VISUAL-INERTIAL CURVE SLAM

BY

KEVIN C. MEIER

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Professor Seth Hutchinson, Chair and Co-Director
Associate Professor Soon-Jo Chung, Co-Director
Assistant Professor Alexander Schwing
Professor Minh Do

ABSTRACT

In this dissertation, we present a simultaneous localization and mapping (SLAM) algorithm that uses Bézier curves as static landmark primitives rather than feature points. Our approach allows us to estimate the full 6-DOF pose of a robot while providing a sparse structured map which can be used to assist a robot in motion planning and control. We demonstrate how to reconstruct the 3-D location of curve landmarks from a stereo pair and how to compare the 3-D shape of curve landmarks between chronologically sequential stereo frames to solve the data association problem. We also present a method to combine curve landmarks for mapping purposes, resulting in a map with a continuous set of curves that contain fewer landmark states than conventional point-based SLAM algorithms. We demonstrate our algorithm's effectiveness with numerous experiments, including comparisons to existing state-of-the-art SLAM algorithms.

A notable contribution of this dissertation is to apply our SLAM algorithm to a river setting to localize a canoe and create a sparse structured map of the border of a river. To accomplish this task, the dissertation presents a novel vision-based algorithm that identifies the boundary separating water from land in a river environment containing specular reflections. Our approach relies on the law of reflection. Assuming the surface of water behaves like a horizontal mirror, the border separating land from water corresponds to the border separating 3-D data which are either above or below the surface of water. We detect a river by identifying this border in a stereo camera. We start by demonstrating how to robustly estimate the normal and height of the water's surface with respect to a stereo camera. Then, we segment water from land by identifying the boundary separating dense 3-D stereo data which are either above or below the water's surface. With the border of water identified, we validate the proposed river boundary detection algorithm by applying it to a chronologically sequential video sequence obtained from

the visual-inertial canoe dataset. Additionally, we use our SLAM algorithm to create a sparse structured map of the shoreline of a river.

Dedicated to Summer, Taysom, and Kinzley

ACKNOWLEDGMENTS

No person has been more instrumental in supporting the completion of my degree than my wife, Summer Meier. I want to thank her for her relentless patience and undeviating confidence in me throughout this entire experience. Her constant encouragement, support, and technical acuity were tremendous assets as I worked on my Ph.D. I want to thank Seth Hutchinson and Soon-Jo Chung for the extensive personal and professional guidance they provided, they taught me a great deal about engineering research. Their patience and the many learning opportunities they provided were instrumental in completing this dissertation. I also want to thank Alex Schwing and Minh Do for reviewing my work and providing insightful feedback that improved the research in this dissertation.

I would also like to acknowledge the financial support provided by the SMART scholarship for service program (Office of Secretary Defense-Test and Evaluation, Defense-Wide/PE0601120D8Z National Defense Education Program (NDEP)/BA-1, Basic Research), and the Office of Naval Research (grant number N00014-14-1-0265).

I want to thank the many friends and collaborators I had the pleasure to work and associate with at the University of Illinois at Urbana-Champaign, and at my local church. I also want to thank all the members of my family. I would like to thank my parents for their support and encouragement; I cannot give them enough credit for the many sacrifices they made throughout my life which allowed me to pursue and complete this degree. I want to thank my two children, Taysom and Kinzley, for filling my life with happiness and joy as I worked on this degree.

Finally, I want to thank God and his son, Jesus Christ, for blessing me with the motivation, persistence, abilities, and opportunity to complete this dissertation.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Simultaneous localization and mapping (SLAM) refers to an algorithm that is used to create a map of a previously unknown environment while simultaneously estimating the location of a mobile platform within this map. At the forefront of the vision-based SLAM approach, feature points are usually selected to represent landmarks in the map. Although point-based approaches have produced accurate SLAM systems that run in real-time, point-based SLAM algorithms are subject to a number of drawbacks: Points ignore structural information between sampling points belonging to the same surface or edge, making it difficult for a robot to determine how it should interact with its surroundings. Creating dense maps with feature points requires a large state space. Many map points do not represent anything physically significant and are not needed because they belong to a structured object that can be represented compactly using parameterized shapes. In settings that lack distinguishable feature points, point-based detector/descriptor algorithms will fail to track enough feature points for a robot to accurately estimate its pose. Settings that contain a river pose a particular challenge as depth information from water reflections will likely approximate the depth of the object flipped about a symmetric plane, and not the true depth to the water's surface [1]. Thus, blindly applying a point-based algorithm to a river setting will result in an erroneous map.

1.1 Problem Statement and Motivation

In this dissertation, we overcome these shortcomings by solving two main problems:

(i) We solve the visual-inertial SLAM problem using parameterized Bézier curves as landmark primitives as opposed to feature points.

(ii) We present a novel vision-based algorithm that identifies the border of a river containing specular reflections in a stereo pair.

Our aim is to create sparse structured maps of unique objects that are located in the environment, e.g., road lanes, sidewalks, a road, or the shoreline of a river. We intend to exploit the structured nature of these settings, allowing our SLAM algorithm to operate in settings that lack distinguishable feature points or where reflections inhibit the map making process. A few example settings demonstrating the motivation for our approach are shown in Figure 1.1. By using parameterized Bézier curves as landmark primitives in these settings, we are able to construct a map that recovers the dimensions, shape, and outline of a sidewalk or yellow road lanes. Furthermore, the constructed map will be compactly represented with a small number of curve landmarks. In fact, in our experimental evaluations, we have observed that Curve SLAM reduces the required number of landmark primitives by several orders of magnitude relative to state-of-the-art point-based methods. Additionally, because a number of images in Figure 1.1 contain few distinguishable feature points, at times, our attempt to apply the stereo parallel tracking and mapping algorithm [2] to these settings failed because we were unable to match feature points across several chronologically successive image views. Furthermore, even when feature points are available and used in these types of settings, the dimensions, shape, and outline of the sidewalk and yellow road lanes will be ignored.

In addition to Curve SLAM, this dissertation presents a novel vision-based algorithm to detect the border of water in a river setting containing specular reflections, allowing us to use Curve SLAM to create a sparse structured map of the shoreline of a river. Identifying the border of water is vital to a robot mapping a river setting; otherwise, as noted above, blindly applying a vision-based SLAM algorithm will create an erroneous map because depth information from water-reflected feature points will approximate the depth of a feature point flipped about a symmetric plane, and not the true depth to the water's surface [1]. In fact, recognizing the border of water is vital for purposes other than creating a map, e.g., for an autonomous surface vehicle (ASV) to plan a collision-free course as it navigates a river, the ASV must identify where water and land are located to avoid colliding with land.

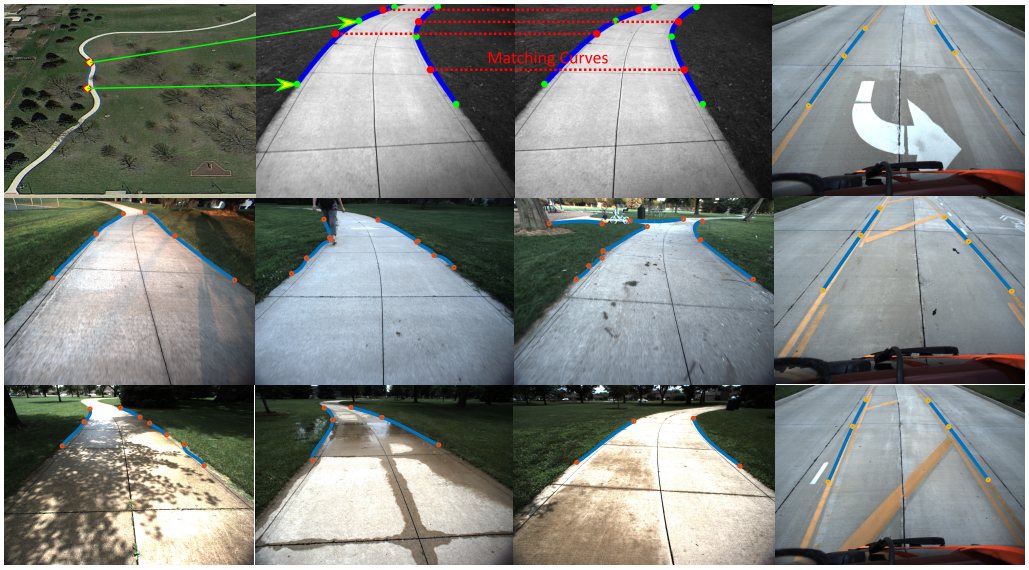


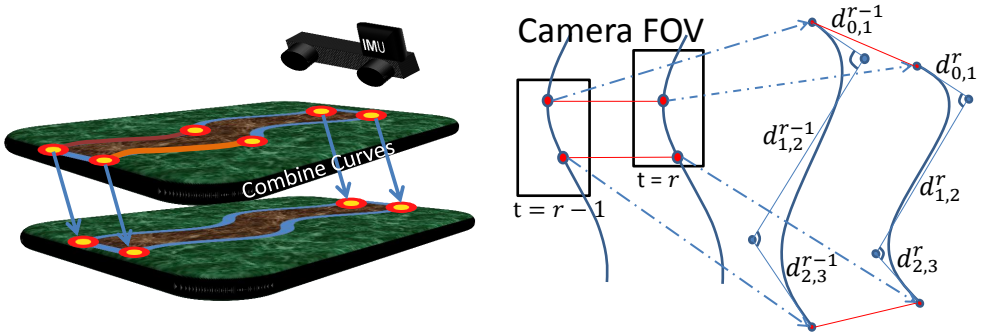
Figure 1.1: The images demonstrate the proposed Curve SLAM algorithm applied to various settings under different environmental conditions. Curve SLAM relies on a stereo camera and IMU to solve the SLAM problem in a previously unknown environment using parameterized curves as landmarks. The images show curve landmarks interpolated to yellow road lanes and the outline of a sidewalk.

1.2 Contributions

In this dissertation, we develop, extend, and combine methods in computer vision and computer-aided geometric design to solve the SLAM problem. We exploit the structured nature of many different types of environments by using Bézier curves to represent the edges of a path. Then, with a stereo camera and IMU, we reconstruct the 3-D location of these curves while simultaneously estimating the 6-DOF pose of a robot. The specific contributions and benefits of the proposed approach are summarized as follows:

- We present an algorithm that interpolates, splits, and matches curves in a stereo pair. This allows us to reconstruct the 3-D location of curves, parameterized by a small number of control points.

- We present a data association algorithm that compares the physical dimensions of curve landmarks between chronologically sequential stereo image frames to remove curve outliers, see Figure 1.2. When the dimensions of a curve do not match between sequential image frames, the curve is pruned. The algorithm relies on heuristics and mild assump-



(a) Combining Curves. (b) Comparing Curves.

Figure 1.2: Features of the proposed Curve SLAM algorithm. Figure 1.2a shows the proposed algorithm provides a method that combines curves to reduce the data representing the map. Figure 1.2a demonstrates a before (top layer) and after (bottom layer) effect of our curve combining algorithm. Curve SLAM also provides a method to compare the physical dimensions of curve landmarks (see Figure 1.2b), allowing Curve SLAM to operate in settings that lack distinguishable feature points. Figure 1.2a is further explained in Section 3.3.8. Figure 1.2b is further explained in Section 3.3.1.

tions, but is designed to find false associations when a small number of landmark curves (usually fewer than four) are consistently tracked between image frames. Tracking such a small number of landmarks allows our algorithm to operate in settings that lack distinguishable feature points, and reduces the state-space size of our SLAM estimator. Our approach to the data association problem is quite different from point-based algorithms that track hundreds of feature points between image frames.

- We present a curve combining algorithm that reduces the number of curve landmarks representing the map, see Figure 1.2. Bézier curves are useful in this process because they can be represented compactly by the location of parameterized control points, allowing us to construct large maps with fewer landmark states than conventional point-based SLAM algorithms. Additionally, curves are used to represent the outline, shape, and dimensions of unique objects in the environment. This provides structure and information which can aid a robot in path planning and control.

- We present an algorithm that segments water in a river environment containing specular reflections. Our algorithm is intended to run online and requires no offline training.

- We show how to formulate and solve an optimization problem that allows us to robustly estimate the normal and height of the water's surface, as well as the 3-D location of symmetric feature points.

- We demonstrate how the normal and height of the water's surface can be used to obtain a precise measurement of roll, pitch, and height at every stereo frame, and we show how to fuse these measurements directly into our SLAM estimator to enhance the localization and mapping accuracy of an autonomous robot.

- We present an algorithm to find, match, and remove symmetric feature point outliers in a stereo camera that contains images of a river setting.

- We present an algorithm to find the boundary separating 3-D data which are either above or below the surface of water. This boundary corresponds to the border separating land from water in the image plane, allowing us to extract the border of water from a stereo pair.

- We validate Curve SLAM and our river detection algorithm with experimental hardware in real-world settings. We apply Curve SLAM to multiple outdoor environments, and we rigorously compare Curve SLAM against the open keyframe-based visual-inertial SLAM (OKVIS) algorithm [3] and a stereo version of the parallel tracking and mapping (SPTAM) algorithm [2, 4].

1.3 Outline of the Dissertation

The remainder of this chapter presents related work. In Chapter 2, we present the proposed river border detection algorithm. Chapter 3 presents the Curve SLAM algorithm. Notice that we present the river detection algorithm before Curve SLAM because Curve SLAM depends on an object recognition algorithm (see Figure 1.3); the river detection algorithm is one such object recognition algorithm that is used by Curve SLAM in this dissertation.

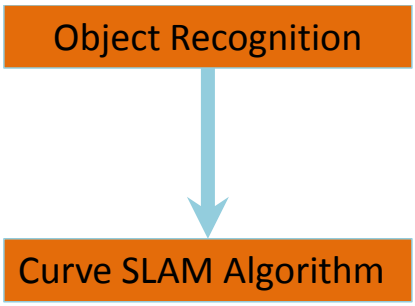


Figure 1.3: A block diagram demonstrating the dependence that Curve SLAM has on an object recognition algorithm. The proposed river detection algorithm is one such object recognition algorithm that is used by Curve SLAM in this dissertation.

Chapter 4 presents experimental results and compares Curve SLAM against SPTAM and OKVIS in an environment that contains a sidewalk, road lanes, the outline of a road, or a river. Chapter 5 presents a conclusion to this dissertation along with suggestions for future work.

1.4 Related Work

The related work is divided into two topics: water detection algorithms, and SLAM algorithms, particularly SLAM solutions that represent the environment using high-level structure. A subsection on each of these topics is included in this section.

1.4.1 SLAM

Various SLAM algorithms have incorporated high-level features in order to overcome drawbacks associated with point-based SLAM. Examples of high-level features include planes [5–7], image moments [8], line segments [9–11], objects such as office chairs and tables [12], or a river [13, 14]. A desirable characteristic of high-level structure is that it provides a compact structured map of the environment. For instance, in [5] orthogonal planes are used to represent structured landmarks in a compact manner. The orthogonal planes represent objects such as walls, the ceiling, windows, and doors of an indoor office setting. In a similar fashion, the work in [10] and [9] use line segments to localize a camera, and to map an environment with a vision-

based sensor. Lines represent the structure of objects one would expect to find in the mapped location, e.g., a computer monitor, the structure of an indoor hallway, or the outline of a door. Lines also provide a sparse representation of the environment. For example, in [10], only two points, the start and end point of a line segment, are used to represent each line. The work in [11] presents a systems-level SLAM approach that uses line segments, ideal lines, vanishing points, and primary planes that are used in conjunction with feature points. In addition to providing a structured map, their algorithm demonstrates that high-level features can improve the localization accuracy of a SLAM algorithm when used in conjunction with sparse feature points, and they demonstrate that high-level features are able to operate in settings that lack distinguishable feature points. Later in this dissertation, we demonstrate that Curve SLAM shares this ability to operate in settings that lack conspicuous feature points.

By creating compact structured maps of the environment, the Curve SLAM algorithm incorporates some of the ideas represented in the previously mentioned papers on high-level structure, and the systems-level approach we take is similar in form to the recent publications [9, 11]. However, Curve SLAM differs from the previously mentioned algorithms on high-level structure because Curve SLAM is intended for settings that contain curved features, e.g., settings where a path or road is present. Applying Bézier curves as landmark primitives in these settings allows us to create maps that are more sparse than the high-level feature primitives previously mentioned.

The use of curves as vision primitives has been studied in the computer vision literature. Methods have been devised to match curves across multiple image views [15], not necessarily closely spaced or specifically in stereo images [16]. The work in [17] presents a method to reconstruct the 3-D location of non-uniform rational B-spline curves from a single stereo pair without matching point-based stereo correspondences. We incorporate the idea contained in this paper [17] in Section 3.2.

The algorithm in [18] uses B-spline curves as landmark primitives to localize a robot and create a structured map of an environment in a compact fashion. They use a single plane 2-D scanning laser sensor and an extended Kalman filter (EKF) to jointly estimate the 3-DOF pose of a planar ground robot and the location of each B-spline curve. They demonstrate how to add curves to their filter state and extend the length of curves in their filter state

by modifying the location of parameterized curve control points representing each B-spline curve. Our work differs from the work in [18] in a number of ways. The algorithms in this dissertation are designed around a vision sensor as its primary sensing input as opposed to a 2-D scanning laser. Thus most of the constituent algorithms in this dissertation, such as reconstructing 3-D curves or solving the data association step, require an entirely different approach. Additionally, our state is more general than the filter state in [18]; we include the full 6-DOF pose of a robot, making it applicable to various robotic platforms such as small UAVs. Furthermore, because one of our sensing inputs is a stereo camera, we are able to locate curves in settings where a laser sensor will fail, e.g., in one of our experiments, we use yellow road lanes as curve landmarks.

Various place recognition algorithms have been developed that can aid a SLAM system that is occasionally unable to track feature points. A recent and thorough review of the place recognition problem is given in [19]. Unfortunately, most approaches rely on feature points or likely require offline training. Additionally, feature points will likely fail when the appearance of the scene changes drastically, e.g., unexpected weather or changes in lighting conditions [20]. The work in [21] presents an algorithm that is invariant to lighting conditions, but relies on feature points and requires multiple stereo cameras. The work in [22,23] presents a place recognition algorithm that is invariant to seasonal and lighting changes, and does not require feature points. Their work uses mid-level image patches, and a support vector machine to train an image classifier offline. They demonstrate that their algorithm is invariant to extreme changes in the environment. Unfortunately, their work requires at least one pass of the environment and offline training.

Before proceeding, we emphasize that we do not believe that point-based approaches are necessarily inferior. Indeed, recent point-based SLAM approaches demonstrate remarkably accurate results that run in real-time [4, 24–31]; this is true both in estimating the motion of a vehicle and in creating maps of an environment. In fact, Curve SLAM can be modified rather easily to include feature points so that curves and feature points can be used simultaneously to solve the SLAM problem. However, we believe alternative approaches are required to overcome the aforementioned shortcomings inherent with feature points.

1.4.2 Water Detection

Previous approaches to detect water rely on locating the horizon and sky prior to detecting water [32, 33]. For instance, the approach in [33] locates water by finding image regions below the horizon that appear differently from above-the-horizon land and vegetation. The approach in [32] locates water by finding image regions below the horizon that have the same color as the sky. This latter approach assumes the color of below-the-horizon terrain is different from the color of the sky, and thus sky-colored pixels below the horizon originate from water reflections. Other approaches to detect water scan the image for cues that characterize water. For instance, the approach in [1] demonstrates that the depth (obtained from a stereo camera) to a reflected object is equal to the depth of the object flipped about a symmetric plane (see Figure 2.1). They detect water reflections by searching for image regions on the ground plane that have a dramatic change in depth. The approach in [34] locates water in a puddle by assuming the saturation-to-brightness ratio from the leading-edge to the trailing-edge of a puddle is uniform and distinct from other terrain. Unfortunately, these approaches are application specific as the algorithms in [32,34] are intended for wide-open terrain that is not cluttered with trees or other objects. Additionally, good water detection results depend on a camera-to-water distance of 35-50 meters for the work in [34], and at least seven meters for the work in [32]. Finally, the approach in [1] depends on thresholding an HSV image and a texture image of the scene.

The symmetric nature of reflections has been exploited to detect a river. These approaches rely on finding symmetric pairs [35,36]. A symmetric pair is a set of matching feature points that are symmetric about a plane. Since one of the feature points in a symmetric pair comes from a reflection, water can be detected by finding the water-reflected feature point in the symmetric pair [14,37]. In fact, the symmetric nature of reflections is beneficial for purposes other than detecting water. For instance, the approach in [14] exploits the multi-view geometry of symmetric pairs in order to enhance the observability of their SLAM estimator. Similarly, the algorithm in [38] demonstrates that depth (up to a scale factor) can be obtained from a single image with a symmetric pair. Furthermore, this algorithm [38] demonstrates that when the normal vector of the water's surface is orthogonal to the optical axis of the camera, all symmetric pairs lie on a corresponding scan line. They

demonstrate this by flipping a reflected object about a horizontal axis in the image plane so that the reflected object and its source are aligned and appear almost identical. While the above approaches on symmetry are useful for detecting water, they are not intended to detect the shoreline of a river.

The work in [39] detects water by deriving image descriptors designed to find specific properties of water, e.g., light waves or water ripples that enter and exit a local area at regular intervals and with small appearance variations. The work in [39] is an application-specific approach of a more general class of algorithms designed to find dynamic textures [40–44]. Dynamic textures are objects with visual patterns or statistical stationarity properties such as ripples or small waves in water, fire, smoke, or flags. However, these approaches are not suitable for an ASV as they require a video sequence captured from a static location. Furthermore, these approaches are not intended for a river containing specular reflections. Similar to the approaches that find dynamic texture, the algorithm in [45] relies on the appearance variation of water to detect a river, but their approach relies on feature points. They assume that since the appearance of feature points on the water's surface changes frequently, they can detect water by identifying feature points that are difficult to track, have non-rigid dynamics, or they identify image regions that have a low density of feature points over time. However, their approach is neither intended to map the border of a river nor is it suitable for a river containing specular reflections.

