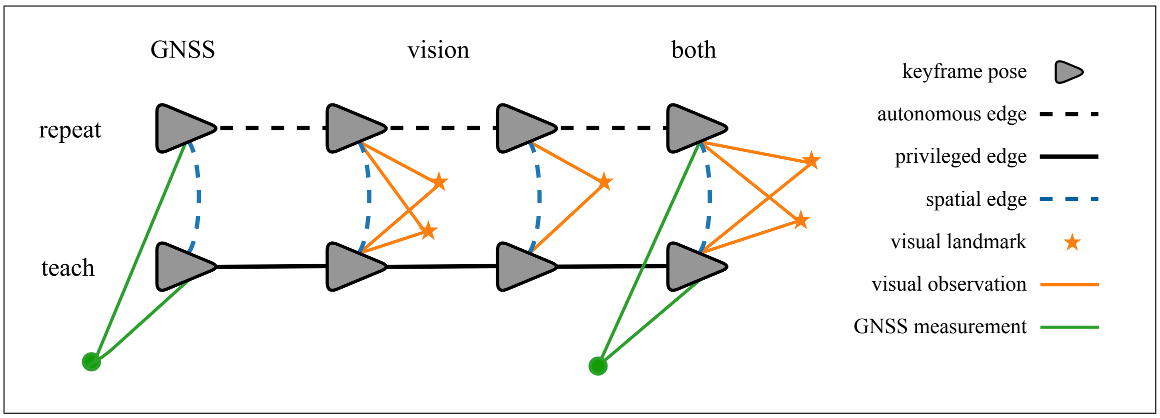

Figure 4.1: Diagram illustrating how path-tracking error can be calculated using either vision or GPS or a fusion of both. Privileged edges represent transformations calculated from the teach run VO while autonomous edges are calculated from repeat VO. Spatial edges come from localization. Path-tracking error is calculated for each sensor independently in the local frame. No single privileged frame is used, and the map does not need to be globally consistent.

4.2.2 GPS Path-Tracking Error

To use a global sensor for localization, an estimate of the vehicle's pose in a global frame is typically required. Drift from dead-reckoning can produce arbitrarily poor pose estimates even if a full batch optimization is performed on the pose graph. Because we estimate the GPS path-tracking error independently, translational bias in the map is cancelled out when subtracting the live position from the map position. Consequently, place-specific bias due to complex factors such as multipath reflections does not reduce the performance of our algorithm.

After subtraction, the error vector is still in the global frame and needs to be rotated to the local frame. We estimate the orientation of the local frame with respect to the global frame using linear regression on a local window of GPS measurements. The result is also reused to calculate the GPS receiver position from the potentially noisy observations. While vision performs 3D metric localization, we make a planar assumption in estimating our GPS path-tracking error. During logging, GPS measurements are projected to Universal Transverse Mercator (UTM) coordinates resulting in data points with an x and y position and a timestamp. On repeat, a small local window of these data points is recalled for both the teach and repeat runs. For each window, least-squares regression is used to model the estimated position, $(\hat{x}, \hat{y})$, as a function of time, t:

$$\hat{x} = \bar{x} + \hat{\beta}_{1,x}(t - \bar{t}), \tag{4.3}$$

$$\hat{y} = \bar{y} + \hat{\beta}_{1,y}(t - \bar{t}), \tag{4.4}$$

$\hat{\beta}_{1,x}$ and $\hat{\beta}_{1,y}$ are the estimated linear regression slope parameters while $\bar{x}$, $\bar{y}$, and $\bar{t}$ denote the mean measurements within the current window. This parameterization allows extrapolation to the current keyframe time for estimating (x_m, y_m) and (x_q, y_q), the map and repeat positions, respectively. The model assumes a constant velocity for the vehicle — a reasonable approximation for the small windows used (typically on the order of 1m). More sophisticated models such as Gaussian processes could be employed but are outside the scope of this thesis. We also assume the vehicle velocity is parallel to the vehicle frame's x-axis (forwards), which is reasonable for nonholonomic robots. The estimated heading, $\hat{\theta}$, is calculated using the slope parameter estimates from regression:

$$\hat{\theta} = \operatorname{atan2}(\hat{\beta}_{1,y}, \hat{\beta}_{1,x}). \tag{4.5}$$

The heading estimate, $\hat{\theta}_{q0}$, for the live frame, $\underrightarrow{\mathcal{F}}_q$, with respect to the UTM frame, $\underrightarrow{\mathcal{F}}_0$, is used to generate the rotation matrix, $\hat{\mathbf{C}}_{q0}$. We calculate the error between teach and repeat positions in the UTM frame, $\mathbf{r}_0^{mq}$, and rotate it to get an estimated position error in the live frame:

$$\hat{\mathbf{r}}_q^{mq} = \hat{\mathbf{C}}_{q0}\left(\hat{\mathbf{r}}_0^{m0} - \hat{\mathbf{r}}_0^{q0}\right) = \hat{\mathbf{C}}_{q0}\begin{bmatrix} \hat{x}_m - \hat{x}_q \\ \hat{y}_m - \hat{y}_q \\ 0 \end{bmatrix}. \tag{4.6}$$

The result is assembled into the transformation matrix with a value of 0 set for the error in roll, pitch, and z-direction:

$$\hat{\mathbf{T}}_{qm} = \begin{bmatrix} \cos\hat{\theta}_{qm} & \sin\hat{\theta}_{qm} & 0 & \hat{r}_{q,1}^{mq} \\ -\sin\hat{\theta}_{qm} & \cos\hat{\theta}_{qm} & 0 & \hat{r}_{q,2}^{mq} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.7}$$

The state we are optimizing for in (4.2) is $\mathbf{T}_{qm}$, the transformation between the vehicle pose at the current repeat frame and the vehicle pose at the nearest keyframe in the teach run. Using the logarithmic map we define the $SE(3)$ error term [57] for our GPS error, $\mathbf{e}_{\text{GPS}}$, as a function of our state and the transformation matrix calculated in (4.7) using the GPS measurements:

$$\mathbf{e}_{\text{GPS}} = \ln(\hat{\mathbf{T}}_{qm}\mathbf{T}_{qm}^{-1})^{\vee}. \tag{4.8}$$

The covariance on our estimate, $\mathbf{\Sigma}_{\text{GPS}}$, could be calculated analytically from the linear regression. However, this involves certain assumptions on the error properties of GPS that may not be satisfied. Instead, we find better results by estimating the uncertainties through empirical trials. A liberal uncertainty is added in the roll, pitch, and z direction as a weak prior.

4.2.3 Outlier Rejection

GPS measurements often suffer from outliers due to nonlinear effects and biases. To reject outlying measurements within our local regression window we use RANSAC [55]. After the path-tracking error calculation, we use M-estimation to account for estimates that disagree significantly with the prediction from VO or, if available, visual localization. The robust cost function from the DCS family [52] that we utilize is

$$\rho(u) = \begin{cases} \frac{1}{2}u^2 & u^2 \leq k^2 \\ \frac{2k^2u^2}{k^2+u^2} - \frac{1}{2}u^2 & u^2 > k^2 \end{cases}. \tag{4.9}$$

We found the robust cost function is rarely a factor when a strong GPS fix is available but can be helpful when fewer satellites are seen. It also prevents GPS from degrading the performance of VT&R when good vision is available.

4.3 Evaluation

4.3.1 Experimental Setup

To validate our approach, we conducted a set of online experiments on a live robot. The 350m path, shown in Figure 4.2, was intentionally designed to include a 120m section inside the UTIAS MarsDome mid-

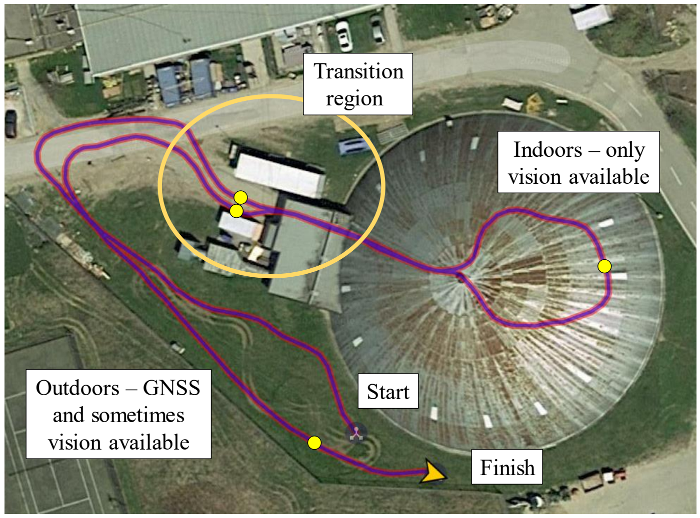

Figure 4.2: Overhead view of the approximate path driven in the experiments. Yellow dots denote the placement of key locations for independent ground-truth measurement of the path-tracking error. Inside the dome, the robot was forced to rely on only visual localization. Outside, the robot relied on mainly GPS and used vision when available.

way through. The terrain inside the MarsDome is a highly non-planar mix of gravel and dirt. Outside, the environment is a mix of pavement and grass. The path requires at least one transition from GPS being available to unavailable and vice-versa. As day turns to night, lighting conditions inside the MarsDome remain relatively consistent but appearance change impedes the robot's ability to visually localize outdoors as all repeats rely on the teach run as a single experience for localization; MEL was deliberately not used in order to prompt vision failures.

The path was first manually driven at midday on October 23, 2020. An initial repeat was conducted directly after the teach run to verify our ability to immediately re-drive paths. Subsequent repeat runs were performed on the evening of October 26, beginning at 5:19pm. They continued until after twilight with the final run beginning at 7:07pm. Sunset occurred at 6:16pm. The timing was chosen to demonstrate the algorithm's ability to both fuse vision and GPS in the earlier repeats and transition to only GPS navigation in the later repeats.

The same Clearpath Grizzly UGV used to collect data in Chapter 3 was used for these online experiments. We used the same onboard NovAtel SMART6-L GPS receiver and Bumblebee XB3 stereo camera as described in Section 3.3.1. A second SMART6-L receiver served as a stationary base station for RTK corrections. The receivers were configured for GPS satellites only in this experiment; sensor availability could be improved with additional GNSS constellations. LED headlights are mounted below the stereo camera. They were turned on for all runs to allow VO outdoors at nighttime and provide consistent conditions inside throughout the experiment. The algorithm ran on an onboard Lenovo laptop and interfaced with the Grizzly via Robot Operating System (ROS) [58]. All estimation is done in the vehicle frame with the fixed sensor-to-vehicle transformations explicitly handled for both the camera and GPS receiver.

To provide an additional measurement of path-tracking error independent from either sensor, jigs were placed at four locations along the path as indicated in Figure 4.2. The jigs, seen in Figure 4.3, consisted of a ruled board affixed to the ground over which the robot drives. By comparing the tire positions of each repeat run to that of the teach run, a lateral and heading error could be estimated. This simple method has the property of working well both indoors and outdoors.

4.3.2 Results

A total of 10 autonomous repeat runs of the experiment path were conducted totalling 3.5km of driving. The first began immediately after the path was manually driven. Immediate repeating remained easy and path tracking was highly accurate with the largest measured lateral error at the jigs just 2cm and all measured heading errors less

Figure 4.3: Jig used for measuring the ground-truth path-tracking error at several locations along the path. Lines are marked every 2cm and the board is affixed to the ground in the robot's path.

than 1°. A summary of measured path-tracking errors for all repeats is provided in Table 4.1.

In general, the runs before sunset achieved good visual localization in most sections despite the different lighting conditions and the passing of several days. As darkness set, visual localization was very difficult outdoors but remained relatively consistent indoors as expected. All runs were successfully completed aside from the tenth in which the robot's headlights failed approximately 240m along the path in a section without a GPS fix. Lack of lighting led to failures in VO and the absence of GPS increased the localization uncertainty beyond a threshold VT&R deemed safe for continued driving so the robot stopped. This unexpected event demonstrated the safe failure mode of VT&R.

All runs experienced long sections of GPS dropout during the indoor driving section. Several runs experienced prolonged vision dropout for up to 90m due to outdoor appearance change. The conditions across all repeats are summarized in Figure 4.4. The results in this figure

Table 4.1: Measured Error at Key Locations (see yellow dots in Fig. 2)

Repeat Start Time	*Error*	*Entering Dome*	*In Dome*	*Exiting Dome*	*Grass*
1:54pm	Lateral (m)	0.020	-0.005	-0.010	0.005
	Yaw (deg)	0.0	-0.6	-0.6	-0.6
5:19pm	Lateral (m)	0.015	-0.010	-0.030	-0.005
	Yaw (deg)	0.6	0.0	0.6	0.6
5:30pm	Lateral (m)	0.005	-0.005	-0.020	-0.005
	Yaw (deg)	0.6	0.6	0.6	0.6
5:43pm	Lateral (m)	0.010	-0.005	-0.020	-0.005
	Yaw (deg)	0.0	0.6	0.0	0.6
5:58pm	Lateral (m)	0.010	-0.005	0.000	0.000
	Yaw (deg)	0.0	0.6	0.0	0.0
6:11pm	Lateral (m)	0.020	-0.010	-0.020	0.010
	Yaw (deg)	0.0	0.0	0.6	0.0
6:23pm	Lateral (m)	0.040	-0.010	-0.030	0.125
	Yaw (deg)	0.0	1.3	0.6	0.6
6:38pm	Lateral (m)	0.040	-0.005	-0.080	0.135
	Yaw (deg)	0.0	0.6	0.6	0.6
6:51pm	Lateral (m)	0.065	-0.010	-0.040	0.155
	Yaw (deg)	0.6	0.0	0.6	0.6
7:07pm	Lateral (m)	0.080	0.00	-0.080	–
	Yaw (deg)	0.0	0.0	0.6	–

illustrate the importance of multiple sensors for robustness in difficult environments. Our combined algorithm has to rely on VO alone for much shorter and less frequent sections.

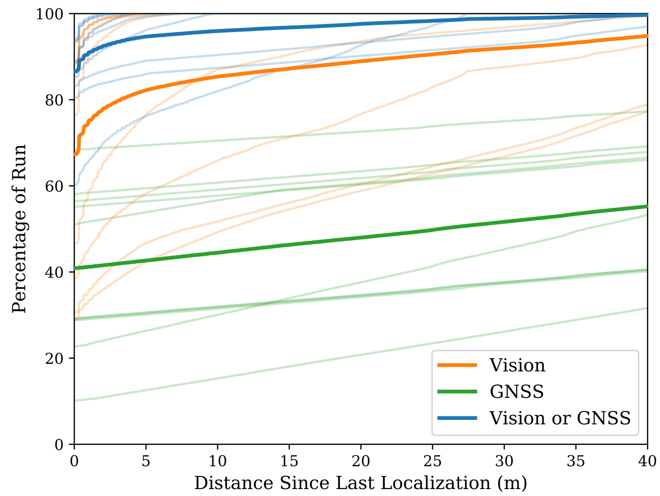

Figure 4.4: Plot showing the percentage of each run in which the robot has successfully localized within the last x metres travelled with a particular sensor. Individual runs are shown in lighter colours while the mean is plotted in a darker colour. Top-left is better as it means the robot had to rely on dead-reckoning for shorter periods. Having both sensors available leads to a better chance at localization.

The mean absolute path-tracking error across all runs, as measured at the four jig locations is just 2.9cm laterally and 0.4° in heading. This measured error is a combination of both the localization error and the path-tracking controller error, as well as measurement error from the jig itself. The metric localization error estimates from our algorithm are correlated with the measured error implying the mean absolute localization error is smaller than the total path-tracking error. The only setting in which we see a noticeable inaccuracy is at the fourth measurement location during runs after sunset. Here, the error is consistently 12–15cm suggesting that there may be some bias in the teach run GPS measurements. Another factor may be the manual driving of the teach path itself, which was slightly more erratic in this area of the path as the human pilot tried to target the centre of the jig. That the error does not occur in earlier runs suggests our method is robust to GPS bias when good vision is available and our sensors are weighted appropriately in (2), the localization cost function.

Equally important to path-tracking accuracy is the smoothness of the path following, especially in transition regions. Figure 4.5 illustrates the relationship between the localization error estimated by the robot and the availability of each sensor. The first 100m of the path corresponds to outdoor navigation where GPS is typically available. It is followed by 120m indoors where the robot must rely on vision. After exiting the MarsDome, the robot typically drives about 30m before regaining a GPS fix for the final 100m outdoors. The entrance to the MarsDome allows only a few centimetres of clearance for the Grizzly so smooth path following was not only a theoretical consideration but was of practical consequence.

Qualitatively, the path-tracking was very smooth throughout the experiment and no difference was noticed compared to when VT&R has good vision consistently. From the plots, we see the estimated localization error stays stable throughout the transition zones. The only spike that occurs is in the worst-case run when a GPS fix is obtained after a prolonged section of VO reliance. Crucially, this spike corresponds to the actual path-tracking error due to dead-reckoning, not a discontinuity in our map due to offset from the global frame. The controller is able to smoothly utilize this new information to recenter the robot on the path. We note the smaller estimated localization errors seen in the bottom right plot of Figure 4.5 from approximately 210m to 260m are due to the lack of corrections from either sensor in this section, not because the true path-tracking accuracy is better.

Without our addition of GPS, the later repeat runs would almost certainly have failed. Consider the 6:51pm run whose sensor availability

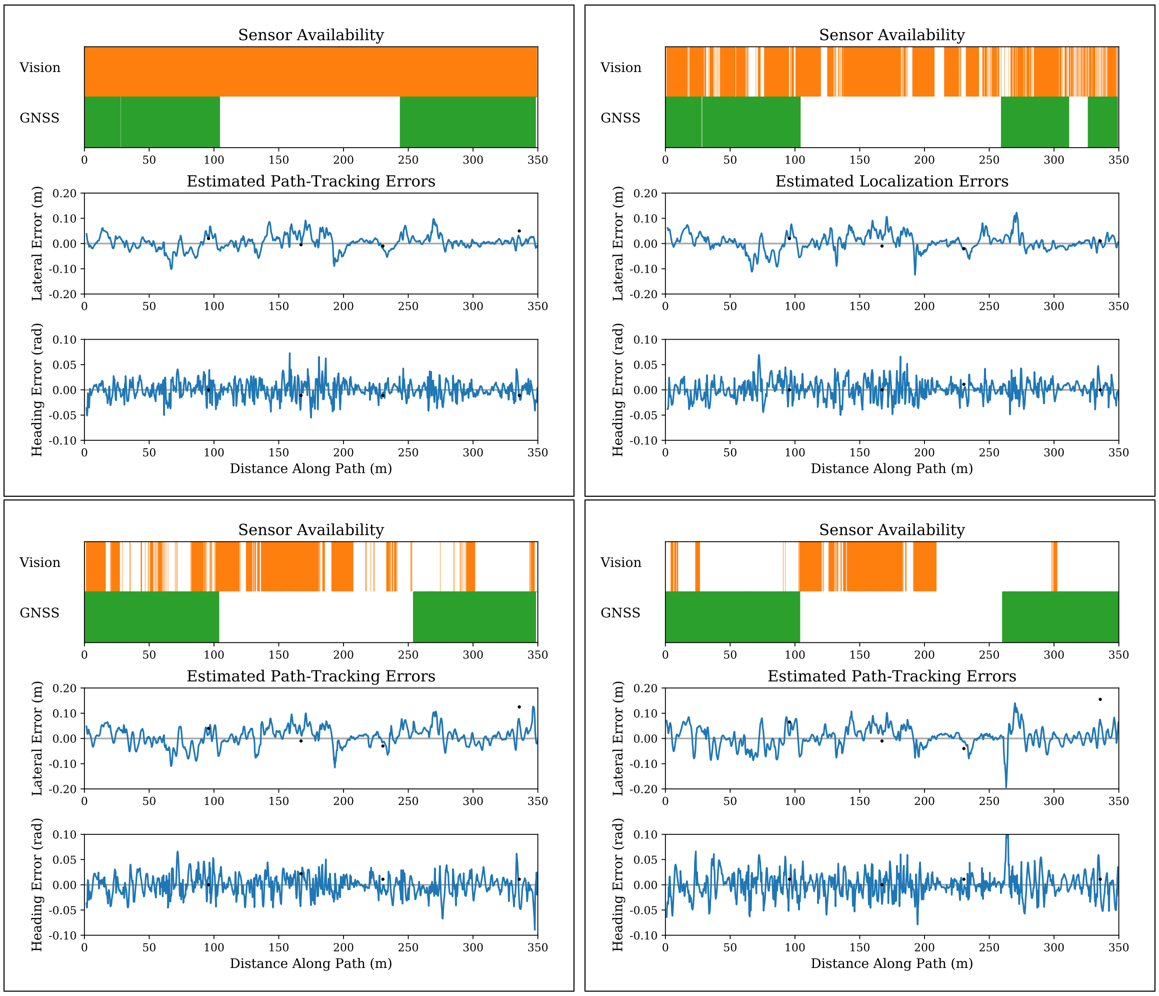

Figure 4.5: Sensor availability versus error estimates for four repeats. Top left: the 1:54pm run representing a best-case run. Top right: The 6:11pm run during which sunset occurred. Bottom left: The 6:23pm run representing a typical repeat in the experiment. Bottom right: the 6:51pm run representing the worst-case situation. The coloured bar shows when a sensor was used for localization. For example, the large gaps in the GPS bars correspond to the period when the robot was indoors. Black dots on the error plots correspond to the jig measurements in Table 4.1.

is shown in the bottom right plot of Figure 4.5. During the first 100m, the robot has little success at visual localization meaning it would have to rely on VO nearly exclusively. MacTavish et al. [44] showed the mean drift rate for our VO pipeline in nighttime conditions is 2.38% suggesting it is highly unlikely the robot could have safely navigated the dome entrance after 100m of dead reckoning. In this scenario, the final third of the repeat would also need to rely exclusively on VO leading to expected final errors in excess of 3m compared to the few-centimetre-level errors we measured. Our method allowed the robot to accurately and efficiently repeat the path in all experimental conditions. Finally, we note that the small additional computation required to estimate GPS error from the local windows of observations did not have a significant effect on the speed of the VT&R algorithm and the runtime was well within the requirements to operate in real-time.