To detect the shoreline, the approach in [33] trains and tests a self-supervised binary support vector machine (SVM) to classify small image patches as either river or land. This algorithm [33] was later used in conjunction with a 3-D lidar sensor to map river environments, and detect obstacles and the shoreline using a quadrotor [13, 37]. However, the approach in [13], which is an extension of the approach in [37], only applies the shoreline detector in [33] at long ranges (greater than 15 meters) because the shoreline detector is not accurate at close range [13]. In fact, even at long range the shoreline detector's precision rate is only roughly 80% [13]. Instead, they have to rely on a 3-D lidar sensor to detect water. The machine learning approach in [46] estimates the collision-free space of a river environment containing boats, and presents experimental results in which their algorithm correctly detects the shoreline. But, their algorithm requires offline training, big-data with labeled ground-truth, and was only validated on a video sequence of

100 images.

Finally, we mention various other approaches that have been implemented to detect water, measure water resources, measure greenhouse gas emissions, perform flood disaster mitigation, or check the depth of water at buoys [47–51]. These approaches are not useful for our application because they rely on GPS, an expensive near-IR camera, a high-altitude UAV, or do not consider how to detect the shoreline.

CHAPTER 2

DETECTING THE BOUNDARY OF A RIVER WITH A STEREO CAMERA

In this chapter, we present a vision-based algorithm to detect the border of a river containing specular reflections. In Chapter 3, this algorithm is used by Curve SLAM to create a sparse structured map of the shoreline of a river. Our approach to detect water relies on the law of reflection. The key observation that allows our algorithm to operate successfully is that 3-D data located below the surface of water represents reflections not physically located in the world frame, while 3-D data above the surface of water represents land; these data are physically present in the 3-D world frame. We identify water by identifying the boundary separating data that are above or below the surface of water. With this observation in mind, we break the proposed water detection algorithm into two steps: first, we estimate the normal and height of the surface of water with respect to the camera frame. Second, we identify the boundary separating dense 3-D stereo data which are either above or below the surface of water.

The remainder of this chapter is organized as follows: Section 2.1 presents an outline and assumptions of the water detection algorithm. Section 2.2 describes how to estimate the normal and height of the surface of water. Additionally, Section 2.2 explains how to extract a precise measurement of roll, pitch, and height from the normal and height estimate. Section 2.3 explains how to use the normal and height to segment water from land. In Section 2.4, we validate the proposed river detection algorithm by applying it to a real-world setting containing a river. Section 2.5 describes cases where the river detection algorithm fails to find the river. Section 2.6 provides a conclusion to this chapter.

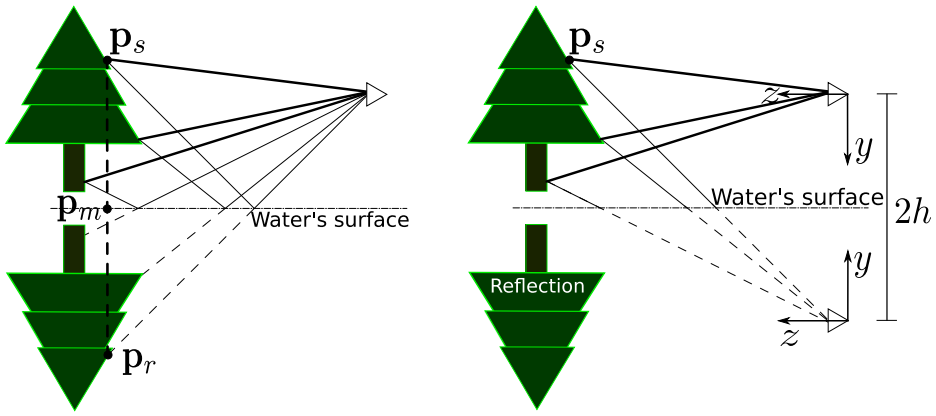
2.1 General Outline and Assumptions

We segment water from land by exploiting a number of cues that characterize a river setting. Throughout this chapter, we assume the river contains specular reflections with distinguishable symmetric feature points that can be extracted and matched with a standard point-based algorithm, e.g., SIFT [52]. We also assume a calibrated, time-synchronized, stereo camera was used to collect images of the river setting. Our approach is to solve the two following sub-problems in order:

- Estimate the normal and height of the surface of water with respect to the 3-D camera frame.
- Segment water from land by identifying where land intersects with the water's surface.

2.2 Estimating the Normal and Height of the Surface of Water

In this section, our aim is to solve an optimization problem in which we jointly estimate the 3-D location of symmetric feature points with respect to the body frame of a camera, and the normal and height of the water's surface. In this process, the symmetric nature of the environment is of paramount importance. We start by demonstrating how symmetric feature points are functionally dependent on the normal and height of the water's surface. This functional dependence has a dual interpretation that allows us to transform a single view into two, allowing us to formulate and solve the aforementioned optimization problem with greater accuracy. Next, we use symmetry to robustly match symmetric feature points within a monocular view and between a stereo pair. Finally, with both single-view and stereo-view symmetric pairs, we demonstrate how to formulate and solve the aforementioned optimization problem.



(a) Ideal scene of a river containing specular reflections.

(b) Dual interpretation of the ideal scene in Figure 2.1a.

Figure 2.1: Dual interpretation of an ideal river scene containing specular reflections. As shown in Figure 2.1a, the river acts like a horizontal mirror orthogonal to the gravity vector. In this ideal configuration, the law of reflection requires that a point $\mathbf{p}_s$ and its reflection $\mathbf{p}_r$ are symmetric about the water's surface. As described by the mathematical developments in Sections 2.2.1 and 2.2.2, Figure 2.1b demonstrates that the symmetry of a river setting allows a single view to be treated as two.

2.2.1 Symmetric Geometry

To define the symmetric geometry, we exploit the fact that the river behaves like a horizontal mirror that is orthogonal to the gravity vector [32]. In this configuration, the law of reflection requires that a 3-D source point $\mathbf{p}_s^{\mathcal{W}}$ in the world frame and its reflection $\mathbf{p}_r^{\mathcal{W}}$ (the subscript s refers to the source point, the subscript r refers to reflected point, and the subscript $\mathcal{W}$ refers to the world frame) are symmetric about a symmetry plane. In this case, the symmetry plane is the water's surface (see Figure 2.1a), the points $\mathbf{p}_s^{\mathcal{W}}$ and $\mathbf{p}_r^{\mathcal{W}}$ are referred to as a symmetric pair, and the 3-D position of the midpoint $\mathbf{p}_m^{\mathcal{W}}$ between $\mathbf{p}_s^{\mathcal{W}}$ and $\mathbf{p}_r^{\mathcal{W}}$ is located on the water's surface [38]. The water's surface can be represented as a planar function with respect to the camera: $n_1x + n_2y + n_3z + h = 0$, where $\mathbf{n} = (n_1, n_2, n_3)^\top$ represents the normal vector of the plane (in this dissertation, it is assumed $\mathbf{n}$ is a unit vector), and h represents the height of the plane. Given the location of $\mathbf{p}_s^{\mathcal{W}}$ in homogeneous coordinates, the 3-D location of its reflection is given by

$$\mathbf{p}_r^{\mathcal{W}} = \mathbf{T}_{rs}\mathbf{p}_s^{\mathcal{W}} \tag{2.1}$$

where

$$\mathbf{T}_{rs} = \begin{pmatrix} \mathbf{I}_{3\times 3} - 2\mathbf{n}\mathbf{n}^{\top} & -2h\mathbf{n} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{2.2}$$

Assuming an intrinsically calibrated camera with camera matrix $\mathbf{K}$, the work in [36] demonstrates that the matrix $\mathbf{T}_{rs}$ can be absorbed into $\mathbf{K}$ to create a second camera, within the same view, that has a valid projection matrix given by $\mathbf{K}_r = \mathbf{K}\mathbf{T}_{rs}$ (see Figure 2.1b). In other words, given the point $\mathbf{p}_s^{\mathcal{W}}$, $\mathbf{p}_s^{\mathcal{W}}$ can be mapped to the same image plane with two different perspective transformations:

$$\mathbf{p}_s^o = \mathbf{K}\mathbf{p}_s^{\mathcal{W}} \tag{2.3}$$

$$\mathbf{p}_r^o = \mathbf{K}\mathbf{p}_r^{\mathcal{W}} = \mathbf{K}\mathbf{T}_{rs}\mathbf{p}_s^{\mathcal{W}} = \mathbf{K}_r\mathbf{p}_s^{\mathcal{W}} \tag{2.4}$$

In Section 2.2.6, the mathematical developments of Equations (2.1)-(2.4) are used to transform a standard two-view stereo triangulation into a four-view triangulation using a single two-view stereo frame.

2.2.2 Comparisons with Conventional Binocular Stereo

Assuming the optical axis of the camera is orthogonal to the normal of the water's surface. Additionally, assuming a conventional camera frame, i.e., the z-axis is parallel to the optical axis, the x-axis is directed out the right of the camera, and the y-axis is directed out the bottom of the camera (see Figure 2.1b), the matrix $\mathbf{T}_{rs}$ has the following form:

$$\mathbf{T}_{rs} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -2h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

In this ideal configuration, the matrix $\mathbf{T}_{rs}$ demonstrates that the newly created camera has a baseline of $2h$ with a y-axis that is flipped with respect to the original camera.

Furthermore, similar to conventional binocular stereo, the search space for finding a symmetric point correspondence is limited to a single scan line.

Indeed, in the 3-D world, lines joining symmetric pairs are parallel to one another, and the normal $\mathbf{n}$. In the image plane, these lines intersect at a vanishing point. Thus, given the vanishing point and a feature point $\mathbf{p}_s$ in the image plane, the symmetric counterpart of $\mathbf{p}_s$ must be located on the line joining the vanishing point and $\mathbf{p}_s$. From hereon, we refer to this constraint as the symmetric epipolar constraint. We apply the symmetric epipolar constraint in Section 2.2.4 to remove outliers when matching symmetric pairs within a single view. For more information regarding symmetric epipolar geometry, see the algorithms in [36, 38, 53].

2.2.3 Finding Symmetric and Stereo Symmetric Correspondences

We detect and match symmetric correspondences and conventional stereo correspondences with SIFT feature points. When solving the optimization problem in Section 2.2.6, we consider two types of symmetric correspondences: symmetric pairs, i.e., symmetric correspondences within a single view, and stereo symmetric correspondences, i.e., symmetric pairs matched between the stereo pair. Both types of symmetric correspondences are used to estimate the normal and height of the surface of water in Section 2.2.6. When matching SIFT feature points in the steps that follow, it is assumed that descriptors are matched in the conventional way: matches in which the ratio of distance between the closest and second closest match is greater than 0.8 are rejected. Additionally, a matching correspondence can only be different, in terms of distance, from a perfect match by less than 5 %. It is also important to note that we changed the SIFT contrast threshold from the default value of 0.04 to 0.004 to obtain the experimental results in Section 2.4. We match symmetric and stereo symmetric correspondences with the following scheme:

Step 1 Detect and extract SIFT feature points in the left and right image of the stereo pair.

Step 2 With the SIFT descriptors from Step 1, we match symmetric pairs within the left image $\{\mathbf{p}_s^L, \mathbf{p}_r^L\}$ and within the right image $\{\mathbf{p}_s^R, \mathbf{p}_r^R\}$ using the algorithm in [35]. Typical results of this step are demonstrated in Figure 2.2.

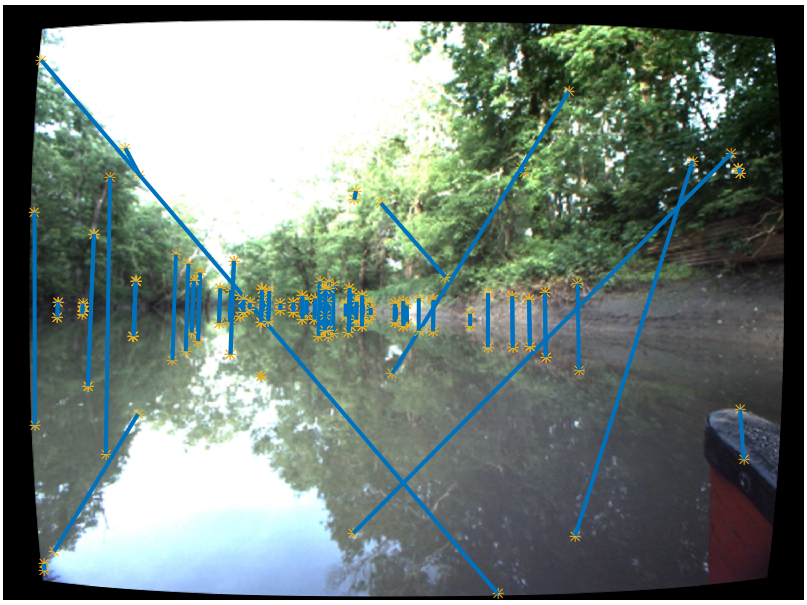


Figure 2.2: Symmetric pairs detected and matched in a monocular view using SIFT descriptors and the algorithm in [35].

Step 3 Locate the vanishing point in both images of the stereo pair, and remove monocular-view symmetric pair outliers with the method described in Section 2.2.4.

Step 4 For each symmetric pair $\{\mathbf{p}_s^L, \mathbf{p}_r^L\}$ in the left image obtained from Step 2, we find a stereo correspondence for $\{\mathbf{p}_s^L, \mathbf{p}_r^L\}$ in the right image using the SIFT descriptors from Step 1. If a correspondence $\{\mathbf{p}_1^R, \mathbf{p}_2^R\}$ is found for the pair $\{\mathbf{p}_s^L, \mathbf{p}_r^L\}$, we check if the pair $\{\mathbf{p}_1^R, \mathbf{p}_2^R\}$ satisfies the symmetric epipolar constraint within in the right image. If it does not, the match is removed as an outlier. If only the source or reflection of the symmetric pair $\{\mathbf{p}_s^L, \mathbf{p}_r^L\}$ is matched to a feature point in the right image, the final correspondence is located at the point in which the epipolar line from the symmetry constraint and the epipolar line from the conventional stereo scan line constraint intersect.

Step 5 Repeat Step 4, swapping the left image with the right image, and by replacing $\{\mathbf{p}_s^L, \mathbf{p}_r^L\}$ with each symmetric pair $\{\mathbf{p}_s^R, \mathbf{p}_r^R\}$ in the right image obtained from Step 2. Typical results of matching stereo symmetric pairs are demonstrated in Figure 2.3.

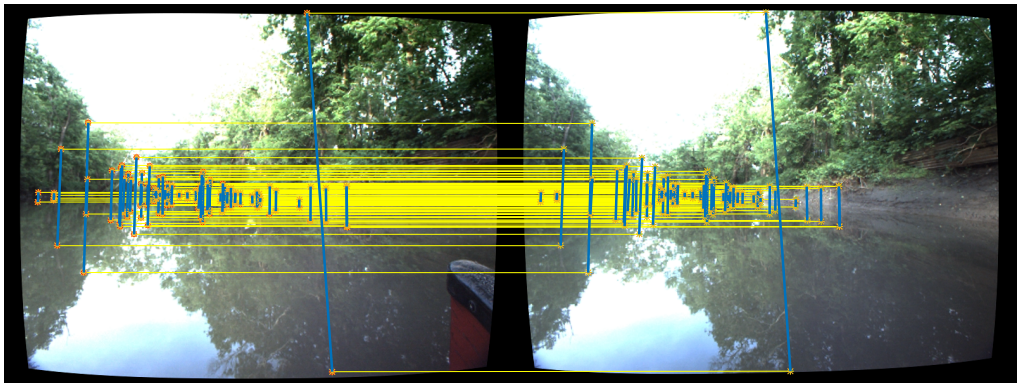


Figure 2.3: Stereo symmetric pairs matched between the left and right image of a conventional stereo pair.

To decrease computational complexity when solving the optimization problem in Section 2.2.6, we limit the number of symmetric pairs so that there is not more than n_s single-view or n_s stereo symmetric pairs that are approximately uniformly distributed across the image. To ensure this condition holds, we divide the image into n_r regions, and count the number of symmetric pairs in each region. Then, we divide these image regions into two sets: the set I_l represents the set of all image regions having less than n_s/n_r symmetric pairs per region, while the set I_m represents the set of all image regions having more than n_s/n_r symmetric pairs per region. We keep all the symmetric pairs in image regions belonging to I_l, and keep track the total number of symmetric pairs belonging to I_l with the variable n_l. We allow each of the image regions that belong to I_m to keep n_m/n_i randomly selected symmetric pairs, where n_i is the number of image regions in the set I_m, and $n_m = n_s - n_l$.

2.2.4 Identifying Symmetric Pair Outliers in a Monocular View with the Vanishing Point

In the 3-D world frame, lines joining symmetric pairs are parallel to each other, and the normal $\mathbf{n}$ of the water's surface. In the image plane, these lines intersect at a vanishing point. In this subsection, we apply RANSAC to determine which symmetric pairs do not intersect with the same vanishing point, allowing us to determine mismatched symmetric pairs. To apply RANSAC, we must supply RANSAC with two inputs. The first input is a

randomly selected hypothesis vanishing point using the minimum number of parameters required to estimate the vanishing point. In this case, the minimum number of parameters is two symmetric pairs. The second input is an error measure that is used to identify and remove outliers.

To calculate a hypothesis vanishing point, we randomly select two symmetric pairs $\{\mathbf{p}_s^{L,1}, \mathbf{p}_r^{L,1}\}$, and $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$, and calculate equations for the lines $\mathbf{l}_1$ and $\mathbf{l}_2$ that join $\{\mathbf{p}_s^{L,1}, \mathbf{p}_r^{L,1}\}$, and $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$, respectively (assuming homogeneous coordinates, $\mathbf{l}_i$ can be calculated by taking the cross product of the source point and reflection point in each symmetric pair, and normalizing appropriately). The hypothesis vanishing point is located at the intersection of $\mathbf{l}_1$ and $\mathbf{l}_2$, which can be calculated by taking the cross product of $\mathbf{l}_1$ and $\mathbf{l}_2$, and normalizing appropriately so that the location of the vanishing point is in homogeneous coordinates (see Figure 2.4).

With the method in [54], we obtain an error measure for the remaining symmetric pairs in two steps. First, we calculate the line $\mathbf{l}_m$ that joins the hypothesis vanishing point to the midpoint of the line $\mathbf{l}_j$ joining one of the remaining symmetric pairs (see Figure 2.4). Second, the error measure is calculated as the angle α subtending the lines $\mathbf{l}_m$ and $\mathbf{l}_j$. Any angle α that is greater than one degree is marked as an outlier. Results of our monocular-view outlier rejection algorithm are demonstrated in Figure 2.5. Before proceeding, it should be noted that when the optical axis is orthogonal to gravity, the vanishing point is located at infinity, and could potentially cause the outlier rejection algorithm described in this subsection from functioning properly. However, this condition almost never occurs because SIFT returns floating point values for the location of its feature points. Additionally, we checked for this condition and included it as a degenerate case when implementing RANSAC.

2.2.5 Identifying Stereo Symmetric Pair Outliers with the 3-D Location of Midpoints

To remove stereo symmetric outliers, we calculate the error measure defined in the previous paragraph between each stereo symmetric pair in the left or right image and its corresponding left-view/right-view vanishing point (these vanishing points are an output of the RANSAC vanishing point algorithm in

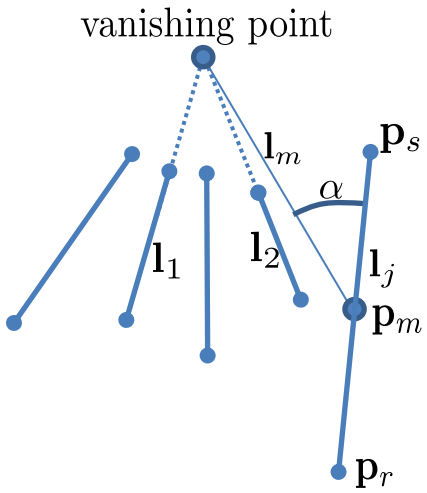


Figure 2.4: Definition of the variables that describe how to eliminate symmetric pair outliers in a monocular view. As described in Section 2.2.4, lines joining symmetric pairs should intersect at the same vanishing point. Using RANSAC, we identify and eliminate symmetric pair outliers that do not satisfy this constraint.

Section 2.2.4). Any error measure angle that is greater than one degree in either the left-view or right-view is marked as a stereo symmetric outlier.

As a second step to remove outliers, we measure the reprojection error of all stereo symmetric pairs using an estimate of $\mathbf{n}$ and h obtained from a small number of robustly matched stereo symmetric pairs. We obtain robustly matched stereo symmetric pairs by triangulating the 3-D location of all stereo symmetric pairs, calculating their corresponding 3-D midpoint coordinates, and fitting a plane to the 3-D midpoint coordinates using RANSAC and least-squares (the location of this plane provides a rough estimate of the 3-D location of the water's surface). Any midpoint whose distance is greater than 0.1 meters from the plane is considered an outlier. Next, we apply the optimization in Equation (2.6) to the inlier stereo symmetric pairs, which outputs an estimate of $\mathbf{n}$ and h. Then, with this newly acquired estimate of $\mathbf{n}$ and h, we use Equation (2.1) to reflect the 3-D position of all stereo symmetric source points about the water's surface, followed by a projection to the image plane. We remove any stereo symmetric pairs whose reprojection error is greater than four pixels. Results of our stereo outlier rejection algorithm are demonstrated in Figure 2.6. Additionally, the plane parameters (normal and plane height) that are calculated in this process will be used as an initial guess for the optimization problem described in Section 2.2.6.