4.4 Summary

In this chapter, we presented a robust system for path following utilizing both vision and GPS for localization. Unlike related methods fusing vision and GPS, we do not attempt to reconcile measurements from the two sensors into a single global coordinate frame. By delaying sensor fusion until the path error is calculated, we avoid requiring a costly optimization for map updates, even after prolonged sensor dropout. We validated our approach through an extensive field trial on a real robot. We emphasize three key results: a) the system maintains high path-following accuracy on the order of centimetres, b) the vehicle was able to overcome long sections of dropout of one or both sensors, and c) there was no spike in error signal due to frame offset during the transitions between sensors and the vehicle continued to drive smoothly.

Our experiments were performed using only RTK-corrected GPS measurements. However, the method presented here could transfer to less accurate GPS setups provided the increased path-tracking error does not exceed the convergence region for visual localization. This would be appropriate in many applications where the acceptable path-tracking error is larger outdoors in open space than indoors.

We assumed VO estimates are available for the prediction step but we could substitute other sensors or rely on GPS alone in the event of total vision failure. We are also not limited to GPS; the “relatively lazy” approach holds for integrating other absolute sensors into relative navigation systems. Finally, our work need not be a replacement for multi-experience localization. In Section 6.2.2, we briefly discuss how we might benefit from MEL by later adding in GPS to sections where a fix was unavailable during the teach run or by using multiple experiences to average out GPS noise. We believe this work will be beneficial in a number of applications such as mining that involve both natural environments where appearance change is a factor and more structured but confined space where a GPS signal is not guaranteed. It could also have applications in marine robotics where visual features are sparse on open water but the higher accuracy and robustness to satellite occlusion afforded by vision is required near shore.

Chapter 5

Visual Teach & Repeat 3 with GPS

In this chapter, we explore the benefits of adding a single GPS receiver to an already successful relative navigation algorithm. We first describe the Visual Teach and Repeat 3 (VTR3) project and its myriad uses for GPS. We then discuss extensions made to the work of Chapter 3 to implement an easy-to-use GPS odometry software package that may be utilized alone or in combination with other navigation algorithms. Finally, we show how this package is used in VTR3 as an alternative or complement to VO.

5.1 Visual Teach & Repeat 3

VT&R first demonstrated its utility in the robotics world over 10 years ago. Its success has spurred many extensions [39], [37], [41], [42] and is still bearing fruitful research projects today [59], [56]. Software evolves quickly as does the robotics research landscape, so it is important to keep the VT&R codebase well-maintained and agile enough to be conducive to new projects and use cases. With this in mind, we launched the VTR3 project with the aim of revitalizing the software stack and setting up VT&R for another five years of success.

VT&R relies on a number of dependencies including Ubuntu, CUDA, OpenCV, Eigen, lgmath, and STEAM. The first milestone of the project was to upgrade to the latest versions of these dependencies and ensure VT&R could be run on the latest hardware. A major component of the project was swapping out the ROS1 [58] backbone for the emerging ROS2 framework and the custom Robochunk logging system of VTR2 for the more standard rosbag2. ROS2 provides a distributed communication system and upgraded client libraries while rosbag2 provides fast access based on the SQLite database engine. A further goal was to provide a flexible architecture that allows easily adding new sensors and navigation modules. Future use cases include pipelines that use lidar or radar as the primary navigation sensor, as well as stereo camera pipelines that swap out point features for deep-learning techniques. Finally, the project will conclude with the open-sourcing of VTR3, expected in the latter half of 2021. Rigorous field-testing has been conducted to detect issues in preparation for this goal.

There are several ways GPS fits into the VTR3 project. The two major examples are through using single-receiver GPS measurements as a prior for localization and using high-quality GPS in localization directly. The former will be discussed in the sections that follow. The latter is quite similar algorithmically to the work presented in Chapter 4 (which was originally implemented in VTR2) so will not be expanded on further here.

One less involved way GPS can help in VTR3 is in topological localization. In VTR2, the user had to manually specify the starting location of the robot in the pose graph. If the robot was not actually in that area, it would not be able to get an initial localization and therefore could not perform the repeat. VTR3 will log absolute GPS positions, when available, and store them in each vertex. If a GPS signal is available when the system is restarted, it may be used to place the robot on the map automatically and thus eliminating a potential source of user error. The GPS coordinates stored in the map may also be used to fetch the satellite image used as the background for the user interface, seen in Figure 5.1. Previously, this too had to be manually specified by the user. An accurate background will make it easier and faster for the end-user to specify goals for the robot. The absolute GPS references now stored in the pose graph may even enable future features that have yet to be foreseen.

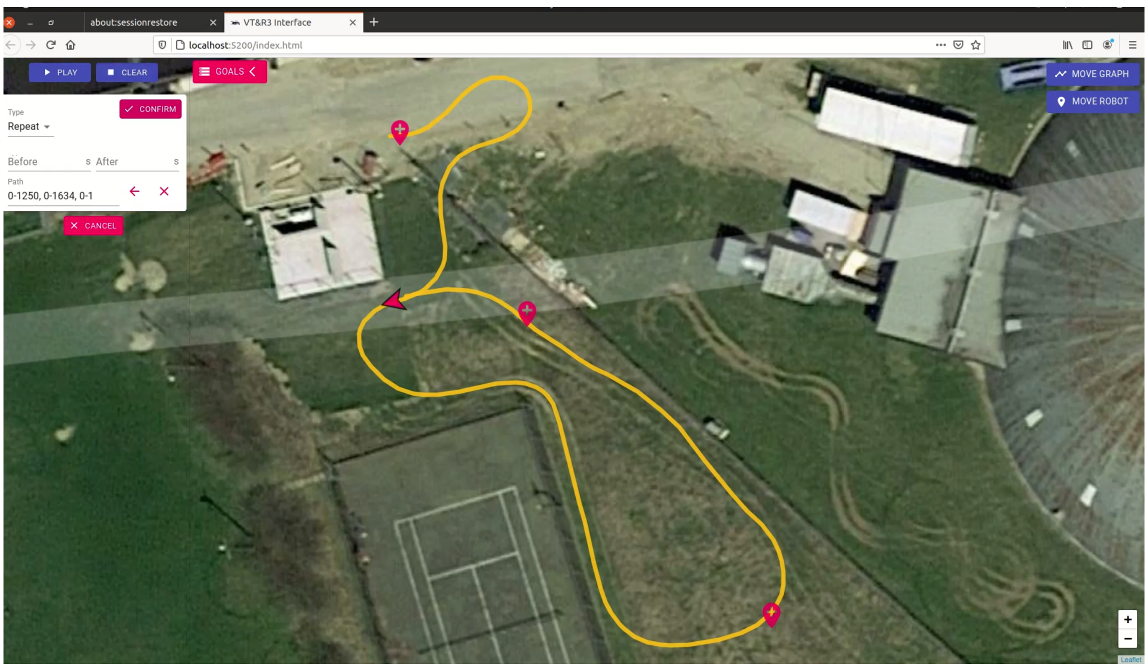

Figure 5.1: The revamped user interface used in VTR3. GPS can be helpful for both placing the robot (red arrow) at the correct location on the map (yellow curve) and for placing the map on the correct satellite image.

5.2 Carrier Phase Odometry Package

The results of Chapter 3 showed single-receiver GPS measurements could be an accurate source for odometry. While the estimation loop itself was fast enough to run online, there were some manual steps involved in pre-processing the GPS data that prevented directly running that code on the robot. The goal of the work presented in this section was to create an easy-to-use package that would provide live GPS odometry estimates to be used alone or as part of a larger navigation stack. The result is the open-source Carrier Phase Odometry (CPO) project found at `https://github.com/utiasASRL/cpo`.

CPO is a ROS2 project designed to work with the majority of modern GPS receivers. It consists of four packages. The `cpo_frontend` package acts as a driver and preprocessor for the carrier phase measurements. The input is standard RTCM1004 (GPS observables) and RTCM1019 (GPS ephemerides) messages logged over serial. The output of this package is a stream of custom TDCP messages, defined in the `cpo_interface` package, published to a ROS2 topic. These messages act as pseudomeasurements pairing a set of satellites observed at two consecutive time points. The front-end node parses the binary RTCM messages, calculates approximate pseudorange GPS solutions, and extracts vectors relevant to the estimation problem. It also estimates and corrects for the tropospheric delay difference as discussed in 3.2.2.

The `cpo_backend` package is responsible for state estimation. Each pseudomeasurement message received from the front end is used to construct $n-1$ TDCP error terms, where n is the number of observed satellites. These are combined with the WNOA motion prior and nonholonomic factors discussed in Chapter 3 to form a nonlinear least squares cost function. Optimization is handled by STEAM over a sliding window. The user may use ROS2 parameters to easily configure the relative importance of these factors as well as the size of the sliding window to fit their application. The result is full SE(3) pose estimates of the vehicle in the ENU frame. These are published with a standard ROS2 `PoseWithCovariance` message either at a fixed rate or with each new incoming pseudomeasurement. This node also provides a query trajectory service. The service accepts two timestamps and returns the relative pose and its covariance over the interval. The two query times do not need to be at GPS measurement times as the continuous-time trajectory can be sampled at any point [60]. For more details on how this relative pose and covariance is estimated, see Appendix A.

Both the message format and the service are defined in the `cpo_interfaces` package. A few Python scripts for visualization and analysis are provided in `cpo_analysis`, one of which can be seen in Figure 5.2. In addition to online operation, the project supports using ROS2's simulation time for offline testing using saved data. This feature combined with the provided sample data allows new users to quickly try out the project for themselves.

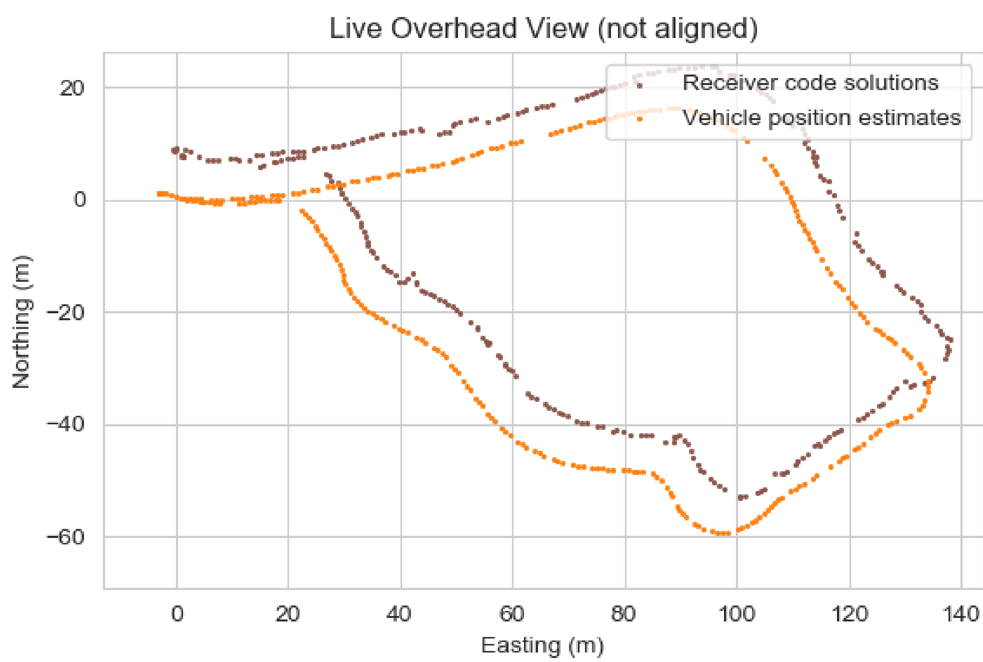

Figure 5.2: Optional interactive plot that can be run while running CPO to plot the vehicle trajectory in real-time. Qualitatively, we see that the orange CPO estimates using TDCP are smoother than the brown pseudorange position estimates.

5.2.1 Performance Evaluation

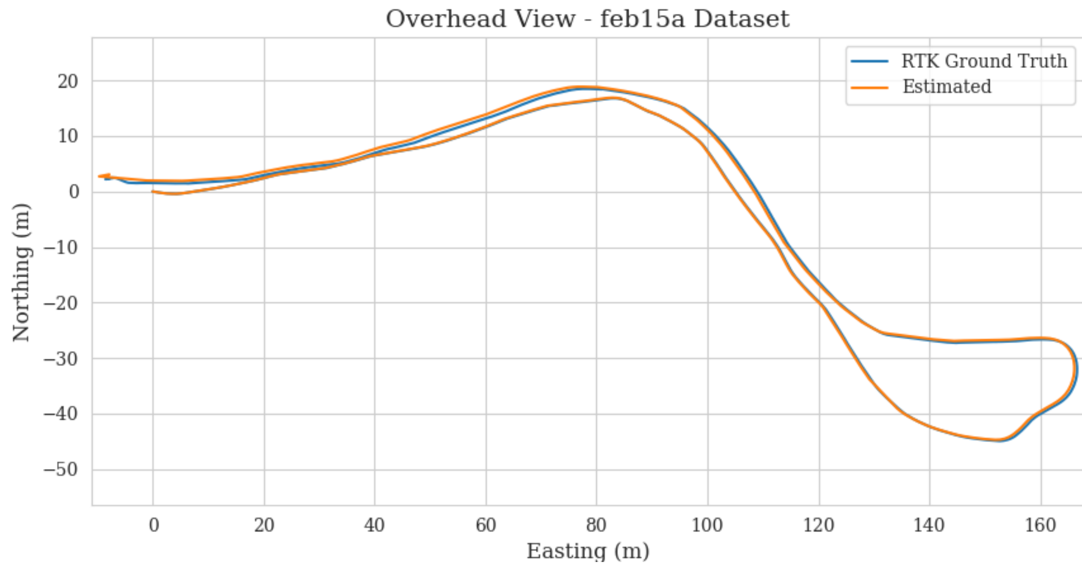

Figure 5.3: Overhead view of one path from the dataset that begins at $(0, 0)$ and proceeds counterclockwise. The estimates from CPO (orange) smoothly and accurately track the RTK ground truth.

To validate the effectiveness of our CPO project, we ran the pipeline on 1.4km of previously collected data. Figure 5.3 shows our GPS odometry next to the RTK ground truth for the run in our dataset with the largest final error. Qualitatively, we see our estimates smoothly track both the shape and scale of the driven path.

Quantitatively, we confirm the estimates are of high accuracy. The average final translational error is just 0.27% of distance travelled (drift rate). Figure 5.4 plots the errors with respect to ground truth for the four independent runs. We also notice that, as before, the errors vary smoothly and linearly. The average execution time of the backend node from receiving a new TDCP message to publishing an updated pose estimate is just 2.56ms with the optimization itself requiring an average of 2.15ms. This could likely be reduced further with a smaller window size and still provide accurate results but our GPS messages are set to log every 100ms so we are already well within the real-time

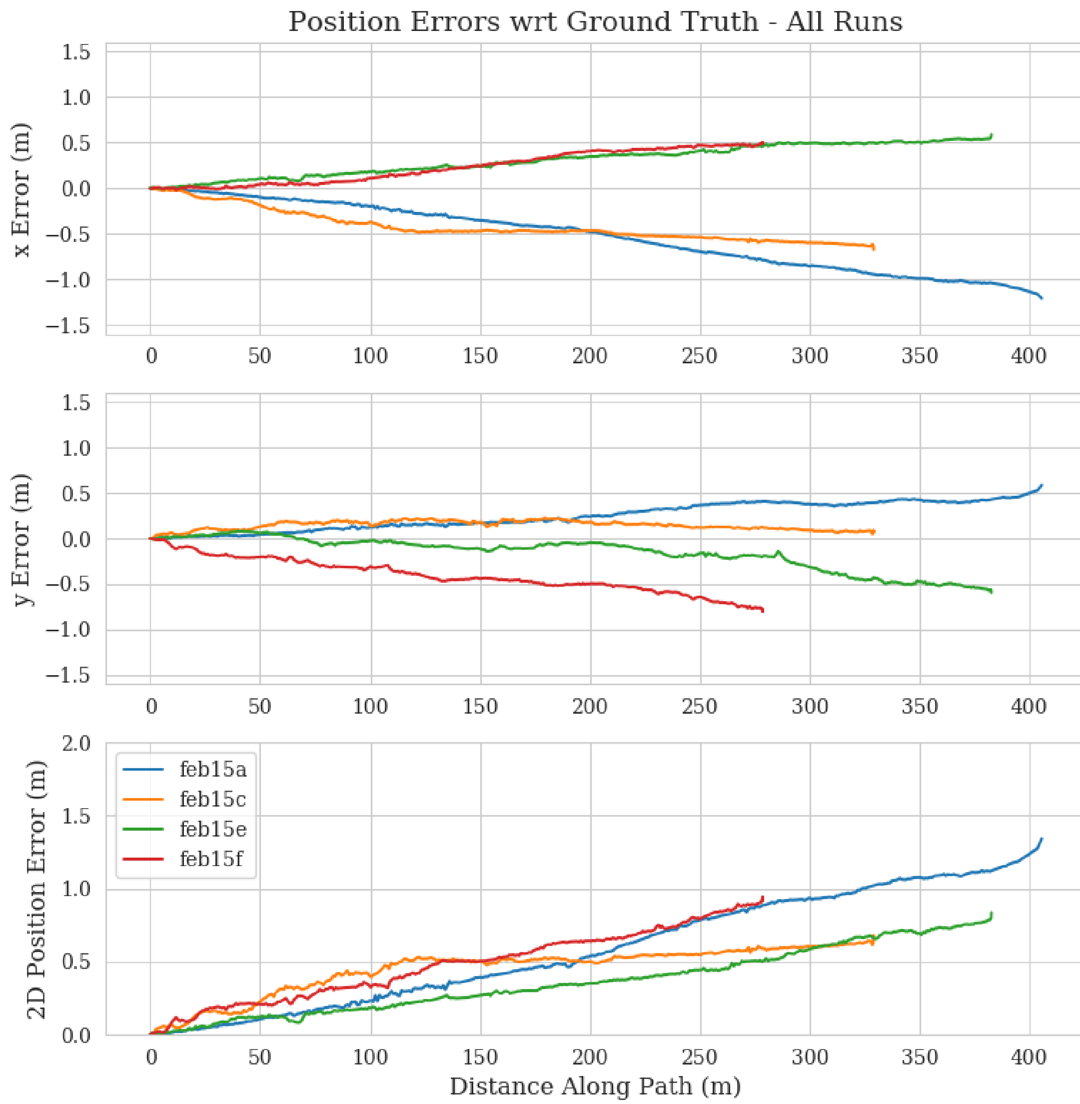

Figure 5.4: Plot of position errors with respect to ground truth versus distance along the path. Total drift rate is less than 0.5% in all runs.

constraints.

5.2.2 Evaluation of Atmospheric Effects

There is some discrepancy among prior works that use TDCP on the importance of correcting for atmospheric delays. Some works [8] correct for it while others [61] assert that over the small time period the carrier phase difference is calculated the delays can be considered constant. To test this empirically we also run our algorithm on the data without the tropospheric correction. We find on all runs the (relatively easy to calculate) correction improves the final error, thus supporting its use. Interestingly, the performance gain varies between runs, likely due to differences in satellite geometry. The tropospheric delay is roughly proportional to $\frac{1}{\sin \epsilon}$ where ϵ is the satellite elevation. Therefore, the delays are more variable and the correction is more important when using measurements from satellites lower in the sky.

Dataset	Distance Travelled (m)	Final 2D Translation Error (m)	
		Without Tropospheric Correction	With Correction
A	409	1.76	**1.33**
B	329	0.69	**0.68**
C	384	1.31	**0.85**
D	280	0.98	**0.95**

Table 5.1: Effect of correcting for tropospheric delay difference on final translational error across four independent runs. In all cases, the correction improves final error. The magnitude of the improvement is dependent on satellite geometry.

5.3 Incorporating GPS Odometry into VTR3

5.3.1 Architecture

The intuitive strategy for integrating our GPS odometry within VT&R is to add TDCP terms into the sliding window bundle adjustment stage, similar to what was done in the experiments of Section 3.4. In this tightly-coupled setup, the front-end node of CPO would run, publishing ROS2 messages that contain all the information needed to construct a TDCP factor. VT&R could then subscribe to these messages and easily accommodate these new factors in the existing nonlinear least squares optimization problem. In theory, the careful combination of two reliable measurement sources should produce a probabilistic estimator that outperforms either source alone. This was our first approach; however, it was found to have several disadvantages in practice that will be detailed shortly. Even after careful tuning, we could not produce an estimator that consistently outperformed both single-sensor odometry algorithms.

Our second approach recognizes that accurate odometry is not a direct requirement for VT&R and the path-tracking problem. Rather, odometry is utilized as a prior for the localization problem. In this "lazy" approach, we calculate odometry using both sensors (when available) independently in separate threads. GPS odometry is computed in the `cpo_backend` node while VO is computed in VT&R's navigator node as normal. We then let the localizer decide the best source to use as its prior when the time comes. A diagram of the setup can be seen in Figure 5.5.