Figure 2.5: Symmetric pairs detected and matched within a monocular view after identifying and removing outliers using RANSAC, and the constraint that, in the image plane, symmetric pairs intersect with the same vanishing point. These symmetric pairs are the same as those shown in Figure 2.2 after removing outliers.

2.2.6 Jointly Estimating the Parameters of the Water's Surface and the 3-D Location of Feature Points

With the feature points that are not outliers, we are ready to formulate and solve a nonlinear least-squares optimization problem to jointly estimate the normal and height of the water's surface, as well as the 3-D location of symmetric feature points in the 3-D world. Defining $\boldsymbol{\beta} \triangleq [\mathbf{n}\ h\ \mathbf{p}_s^{\mathcal{B},1} \dots \mathbf{p}_s^{\mathcal{B},n}]^\top$ as a parameter vector (here, the superscript $(\mathcal{B}, i)$ represents the 3-D body frame $\mathcal{B}$ of the i^{th} symmetric pair), we estimate $\boldsymbol{\beta}$ by minimizing the following equation:

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \sum_{\mathcal{O}\in\{L,R\}} \sum_{j\in\{s,r\}} \boldsymbol{1}_{\mathcal{O}}(\mathbf{y}^{i,\mathcal{O},j}) ||f(\mathbf{y}^{i,\mathcal{O},j} - \hat{\mathbf{y}}^{i,\mathcal{O},j}(\boldsymbol{\beta}))||_1^2 \qquad (2.6)$$

where i indicates the feature point, $\mathcal{O}$ indicates the left or right stereo image, and j indicates if the feature point is a source feature point or a reflection. The function $\hat{\mathbf{y}}^{\mathcal{O},j,i}(\cdot)$ represents a measurement predicted by the parameters in $\boldsymbol{\beta}$. Letting $\mathbf{M}^{\mathcal{O}}(\cdot)$ represent the function that maps a 3-D point from

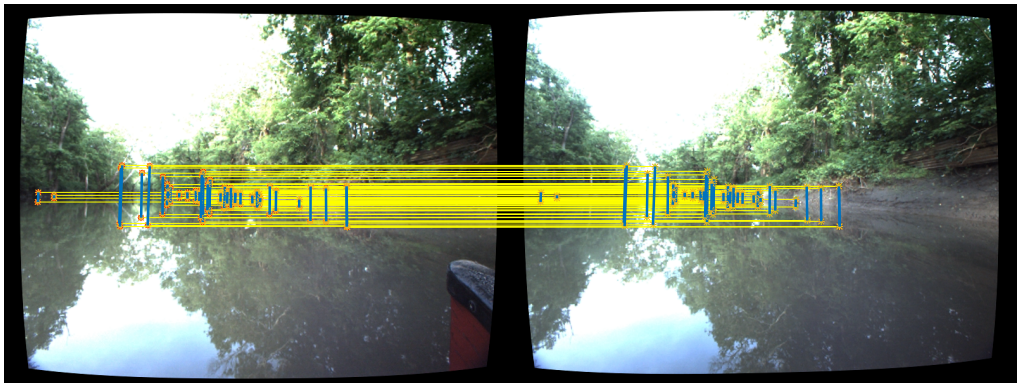


Figure 2.6: Stereo symmetric pairs matched between the stereo pair. These symmetric pairs are the same as those shown in Figure 2.3 after removing outliers.

the body frame to the image plane of the left or right camera, $\hat{\mathbf{y}}^{\mathcal{O},j,i}(\cdot)$ has one of four functional forms for the i^{th} feature point, these functional forms are defined in Table 2.1. The 2-D vector $\mathbf{y}^{i,\mathcal{O},j}$ represents measured image coordinates. Equations for $\mathbf{y}^{i,\mathcal{O},j}$ are defined in Table 2.1 for the i^{th} feature point. Notice that the measurement $\mathbf{y}^{i,\mathcal{O},j}$ corresponds to the projection of $\mathbf{p}_s^{\mathcal{B},i}$ onto the image plane with one of four camera matrices. The indicator function $\boldsymbol{1}_{\mathcal{O}}(\mathbf{y}^{i,\mathcal{O},j})$ indicates whether or not a symmetric pair is in the field of view of the left or right image, i.e., $\boldsymbol{1}_{\mathcal{O}}(\mathbf{y}^{i,\mathcal{O},j}) = 1$ (this symmetric pair is in the camera's field of view) or $\boldsymbol{1}_{\mathcal{O}}(\mathbf{y}^{i,\mathcal{O},j}) = 0$ (this symmetric pair is not in the camera's field of view). The function $f(\cdot)$ represents a robust loss function, and is applied element-wise. We assume $f(\cdot) = 2\arctan(\cdot)$. Additionally, it is important to stress that the l_1-norm is used in Equation (2.6), we found its performance superior to the l_2-norm, and the the l_1-norm allowed us to obtain the experimental results presented in Section 2.4.

Table 2.1: The Measurement and Predicted Measurement Equation for the i^{th} Feature Point

Predicted Measurement	Measurement
$\hat{\mathbf{y}}^{i,L,r}(\boldsymbol{\beta}) = \mathbf{M}^L(\mathbf{T}_{rs}\mathbf{p}_s^{\mathcal{B},i})$	$\mathbf{y}^{i,L,r} = \mathbf{p}_r^{L,i}$
$\hat{\mathbf{y}}^{i,L,s}(\boldsymbol{\beta}) = \mathbf{M}^L(\mathbf{p}_s^{\mathcal{B},i})$	$\mathbf{y}^{i,L,s} = \mathbf{p}_s^{L,i}$
$\hat{\mathbf{y}}^{i,R,r}(\boldsymbol{\beta}) = \mathbf{M}^R(\mathbf{T}_{rs}\mathbf{p}_s^{\mathcal{B},i})$	$\mathbf{y}^{i,R,r} = \mathbf{p}_r^{R,i}$
$\hat{\mathbf{y}}^{i,R,s}(\boldsymbol{\beta}) = \mathbf{M}^R(\mathbf{p}_s^{\mathcal{B},i})$	$\mathbf{y}^{i,R,s} = \mathbf{p}_s^{R,i}$

With the measurement and predicted measurement defined, we solve Equa-

tion (2.6) with the Levenberg-Marquardt algorithm [55–57]. An initial guess for the parameters in $\boldsymbol{\beta}$ is given in the following ways: An initial estimate for the normal, and water surface height is given in Section 2.2.5. An initial estimate for a feature point viewed in both images of the stereo pair is obtained by stereo triangulation. Additionally, we obtain an initial estimate of the 3-D location of the remaining symmetric pairs, i.e, symmetric pairs for which no conventional stereo correspondence was found, by triangulating their locations with the normal and height estimated in Section 2.2.5. The triangulation procedure for these remaining symmetric pairs is given by the predicted measurement equations in Table 2.1, where the camera matrices are obtained by the mathematical developments of Equations (2.1)-(2.4). It is important to mention that without the conventional stereo measurements, the triangulation given by Equations (2.3) and (2.4) can only be determined up to scale.

Once the optimization is complete, the output of Equation (2.6) provides a robust estimate of the normal and height of the water's surface.

2.2.7 Extensions to the SLAM Problem: Augmenting SLAM Measurements with the Normal and Height of the Water's Surface

Obtaining the normal and height of the water's surface is particularly useful because it allows us to obtain a measurement of roll, pitch, and height. Indeed, since the normal is approximately parallel to gravity, we are able to obtain the following mathematical expressions:

$$g\mathbf{n} = \mathbf{g}^{\mathcal{B}} = \boldsymbol{R}_{\mathcal{BW}}(\phi, \theta)\mathbf{g}^{\mathcal{W}} = \begin{pmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{pmatrix} \tag{2.7}$$

where g represents the gravitational constant, the vectors $\mathbf{g}^{\mathcal{W}}$, and $\mathbf{g}^{\mathcal{B}}$ represent gravitational acceleration in the world frame and body frame, respectively, and the matrix $\boldsymbol{R}_{\mathcal{BW}}$ is a rotation matrix that changes the representation of a point from the world frame to the body frame. Solving Equation

(2.7) for ϕ and θ, we obtain the following expressions:

$$\tan\phi = \frac{n_2}{n_3}$$

$$\tan\theta = \frac{-n_1}{\sqrt{n_2^2 + n_3)^2}}$$

In Section 2.4, We demonstrate that these measurements provide a drift-free solution for the roll, pitch, and height of an unmanned surface vehicle navigating a river setting, and can easily be implemented as a measurement in a filter-based SLAM algorithm, e.g., see [58, 59]. We include them as measurements in our SLAM estimator when we present experimental results of the Curve SLAM algorithm in Chapter 4.

2.3 Segmenting Water from Land

With the normal and height of the water's surface, we are ready to describe an algorithm that allows us to locate water in an image. To detect water, our algorithm relies on the following observation: 3-D objects triangulated below the water's surface are reflections, not physically located in the world, while 3-D objects triangulated above the water's surface are the source of these reflections. Furthermore, the boundary that separates water from land corresponds to the intersection in which dense 3-D stereo data are either above or below the water's surface; we locate water in an image by identifying this boundary. In fact, assuming the shoreline contains no occlusions blocking it from the field of view of the camera, a simple way to find the shoreline is to calculate the height of dense 3-D stereo data with respect to the surface of water, where data with a height of zero corresponds to the boundary separating water from land. However, this approach is not robust to occlusions blocking the shoreline. Our water detection algorithm is designed to account for occlusions.

After obtaining an initial estimate of the location of the shoreline in the stereo pair, our water detection algorithm works by iteratively sliding an image patch inside the left image. At each iteration, we perform two operations within this image patch: first, we calculate dense 3-D stereo data near the

shoreline. Second, using the sign and height of the dense 3-D data, we find the boundary that separates water from land by formulating and solving a bi-directional graph-based optimization problem, which outputs the boundary separating water from land. When estimating dense 3-D stereo data, we rely on the normalized cross correlation (NCC) algorithm [60, 61]. Although the dense stereo correspondence problem is a widely studied topic in computer vision [60–66, 66–88], we found that NCC produced sufficient accuracy for our purposes and is simple to implement.

To guide our selection of the image patch at each iteration, we generate a large number symmetric pairs distributed across the image. We use symmetric pairs because symmetric pairs possess important characteristics that can be used to constrain our water detection algorithm. In particular, any two symmetric pairs should not intersect with each other inside the image plane. Furthermore, the line connecting a symmetric pair intersects with the boundary separating land and water. These two characteristics are used to select an image patch in two steps: first, we sort the symmetric pairs according to the distance between their midpoints. Second, we apply our shoreline detection algorithm by iteratively traversing the image plane using two sorted symmetric pairs at a time. At each iteration, an image patch is selected such that it contains a segment of both lines connecting the two symmetric pairs. Furthermore, the image patch is approximately centered around the midpoints of the two symmetric pairs. We center the image patch around the midpoints because the midpoint of a symmetric pair provides an approximate guess of the location of the shoreline. We now explain each step of the water detection algorithm in greater detail:

Step 1 Our first step is to generate a large number of matching stereo symmetric pairs distributed across the stereo pair. We do so by extracting, matching, and triangulating SIFT correspondences between the stereo pair (these SIFT feature points were already extracted in Step 1 of Section 2.2.3), stereo matches that are not row aligned, or whose disparity is not between 0 and 125 pixels are outliers. To generate symmetric pairs, we use Equation (2.1), where $\mathbf{n}$ and h are obtained from the optimization performed in Section 2.2.6, to reflect the 3-D position of the SIFT correspondences about the water's surface, followed by a projection to the

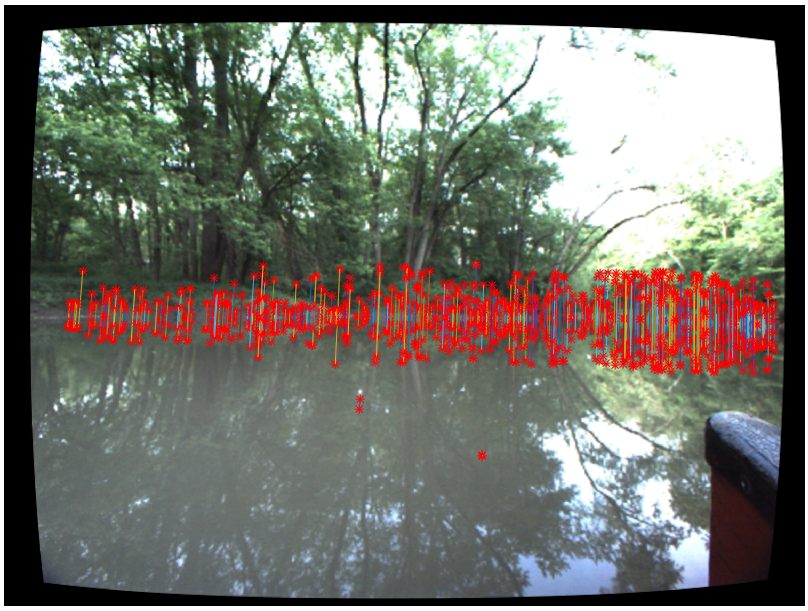


Figure 2.7: The image demonstrates results from our approach to generate stereo symmetric pairs distributed across the left and right image plane, only results from the left image are shown here. We generate symmetric pairs by extracting, matching, and triangulating SIFT correspondences between the stereo pair. Then, we use Equation (2.1) to reflect the 3-D position of the SIFT correspondences about the water's surface, followed by a projection to the image plane.

image plane. Finally, we sort all symmetric pairs according to the distance between their midpoints, and remove any symmetric pair if the line connecting it exceeds a distance of 100 pixels. Figure 2.7 displays the symmetric pairs generated in this step.

Step 2 Obtain an initial estimate of the shoreline. We do so by finding the midpoint $\mathbf{p}_m^{L,1}$, $\mathbf{p}_m^{R,1}$ (in the left and right image plane) of the stereo symmetric pair $\{\mathbf{p}_s^{L,1}, \mathbf{p}_r^{L,1}\}$, $\{\mathbf{p}_s^{R,1}, \mathbf{p}_r^{R,1}\}$ that is closest to the camera. To ensure this symmetric pair is correctly matched, we calculate the disparity D^1 between the source points $\mathbf{p}_s^{L,1}$ and $\mathbf{p}_s^{R,1}$ along with the disparity of 100 symmetric pair source points that are closest to $\mathbf{p}_s^{L,1}$. If D^1 is between the 5th and 95th percentiles of all 100 disparities, we take $\mathbf{p}_m^{L,1}$ as an initial estimate of the shoreline. If not, we reassign $\{\mathbf{p}_s^{L,1}, \mathbf{p}_r^{L,1}\}$, $\{\mathbf{p}_s^{R,1}, \mathbf{p}_r^{R,1}\}$ as the symmetric pair with the median disparity value, and use the newly reassigned

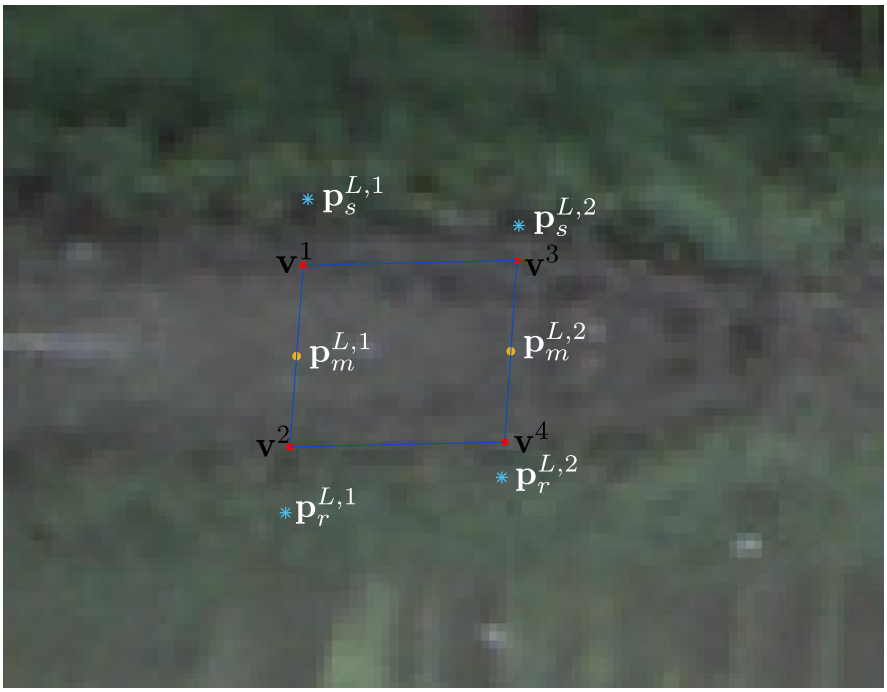


Figure 2.8: The variables used to explain how to extract an image patch in the left image. We extract an image patch in the left image by finding the symmetric pair $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$ whose midpoint $\mathbf{p}_m^{L,2}$ is spaced at least 20 pixels from $\mathbf{p}_m^{L,1}$. Then, we find four points $\mathbf{v}^1$, $\mathbf{v}^2$, $\mathbf{v}^3$, $\mathbf{v}^4$ that are located on the line connecting the symmetric pairs, and that are spaced by 11 pixels on either side of $\mathbf{p}_m^{L,1}$ and $\mathbf{p}_m^{L,2}$. For simplicity, we make the patch rectangular by using the following four points as vertices: (x_1,y_1), (x_1,y_2), (x_2,y_1), (x_2,y_2) where $x_1 = \min(\mathbf{v}_x^1, \mathbf{v}_x^2)$, $x_2 = \max(\mathbf{v}_x^3, \mathbf{v}_x^4)$, $y_1 = \min(\mathbf{v}_y^1, \mathbf{v}_y^3)$, $y_2 = \max(\mathbf{v}_y^2, \mathbf{v}_y^4)$.

value of $\mathbf{p}_m^{L,1}$ as an initial estimate of the shoreline.

Step 3 In the left image, we extract an image patch by finding the symmetric pair $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$ whose midpoint $\mathbf{p}_m^{L,2}$ is spaced at least 20 pixels from $\mathbf{p}_m^{L,1}$. Then, we find four points $\mathbf{v}^1$, $\mathbf{v}^2$, $\mathbf{v}^3$, $\mathbf{v}^4$ that are located on the line connecting the symmetric pairs, and that are spaced by 11 pixels on either side of $\mathbf{p}_m^{L,1}$ and $\mathbf{p}_m^{L,2}$, see Figure 2.8. We extract a rectangular image patch in the left image using the following four points as vertices: (x_1,y_1), (x_1,y_2), (x_2,y_1), (x_2,y_2) where $x_1 = \min(\mathbf{v}_x^1, \mathbf{v}_x^2)$, $x_2 = \max(\mathbf{v}_x^3, \mathbf{v}_x^4)$, $y_1 = \min(\mathbf{v}_y^1, \mathbf{v}_y^3)$, $y_2 = \max(\mathbf{v}_y^2, \mathbf{v}_y^4)$, and the subscript attached to the vertex coordinate denotes the x-coordinate or y-coordinate of the vertex point.

Step 4 For each of the image coordinates in the left image patch, we calculate their 3-D location by finding a stereo correspondence in the right image using normalized cross correlation, followed by stereo triangulation. Letting D^1 represent the disparity between $\mathbf{p}_s^{L,1}$ and $\mathbf{p}_s^{R,1}$, we limit the stereo correspondence search space in this step by extracting a rectangular image patch in the right image whose top left corner is located at $(x_1 - D^1,\ y_1)$, and whose height and width are equal to $y_2 - y_1$ and $x_2 - x_1$, respectively. Notice that the right image patch coordinates (i, j) of this newly formed image patch provide a good initial guess as a stereo correspondence for the left image patch coordinates (i, j). Thus, given left patch coordinates (i, j), our search for a stereo correspondence is constrained to all the right image coordinates that are centered on right patch coordinates (i, j), and that are shifted to the left or right of i by eleven pixels.

Step 5 Formulate a bi-directional graph-based optimization framework using the dense 3-D stereo data calculated in Step 4. Each node in the graph represents the image coordinates of a single pixel in the left image patch. Additionally, each node represents the distance of a single pixel, in the left image patch, to the surface of water or to the camera. An edge represents a connection between two nodes (or pixels) that are adjacent to one another in any one of the cardinal or ordinal compass directions, i.e., north, south, east, west, northeast, northwest, southeast, or southwest (see Figure 2.9 which demonstrates a 4x4 image patch).

Step 6 Assuming node $N_{i,j}$ and node $N_{k,n}$ are connected by an edge, the cost to travel from node $N_{i,j}$ to node $N_{k,n}$ is given by the 3-D distance $d_{k,n}$ of node $N_{k,n}$ to the water's surface, where $d_{k,n}$ is measured along the normal of the surface of water (see Figure 2.9, notice that all edges leading into a node have the same cost or 3-D distance $d_{k,n}$).

Step 7 To account for occlusions, let $N_{k,n}$ represent the neighbor of $N_{i,j}$ in the north, south, east, or west direction. We modify the cost to travel between nodes as follows: for each node $N_{i,j}$, we compare the

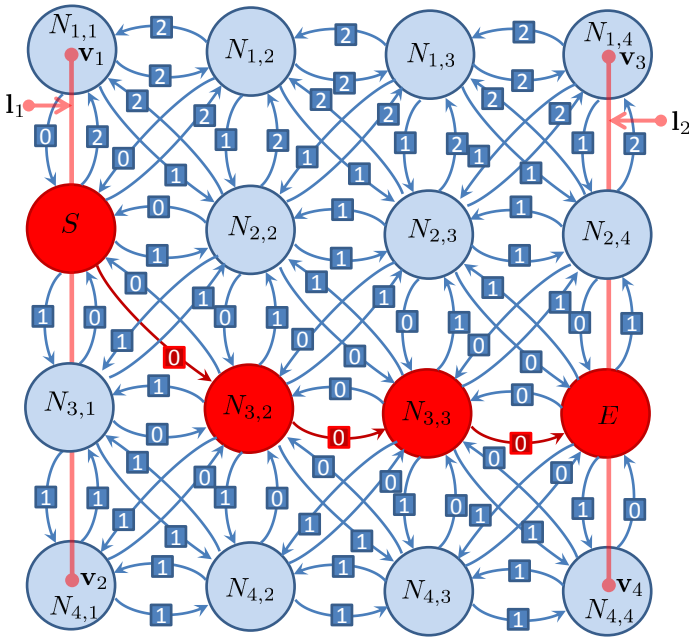


Figure 2.9: A bi-directional graph-based optimization framework using dense 3-D stereo data. Each node in the graph represents image coordinates of a single pixel in a 4x4 image patch. Additionally, each node represents the distance of a single pixel, in the 4x4 image patch, to the surface of water or to the camera. An edge represents a connection between two nodes (or pixels) that are adjacent to one another. The cost to travel from node $N_{i,j}$ to node $N_{k,n}$ is given by the distance $d_{k,n}$ of node $N_{k,n}$ to the plane representing the water's surface, where $d_{k,n}$ is measured along the normal vector of the plane representing the water's surface. The node with the smallest distance that is located on line $\mathbf{l}_1$ is selected as the start node S, while the node with the smallest distance that is located on line $\mathbf{l}_2$ is selected as the end node E. The minimum cost path from node S to node E provides an estimate of the location of the shoreline. We solve for the minimum cost path using Dijkstra's algorithm.

sign of $d_{i,j}$ to the sign of $d_{k,n}$. If the sign of $d_{i,j}$ is opposite to $d_{k,n}$ and the magnitude of $d_{i,j}$ is less than $d_{k,n}$, we set the cost to travel to $N_{i,j}$, from any one of its eight neighboring nodes, to be 1e-6.

Step 8 Now, our objective is to find the lowest cost path that cuts across the left image patch. Doing so will identify the boundary that separates water from land. To find the lowest cost path, we define the start node S as the node with the smallest cost that is located on the line $\mathbf{l}_1$ joining $\mathbf{v}^1$ and $\mathbf{v}^2$, and we define the end node E as the node with the smallest cost that is located on the line $\mathbf{l}_2$ joining

$\mathbf{v}^3$ and $\mathbf{v}^4$ (see Figure 2.9).

Step 9 Using Dijkstra's algorithm [89], we find the minimum cost path from node S to node E.

Step 10 Once the shortest path has been determined for this image patch, we extract a new image patch P^L, P^R essentially by repeating Step 3 and Step 4. Indeed, notice that E is located on the line joining the symmetric pair $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$ (see Figure 2.10). Thus, we find the symmetric pair $\{\mathbf{p}_s^{L,3}, \mathbf{p}_r^{L,3}\}$ whose midpoint $\mathbf{p}_m^{L,3}$ is spaced at least 20 pixels from E and is on the side of $\mathbf{l}_2$ opposite to S. Then, we reassign the four points $\mathbf{v}^1$, $\mathbf{v}^2$, $\mathbf{v}^3$, $\mathbf{v}^4$ to be on the line connecting $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$ and $\{\mathbf{p}_s^{L,3}, \mathbf{p}_r^{L,3}\}$, and that are spaced 11 pixels on either side of E and $\mathbf{p}_m^{L,3}$, see Figure 2.10. Next, we reassign $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$ as $\{\mathbf{p}_s^{L,1}, \mathbf{p}_r^{L,1}\}$, and $\{\mathbf{p}_s^{L,3}, \mathbf{p}_r^{L,3}\}$ as $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$, respectively, and we extract a rectangular image patch in the left image by reassigning the following points as vertices: (x_1,y_1), (x_1,y_2), (x_2,y_1), (x_2,y_2) where $x_1 = \min(\mathbf{v}_x^1, \mathbf{v}_x^2)$, $x_2 = \max(\mathbf{v}_x^3, \mathbf{v}_x^4)$, $y_1 = \min(\mathbf{v}_y^1, \mathbf{v}_y^3)$, $y_2 = \max(\mathbf{v}_y^2, \mathbf{v}_y^4)$. Finally, letting D^2 represent the disparity between E and the stereo correspondence for E, we extract a rectangular image patch in the right image by assigning the top left corner as $(x_1 - D^2, y_1)$, and setting the patch height and width equal to $y_2 - y_1$ and $x_2 - x_1$, respectively. Note, as in Step 4, this right image patch is used to limit the stereo correspondence search space.

Step 11 For each of the image coordinates in the left image patch obtained in Step 10, we calculate their 3-D location by finding a stereo correspondence in the right image using normalized cross correlation, followed by stereo triangulation. In this step, we again limit the search space with the approach described in Step 4, and the right image patch found in Step 10.

Step 12 Repeat Steps 5-11 on image patches P^L and P^R until the estimated boundary that separates water from land intersects the outer edge of the image plane, or the median value of the camera to patch distance is greater than 125 meters.

Step 13 Completing Step 12 gives an estimate of the boundary that sepa-

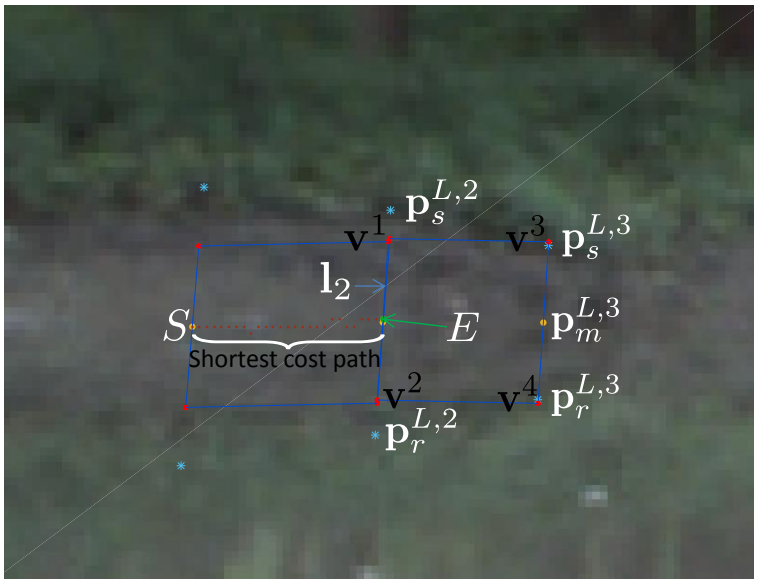


Figure 2.10: The variables used to explain how to slide an image patch across the left image. Once the shoreline has been determined for the previous image patch, we extract a new image patch P^L. To do so, notice that node E is on the shoreline, and is located on the line joining the symmetric pair $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$. We find the symmetric pair $\{\mathbf{p}_s^{L,3}, \mathbf{p}_r^{L,3}\}$ whose midpoint $\mathbf{p}_m^{L,3}$ is spaced at least 20 pixels from E and is on the side of $\mathbf{l}_2$ opposite of S. Then, we reassign the four points $\mathbf{v}^1$, $\mathbf{v}^2$, $\mathbf{v}^3$, $\mathbf{v}^4$ to be on the line connecting $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$ and $\{\mathbf{p}_s^{L,3}, \mathbf{p}_r^{L,3}\}$, and that are spaced 11 pixels on either side of E and $\mathbf{p}_m^{L,3}$. For simplicity, we extract a rectangular image patch by reassigning the following points as vertices: (x_1,y_1), (x_1,y_2), (x_2,y_1), (x_2,y_2) where $x_1 = \min(\mathbf{v}_x^1, \mathbf{v}_x^2)$, $x_2 = \max(\mathbf{v}_x^3, \mathbf{v}_x^4)$, $y_1 = \min(\mathbf{v}_y^1, \mathbf{v}_y^3)$, $y_2 = \max(\mathbf{v}_y^2, \mathbf{v}_y^4)$.

rates water from land between the first image patch and the outer edge of the camera's sensing capabilities. However, since the first image patch could be located any where along the border of water, we continue our search for the boundary of water by repeating Steps 1-12. However, when reaching Step 2, we replace the symmetric pair $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$ with a symmetric that that is on the side of the line joining $\{\mathbf{p}_s^{L,1}, \mathbf{p}_r^{L,1}\}$ opposite of the first found symmetric pair $\{\mathbf{p}_s^{L,2}, \mathbf{p}_r^{L,2}\}$ and whose midpoint is spaced at least 20 pixels from $\mathbf{p}_m^{L,1}$. Once completed, Steps 1-12 provide an estimate of where land intersects with water on one side of the river.

Step 14 To find where water intersects with land on the other side of the river, we locate remaining symmetric pairs in which the line connecting these remaining symmetric pairs does not intersect with the

border of water estimated in Steps 1-13, and that are within 125 meters from the camera.

Step 15 Repeat Steps 2-13 with this remaining set of symmetric pairs.

Once the previous steps are complete, the above algorithm provides an estimate of the boundary of water on both sides of a river.

2.4 Experimental Validation

To validate the effectiveness of the proposed algorithm, we apply our water detection algorithm to a video sequence taken from the the visual-inertial canoe dataset [90]. Portions of the canoe dataset contain images of a river with specular reflections and distinguishable symmetric pairs, making it applicable to the algorithm proposed in this chapter. The entire dataset was captured using a GPS-INS track that is time-synchronized with stereo camera measurements and IMU measurements. The entire sensor package was mounted to the front of canoe as it navigated a river setting. During data collection, the stereo camera had a 60 cm baseline with 3.5 mm focal length lens, and images were sampled at 20 Hz. To validate the proposed water detection algorithm, we apply it to a chronological sequence of 6400 images, i.e., a video sequence 320 seconds long. During this time, the canoe traveled approximately 318.9 meters. Results of our river segmentation algorithm applied to 32 images in this dataset are demonstrated in Figure 2.11. These images provide a good representation of the results over the entire dataset, and, due to the accuracy of these results, we apply our river detection algorithm as an object recognition module for Curve SLAM in Chapter 4 to create a map of the border of a river.

The visual-inertial canoe dataset also provides a precise roll and pitch ground-truth, allowing us to compare the roll, and pitch estimate we obtained in Section 2.2.7 against this ground-truth signal. This comparison is plotted in Figure 2.12. We could not compare our height estimate because the accuracy of the ground-truth height provided in this dataset is not accurate. All the measurements match ground-truth almost perfectly. It is due to this precision that we include them as measurements in the Curve SLAM algorithm in Chapter 4. Indeed, the roll estimate almost perfectly

Figure 2.11: Results of our water segmentation algorithm applied to a stereo pair. The red line indicates the boundary that separates water from land, and is the output obtained from the river segmentation algorithm proposed in this chapter. The images are in chronological order from left to right and top to bottom. The timestamp of the top left image is zero, the timestamp of the bottom right image is 320 seconds, and the images in the middle are spaced 10 seconds apart.

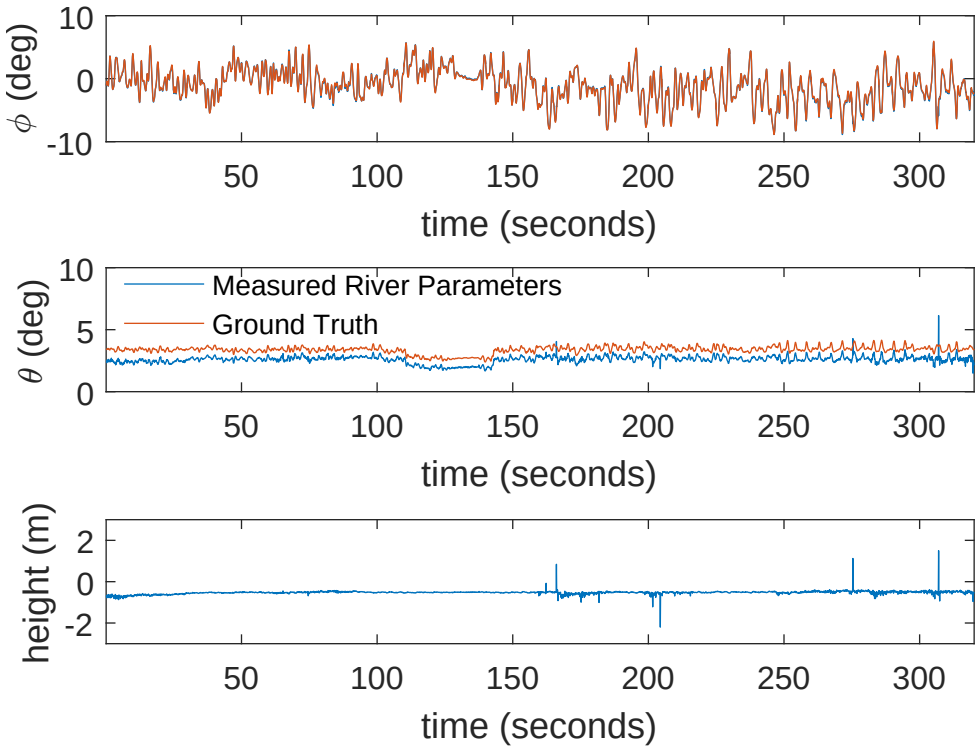


Figure 2.12: Roll, pitch, and height measurements that we estimated using the normal and height of the water's surface compared against precise roll and pitch ground-truth. Due to the precision of our estimates, we include them as measurements in the Curve SLAM algorithm in Chapter 4.

matches the ground-truth signal, while our pitch estimate appears to have a near-constant offset of about 0.7 degrees. We hypothesize that this offset exists because the normal of the water's surface is not perfectly parallel to the gravity vector. It is important to note that when we incorporate the pitch measurements into the Curve SLAM algorithm in Chapter 4, we add a small offset of roughly 0.7 degrees to the pitch measurement. With the exception of a few spurious measurements, the estimated height of the camera above the surface of water matches what we predict, i.e., the height measures 0.5 meters and is approximately constant over the entire dataset.

2.5 Cases Where Our Algorithm Fails to Find the Boundary of a River

We now describe a few cases in which our algorithm has problems detecting the border of a river and the reason for these problems. The first case happens when symmetric feature points are not available in the environment, this case occurs when the main assumption of our algorithm is violated: the river does not contain specular reflections. This problem can be detected by checking the number of matched symmetric pairs that are returned from our symmetric pair matching algorithm, see Sections 2.2.3 - 2.2.5. A related problem is when only a small number of matching symmetric pairs are found. In this case, there are not enough symmetric pairs to robustly eliminate outliers. Consequently, incorrectly matched symmetric pairs are used to estimate the normal and height of the water's surface, which causes our algorithm to estimate the boundary of water incorrectly, see Figure 2.13, and notice in some of these images the estimated shoreline is located on an incorrectly oriented plane. Another case in which our algorithm struggles is when it fails to match dense stereo correspondences correctly. These matching problems are due to typical dense stereo correspondence failures: the stereo pair fails to satisfy the brightness constancy assumption, different image patches appear too similar to one another, or occlusions block one of the camera's FOV; these cases are shown in Figure 2.13; notice in some of these images the estimated shoreline unexpectedly dips up or down from the true shoreline.

However, simply applying open-source computer-vision algorithms will completely fail to correctly find the boundary of a river. We applied several active contour methods [91–97] and the pre-trained convolutional neural network [98] described in Section 3.2.1, but all these algorithms were unable to correctly recognize the shoreline, these results are shown in Figures 2.14 and 2.15. Of course, this is an expected result because all these algorithms are not designed to identify the border of a river.

2.6 Conclusion

In this chapter, we presented an algorithm to segment water from land in a river setting containing specular reflections and distinguishable symmetric

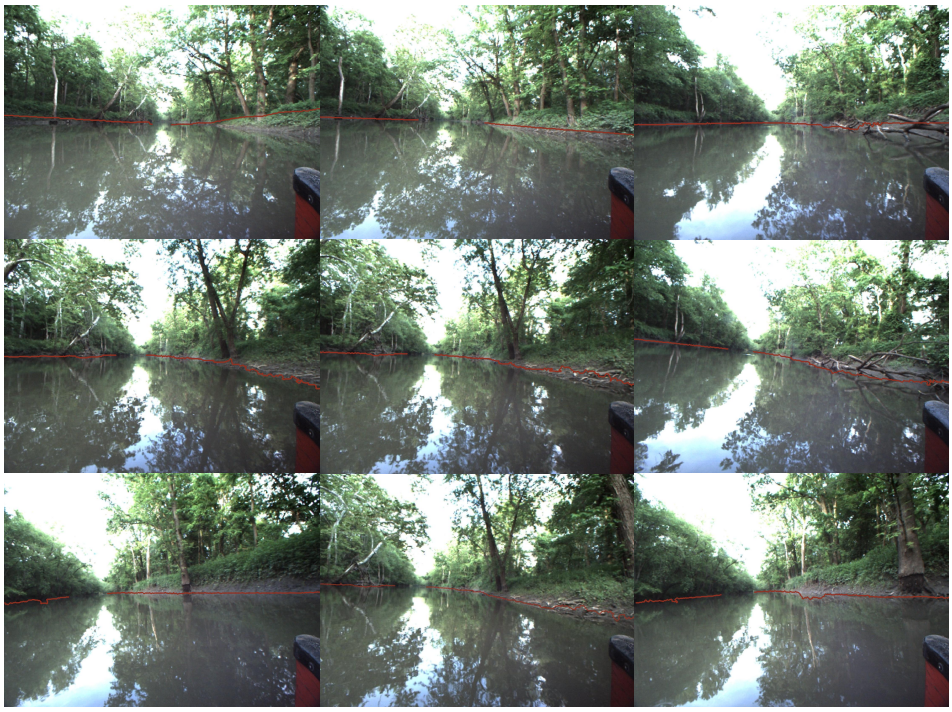
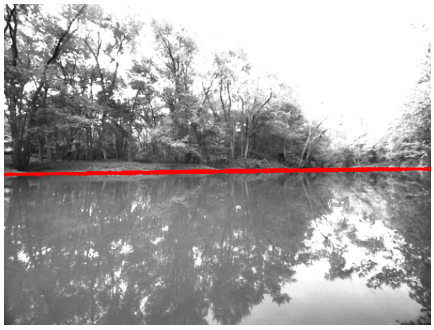


Figure 2.13: Images in which our algorithm has trouble detecting the border of a river. Our algorithm has problems when the environment lacks distinguishable symmetric pairs, or when it fails to match dense stereo correspondences correctly.



(a) Near perfect estimate of the shoreline hand-picked by a human.

(b) Shoreline estimate of an active contour algorithm using the estimate in Figure 2.14a.

Figure 2.14: River segmentation results of active contour algorithms. These algorithms are unable to correctly recognize the shoreline, and require an initial shoreline estimate. Notice that even with a near perfect shoreline estimate (see Figure 2.14a), these algorithms are unable to detect the shoreline (see Figure 2.14b).

(a) Left image.

(b) Right image.

Figure 2.15: River segmentation results of the pre-trained convolutional neural network [98] described in Section 3.2.1. This algorithm is unable to correctly recognize the shoreline. This is an expected result because the algorithm is designed to recognize a road (see Figure 2.15b) and not the border of a river (see Figure 2.15a).

feature points. Our algorithm works by identifying the boundary separating 3-D data which are either above or below the surface of water. To locate this boundary, we presented an algorithm to robustly estimate the normal and height of the surface of water. Then, with the normal and height, we presented an algorithm to identify where dense 3-D stereo data are above or below the surface of water. We demonstrated the robustness of our algorithm by applying it to a video sequence obtained from the visual-inertial canoe dataset that is over 320 seconds long and roughly 319 meters in length. In fact, the robustness of our river detection algorithm allows us to use it as an object recognition module for Curve SLAM in Chapter 3. Finally, we showed how to obtain a precise measurement of roll, pitch, and height from the normal of the water surface plane, and we demonstrated the accuracy of the roll and pitch measurements by comparing them against a precise ground-truth signal. Due to the precision of these measurements, we include them as measurements in the Curve SLAM algorithm in Chapter 4.

CHAPTER 3

THE CURVE SLAM ALGORITHM

In this chapter, we show how to solve the SLAM problem by using Bézier curves as landmark primitives in the SLAM framework as opposed to feature points. Our approach allows us to create a sparse map that recovers the outline, shape, and dimensions of unique objects in the environment, e.g., a river, road, or sidewalk. After obtaining the boundary of these objects in an image, we present an algorithm that interpolates, splits, and matches curves to boundary of these objects in a stereo pair. This allows us to reconstruct the 3-D location of curves, parameterized by a small number of control points. Then, we show how to formulate and solve the SLAM problem using Bézier curves and an extended Kalman filter. To solve the SLAM problem, we present a data association algorithm that compares the physical dimensions of curve landmarks between chronologically sequential stereo image frames to remove curve outliers. Additionally, we show how to determine the polynomial order of curves, and how to add curves to the SLAM estimator. Finally, we present a curve combining algorithm that further reduces the number of landmark states representing the map.

The remainder of this chapter proceeds as follows: In Section 3.1, we provide a high-level overview of the algorithm, and introduce notations and definitions used to describe the algorithm. Section 3.2 explains how to estimate 3-D body-frame curves from a single stereo pair. In Section 3.3, we formulate and solve the SLAM problem. Section 3.4 describes cases where Curve SLAM has trouble producing accurate localization and mapping results. Section 3.5 provides a conclusion to this chapter. Experimental results of Curve SLAM are presented in Chapter 4.