There are several advantages this strategy has over the tightly-coupled solution. Perhaps the largest is that, from a practical point of view, the latter strategy is much more easily extended to versions of

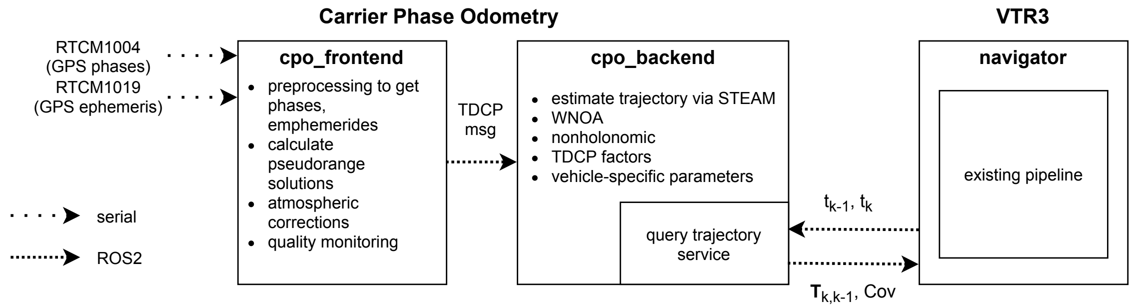

Figure 5.5: Architecture diagram for our chosen approach to include GPS odometry in VT&R. All GPS estimation is done separately from VT&R which can query the estimator to receive transformation estimates for its intra-run edges.

VT&R that swap the stereo camera used in this work for other primary sensors such as lidar or radar. The tightly-coupled strategy required careful handling of edge cases in the stereo bundle adjustment pipeline. In the lazy strategy, the bundle adjustment pipeline does not need to know that GPS exists. Another advantage is that we do not have to balance the size of our sliding window between the two sensors. We are free to optimize the sparse GPS factors over a larger window while keeping the visual bundle adjustment over a small window containing many stereo landmark factors. As well, parameter tuning becomes less important when we are not fusing the two sensors directly. VT&R can get away with unrealistic covariances on its stereo landmark terms to some degree because only the relative weighting of terms is important. When adding GPS terms to the optimization problem, suddenly the results become more dependent on our choice of parameters. Finally, the lazy approach avoids having to estimate and bookkeep the global orientation in VT&R. In the tightly-coupled approach, the global orientation becomes an extra state variable that must be included in our state vector if and only if we are using GPS factors. In the lazy approach, it is encapsulated in CPO.

5.3.2 Implementation Details

Because of our chosen approach, our algorithm runs very similarly to VT&R without GPS with a couple of key differences. After a new keyframe is created and bundle adjustment has run, a request is sent to CPO's query trajectory service with the timestamp of the current keyframe, t_k and the previous keyframe, t_{k-1}. The response, $\mathbf{T}_{k,k-1}$ and its covariance, is stored in the pose graph edge as a second, separate transformation. As the current keyframe was just captured and the GPS measurements are asynchronous with respect to the camera images, the query trajectory service typically has to extrapolate slightly past the latest GPS measurement. However, this is easily handled by STEAM's continuous-time estimation. To obtain a better estimate of $\mathbf{T}_{k,k-1}$ utilizing GPS measurements received after t_k, the service is called again after a fixed delay and the VT&R pose graph is updated accordingly.

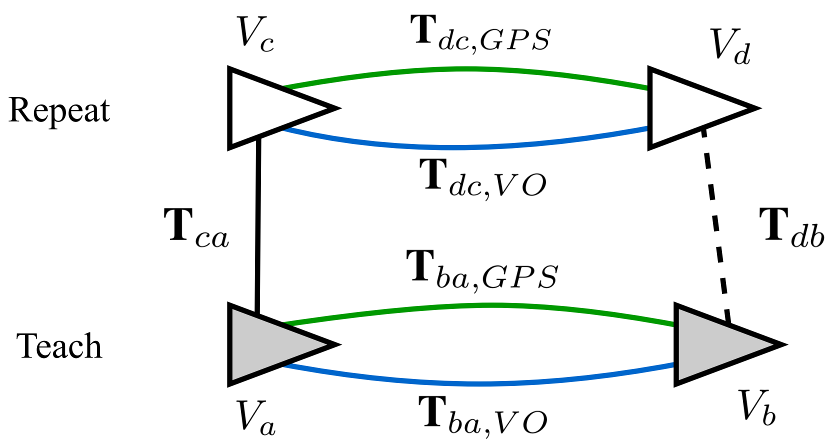

Figure 5.6: Diagram of the "localization chain" showing the part of the pose graph relevant to the localization problem. Transformation estimates from odometry within each run may be composed with the previous localization result, $\mathbf{T}_{ca}$, to provide a prior for the current estimation problem.

The new transformations are depicted as green edges in Figure 5.6. In the localization problem, we are required to estimate the transformation between the current repeat vertex and the closest teach vertex, V_d and V_b in Figure 5.6, respectively. The standard version of VT&R composes the edges from VO with the most recent localization result to generate a prior:

$$\check{\mathbf{T}}_{db,vo} = \mathbf{T}_{dc,vo}\mathbf{T}_{ca}\mathbf{T}_{ba,vo}^{-1}. \tag{5.1}$$

With GPS odometry available, we now have an alternative method for generating this prior:

$$\check{\mathbf{T}}_{db,gps} = \mathbf{T}_{dc,gps}\mathbf{T}_{ca}\mathbf{T}_{ba,gps}^{-1}. \tag{5.2}$$

In the event of VO failure, we can now still calculate a prior via the GPS edges. If no GPS is available, we can still use VO as before. If both are available, we can either compare their covariances to determine which is likely to be more accurate or the user may decide to always prefer one sensor or the other.

One reason a good prior is important is that it provides a good initial condition for localization. Perhaps a more significant reason is that in the case where localization fails, the prior becomes our estimate for $\mathbf{T}_{db}$. When VO fails, visual localization is also likely to fail so the GPS odometry prior becomes very important.

5.3.3 Experimental Setup

We designed three experiments with each one validating a goal for our algorithm:

1. Replacing the VO prior used by VTR3 localization with a GPS odometry prior provides comparable performance under nominal conditions. In theory, the more accurate GPS odometry could provide a better prior than VO and therefore more accurate local-

ization. However, VO has shown it provides a sufficient prior for this task in normal circumstances, so we only need to show that GPS can do at least as well.

2. The GPS prior can be relied on for dead-reckoning when we are not able to localize visually. VTR3 often faces small sections where it cannot visually localize due to factors such as appearance change. To test this we will create a difficult scenario for the robot by not allowing the localization pipeline to use any feature matches for certain sections.

3. GPS odometry can be used to maintain path-following in the event of total VO failure. We will test this by replacing the left image from the stereo camera with a blank image simulating, for example, a total blockage of the camera. With GPS odometry, VTR3 should be able to regain normal localization once the camera recovers. We will also show that without GPS, this would result in a failed repeat.

We run each experiment first using a version of VTR3 without GPS, and then using our method configured to replace VO with GPS odometry. We test on three independent paths approximately 60m in length (one minute of driving per run). For Experiments 2 and 3, the sensor dropout is applied to a 20-second section midway through the repeat run to illustrate the behaviour both when the dropout begins and when the sensor recovers.

Data for the experiments in this chapter are once again collected via a Clearpath Grizzly UGV. For more information on the vehicle and its sensors, see Section 3.3.1. Figure 5.7 shows the network diagram for our GPS setup on the Grizzly. The NovAtel SMART6-L GPS receiver provides three serial communication lines. To provide ground truth for these experiments, COM2 is used to relay RTK corrections via the Grizzly radio to the receiver which then sends the RTK solutions to be logged via COM1. COM3 is used as an independent channel to log the observables needed for TDCP that will be used in the estimation in this section. This setup handles all the ways GPS is used in VTR3 with COM1 providing the position measurements used in relative localization (Chapter 4).

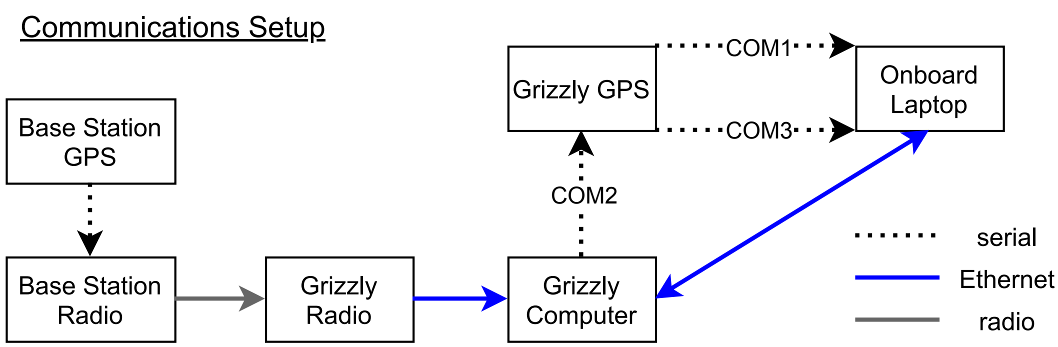

Figure 5.7: Network diagram showing how our robot is configured for logging both RTK ground truth (COM1) and the measurements needed for GPS odometry (COM3).

5.3.4 Results

Figure 5.8 shows the paths on which data was collected for these experiments. Repeat runs for Paths 1 and 2 were autonomously driven in good visual conditions (a sunny afternoon). Path 3 consists of a manually-driven route collected at nighttime. As a result, this path has a larger path-tracking error our localization pipeline must attempt to estimate. During these runs, we logged both the full stereo image stream and the raw GPS serial data.