3.1 Curve SLAM Overview

3.1.1 Goals and Assumptions

The aim of this chapter is to estimate the pose of a robot equipped with a stereo camera and IMU while providing a sparse structured map of a previously unknown environment using static curved features as landmarks. Letting $\mathbf{x} \in \mathbb{R}^{15+12N}$ represent the state vector, our goal is to estimate the following variables:

$$\mathbf{x} \triangleq [\mathbf{p}^{\mathcal{W}}, \mathbf{v}^{\mathcal{B}}, \boldsymbol{\Theta}, \mathbf{b}_a, \mathbf{b}_g, (\mathbf{C}_1^{\mathcal{W}})^\top, \ldots, (\mathbf{C}_N^{\mathcal{W}})^\top]^\top \tag{3.1}$$

where $\mathbf{p}^{\mathcal{W}}$ represents the robot's position with respect to the world frame, $\mathbf{v}^{\mathcal{B}}$ represents the body-frame velocity of the robot, $\boldsymbol{\Theta}$ represents the robot's attitude, $\mathbf{b}_a$ represents accelerometer bias, and $\mathbf{b}_g$ represents gyroscope bias. The N variables $\mathbf{C}_1^{\mathcal{W}}, \ldots, \mathbf{C}_N^{\mathcal{W}}$ represent the location of curved features defined with respect to the world frame. Each curved feature $\mathbf{C}_j^{\mathcal{W}} \in \{\mathbf{C}_1^{\mathcal{W}}, \ldots, \mathbf{C}_N^{\mathcal{W}}\}$ is represented as a Bézier curve and is defined by the location of its control points, i.e., $\mathbf{C}_j^{\mathcal{W}} \triangleq [\ (\mathbf{P}_{j,0}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,1}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,2}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,3}^{\mathcal{W}})^\top]^\top \in \mathbb{R}^{12}$ where the variables $\mathbf{P}_{j,0}^{\mathcal{W}} \in \mathbb{R}^3$, $\mathbf{P}_{j,1}^{\mathcal{W}} \in \mathbb{R}^3$, $\mathbf{P}_{j,2}^{\mathcal{W}} \in \mathbb{R}^3$, and $\mathbf{P}_{j,3}^{\mathcal{W}} \in \mathbb{R}^3$ represent the 3-D coordinates of control points defined with respect to the world frame (an image of a cubic Bézier curve, along with its control points, is shown in Figure 3.1). Each curved feature is fixed to a larger curved object naturally

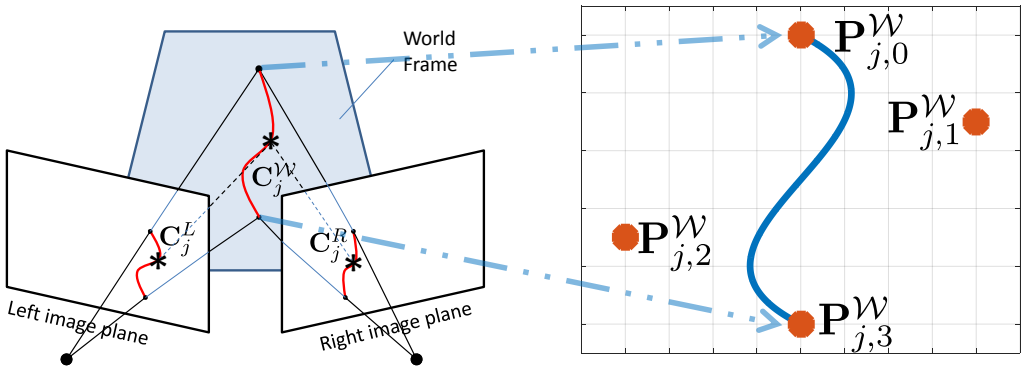


Figure 3.1: The goal of Curve SLAM is to simultaneously estimate the pose of a robot and the 3-D location of curved features. Each curved feature is represented as a Bézier curve and is defined by the location of its 3-D control points: $\mathbf{P}_{j,0}^{\mathcal{W}}$, $\mathbf{P}_{j,1}^{\mathcal{W}}$, $\mathbf{P}_{j,2}^{\mathcal{W}}$, and $\mathbf{P}_{j,3}^{\mathcal{W}}$.

occurring in the scene, e.g., a long curved sidewalk. Additionally, because the work in this dissertation is focused on mapping environments where a road or path dominates the scene, we assume at least one static curve is in the image corresponding to the left or right edge of a road or path.

3.1.2 Outline of Curve SLAM Algorithm

The functional components of Curve SLAM are illustrated in Figure 3.2, and the pseudocode of the proposed Curve SLAM algorithm is provided in Algorithm 1. As shall be demonstrated in Section 3.2, the algorithm uses a single stereo pair to determine the 3-D coordinates of curve landmark parameters, i.e., control points, relative to the body frame of the stereo camera. To reconstruct the 3-D location of control points, we do not rely on matching point-to-point stereo correspondences. Instead, we use the projection of curves in the stereo image plane to formulate a least-squares problem that optimizes the 3-D location of control points. Section 3.3 explains how to track curve landmarks between chronologically sequential image frames in order to solve the data association problem. Section 3.3 also explains how the IMU measurements and the control point measurements captured from the stereo camera are fused together with an EKF to simultaneously localize the stereo camera and create a structured map. Once the location of a 3-D curve has been estimated, Section 3.3.8 shows how to combine this curve with previously estimated curves to further reduce the number of curve landmarks representing the map.

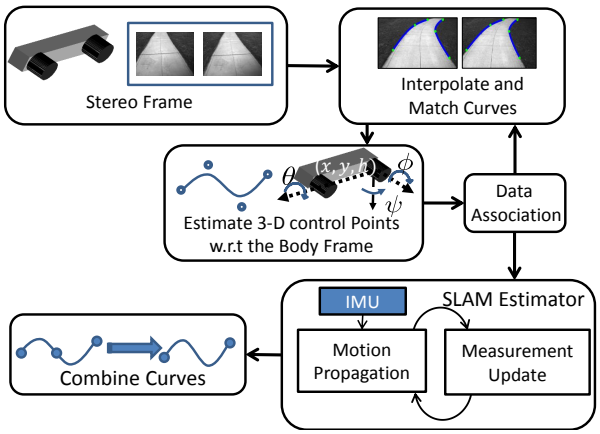


Figure 3.2: Functional components of the Curve SLAM algorithm.

Algorithm 1 Curve SLAM

1: Initialize the camera's pose and breakpoints (Sections 3.3.2 and 3.3.3).
2: **while** A new stereo image pair is available **do**
3: Extract the boundary of the path in the left and right stereo pair (Section 3.2.1).
4: With the breakpoints, interpolate and match curves between the left and right stereo pair (Section 3.2.2).
5: With the Levenberg-Marquardt algorithm, reconstruct the 3-D location of curve control points (Section 3.2.3).
6: Track curves between the current and previous frame (Section 3.3.1).
7: Verify that physical dimensions of curves match between the current and previous frame. Remove outliers as necessary (Section 3.3.1).
8: If necessary, add curves in the image plane, and determine the curve's polynomial order. (Sections 3.3.2 and 3.3.3).
9: Assign breakpoints as the start and end control points of the tracked curves from Step 6, and newly added curves from Step 7.
10: Compute the EKF prediction using the IMU (Section 3.3.5).
11: Compute the EKF update with the reconstructed control points obtained in Step 4 (Section 3.3.6).
12: If necessary, add newly found image curves from Step 7 to the filter state (Section 3.3.7).
13: When a curve leaves the camera's field of view, combine it with previously estimated curves (Section 3.3.8).
14: **end while**

3.1.3 Properties of Bézier Curves

The notations and symbols used in Chapter 3 and Chapter 4 are defined in Table 3.1. The following properties of Bézier curves [99] make them useful for Curve SLAM:

- A Bézier curve is defined by its control points, which define the curve's shape. The start control point and end control point are located at the start point and end point of the curve. Letting t_i be the i^{th} element of $\mathbf{t} = [0, \Delta t, 2\Delta t, \ldots, 1]^\top$, the linear transformation that maps t_i to a point that lies on the Bézier curve $\mathbf{C}_j^{\mathcal{W}}$ is given by

$$\mathcal{A}(t_i, \mathbf{C}_j^{\mathcal{W}}) = [\ \mathbf{P}_{j,0}^{\mathcal{W}},\ \mathbf{P}_{j,1}^{\mathcal{W}},\ \mathbf{P}_{j,2}^{\mathcal{W}},\ \mathbf{P}_{j,3}^{\mathcal{W}}\ ]\ \mathbf{B}_{\mathcal{O}_j}\mathbf{U}_{i,\mathcal{O}_j} \tag{3.2}$$

  where $\mathbf{U}_{i,\mathcal{O}_j} = \begin{bmatrix} t_i^{\mathcal{O}_j} & t_i^{\mathcal{O}_j - 1} & \ldots & 1 \end{bmatrix}^\top$, and the matrix of constant coefficients $\mathbf{B}_{\mathcal{O}_j}$ is obtained from the Bernstein polynomial coefficients (see [99]). In fact, Equation (3.2) is true for any $t \in [0, 1]$, but in this dissertation, we are only concerned with mapping the uniformly spaced vector $\mathbf{t}$ onto a Bézier curve.

Table 3.1: Notations and Definitions

Name	Description
$\mathcal{W}$	The 3-D world frame.
$\mathcal{B}$	The body frame of the stereo camera. When a subscript is attached to the variable, e.g., $\mathcal{B}_r$, it is used to denote the body frame at the r^{th} stereo image frame.
L	The left camera image.
R	The right camera image.
$\mathbf{p}^{\mathcal{W}}$	$\mathbf{p}^{\mathcal{W}} \triangleq (x, y, z)$ is defined as the 3-D displacement of $\mathcal{B}$ with respect to $\mathcal{W}$. The variable $h = -z$ represents the camera's height.
$\mathbf{\Theta}$	$\mathbf{\Theta} \triangleq (\phi, \theta, \psi)$ is the orientation of $\mathcal{B}$ in ZYX euler angles.
$\mathbf{v}^{\mathcal{B}}$	$\mathbf{v}^{\mathcal{B}} \triangleq (u, v, w)$ is defined as the linear velocity of the camera with respect to the body frame.
$\mathbf{T}_{AB}$	$\mathbf{T}_{AB} \in SE(3)$ is the transformation that changes the representation of a point defined in the coordinates of frame B to a point defined in the coordinates of frame A.
$\mathbf{P}_{j,l}^{\mathcal{W}} \in \mathbb{R}^3$	We define the variables $\mathbf{P}_{j,0}^{\mathcal{W}},\ \mathbf{P}_{j,1}^{\mathcal{W}},\ \mathbf{P}_{j,2}^{\mathcal{W}},\ \mathbf{P}_{j,3}^{\mathcal{W}}$ to represent the control points of the j^{th} cubic Bézier curve. The variables $\mathbf{P}_{j,1}^{\mathcal{W}}$ and $\mathbf{P}_{j,2}^{\mathcal{W}}$ are the middle control points, and $\mathbf{P}_{j,0}^{\mathcal{W}}$ and $\mathbf{P}_{j,3}^{\mathcal{W}}$ are the start and end control points, respectively. The superscript denotes the frame where the variable is defined. When the curve is linear or quadratic, $\mathbf{P}_{j,0}^{\mathcal{W}}$ and $\mathbf{P}_{j,3}^{\mathcal{W}}$ are the start and end control points, respectively. When the curve is quadratic, $\mathbf{P}_{j,1}^{\mathcal{W}}$ is the middle control point.
$\mathcal{O}_j$	$\mathcal{O}_j$ denotes the order of the j^{th} curve. In this dissertation, $\mathcal{O}_j$ is 1 (linear), 2 (quadratic), or 3 (cubic).
$\mathbf{C}_j^{\mathcal{W}}$	$\mathbf{C}_j^{\mathcal{W}} \triangleq [\ (\mathbf{P}_{j,0}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,1}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,2}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,3}^{\mathcal{W}})^\top]^\top \in \mathbb{R}^{12}$ is defined as the j^{th} cubic Bézier curve. A similar expression follows for a linear or quadratic curve. The superscript denotes the frame where the variable is defined. Throughout this dissertation, it should be remembered that each curve has an associated polynomial order, which can easily be determined using a lookup table associated with the variable j.
$\mathbf{t}$	We define $\mathbf{t}$ as an ordered vector with n elements that are evenly spaced between zero and one, i.e., $\mathbf{t} = [0, \Delta t, 2\Delta t, \ldots, 1]^\top$.
$\mathbf{U}_{i,\mathcal{O}_j}$	Let t_i be the i^{th} element of $\mathbf{t}$, then $\mathbf{U}_{i,3} = \begin{bmatrix} t_i^3 & t_i^2 & t_i & 1 \end{bmatrix}^\top$, where the subscript denotes the order of the j^{th} curve . A similar expression follows for a linear or quadratic Bézier curve.
$\mathcal{A}(\mathbf{t}, \mathbf{C}_j^{\mathcal{W}})$	The linear transformation that maps $\mathbf{t}$ to points that lie on the bezier curve $\mathbf{C}_j^{\mathcal{W}}$, $\mathcal{A}(\mathbf{t}, \mathbf{C}_j^{\mathcal{W}})$ is defined in Section 3.1.3.
$\boldsymbol{\beta}$	$\boldsymbol{\beta}$ represents an estimated parameter vector of m Bézier curves, i.e., $\boldsymbol{\beta} \triangleq [(\mathbf{C}_1^{\mathcal{B}})^\top \ldots (\mathbf{C}_m^{\mathcal{B}})^\top]^\top$.

- Bézier curves are invariant under affine transformations, i.e., any affine transformation on a Bézier curve is equivalent to an affine transformation on the control points [100].

- If a Bézier curve cannot be degree reduced, the control points are unique and the weights can only be varied in a known way by a perspective transformation [101, 102].

3.2 Estimating 3-D Body-Frame Curves with a Stereo Pair

The purpose of this section is to demonstrate how to reconstruct the 3-D location of the path boundary using a single stereo pair. This task is accomplished by formulating and solving a nonlinear least-squares optimization problem that minimizes the reprojection error of the image coordinates comprising the path boundary. The optimization problem depends on two inputs, and the purpose of this section is to explain how to construct these two inputs. The first input is represented by the variable $\mathbf{y}^{o,j,i}$, which represents 2-D image coordinates located on the path boundary, where $o \in \{L, R\}$, j represents the j^{th} curve, and i represents the i^{th} discretized point belonging to curve j. The second input is the predicted measurement function $\hat{\mathbf{y}}^{o,j,i}(\cdot)$ that represents the projection of a 3-D curve from the body frame to the image plane. A high-level overview of the steps taken to construct these two inputs is as follows (these steps and two inputs are illustrated in Figure 3.3):

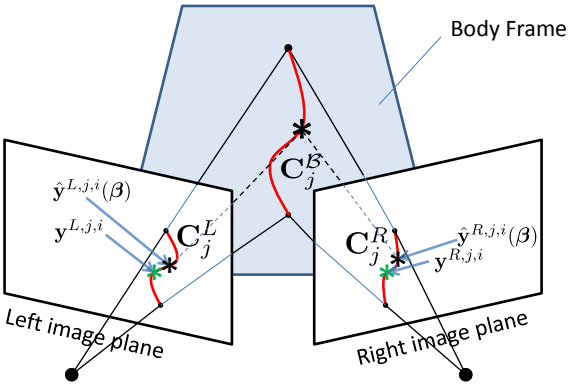
1. Locate the path boundary in each image of the stereo pair.

2. Interpolate and match m Bézier curves $\mathbf{C}_1^L \dots \mathbf{C}_m^L$, $\mathbf{C}_1^R \dots \mathbf{C}_m^R$ to the path boundary in the stereo images; a single point located on these interpolated and matched curves represents the measurement $\mathbf{y}^{o,j,i}$.

3. Obtain the predicted measurement function $\hat{\mathbf{y}}^{o,j,i}(\cdot)$ by projecting points located on 3-D body-frame curves $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$ to the image plane. The curve $\mathbf{C}_j^{\mathcal{B}}$ is related to the j^{th} curve in the world frame $\mathbf{C}_j^{\mathcal{W}}$, an element of the SLAM state defined in Equation (3.1), as follows: $\mathbf{C}_j^{\mathcal{B}} = \mathbf{T}_{\mathcal{BW}}\mathbf{C}_j^{\mathcal{W}}$, where $\mathbf{T}_{\mathcal{BW}} \in SE(3)$. Additionally The curves $\mathbf{C}_1^L \dots \mathbf{C}_m^L$, $\mathbf{C}_1^R \dots \mathbf{C}_m^R$ correspond to the projection of $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$ in the stereo pair.

(a) Step 1. Extract the boundary of the path.

(b) Step 2. Interpolate and match curves.



(c) Step 3. Reconstruct the 3-D location of control points.

Figure 3.3: The steps taken to reconstruct the 3-D location of control points with respect to the body frame. Also, see Proposition 1 in Appendix A.

This projection is uniquely defined. Furthermore, with the correct correspondence between the curves in $\mathbf{C}_1^L \dots \mathbf{C}_m^L$ and $\mathbf{C}_1^R \dots \mathbf{C}_m^R$, we are able to estimate the 3-D location of $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$ in the body frame. A proof of these facts is presented in Appendix A.

3.2.1 Extraction of Path Boundary

In this dissertation, we employ three methods to extract the boundary of the path. In the first method, we identify the boundary of a river containing specular reflections. This method was presented in Chapter 2. In the second

method, we filter the image with an averaging filter to remove noise, threshold the image to locate the path, and apply a contour detector [103] that finds and sorts the path boundary according to spatial proximity. We repeat this procedure for both the left and right image of each stereo frame. The size ρ_n of the average filter window and image threshold are discussed in Section 4.5. The third method relies on a pre-trained convolutional neural network to detect pixels that represent the road [98]. Once the road is detected, we apply a contour detector [103] that finds and sorts the road boundary according to spatial proximity. We repeat this procedure for both the left and right images of each stereo frame.

3.2.2 Interpolating and Matching Curves in the Image Plane

To find $\mathbf{y}^{o,j,i}$, we interpolate and match a set of m Bézier curves $\mathbf{C}_1^L \dots \mathbf{C}_m^L$, $\mathbf{C}_1^R \dots \mathbf{C}_m^R$ to the path boundary in the stereo pair using linear least squares by modifying the algorithm in [104]. During this process, we must be able to quickly match a measurement $\mathbf{y}^{o,j,i}$ to a predicted measurement $\hat{\mathbf{y}}^{o,j,i}(\cdot)$, and determine where to split curves when the boundary is not sufficiently smooth. In this subsection, we explain how to construct $\mathbf{y}^{o,j,i}$ to accomplish these tasks. We define B as the set of image coordinates comprising the path boundary, and a break point as a single image coordinate belonging to the set B. A break point designates a desired start or end control point of the j^{th} curve $\mathbf{C}_j^o$ in the image plane. Between two break points, we attempt to interpolate $\mathbf{C}_j^o$. Break points are determined when a curve is added to the state (Section 3.3.2), or from the data association step (Section 3.3.1). The steps to interpolate and match curves are as follows:

Step 1 Let $(u^r_{b_{2j-1}}, v^r_{b_{2j-1}})$ and $(u^r_{b_{2j}}, v^r_{b_{2j}})$ be the break points of curve j in image frame r and $B_c = \{(u^r_{b_{2j-1}}, v^r_{b_{2j-1}}), \dots, (u^r_{b_{2j}}, v^r_{b_{2j}})\} \subset B$ be the image coordinates between the break points $(u^r_{b_{2j-1}}, v^r_{b_{2j-1}})$ and $(u^r_{b_{2j}}, v^r_{b_{2j}})$. We interpolate a single Bézier curve $\mathbf{C}_j^L$ to B_c by fixing the start control point $\mathbf{P}_{j,0}^L$ and end control point $\mathbf{P}_{j,3}^L$ at the break points. In other words, $\mathbf{P}_{j,0}^L = (u^r_{b_{2j-1}}, v^r_{b_{2j-1}})$, $\mathbf{P}_{j,3}^L = (u^r_{b_{2j}}, v^r_{b_{2j}})$, and depending on the order of the curve (determined in Section 3.3.3), we find the middle control points $\mathbf{P}_{j,1}^L$ and $\mathbf{P}_{j,2}^L$ with least-squares. We repeat this process for all the break points in the left image.

Step 2 Match curves between the left and right images. To do so, we obtain $\mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$, which has the effect of discretizing $\mathbf{C}_j^L$ into a set of spatially ordered points that lie on curve $\mathbf{C}_j^L$. In general, the spacing of these discretized points is not uniform, but is obtained by mapping the uniformly spaced vector $\mathbf{t}$ onto a nonlinear polynomial (see Equation (3.2)). Other parameterizations exist for the vector $\mathbf{t}$ that allow the discretized curve points to be more uniformly spaced [105, 106], but these approaches are more computationally complex. Each point $\boldsymbol{\mathcal{P}}^L \in \mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$ must satisfy the epipolar constraint in the right image and must lie on the path boundary. These two constraints combined give a good initial estimate of where the discretized curve $\mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$ is located in the right image. Then, for each point $\boldsymbol{\mathcal{P}}^L \in \mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$, we implement a template matcher to further refine the curves location in the right image. The template matcher compares two small image patches between the left and right image. One image patch is centered around each point $\boldsymbol{\mathcal{P}}^L \in \mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$, while the second, slightly larger image patch is centered around the estimated location of $\boldsymbol{\mathcal{P}}^L$ in the right image. The size ρ_t of the template window is described in Section 4.5.

Step 3 Our last step is to interpolate a Bézier curve $\mathbf{C}_j^R$ to the refined location of $\mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$ in the right image.