Figures 5.9, 5.10, and 5.11 show the results for Experiments 1, 2, and 3 respectively on Path 3. Path 3 had the most challenging conditions and shows the worst-case performance of our method. The error plotted

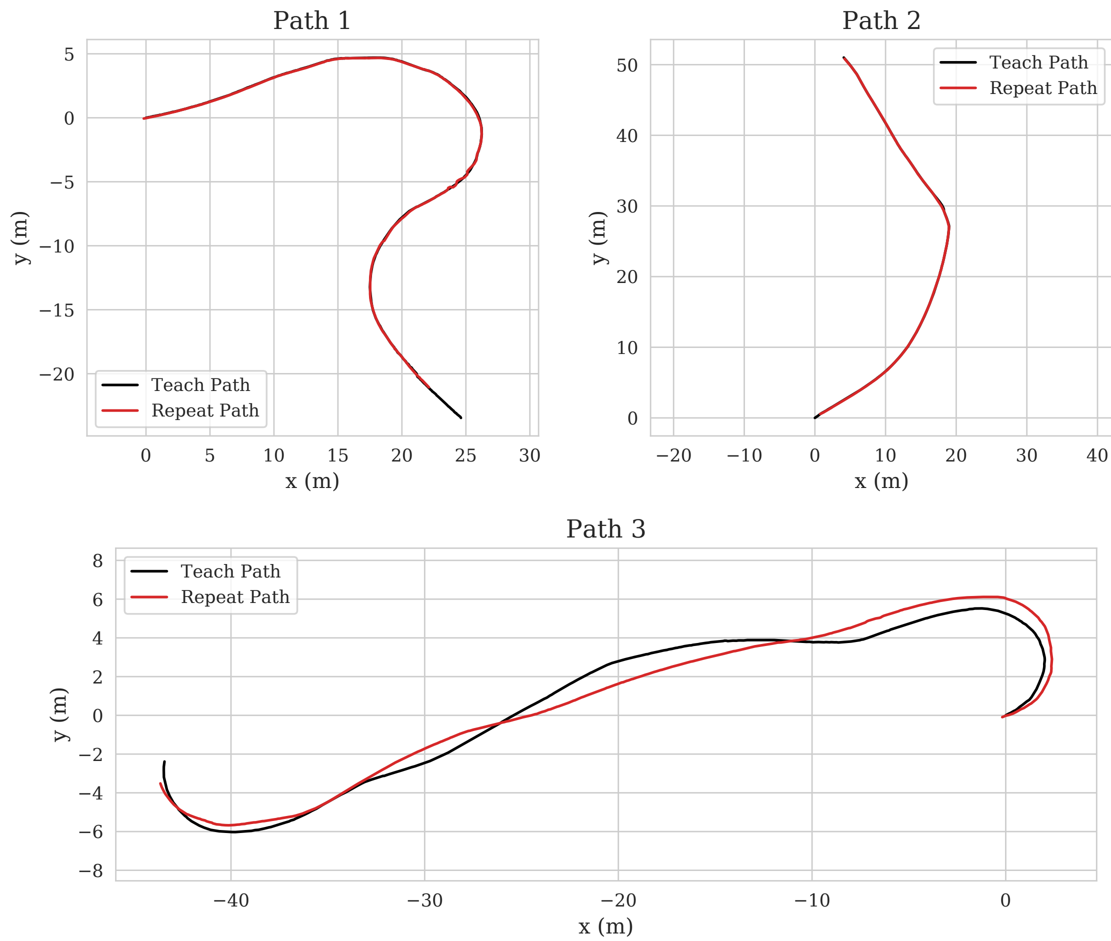

Figure 5.8: Overhead view of ground truth for the paths used in the VTR3 + GPS experiments. Paths 1 and 2 were autonomously collected using VT&R in good conditions. Path 3 was manually driven twice at nighttime. Note the larger, varying offset between the two instances Path 3 was driven. This was intentional to create more challenging baseline conditions.

in this section is the difference between our estimated path-tracking offset from localization compared to the ground truth path-tracking error. We show the lateral error in the vehicle frame as this is most important for maintaining accurate path-tracking. Table 5.2 summarizes the results across all experiments and paths.

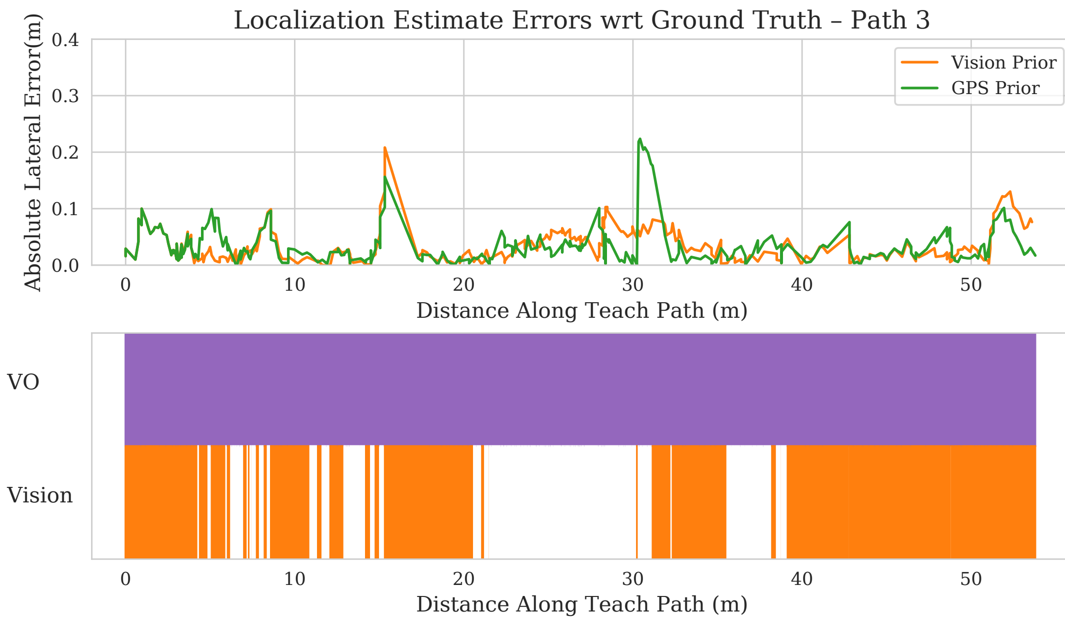

Figure 5.9: Experiment 1 results for Path 3. No sensor dropout was simulated in this experiment though some naturally occurring localization failures were seen in the nighttime dataset (Path 3).

In Experiment 1, both the VO and the GPS odometry methods provided similar performance. In good conditions (e.g., Path 1 collected during daylight), the localization error was almost identical. This was expected as with many visual features available for localization, the relative weight of the prior is reduced. Therefore, under conditions where VT&R is already highly successful, our addition does not degrade that performance. For Path 3, the manual driving and nighttime conditions created some natural localization failures, as seen in the bottom subplot of Figure 5.9. However, both methods were able to bound errors to a reasonable level for path-tracking.

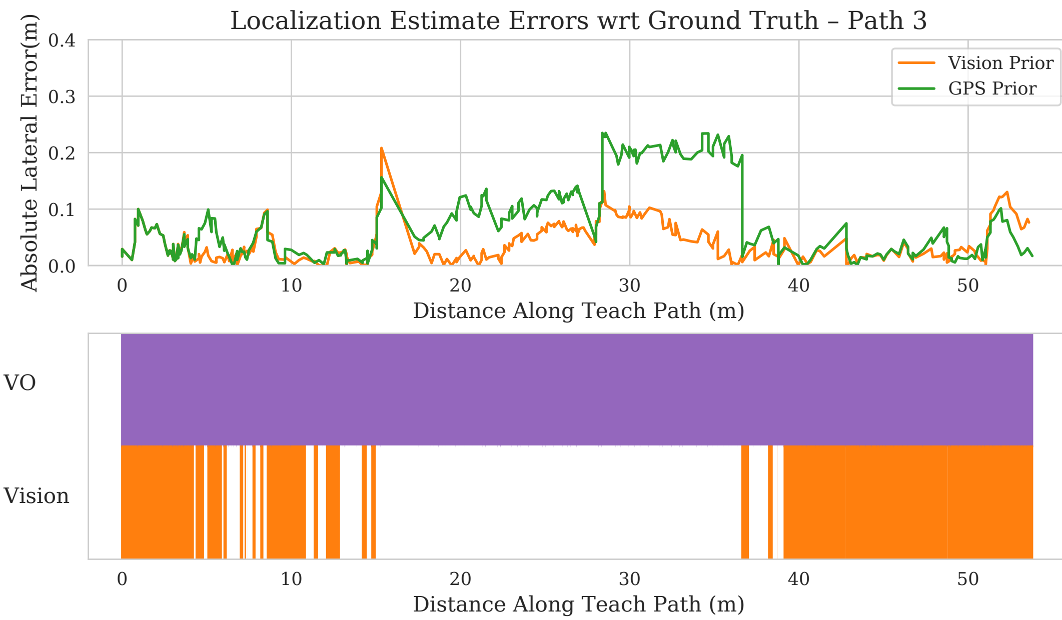

Figure 5.10: Experiment 2 results for Path 3. Both algorithms were prevented from using visual localization for a 20-second section midway through.

In Experiment 2, we simulate a 20-second section where visual localization is not possible, so the algorithm must rely heavily on its prior. For both methods and on all paths, the localization error grows larger in the section in which we prevent visual localization. For two of the three paths, the mean error with the GPS prior is higher than the VO prior but, they are of similar magnitudes for all paths. Both algorithms do well enough that when the robot is allowed to localize again, it still knows where it is along the path and the localization error quickly drops.

In Experiment 3, summarized in Figure 5.11, the vision-only pipeline is less successful. We again create a 20-second vision dropout, but this time the left camera images are replaced by black images to trigger VO failures. When using the GPS prior, we do not rely on VO while GPS odometry is available. As a result, errors are similar in magnitude to Experiment 2. Without GPS, the robot can rely on its motion model for a short period but quickly becomes lost without the proper means for

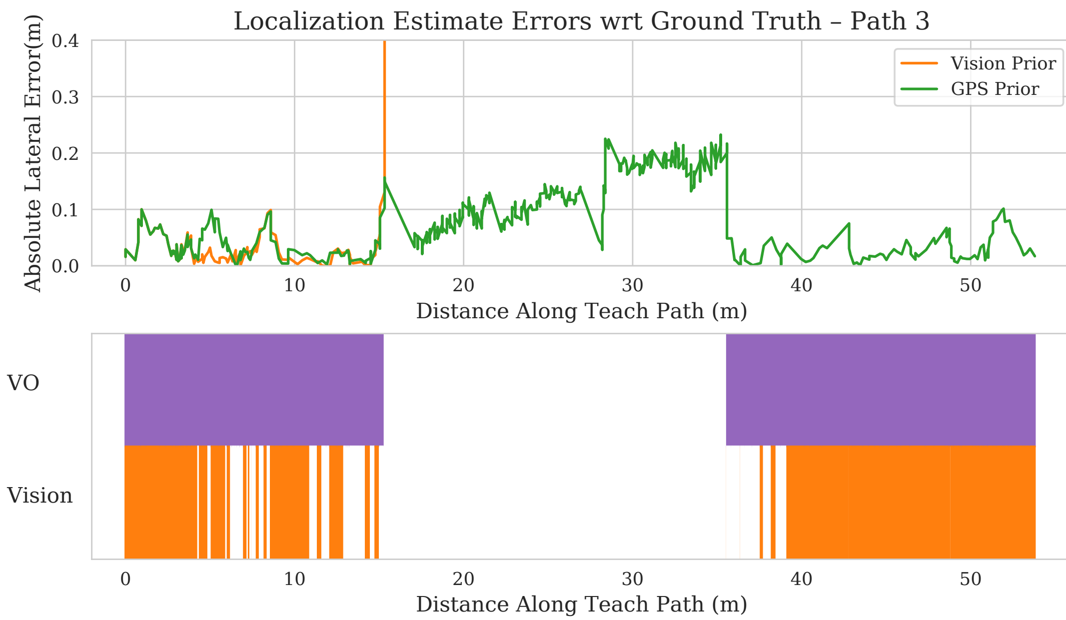

Figure 5.11: Experiment 3 results for Path 3. A 20-second total camera failure was simulated beginning approximately 15m along the teach path.

Path	Experiment	Mean Lateral Localization Error (m)	
		VO Prior	GPS Prior
1	1	**0.022**	**0.022**
	2	**0.031**	0.054
	3	6.767	**0.077**
2	1	0.081	**0.020**
	2	0.104	**0.030**
	3	3.704	**0.036**
3	1	0.036	**0.035**
	2	**0.042**	0.079
	3	1.525	**0.092**

Table 5.2: Summary of results for all three experiments across all three paths. Our GPS prior and the VO prior have similar performance in Experiments 1 and 2. Experiment 3 simulated a camera failure. Here the addition of GPS odometry made a large impact and would have been the difference between a successful run and an unrecoverable failure.

dead-reckoning. Even when VO recovers, the robot is unsure of where it is in relation to the path and is not able to relocalize. With GPS, the robot quickly recovers and accurately localizes when the camera images are restored.

5.4 Summary

In this chapter, we built on the contributions of previous chapters to demonstrate how GPS could benefit a successful relative navigation system. We briefly introduced the VTR3 project which was developed concurrently and had high-level goals that were important to keep in mind while adding GPS. At its completion, this project will provide an open-sourced, state-of-the-art navigation system to the robotics community. We mentioned several minor uses for GPS in VTR3 such as for determining the robot's start position in the map. These will make operating VT&R easier for future users.

We then described our Carrier Phase Odometry package that is the basis for using GPS as a backup or alternative for VO in VTR3. As this is run in a separate node, the details are abstracted from the main stereo camera pipeline allowing our changes to easily be used in future varieties of VT&R. Beyond VT&R, CPO can be run as a standalone GPS odometry package and is available open-source. To the best of our knowledge, this is the only widely available TDCP software package and one of the few open-source navigation projects developed in ROS2. The project provides accurate odometry with final drift rates averaging just 0.27% in our experiments.

When used in VT&R, GPS odometry provides an accurate prior allowing localization to achieve similar performance as with VO in good conditions and preventing total localization failures in challenging scenarios. Like the results of Chapter 4, this contribution uses GPS to improve the robustness of our system when vision struggles. However, by focussing on the odometry side, we can achieve this with only a single receiver.

Chapter 6

Conclusion

6.1 Summary of Contributions

In this work, we investigated several ways GPS could be used across a relative navigation pipeline on a mobile robot. In Chapter 3, we saw how relaxing the constraint on absolute accuracy allowed us to cancel errors affecting the GPS signal through TDCP. This generated several novel contributions including a practical TDCP odometry algorithm incorporating a motion model and robust cost functions, and the first direct comparison of TDCP-based odometry and visual odometry. We also showed that combining the two sensors into a single odometry algorithm could deliver the accuracy of our single-receiver GPS odometry with the reliability of VO when GPS availability may not be guaranteed.

In Chapter 4, we combined GPS with vision for relative localization. Unlike related methods, we did not attempt to reconcile measurements from vision and GPS into a single global coordinate frame. By delaying sensor fusion we were able to improve the robustness of VT&R with an absolute sensor without compromising the advantages of our relative map framework When tested on a real robot via 3.5km of autonomous driving, the system maintained high path-tracking accuracy and transitioned smoothly between sensors despite challenging conditions.

Finally, we showed how GPS odometry could be used in VTR3 as an alternative to VO providing a prior for localization. The addition provided an increase in reliability to an already highly successful relative navigation system. As a byproduct, we released the open-source Carrier Phase Odometry available at `https://github.com/utiasASRL/cpo`. It may be used as a comprehensive odometry solution using only a single GPS receiver.

6.2 Future Work

6.2.1 Extensions to GPS Odometry

One interesting observation from the results of Chapter 3 was that the position errors with respect to ground truth vary more linearly than we would expect in a random walk, as odometry is often modelled. We hypothesize this is due to the uncorrected error sources in the carrier phase double difference equation (Equation (3.3)) varying smoothly. We might expect these error sources to be autocorrelated given they are a function of satellites moving at near-constant velocity sending signals through an atmosphere that changes gradually. An interesting extension might be to use other sensors or a motion model to estimate and correct for this linear bias.

On the Carrier Phase Odometry project itself, our biggest goal is to have it tested and used on more receivers and by more roboticists. There are also a few features that could be added to improve it such as correcting for ionospheric effects and adding support for more GNSS constellations.

6.2.2 Benefits of GPS for Multi-Experience Localization

Most of our work assumes we are performing single-experience localization; for VT&R this means the repeat only uses data from the teach path. But VT&R has shown MEL to be quite useful and there is no reason GPS could not be used here too. For instance, our receiver may not have GPS during the original manually driven teach path but may have it on a later repeat and could log those measurements for localization. Currently, the Relatively Lazy algorithm only uses GPS positions from the single teach run but it could use multiple experiences worth of GPS measurements to reduce some of the measurement noise. Finally, with multiple experiences, GPS can actually improve the future success of visual localization. Typically when visual localization struggles, it is due to appearance change in the environment. By relying on GPS odometry or relative GPS localization to get through these regions on the current repeat, we are able to autonomously log images of the changed appearance that may be used during future repeats.

Appendix A

Interpolating Relative Covariances

The method developed in Chapter 5 to incorporate GPS into VT&R relied on sampling the GPS odometry trajectory at asynchronous keyframe times. The trajectory implementation uses the continuous-time estimation tools from Anderson [62] based on Gaussian process regression. As such, it is straightforward to query the trajectory for a pose estimate, $\mathbf{T}_{a0}$ at time t_a with respect to the start of the trajectory, t_0. With two query times, we can generate a relative pose estimate as:

$$\mathbf{T}_{ba} = \mathbf{T}_{b0}\mathbf{T}_{a0}^{-1}. \tag{A.1}$$

Anderson [62] outlines how one may interpolate the covariance on both $\mathbf{T}_{a0}$ and $\mathbf{T}_{b0}$ individually. To find the covariance on $\mathbf{T}_{ba}$, an additional term, the cross-covariance between $\mathbf{T}_{a0}$ and $\mathbf{T}_{b0}$, must be considered. If t_a and t_b are between the same two knots of the trajectory (i.e. they lie on the same piece of our piecewise, locally-linear GP prior) we could likely extend the method for interpolating at one query time to handle two query times. However, this is typically not the case; for our problem there is usually at least one GPS measurement logged between keyframes.

Instead, we take a more pragmatic approach to the problem by setting up a second, smaller optimization problem with our query times as new states and include the original states that bound them. Figure A.1 shows the six possible cases we might have depending on the position of the query times with respect to the knots in the original trajectory.

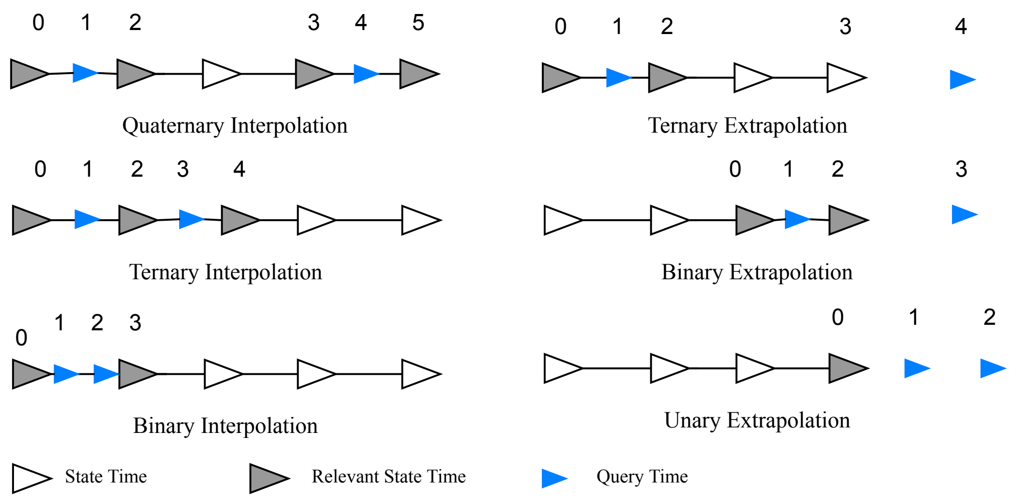

Figure A.1: Illustration of the six cases that may occur while interpolating/extrapolating a relative pose and covariance at two asynchronous query times. The larger black triangles represent the original trajectory while the blue triangles represent the two times our relative pose spans. Only the filled black triangles are needed to estimate the covariance on our relative pose.

This new problem contains two types of factors, as seen in Figure A.2. The first utilizes the posterior, $\hat{\mathbf{P}}$, from the original problem applied as a pseudomeasurement. It is marginalized such that it only contains the posterior blocks corresponding to states kept in the new problem. For this, we developed a new STEAM error evaluator of variable size such that it could be re-used for each of the six cases. The other factors are WNOA smoothing terms used to connect the new states to the original ones.

After optimizing this problem, we extract the posterior like we did for the original problem but now marginalize it for the covariance be-

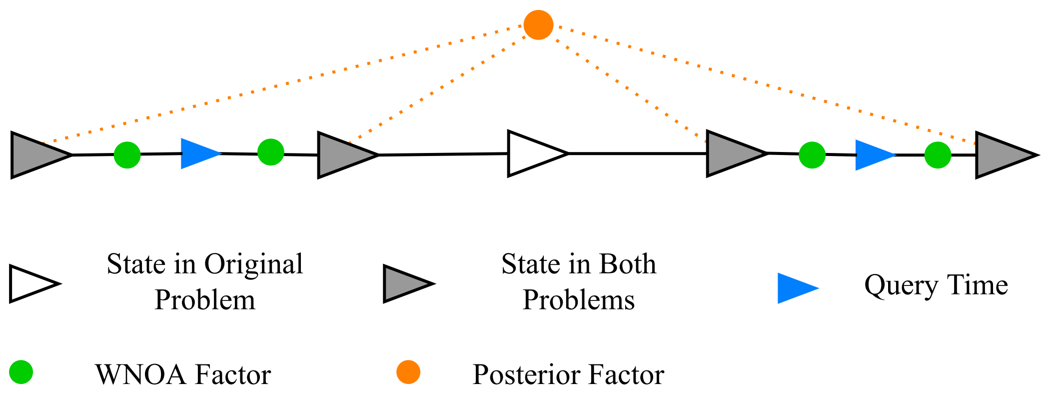

Figure A.2: Factor graph of the small optimization problem used to obtain the covariance on an asynchronous relative pose. The quaternary interpolation case is shown. Five factors are included in the least-squares optimization problem – four from the motion model and one from the original problem's posterior covariance matrix.

tween $\mathbf{T}_{a0}$ and $\mathbf{T}_{b0}$:

$$\hat{\mathbf{P}}_{a0,b0} = \begin{bmatrix} \hat{\mathbf{P}}(t_a, t_a) & \hat{\mathbf{P}}(t_a, t_b) \\ \hat{\mathbf{P}}(t_b, t_a) & \hat{\mathbf{P}}(t_b, t_b) \end{bmatrix}. \tag{A.2}$$

The matrix in Equation A.2 now contains the cross-covariance block needed to calculate the covariance on $\mathbf{T}_{ba}$:

$$\hat{\mathbf{P}}_{ba} = \hat{\mathbf{P}}(t_b, t_b) + \boldsymbol{\mathcal{T}}_{ba}\hat{\mathbf{P}}(t_a, t_a)\boldsymbol{\mathcal{T}}_{ba}^T - \boldsymbol{\mathcal{T}}_{ba}\hat{\mathbf{P}}(t_a, t_b) - (\boldsymbol{\mathcal{T}}_{ba}\hat{\mathbf{P}}(t_a, t_b))^T. \tag{A.3}$$

As the new problem consists of at most six states and, for our problem, the query times are relatively close together, the problem can be solved efficiently. Our implementation has an average execution time of 1 millisecond.