Typical results of the matching process are shown in Figure 3.4. With the interpolated image curves, we find a measurement $\mathbf{y}^{o,j,i}$ by mapping the vector $\mathbf{t}$ with curve $\mathbf{C}_j^o$ to the image plane. A measurement is given by

$$\mathbf{y}^{o,j,i} = \mathcal{A}(t_i, \mathbf{C}_j^o) \tag{3.3}$$

3.2.3 Curve Parameter Optimization

Our next step is to reconstruct the 3-D coordinates of the path boundary with a series of m Bézier curves $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$. When reconstructing the path boundary, it is helpful to remember that the measured curves $\mathbf{C}_1^L \dots \mathbf{C}_m^L$, $\mathbf{C}_1^R \dots \mathbf{C}_m^R$, defined in Section 3.2, correspond to the projection of $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$

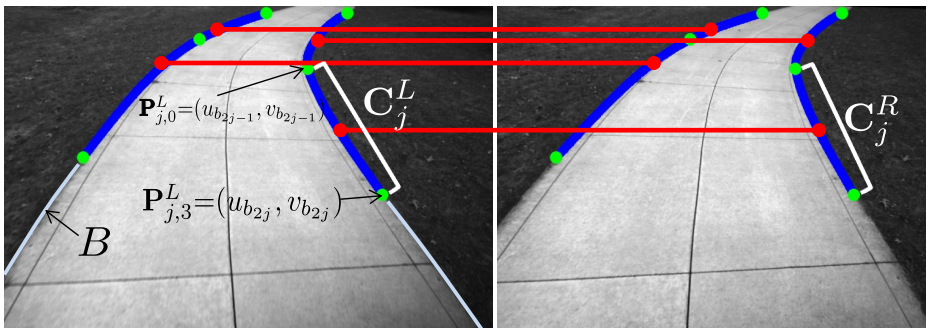


Figure 3.4: Curves matched in the left and right images. The green points represent the start or end point of a single curve. The red lines denote matching sets of curves.

in the stereo pair. Defining $\boldsymbol{\beta} \triangleq [(\mathbf{C}^{\mathcal{B}}_1)^\top \dots (\mathbf{C}^{\mathcal{B}}_m)^\top]^\top$ as a stacked parameter vector of curve control points, we estimate $\boldsymbol{\beta}$ by formulating a nonlinear least-squares optimization problem that minimizes the reprojection error of the image coordinates comprising the path boundary in a single stereo frame. The general form of the optimization problem is given by

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \sum_{o\in\{L,R\}} \sum_{j=1}^{m} \sum_{i=1}^{n} ||\mathbf{y}^{o,j,i} - \hat{\mathbf{y}}^{o,j,i}(\boldsymbol{\beta})||^2 \tag{3.4}$$

where o indicates the left or right stereo image, j indicates the curve, and i indicates the image coordinates belonging to curve j. The 2-D vector $\mathbf{y}^{o,j,i}$ represents measured image coordinates belonging to the path boundary. We showed how to calculate $\mathbf{y}^{o,j,i}$ in Section 3.2.2. The function $\hat{\mathbf{y}}^{o,j,i}(\cdot)$ represents a measurement predicted by the parameters in $\boldsymbol{\beta}$. To find a predicted measurement, we map the vector $\mathbf{t}$, using the j^{th} curve $\mathbf{C}^{\mathcal{B}}_j$, from the body frame to the image plane. Letting t_i be the i^{th} element of $\mathbf{t}$, a predicted measurement is given by the equation

$$\hat{\mathbf{y}}^{o,j,i}(\boldsymbol{\beta}) = \mathbf{M}^o(\mathcal{A}(t_i, \mathbf{C}^{\mathcal{B}}_j)) \tag{3.5}$$

where $\mathbf{M}^o(\cdot)$ is the function that maps a 3-D point from the body frame to the image plane of the left or right camera.

With the measurement and predicted measurement defined, we solve Equation (3.4) with the Levenberg-Marquardt algorithm [55–57]. An initial guess for the parameters in $\boldsymbol{\beta}$ is given by the optimized variables from the previous

image frame. Once the optimization is complete, the reprojection error of each curve is checked. Any curve with a reprojection larger than a threshold ρ_r is discarded. The selection of ρ_r is based on the precision of the stereo camera calibration. The purpose of ρ_r is simply to provide a fail safe when it is obvious the stereo triangulation failed, the size of ρ_r is discussed in Section 4.5. The output of the Levenberg-Marquardt optimization gives an estimate of the curve control points defined with respect to the body frame.

After applying the Levenberg-Marquardt algorithm to estimate $\boldsymbol{\beta}$, we calculate the measurement covariance matrix $\mathbf{V}_r$, as described in Section 4.5 [107]. The covariance $\mathbf{V}_r$ is used as the extended Kalman filter measurement covariance in Section 3.3.6, and will be used in solving the data association step in Section 3.3.1.

3.3 SLAM Estimator Formulation

In this section, we propose a solution to the data association problem and formulate an Extended Kalman filter to solve the SLAM problem. To do so, we determine the order of each Bézier curve and determine when to add curves to the filter state. This section explains how to accomplish these tasks.

3.3.1 Curve-Based Data Association

We solve the data association problem by tracking the start and end points of curves between sequential frames in the left camera's field of view (FOV), and by comparing the 3-D structure of these curves between frames to ensure they were tracked correctly. Tracking is done with the the Lucas-Kanade tracking (KLT) algorithm [108]. In implementing the KLT algorithm, it is important to emphasize that our data association algorithm is different from conventional point-based tracking algorithms, e.g., the type that relies on salient regions in the image to detect, describe, and track a uniform distribution of points between image frames, followed by an outlier rejection algorithm to remove outliers, see [3] as an example. Instead, the KLT algorithm is used to track just the start or end points of curves between two sequential image frames. Additionally, because we remove outliers by comparing the 3-D shape of curves between image frames, it is sufficient to track fewer than five

landmark curves between most image frames. In other words, our algorithm is not dependent on tracking a large number of point-based landmarks. In turn, this allows our algorithm to operate in settings that lack distinguishable feature points and to create maps that are more sparse than point-based SLAM algorithms. This approach is quite different from point-based data association algorithms that track hundreds of feature points between image frames. Finally, our data association step is different from tracking a collection of feature points because curves lie on the edge of a path. Thus, when finding a curve correspondence between two chronologically sequential image frames, the data association search space is limited to a one-dimensional edge.

The KLT algorithm will occasionally track the start and end control points to the wrong location, producing outliers. We remove curve outliers by comparing their 3-D curve shapes between frames. We explain our outlier rejection process assuming cubic ordered curves since similar steps can be applied to linear or quadratic curves. To compare curves, we define $\mathbf{d}_{l,l+1}^{r-1}$ as the estimated distance between two control points, i.e., $\mathbf{d}_{l,l+1}^{r-1} = \mathbf{P}_{j,l}^{\mathcal{B}_{r-1}} - \mathbf{P}_{j,l+1}^{\mathcal{B}_{r-1}}$, see Figure 3.5. Alternatively, $\mathbf{d}_{l,l+1}^{r-1}$ can be written as $\mathbf{d}_{l,l+1}^{r-1} = \boldsymbol{A}\boldsymbol{\beta}$, where $\boldsymbol{A}$ is of the form $\boldsymbol{A} = [0, \ldots, 0, 1, -1, 0, \ldots, 0]$. In this case, $\boldsymbol{\Sigma}_{l,l+1}^{r-1}$ has an estimated covariance given by $\boldsymbol{\Sigma}_{l,l+1}^{r-1} = \boldsymbol{A}\mathbf{V}_{r-1}\boldsymbol{A}^{\top}$. The steps to solve the data association problem are as follows:

Step 1 Track the break points of curves $(u_{b_1}^{r-1}, v_{b_1}^{r-1}), \ldots, (u_{b_{2m}}^{r-1}, v_{b_{2m}}^{r-1})$ from frame $r-1$ to frame r with the the Lucas-Kanade tracking (KLT) algorithm. We represent the image coordinates that tracked from frame $r-1$ to frame r as $(u_{t_1}^{r}, v_{t_1}^{r}), \ldots, (u_{t_{2m}}^{r}, v_{t_{2m}}^{r})$. These image coordinates are not yet confirmed as break points because the KLT algorithm may fail to correctly track break points between image frames.

Step 2 Track the image coordinates $(u_{t_1}^{r}, v_{t_1}^{r}), \ldots, (u_{t_{2m}}^{r}, v_{t_{2m}}^{r})$ from frame r back to frame $r-1$, forming the image coordinates $(u_{t_1}^{r-1}, v_{t_1}^{r-1}), \ldots, \ (u_{t_{2m}}^{r-1}, v_{t_{2m}}^{r-1})$.

Step 3 For each break point in frame $r-1$, measure the Euclidean distance between the location of the break point and the location the break point is tracked to in Step 2, i.e., for the q^{th} breakpoint determine

$||(u_{t_q}^{r-1}, v_{t_q}^{r-1}) - (u_{b_q}^{r-1}, v_{b_q}^{r-1})||$. If a curve's break point is not tracked to its original location then the curve is removed.

Step 4 The remaining tracked points are assigned as break points and the curve fitting algorithm of Section 3.2 is computed for frame r, outputting the 3-D location of curves with respect to the r^{th} body frame.

Step 5 Verify that the 3-D curve structure matches between frame r and frame $r-1$ (see Figure 3.5). We verify this by performing two Mahalanobis distance tests. The first Mahalanobis distance test verifies the distance between all sequential control points with the estimated covariance of $\mathbf{d}_{l,l+1}^{r-1}$ and sample $\mathbf{d}_{l,l+1}^{r}$. The second Mahalanobis distance test verifies the distance between the start and end point of the curve using the estimated covariance of $\mathbf{d}_{0,3}^{r-1}$ and sample $\mathbf{d}_{0,3}^{r}$. Note, in the second Mahalonobis distance test, we include a max threshold tolerance ρ_s. If a curve changes shape by more than ρ_s, it fails the second Mahalonobis distance test. The selection of ρ_s is based on the precision of the stereo camera. The purpose of ρ_s is simply to provide a fail safe when it is obvious the KLT algorithm tracked feature points incorrectly. The size of ρ_s is discussed in Section 4.5.

Step 6 If the curve fails the second Mahalonobis distance test in Step 5, the curve is removed from the state. If the curve passes the second Mahalanobis distance test but not the first, the curve is treated as a linear Bézier curve.

3.3.2 Adding Curves in the Image Plane

Curves are added so that their lengths are approximately equivalent. We add a curve when the end control point of the currently tracked curve is about to leave the camera's FOV. This illustrated in Figure 3.6a by the tracked curve crossing the the yellow line L_e. When this occurs, a desired control point is set approximately at the top of the camera's FOV and at the start control point of the currently tracked curve (see Figure 3.6a). In the initial frame or when a particular side of the boundary is empty of curves (a particular side may be empty of curves due to tracking failures), we arbitrarily set

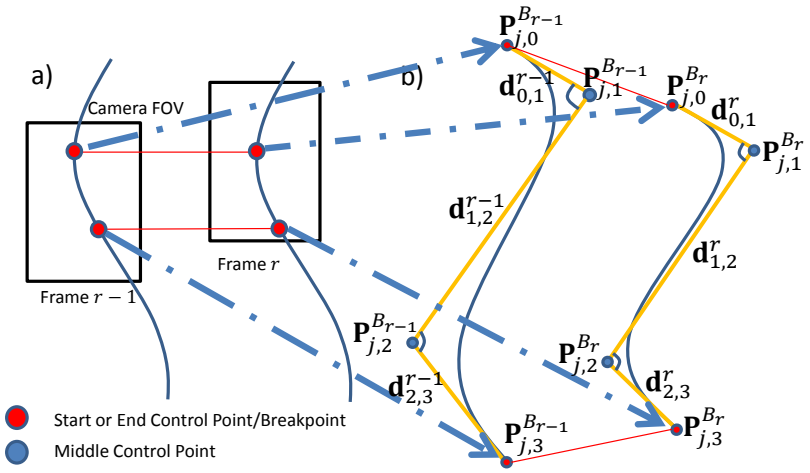


Figure 3.5: a) Tracking endpoints of a curve between two chronologically sequential frames. b) Comparing the structure of matched curves, this particular example demonstrates a tracking failure.

a desired control point approximately at the top of the camera's FOV, the bottom of the camera's FOV, and at half the arc length of the path boundary. These desired control points represent a start or end point of a curve. A region of interest is selected around the desired control points, and the Shi-Tomasi corner detector [109] is used to find good features to track in this region. Corner points whose distance exceeds a threshold ρ_k from the path boundary are rejected. Among the remaining corner points in a single region, the point with the strongest corner feature is selected as the new start or end control point (see Figure 3.6b). The size of the region of interest ρ_w and the parameter ρ_k are discussed in Section 4.5.

3.3.3 Automatic Correction of the Polynomial Curve Order

With the newly determined start or end points (determined in Section 3.3.2), we are ready to determine the polynomial order of these newly added curves. Determining the polynomial order is necessary for two reasons: First, when adding curves to the boundary of the path, care must be taken to avoid over-fitting the path by interpolating a higher-order curve to the boundary when only a lower-order curve is required. Otherwise, it is likely that the fitted 3-D control points will not remain static between image frames, causing localization errors. Second, we need a way to determine when to split these

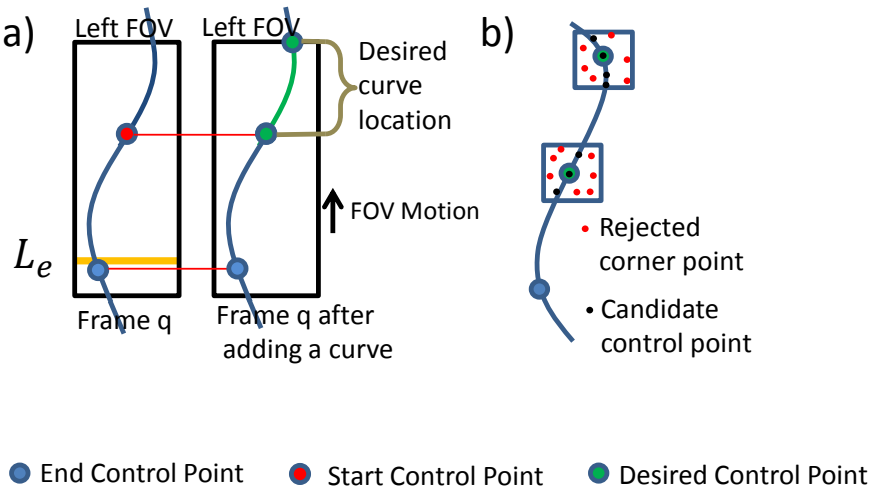


Figure 3.6: Events triggering the addition of a curve to the state and their resulting addition. A curve is added when the the currently tracked curve is about to leave the camera's FOV, see Figure 3.6a. When this occurs, a region of interest is set around the location of desired control points, and the Shi-Tomasi corner detector [109] is used to find good features to track in these regions see Figure 3.6b (the Shi-Tomasi corner detector allows the KLT tracking algorithm to track start/end control points with greater accuracy). The point with the strongest corner feature in this region that is close enough to the boundary of the path is selected as the new start or end control point.

newly added curves when the boundary of the path is not sufficiently smooth (see Figure 3.7). This section explains how to accomplish both of these tasks. To determine the order of a curve, let $\mathbf{P}^L_{j,0}$ $\mathbf{P}^L_{j,3}$ be the start and end points of a newly added curve found in Section 3.3.2. Additionally, let $B_c = \{\mathbf{P}^L_{j,0}, \ldots, \mathbf{P}^L_{j,3}\} \subset B$ represent the image coordinates of the path boundary between $\mathbf{P}^L_{j,0}$ and $\mathbf{P}^L_{j,3}$. Starting with a first-order curve $\mathcal{O}_j = 1$, the following steps are taken to determine the curve's order:

Step 1 Given $\mathcal{O}_j$, interpolate a Bézier curve of order $\mathcal{O}_j$ to the boundary of the path between the newly added control points $\mathbf{P}^L_{j,0}$ $\mathbf{P}^L_{j,3}$.

Step 2 Using the Shapiro-Wilk test [110], check that the residuals of the interpolated curve from Step 1 are normally distributed. If the residuals are normally distributed, the curve order has been determined. If the residuals are not normally distributed, increase the curve order by one, i.e., $\mathcal{O}_j = \mathcal{O}_j + 1$. Note, in this step, we also include

(a) A path with sharp corners.

(b) Curves automatically split and interpolated to the path.

Figure 3.7: Figure 3.7a demonstrates a path that contains sharp corners and is not sufficiently smooth for a single cubic Bézier curve. When this occurs, the Curve SLAM algorithm will automatically split the boundary of the path, see Figure 3.7b.

a minimum threshold tolerance ρ_p to avoid over-splitting the path boundary (splitting is discussed in Step 4 of this section). As long as the maximum residual of the interpolated curve from Step 1 is below ρ_p, the order of the curve has been determined. We discuss the parameter ρ_p in Section 4.5.

Step 3 Repeat Step 1–Step 2, progressing from a first-order curve to a third-order curve. If the curve is not third-order, proceed to Step 4.

Step 4 If the curve is not third-order, we split B_c at the point on B_c where the residual is a maximum, and designate this split point (u_s, v_s) as a new break point. This forms two sets of data points $B_1 = \{\mathbf{P}^L_{j,0}, \ldots, (u^r_s, v^r_s)\}$ and $B_2 = \{(u^r_s, v^r_s), \ldots, \mathbf{P}^L_{j,3}\}$, where B_1 consists of all the ordered points before the break point, and B_2 consists of all the ordered points after the break point.

Step 5 Repeat Step 1–Step 3 recursively on the data points in B_1 and B_2 until the residuals are normally distributed.

The start and end points of curves determined from this process are used as as break points in Section 3.2.2.

3.3.4 SLAM Estimator State and Sensors

We estimate the camera pose, linear velocity, and location of curve control points with an extended Kalman filter (EKF). We define the variable $\boldsymbol{P}$ as

the error covariance matrix, and the variable $\mathbf{x}$ as the filter state:

$$\mathbf{x} \triangleq [\mathbf{p}^{\mathcal{W}}, \mathbf{v}^{\mathcal{B}}, \boldsymbol{\Theta}, \mathbf{b}_a, \mathbf{b}_g, (\mathbf{C}_1^{\mathcal{W}})^\top, \ldots, (\mathbf{C}_N^{\mathcal{W}})^\top]^\top$$

where the variables $\mathbf{p}^{\mathcal{W}}$, $\mathbf{v}^{\mathcal{B}}$, and $\boldsymbol{\Theta}$ are defined in Table 3.1. The variables $\mathbf{b}_a \in \mathbb{R}^3$ and $\mathbf{b}_g \in \mathbb{R}^3$ represent the accelerometer bias and gyroscope bias, respectively.

An image of the sensors used to collect experimental data for Curve SLAM is shown in Figure 3.8. In addition to a stereo camera, our hardware is equipped with a Novatel SPAN-IGM-A1 GNSS inertial navigation system equipped with a GPS and an Analog Devices ADIS 16488 IMU consisting of an accelerometer and gyroscope. The GPS is used only for ground truth. All hardware has been calibrated so that measurements can be transformed to the body frame of the left camera [111–113]. The accelerometer and gyroscope are used to propagate the state equations in the prediction step of the EKF, where the gyroscope bias and accelerometer bias are both propagated as a random walk. The gyroscope measurement $\boldsymbol{\omega}_k^m$ at time step k is given by $\boldsymbol{\omega}_k^m = \boldsymbol{\omega}_k + \boldsymbol{\omega}_k^n + \mathbf{b}_g$, where $\boldsymbol{\omega}_k$ is the true angular rate of the camera, and $\boldsymbol{\omega}_k^n$ represents gyroscope noise. The accelerometer measurement $\mathbf{a}_k^m$ at time step k is given by $\mathbf{a}_k^m = \dot{\mathbf{v}}^{\mathcal{B}} + \boldsymbol{\omega}_k \times \mathbf{v}^{\mathcal{B}} - \boldsymbol{R}_{\mathcal{BW}}(\boldsymbol{\Theta})\mathbf{g}^{\mathcal{W}} + \mathbf{a}_k^n + \mathbf{b}_a$, where $\mathbf{a}_k^n$ represents accelerometer noise. Letting $\mathbf{a}_k = \dot{\mathbf{v}}^{\mathcal{B}} + \boldsymbol{\omega}_k \times \mathbf{v}^{\mathcal{B}} - \boldsymbol{R}_{\mathcal{BW}}(\boldsymbol{\Theta})\mathbf{g}^{\mathcal{W}}$, we define $\mathbf{u}_k = [\mathbf{a}_k^\top \ \boldsymbol{\omega}_k^\top]^\top$ as the input vector, and $\mathbf{w}_k = [\mathbf{a}_k^{n\top} \ \boldsymbol{\omega}_k^{n\top} \ \mathbf{b}_k^{n,a} \ \mathbf{b}_k^{n,g}]^\top$ as the process noise with covariance matrix $\mathbf{W}$. The variables $\mathbf{b}_k^{n,a} \in \mathbb{R}^3$, $\mathbf{b}_k^{n,g} \in \mathbb{R}^3$ represent accelerometer and gyroscope bias noise, respectively. This noise comes as a result of propagating the bias states as a random walk.

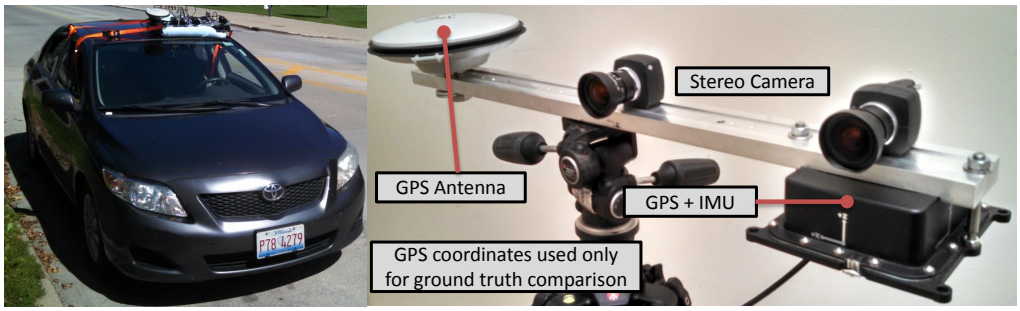


Figure 3.8: Our sensor package consists of a stereo camera, an IMU and GPS.

3.3.5 Estimator Prediction Step

We implement the EKF prediction step by feeding the data from the IMU as a dynamic model replacement, where the gyroscope bias and accelerometer bias are both propagated as a random walk [114]. To explain the process, we adopt standard EKF notation in which the subscript $k|k-1$ represents a predication step, while the subscript $k|k$ represents a measurement update. Using one-step Euler integration, the predicted state $\hat{\mathbf{x}}$ at time step k is given by $\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})\Delta t$ where

$$f(\mathbf{x}, \mathbf{u}_k, \mathbf{w}_k) = \begin{pmatrix} \boldsymbol{R}_{\mathcal{WB}}(\boldsymbol{\Theta})\mathbf{v}^{\mathcal{B}} \\ \mathbf{a}_k^m - \mathbf{b}_a - (\boldsymbol{\omega}_k^m - \mathbf{b}_g) \times \mathbf{v}^{\mathcal{B}} + \boldsymbol{R}_{\mathcal{BW}}(\boldsymbol{\Theta})\mathbf{g}^{\mathcal{W}} \\ \boldsymbol{S}(\boldsymbol{\Theta})(\boldsymbol{\omega}_k^m - \mathbf{b}_g) \\ \mathbf{0}^{3\times 1} \\ \mathbf{0}^{3\times 1} \\ \mathbf{0}^{N\times 1} \end{pmatrix} \tag{3.6}$$

The variable $\boldsymbol{R}_{\mathcal{WB}}(\Theta)$ is a rotation matrix from the body frame to the world frame, and $\boldsymbol{S}(\boldsymbol{\Theta})$ is a rotational transformation that allows the body-frame angular rates to be expressed in terms of the derivatives of the ZYX Euler angles, i.e.,

$$\boldsymbol{S}(\boldsymbol{\Theta}) = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{pmatrix} \tag{3.7}$$

The error covariance is updated as

$$\boldsymbol{P}_{k|k-1} = \mathbf{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\mathbf{F}_{k-1} + \mathbf{L}_{k-1}\mathbf{W}\mathbf{L}_{k-1}^{\top}$$

where $\mathbf{F}_k = \frac{\partial f(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, \mathbf{w}_k)}{\partial \mathbf{x}}$ and $\mathbf{L}_k = \frac{\partial f(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, \mathbf{w}_k)}{\partial \mathbf{w}}$. We compute the expressions for the Jacobians $\mathbf{F}_k$ and $\mathbf{L}_k$ symbolically offline.

3.3.6 Measurement Update

At every stereo frame, we measure m different Bézier curves $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$. These curves are related to the existing map curves by the transformation $\mathbf{C}_j^{\mathcal{B}} = \mathbf{T}_{\mathcal{BW}}(\mathbf{C}_j^{\mathcal{W}})$, where $\mathbf{T}_{\mathcal{BW}} \in SE(3)$ is the transformation that changes

the representation of a point defined in the coordinates of frame $\mathcal{W}$ to a point defined in the coordinates of frame $\mathcal{B}$. The measurement equation is given by $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k$, where

$$\mathbf{h}(\mathbf{x}) = [(\mathbf{T}_{\mathcal{B}\mathcal{W}}(\mathbf{C}_1^{\mathcal{W}}))^\top, \ldots, (\mathbf{T}_{\mathcal{B}\mathcal{W}}(\mathbf{C}_m^{\mathcal{W}}))^\top]^\top \tag{3.8}$$

The variable $\mathbf{v}_k$ represents measurement noise with covariance matrix $\mathbf{V}_k$. The measurement covariance matrix $\mathbf{V}_k$ is defined as stated in Section 3.2.3. We demonstrate how to calculate $\mathbf{V}_k$ in Section 4.5. The remaining EKF update equations are implemented as follows:

$$\begin{aligned}
\tilde{\mathbf{y}}_k &= \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) \\
\mathbf{S}_k &= \mathbf{H}_k \boldsymbol{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{V}_k \\
\mathbf{K}_k &= \boldsymbol{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{S}_k^{-1} \\
\hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \\
\boldsymbol{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \boldsymbol{P}_{k|k-1}
\end{aligned}$$

where $\mathbf{H}_k = \frac{\partial \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})}{\partial \mathbf{x}}$. We compute the expression for the Jacobian $\mathbf{H}_k$ symbolically offline.

3.3.7 Adding Curve Control Points to the Filter State

After a curve has been added in the image plane (see Section 3.3.2), the filter state needs to be updated. With the method in [18], we augment the filter state with the new curve $\mathbf{C}_{N+1}^{\mathcal{B}}$ and augment the error covariance matrix with the necessary initial cross-covariances. Letting $\mathbf{x}^a$ represent the augmented state and $\boldsymbol{P}^a$ represent the augmented error covariance, we perform this operation as follows:

$$\begin{aligned}
\mathbf{x}^a &= \mathbf{g}(\mathbf{x}, \mathbf{z}) \\
\boldsymbol{P}^a &= \mathbf{G}_{\mathbf{x}} \boldsymbol{P} \mathbf{G}_{\mathbf{x}}^\top + \mathbf{G}_{\mathbf{z}} \mathbf{V}_r \mathbf{G}_{\mathbf{z}}^\top
\end{aligned}$$

where $\mathbf{g}(\mathbf{x}, \mathbf{z}) = [\mathbf{x}^\top, \mathbf{T}_{\mathcal{W}\mathcal{B}}(\mathbf{C}_{N+1}^{\mathcal{B}})^\top]^\top$, $\mathbf{G}_{\mathbf{x}} = \frac{\partial \mathbf{g}(\mathbf{x},\mathbf{z})}{\partial \mathbf{x}}$, and $\mathbf{G}_{\mathbf{z}} = \frac{\partial \mathbf{g}(\mathbf{x},\mathbf{z})}{\partial \mathbf{z}}$. We compute the expressions for the Jacobians $\mathbf{G}_{\mathbf{z}}$ and $\mathbf{G}_{\mathbf{x}}$ symbolically offline.

3.3.8 Combining Curves

One of our main objectives is to represent long segments of a path with a small number of curves. However, because the depth measurement accuracy of a stereo camera is limited by range, our sparseness objective interferes with how accurately we are able to localize the camera. Thus, we limit the length of curve segments. To overcome this drawback, we add one additional step while mapping the environment. When the location of a curve $\mathbf{C}^{\mathcal{W}}_{m_c+1}$ is no longer contained in the camera's FOV, we attempt to combine $\mathbf{C}^{\mathcal{W}}_{m_c+1}$ with curves $\mathbf{C}^{\mathcal{W}}_{j_c} \dots \mathbf{C}^{\mathcal{W}}_{m_c}$ that were estimated prior to $\mathbf{C}^{\mathcal{W}}_{m_c+1}$. The steps to combine curves are given as follows:

Step 1 Assume $\mathbf{C}^{\mathcal{W}}_{j_c}, \dots, \mathbf{C}^{\mathcal{W}}_{m_c}$ belong to a single growing cubic Bézier curve $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}}$ (see Figure 3.9).

Step 2 Interpolate a single cubic Bézier curve to $\mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{j_c}) \dots \mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{m_c})$ and $\mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{m_c+1})$ by fixing the start control point at the start point in $\mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{j_c})$ and the end control point at the last point in $\mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{m_c+1})$. The two middle control point are determined with least squares.

Step 3a If the median of the residuals in the least squares fit is less than a threshold d, $\mathbf{C}^{\mathcal{W}}_{m_c+1}$ is added to $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}}$. The size of d is discussed in Section 4.5.

Step 3b If the median of the residuals in the least squares fit is not less than a threshold d, we start a new growing curve $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}+1}$, with $\mathbf{C}^{\mathcal{W}}_{m_c+1}$ as its only element. Meanwhile, the past curve $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}}$ is no longer growing, i.e., no curves that are subsequently estimated are added to $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}}$. Instead, they are checked for addition to curve $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}+1}$.

3.4 Where Curve SLAM Has Problems Producing Accurate SLAM Results

Curve SLAM is dependent on a stereo camera to extract depth information from the environment. As the distance from a stereo camera to a curve increases, the accuracy of a stereo depth estimate decreases. Consequently,

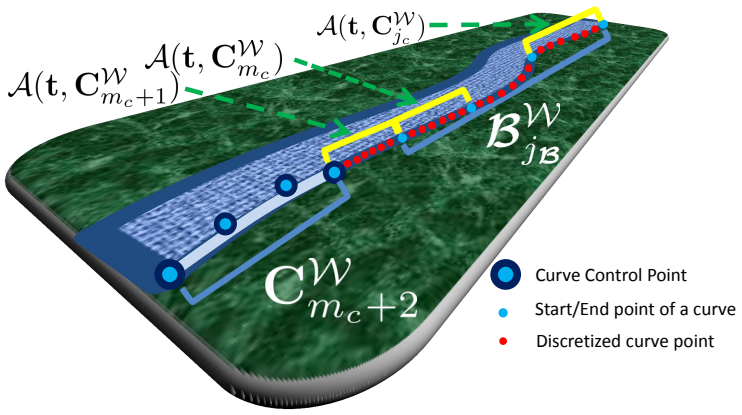


Figure 3.9: The variables used to describe the curve combining algorithm.

when the distance between the stereo camera to all imaged curves becomes large, the accuracy of Curve SLAM suffers. And, if the distance between the stereo camera to all imaged curves becomes too large (roughly eight meters for our stereo camera), Curve SLAM must rely on propagating an IMU to obtain localization results, which, in turn, will produce quickly growing localization and mapping errors. Additionally, Curve SLAM relies on an object detection algorithm to consistently find a curved object in the environment. If the object detection algorithm consistently fails, Curve SLAM must rely on an IMU to obtain localization results, causing localization and mapping errors. In the experimental results, both these problems occur and negatively affect Curve SLAM, see Chapter 4. We suggest ways to improve the accuracy of our curve triangulation method for future work (see Chapter 5).

3.5 Conclusion

In this chapter, we presented a SLAM algorithm that uses Bézier curves as landmark primitives rather than feature points. Our approach allows us to create a sparse map that recovers the outline, shape, and dimensions of unique objects in the environment. We showed how to interpolate, split, and match curves in a stereo pair. This allowed us to reconstruct the 3-D location of curves, parameterized by a small number of control points. Then, we showed how to formulate and solve the SLAM problem using Bézier curves and an extended Kalman filter. To solve the SLAM problem, we presented a data association algorithm that compares the physical dimensions

of curve landmarks between chronologically sequential stereo image frames to remove curve outliers. Additionally, we showed how to determine the polynomial order of curves, we formulated the prediction and measurement step of the extended Kalman filter, and we showed how to add curves to the SLAM state and inside the image plane. Finally, we presented a curve combining algorithm that further reduces the number of landmark states representing the map. In Chapter 4, we apply this algorithm to six different real-world settings and compare it rigorously against two state-of-the-art SLAM algorithms.

CHAPTER 4

EXPERIMENTAL RESULTS OF THE CURVE SLAM ALGORITHM

In this chapter, we compare Curve SLAM against the open keyframe-based visual-inertial SLAM algorithm (OKVIS) [3], and a stereo version of the parallel tracking and mapping (SPTAM) algorithm [2, 4]. We compare algorithms by adopting the metric proposed in [115], and by comparing the number of landmarks each algorithm requires to represent the map in each setting. We present localization and mapping results of Curve SLAM applied to the boundary of a river, yellow road lanes, the outline of a road, and a sidewalk. In these datasets we demonstrate that Curve SLAM is able to create a map that recovers the outline, shape, and dimensions of these objects. Furthermore, we demonstrate that Curve SLAM reduces the required number of landmark primitives by several orders of magnitude relative to OKVIS and SPTAM. We conclude this chapter by explaining how we selected the parameters of the Curve SLAM algorithm.

The remainder of this chapter proceeds as follows: Section 4.1 explains how we obtained the datasets used to compare the algorithms in this chapter, and describes the sensor packages used to collect the datasets. Section 4.2 describes the metric used to compare the localization accuracy of the different algorithms. Section 4.3 compares the localization accuracy of the different algorithms and compares the number of landmarks each algorithm used to represent each setting. Section 4.4 presents mapping results of Curve SLAM. Section 4.5 describes how to select the parameters of the Curve SLAM algorithm. Section 4.6 provides a conclusion to this chapter.

4.1 Sensor Characteristics and a Description of the Datasets

To compare algorithms, we used six real-world datasets of various environments, these datasets were captured at different times of the day under varying environmental conditions. The first three datasets contain images of sidewalks that were obtained from a local park (the sidewalk provided curves for the Curve SLAM algorithm), the fourth dataset contains images of yellow road lanes that were obtained while driving on a nearby road (the yellow road lanes provided curves for the Curve SLAM algorithm), the fifth dataset is a sequence taken from the KITTI dataset [116], and contains images of a road (the road provided curves for the Curve SLAM algorithm), and the sixth dataset is a sequence taken from the visual-inertial canoe dataset [90], and contains images of a river with specular reflections (the shoreline of the river provided curves for the Curve SLAM algorithm).

Sample images of the six datasets are shown in Figures 4.1–4.6. During the first four datasets, the stereo camera had a 36 cm baseline with 3.5 mm focal length lens. Images were sampled at 20 Hz, and IMU measurements were sampled at 100 Hz. During the fifth dataset, the stereo camera had a 54 cm baseline with 4 mm focal length lens. Images were sampled at 10 Hz, and IMU measurements were sampled at 10 Hz. During the sixth dataset, the stereo camera had a 60 cm baseline with 3.5 mm focal length lens. Images were sampled at 20 Hz, and IMU measurements were sampled at 200 Hz. During the first four datasets, we segmented the boundary of the path by thresholding the HSV color channels of an image (our approach is described in Section 3.2.1). During the fifth dataset, we segmented the boundary of the path with a pre-trained convolutional neural network designed to detect image pixels that represent the road (see Section 3.2.1). During the sixth dataset, we segmented the boundary of the river with the water detection algorithm described in Chapter 2.

The first three datasets were collected from a local park at different times of the day under different lighting and weather conditions. All the data in the park were obtained by mounting the sensors onto a cart that was pushed by a person. The edges of a long curved sidewalk in this local park were used as curves in the Curve SLAM algorithm. The first dataset, DS1, was collected at dawn under clear weather conditions. The experiment lasted

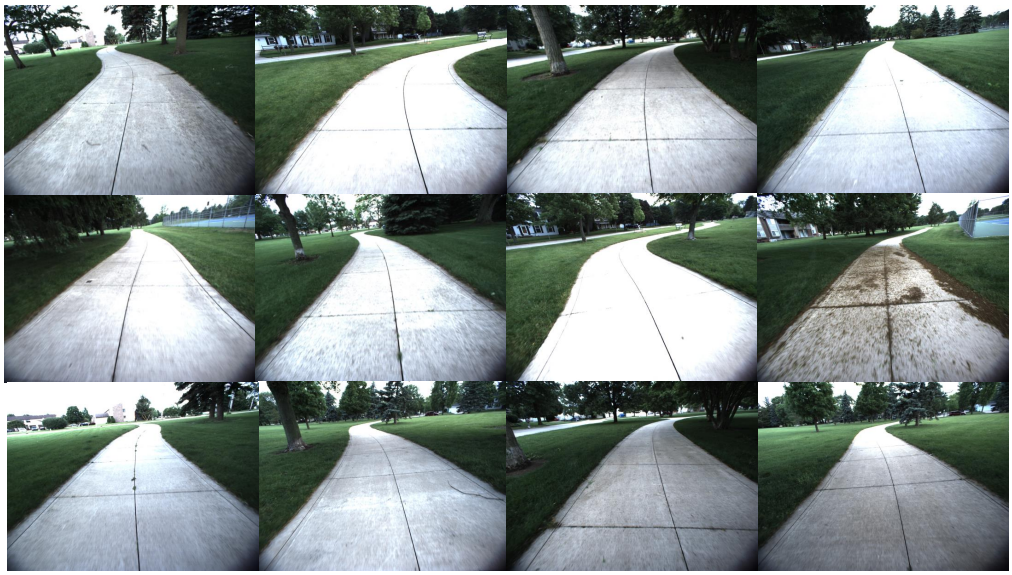


Figure 4.1: Sample images of DS1. The edges of the curved sidewalk provided curves for the Curve SLAM algorithm. Collection time: Dawn. Weather: mostly cloudy and clear. Length: 252.2 meters. Duration: 138.5 seconds.

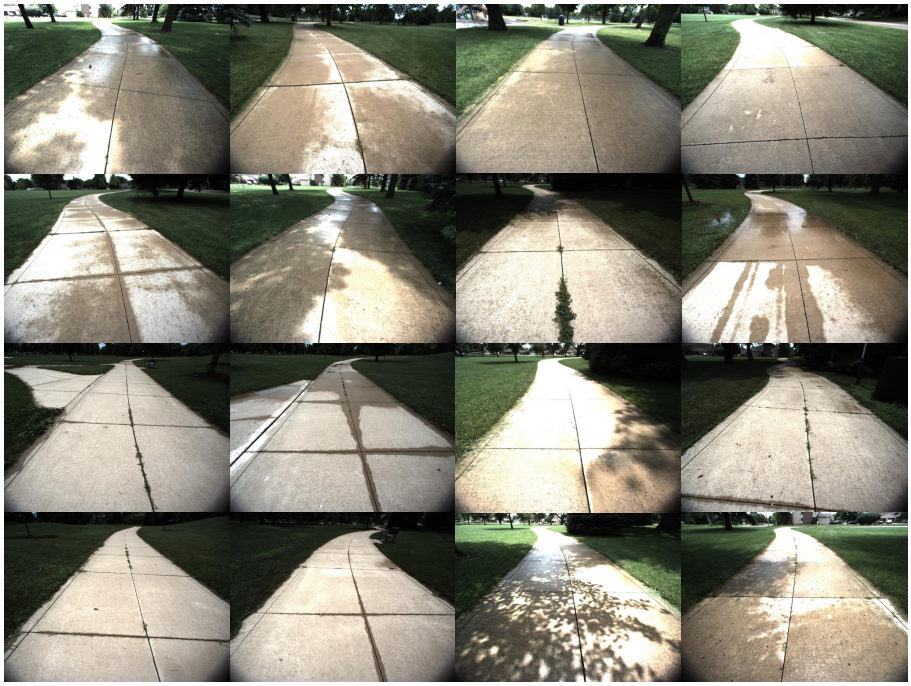


Figure 4.2: Sample images of DS2. The edges of the curved sidewalk provided curves for the Curve SLAM algorithm. Collection time: 2PM. Weather: Part of DS2 was collected during a light rainstorm under cloudy conditions. The other part was collected under mostly sunny conditions and clear weather. Length: 375 meters. Duration: 195.85 seconds.

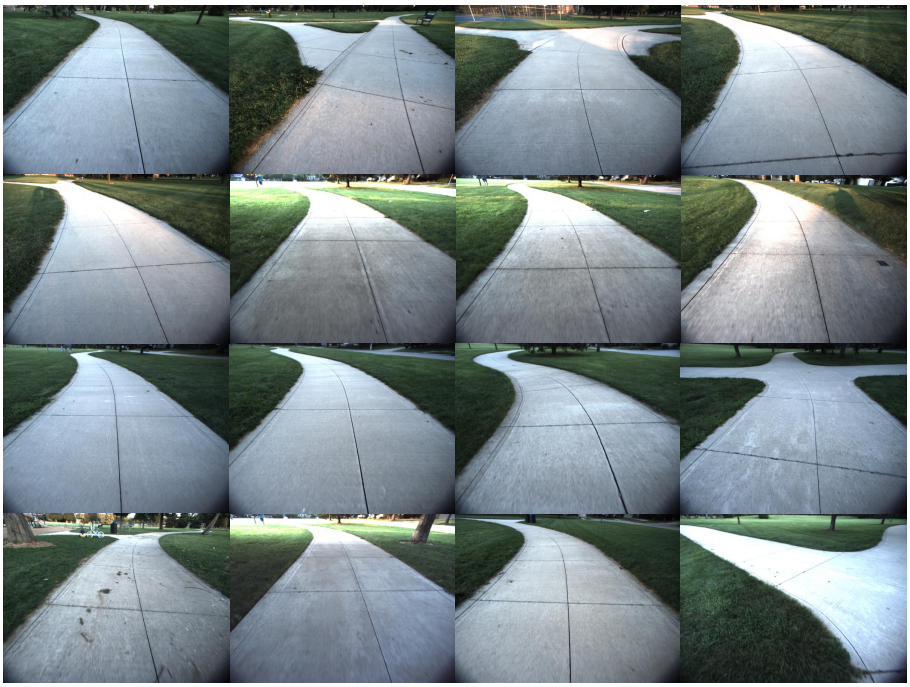


Figure 4.3: Sample images of DS3. The edges of the curved sidewalk provided curves for the Curve SLAM algorithm. Collection time: roughly one hour prior to sunset. Weather: mostly sunny and clear. Length: 603.4 meters. Duration: 437.55 seconds.