Bibliography

[1] G. Seeber, *Satellite geodesy: foundations, methods, and applications*. Walter de gruyter, 2008.

[2] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.

[3] K. Ulmer, P. Hwang, B. Disselkoen, and M. Wagner, "Accurate azimuth from a single plgr+ gls dod gps receiver using time relative positioning," in *Proceedings of the 8th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 1995)*, pp. 1733–1741, 1995.

[4] S. Michaud and R. Santerre, "Time-relative positioning with a single civil gps receiver," *GPS Solutions*, vol. 5, no. 2, pp. 71–77, 2001.

[5] N. Balard, R. Santerre, M. Cocard, and S. Bourgon, "Single gps receiver time-relative positioning with loop misclosure corrections," *GPS Solutions*, vol. 10, no. 1, pp. 56–62, 2006.

[6] W. Travis, *Path Duplication Using GPS Carrier Based Relative Position for Automated Ground Vehicle Convoys*. PhD thesis, 2010.

[7] D. Pierce, S. Martin, and D. M. Bevly, "Opportunistic landmark registration for long distance relative path following," in *Pro-*

ceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017), pp. 2560–2573, 2017.

[8] J. Traugott, *Precise flight trajectory reconstruction based on time-differential GNSS carrier phase processing.* PhD thesis, Technische Universität München, 2011.

[9] B. K. Soon, S. Scheding, H.-K. Lee, H.-K. Lee, and H. Durrant-Whyte, "An approach to aid ins using time-differenced gps carrier phase (tdcp) measurements," *Gps Solutions*, vol. 12, no. 4, pp. 261–271, 2008.

[10] Y. Zhao, "Applying time-differenced carrier phase in nondifferential gps/imu tightly coupled navigation systems to improve the positioning performance," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 992–1003, 2016.

[11] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover.," tech. rep., Stanford Univ CA Dept of Computer Science, 1980.

[12] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[13] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3946–3952, IEEE, 2008.

[14] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2043–2050, IEEE, 2017.

[15] F. Xue, X. Wang, S. Li, Q. Wang, J. Wang, and H. Zha, "Beyond tracking: Selecting memory and refining poses for deep visual odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8575–8583, 2019.

[16] Y. Yu, W. Gao, C. Liu, S. Shen, and M. Liu, "A gps-aided omnidirectional visual-inertial state estimator in ubiquitous environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7750–7755, IEEE, 2019.

[17] M. Schreiber, H. Königshof, A.-M. Hellmund, and C. Stiller, "Vehicle localization with tightly coupled gnss and visual odometry," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 858–863, IEEE, 2016.

[18] H. Kim, K. Choi, and I. Lee, "High accurate affordable car navigation using built-in sensory data and images acquired from a front view camera," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 808–813, IEEE, 2014.

[19] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *CoRR*, vol. abs/1901.03642, 2019.

[20] J.-H. Choi, Y.-W. Park, J.-B. Song, and I.-S. Kweon, "Localization using gps and vision aided ins with an image database and a network of a ground-based reference station in outdoor environments," *International Journal of Control, Automation and Systems*, vol. 9, no. 4, p. 716, 2011.

[21] Y. Li, Z. Hu, Y. Hu, and D. Chu, "Integration of vision and topological self-localization for intelligent vehicles," *Mechatronics*, vol. 51, pp. 46–58, 2018.

[22] K. Vishal, C. Jawahar, and V. Chari, "Accurate localization by fusing images and gps signals," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 17–24, 2015.

[23] O. Vysotska, T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Efficient and effective matching of image sequences under substantial appearance changes exploiting gps priors," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2774–2779, IEEE, 2015.

[24] Y. Shi, S. Ji, Z. Shi, Y. Duan, and R. Shibasaki, "Gps-supported visual SLAM with a rigorous sensor model for a panoramic camera in outdoor environments," *Sensors*, vol. 13, no. 1, pp. 119–136, 2013.

[25] X. Chen, W. Hu, L. Zhang, Z. Shi, and M. Li, "Integration of low-cost GNSS and monocular cameras for simultaneous localization and mapping," *Sensors*, vol. 18, no. 7, p. 2193, 2018.

[26] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[27] K. MacTavish, M. Paton, and T. D. Barfoot, "Selective memory: Recalling relevant experience for long-term visual localization," *Journal of Field Robotics*, vol. 35, pp. 1265–1292, 2018.

[28] D. Brown, "A solution to the general problem of multiple station analytical stereotriangulation," tech. rep., Patrick AirforceBase, Florida, 1958.

[29] H. F. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics Autom. Mag.*, vol. 13, no. 2, pp. 99–110, 2006.

[30] E. Olson, J. J. Leonard, and S. J. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA*, pp. 2262–2269, IEEE, 2006.

[31] K. S. Chong and L. Kleeman, "Feature-based mapping in real, large scale environments using an ultrasonic array," *Int. J. Robotics Res.*, vol. 18, no. 1, pp. 3–19, 1999.

[32] S. B. Williams, *Efficient Solutions to AutonomousMapping and Navigation Problems.* PhD thesis, The University of Sydney, 2001.

[33] J. A. Marshall, T. D. Barfoot, and J. Larsson, "Autonomous underground tramming for center-articulated vehicles," *J. Field Robotics*, vol. 25, no. 6-7, pp. 400–421, 2008.

[34] G. Sibley, C. Mei, I. D. Reid, and P. Newman, "Adaptive relative bundle adjustment," in *Robotics: Science and Systems V, University of Washington, Seattle, USA, June 28 - July 1, 2009* (J. Trinkle, Y. Matsuoka, and J. A. Castellanos, eds.), The MIT Press, 2009.

[35] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *Int. J. Robotics Res.*, vol. 32, no. 14, pp. 1645–1661, 2013.

[36] C. Linegar, W. Churchill, and P. Newman, "Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained

localisation," in *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pp. 90–97, IEEE, 2015.

[37] M. Paton, K. MacTavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1918–1925, Oct 2016.

[38] K. MacTavish, M. Paton, and T. D. Barfoot, "Visual triage: A bag-of-words experience selector for long-term visual route following," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2065–2072, IEEE, 2017.

[39] M. Paton, K. MacTavish, C. J. Ostafew, and T. D. Barfoot, "It's not easy seeing green: Lighting-resistant stereo visual teach amp; repeat using color-constant images," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1519–1526, May 2015.

[40] C. McManus, P. T. Furgale, B. Stenning, and T. D. Barfoot, "Lighting-invariant visual teach and repeat using appearance-based lidar," *J. Field Robotics*, vol. 30, no. 2, pp. 254–287, 2013.

[41] L. E. Clement, J. Kelly, and T. D. Barfoot, "Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery," *J. Field Robotics*, vol. 34, no. 1, pp. 74–97, 2017.

[42] M. Warren, M. Greeff, B. Patel, J. Collier, A. P. Schoellig, and T. D. Barfoot, "There's no place like home: Visual teach and re-

peat for emergency return of multirotor uavs during GPS failure," *IEEE Robotics Autom. Lett.*, vol. 4, no. 1, pp. 161–168, 2019.

[43] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," 2007.

[44] K. MacTavish, M. Paton, and T. D. Barfoot, "Night rider: Visual odometry using headlights," in *2017 14th Conference on Computer and Robot Vision (CRV)*, pp. 314–320, IEEE, 2017.

[45] B. Congram and T. D. Barfoot, "Experimental comparison of visual and single-receiver gps odometry," *arXiv preprint arXiv:2106.02122*, 2021.

[46] J. A. Klobuchar, "Ionospheric time-delay algorithm for single-frequency gps users," *IEEE Transactions on aerospace and electronic systems*, no. 3, pp. 325–331, 1987.

[47] A. Niell, "Global mapping functions for the atmosphere delay at radio wavelengths," *Journal of Geophysical Research: Solid Earth*, vol. 101, no. B2, pp. 3227–3246, 1996.

[48] P. Collins, R. Langley, and J. LaMance, "Limiting factors in tropospheric propagation delay error modelling for gps airborne navigation," *Proc. Inst. Navig. 52nd Ann. Meet*, vol. 3, 1996.

[49] T. Takasu and A. Yasuda, "Development of the low-cost rtk-gps receiver with an open source program package rtklib," in *International symposium on GPS/GNSS*, vol. 1, International Convention Center Jeju Korea, 2009.

[50] S. Anderson and T. D. Barfoot, "Full steam ahead: Exactly sparse gaussian process regression for batch continuous-time trajectory estimation on se(3)," 2015.

[51] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Comput. J.*, vol. 7, no. 2, pp. 155–162, 1964.

[52] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pp. 62–69, IEEE, 2013.

[53] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, IEEE, 2012.

[54] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[55] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[56] B. Congram and T. D. Barfoot, "Relatively lazy: Indoor-outdoor navigation using vision and gnss," in *2021 18th Conference on Robots and Vision (CRV)*, pp. 25–32, 2021.

[57] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[58] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating

system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[59] M. Gridseth and T. D. Barfoot, "Deepmel: Compiling visual multi-experience localization into a deep neural network," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1674–1681, IEEE, 2020.

[60] T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch continuous-time trajectory estimation as exactly sparse gaussian process regression.," in *Robotics: Science and Systems*, vol. 10, Citeseer, 2014.

[61] T. Suzuki, "Time-relative RTK-GNSS: GNSS loop closure in pose graph optimization," *IEEE Robotics Autom. Lett.*, vol. 5, no. 3, pp. 4735–4742, 2020.

[62] S. W. Anderson, *Batch continuous-time trajectory estimation.* PhD thesis, University of Toronto (Canada), 2017.