Figure 4.4: Sample images of DS4. Yellow road lanes provided curves for the Curve SLAM algorithm. Collection time: roughly 10:30 AM. Weather: mostly cloudy and clear. Length: 230 meters. Duration: 70 seconds.

Figure 4.5: Sample images of DS5. The edges of the road provided curves for the Curve SLAM algorithm. Collection time: unknown. Weather: sunny and clear. Length: 435 meters. Duration: 40 seconds.

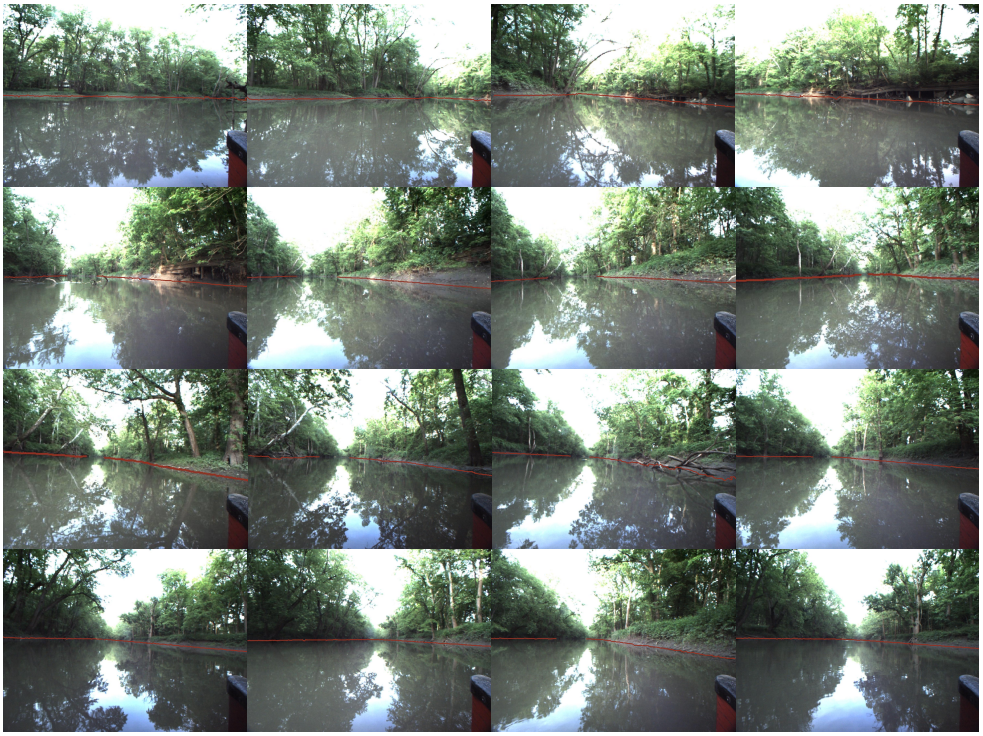


Figure 4.6: Sample images of DS6. The boundary of the river provided curves for the Curve SLAM algorithm. Collection time: unknown. Weather: sunny and clear. Length: 170 meters. Duration: 172.5 seconds.

roughly 138.5 seconds. During this time, our sensors traveled approximately 252.2 meters. The second dataset, DS2, was collected in the mid-afternoon, roughly 2 PM. Part of DS2 was collected during a light rainstorm under cloudy conditions, while the other part of DS2 was collected immediately following this light rainstorm under mostly sunny conditions. DS2 lasted roughly 195.85 seconds. During this time, our sensors traveled approximately 375 meters. DS3 was collected about an hour prior to sunset on a mostly sunny day with clear weather conditions. DS3 lasted roughly 437.55 seconds. During this time, our sensors traveled approximately 603.4 meters. The fourth dataset DS4 was collected by strapping the sensors onto a car and driving on a nearby road. Yellow road lanes were used as curves. DS4 was collected in the morning hours under mostly cloudy conditions. DS4 lasted roughly 70 seconds. During this time, our sensors traveled approximately 230 meters. The fifth dataset DS5 is a sequence obtained from the KITTI dataset [116], and was collected with the sensors attached to the top of a car while driving in a residential area. The edges of a road were used as curves. This particular sequence from the KITTI dataset was selected due to the presence of curves in the environment and the high number of occlusions covering the road. DS5 was collected under sunny conditions. DS5 lasted roughly 40 seconds. During this time, the sensors traveled approximately 435 meters. The sixth dataset DS6 is a sequence obtained from the visual-inertial canoe dataset [90], and was collected with the sensors attached to the front of a canoe as it navigated a river setting containing specular reflections. The shoreline of a river provided curves for Curve SLAM. This particular sequence from the visual-inertial canoe dataset was selected because the shoreline was almost in range of the stereo camera. The closest distance between the stereo camera to the shoreline across the entire dataset was approximately 10 meters. DS6 was collected under sunny conditions. DS6 lasted roughly 172.5 seconds. During this time, the sensors traveled approximately 170 meters.

For a ground-truth comparison of all the datasets, we have included a precise GPS-INS track that is time-synchronized with the stereo camera and IMU measurements. The GPS-INS track includes a ground truth for the location and attitude of the sensor platform. Note, the GPS is only used to provide ground-truth information.

4.2 Evaluation Metric

To compare Curve SLAM against OKVIS and SPTAM, we extend the metric proposed in [115]. Letting d represent the distance traveled between frame i_e and frame j_e, we compare algorithms with the following metric:

$$\Delta\mathbf{T}(d) = \mathbf{T}_{0j_e}^{-1}\mathbf{T}_{0i_e}\hat{\mathbf{T}}_{0i_e}^{-1}\hat{\mathbf{T}}_{0j_e} \tag{4.1}$$

where $\Delta\mathbf{T}(d) \in SE(3)$ represents the error between an estimated pose $\hat{\mathbf{T}}_{i_e j_e} \in SE(3)$ and ground-truth pose $\mathbf{T}_{i_e j_e} \in SE(3)$, and the subscript 0 represents the initial frame. To compute d, we sum the Euclidean distance between all temporally sequential GPS ground-truth coordinates between frame i_e and frame j_e. For a fixed value of d, we compute $\Delta\mathbf{T}(d)$ between numerous poses i_e and j_e in order to obtain error statistics for the orientation error and translation error.

4.3 Localization Results

Figures 4.7-4.12 display the median translation error $\tilde{m}_t$ plotted between the fifth e_{t05} and ninety-fifth e_{t95} percentile for translation error as well as the median orientation error $\tilde{m}_o$ plotted between the fifth e_{o05} and ninety-fifth e_{o95} percentile for orientation error for five fixed values of d. Tables 4.1-4.6 summarize these results. In DS1, DS3, and DS4, Curve SLAM is more accurate or provides less variance in its motion estimates than SPTAM and OKVIS because these environments lacked distinguishable feature points (sample images of the datasets are shown in Figures 4.1–4.6). Consequently, SPTAM and OKVIS occasionally failed to track feature points robustly, and the ability of SPTAM and OKVIS to accurately localize and map these settings declined. In fact, at one point during DS3, OKVIS reported a failure to track feature points. Similarly, while operating in DS3 and DS4, SPTAM occasionally failed to track feature points correctly, causing major localization and mapping errors. In DS1, we were unable to apply SPTAM because SPTAM completely failed to track feature points. A reason that OKVIS operates in DS1, DS3, and DS4 better than SPTAM is because OKVIS relies on an IMU to localize its position, preventing tracking failures from causing major localization errors. However, even in these settings, the performance

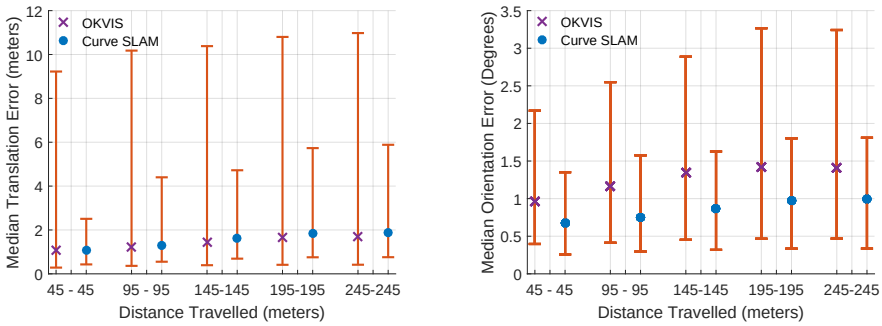


Figure 4.7: The median (x or dot) plotted between the fifth (bottom horizontal line), and ninety-fifth (top horizontal line) percentile for translation error and orientation error of Curve SLAM and OKVIS for DS1.

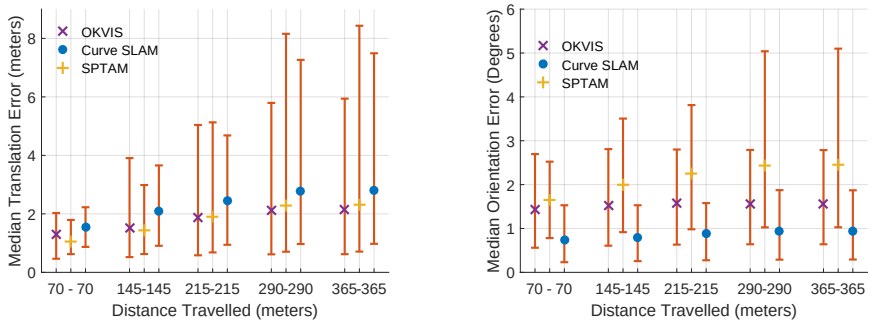


Figure 4.8: The median (x, dot, or plus sign) plotted between the fifth (bottom horizontal line), and ninety-fifth (top horizontal line) percentile for translation error and orientation error of Curve SLAM, OKVIS, and SPTAM for DS2.

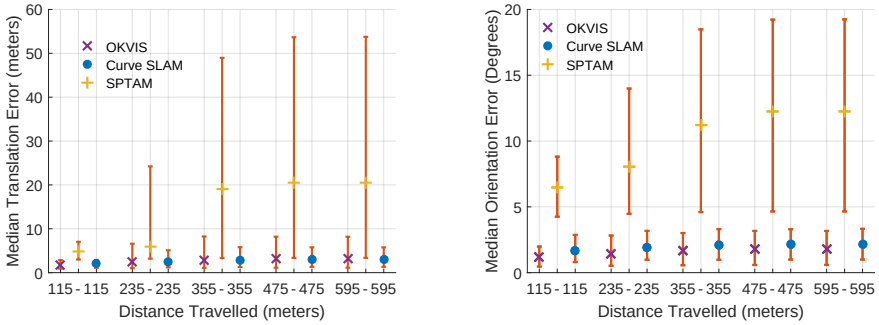


Figure 4.9: The median (x, dot, or plus sign) plotted between the fifth (bottom horizontal line), and ninety-fifth (top horizontal line) percentile for translation error and orientation error of Curve SLAM, OKVIS, and SPTAM for DS3.

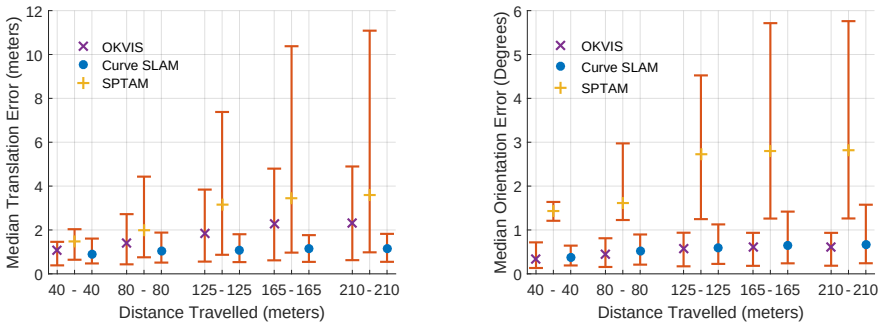


Figure 4.10: The median (x, dot, or plus sign) plotted between the fifth (bottom horizontal line), and ninety-fifth (top horizontal line) percentile for translation error and orientation error of Curve SLAM, OKVIS, and SPTAM for DS4.

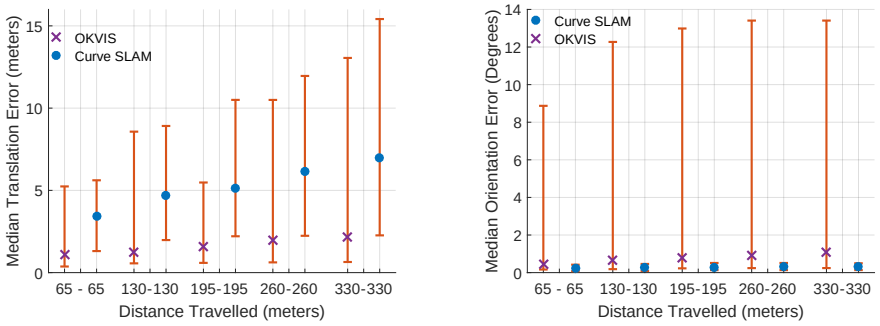


Figure 4.11: The median (x or dot) plotted between the fifth (bottom horizontal line), and ninety-fifth (top horizontal line) percentile for translation error and orientation error of Curve SLAM and OKVIS for DS5.

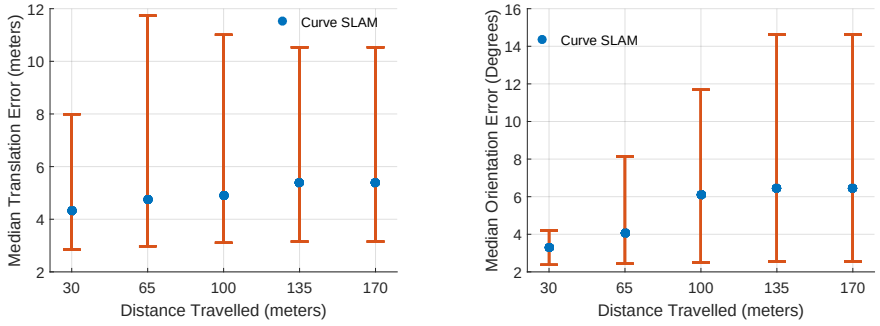


Figure 4.12: The median (x or dot) plotted between the fifth (bottom horizontal line), and ninety-fifth (top horizontal line) percentile for translation error and orientation error of Curve SLAM for DS6.

of OKVIS diminishes. In contrast to OKVIS and SPTAM, Curve SLAM does not depend on tracking large numbers of feature points, allowing it to operate

Table 4.1: Algorithm Comparison for DS1

DS1 was collected at dawn under clear weather conditions. We did not apply SPTAM to DS1 because we were unable to track a sufficient number of feature points.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	NA	160 control points	279690
Distance Traveled: 45 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	1.09 m, 0.67°	1.09 m, 0.97°
e_{t05}, e_{o05}	NA	0.43 m, 0.26°	0.28 m, 0.40°
e_{t95}, e_{o95}	NA	2.51 m, 1.35°	9.22 m, 2.17°
Max error (trans., att.)	NA	2.73 m, 1.75°	10.61 m, 3.56°
$\tilde{m}_t/d$	NA	2.42 %	2.42 %
Distance Traveled: 95 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	1.29 m, 0.76°	1.23 m, 1.16°
e_{t05}, e_{o05}	NA	0.55 m, 0.30°	0.36 m, 0.41°
e_{t95}, e_{o95}	NA	4.40 m, 1.58°	10.18 m, 2.55°
Max error (trans., att.)	NA	5.43 m, 1.90°	10.98 m, 3.91°
$\tilde{m}_t/d$	NA	1.36 %	1.30 %
Distance Traveled: 145 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	1.64 m, 0.86°	1.44 m, 1.34°
e_{t05}, e_{o05}	NA	0.69 m, 0.32°	0.39 m, 0.45°
e_{t95}, e_{o95}	NA	4.72 m, 1.63°	10.38 m, 2.89°
Max error (trans., att.)	NA	6.91 m, 2.10°	11.55 m, 4.14°
$\tilde{m}_t/d$	NA	1.13 %	0.99 %
Distance Traveled: 195 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	1.83 m, 0.97°	1.65 m, 1.42°
e_{t05}, e_{o05}	NA	0.75 m, 0.33°	0.41 m, 0.47°
e_{t95}, e_{o95}	NA	5.73 m, 1.80°	10.80 m, 3.27°
Max error (trans., att.)	NA	6.91 m, 2.29°	12.35 m, 4.76°
$\tilde{m}_t/d$	NA	0.94 %	0.85 %
Distance Traveled: 245 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	1.86 m, 0.99°	1.68 m, 1.41°
e_{t05}, e_{o05}	NA	0.76 m, 0.34°	0.41 m, 0.47°
e_{t95}, e_{o95}	NA	5.88 m, 1.81°	10.98 m, 3.24°
Max error (trans., att.)	NA	6.91 m, 2.29°	12.73 m, 4.76°
$\tilde{m}_t/d$	NA	0.76 %	0.68 %

in these settings better than OKVIS or SPTAM (for further discussion of Curve SLAM's dependence on tracking landmarks see Section 3.3.1).

DS2 contained better conditions for detecting, extracting, and tracking feature points, thus for DS2, Curve SLAM is slightly less accurate than SPTAM and OKVIS. However, the maximum error estimates and attitude estimates provided by Curve SLAM are better than OKVIS and SPTAM over all distances traveled in DS2. In DS5, our algorithm is less accurate than OKVIS for a few reasons: DS5 contained better conditions for detecting, extracting, and tracking feature points. Additionally, in DS5 we relied on a pre-trained convolutional neural network to detect the road. During short periods of operation, Curve SLAM was unable to detect the road, and our algorithm was forced to rely solely on propagating the IMU. Finally, in DS5, the camera to road distance was large, causing larger inaccuracies in our stereo triangulation method. However, in settings similar to DS2 and DS5 where large numbers of feature points are readily available for tracking, we

Table 4.2: Algorithm Comparison for DS2

DS2 was collected at 2 PM with light rain and mostly sunny conditions.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	138403	220 control points	345340
Distance Traveled: 70 Meters			
$\tilde{m}_t, \tilde{m}_o$	1.06 m, 1.64°	1.54 m, 0.74°	1.30 m, 1.44°
e_{t05}, e_{o05}	0.62 m, 0.78°	0.87 m, 0.23°	0.46 m, 0.56°
e_{t95}, e_{o95}	1.79 m, 2.52°	2.23 m, 1.53°	2.03 m, 2.70°
Max error (trans., att.)	2.58 m, 3.29°	2.57 m, 1.86°	4.68 m, 4.97°
$\tilde{m}_t/d$	1.51 %	2.19 %	1.85 %
Distance Traveled: 145 Meters			
$\tilde{m}_t, \tilde{m}_o$	1.43 m, 1.99°	2.09 m, 0.80°	1.52 m, 1.51°
e_{t05}, e_{o05}	0.63 m, 0.92°	0.91 m, 0.26°	0.52 m, 0.61°
e_{t95}, e_{o95}	2.99 m, 3.51°	3.66 m, 1.53°	3.91 m, 2.81°
Max error (trans., att.)	4.91 m, 4.19°	4.24 m, 1.89°	9.35 m, 4.97°
$\tilde{m}_t/d$	0.98 %	1.44 %	1.04 %
Distance Traveled: 215 Meters			
$\tilde{m}_t, \tilde{m}_o$	1.90 m, 2.26°	2.44 m, 0.88°	1.88 m, 1.57°
e_{t05}, e_{o05}	0.68 m, 0.98°	0.94 m, 0.28°	0.58 m, 0.63°
e_{t95}, e_{o95}	5.13 m, 3.81°	4.68 m, 1.58°	5.04 m, 2.80°
Max error (trans., att.)	8.41 m, 4.87°	6.11 m, 2.06°	10.51 m, 4.97°
$\tilde{m}_t/d$	0.88 %	1.13 %	0.88 %
Distance Traveled: 290 Meters			
$\tilde{m}_t, \tilde{m}_o$	2.29 m, 2.44°	2.77 m, 0.95°	2.11 m, 1.57°
e_{t05}, e_{o05}	0.71 m, 1.03°	0.97 m, 0.29°	0.62 m, 0.64°
e_{t95}, e_{o95}	8.16 m, 5.04°	7.27 m, 1.87°	5.79 m, 2.79°
Max error (trans., att.)	12.58 m, 6.78°	9.07 m, 2.76°	14.71 m, 4.97°
$\tilde{m}_t/d$	0.79 %	0.96 %	0.73 %
Distance Traveled: 365 Meters			
$\tilde{m}_t, \tilde{m}_o$	2.31 m, 2.45°	2.80 m, 0.94°	2.16 m, 1.56°
e_{t05}, e_{o05}	0.71 m, 1.03°	0.98 m, 0.29°	0.62 m, 0.64°
e_{t95}, e_{o95}	8.43 m, 5.10°	7.49 m, 1.87°	5.94 m, 2.79°
Max error (trans., att.)	15.24 m, 6.78°	9.63 m, 2.76°	19.44 m, 4.97°
$\tilde{m}_t/d$	0.63 %	0.77 %	0.59 %

expect a robust point-based feature detector/extractor tracking algorithm to provide a better motion estimate than Curve SLAM which tracks a limited number of landmark curves between temporally sequential stereo frames.

DS6 is a particularly hostile environment for SLAM. In DS6, the shoreline was out of range of our stereo camera's depth sensing capabilities; the closest distance between the stereo camera to the shoreline over this entire dataset was approximately 10 meters. To mitigate this problem, we added absolute measurements of roll ϕ, pitch θ, and height h to our SLAM estimator's measurement update step. We obtained these measurements directly from the stereo camera. We showed how to extract these measurements from the normal and height of the water's surface in Section 2.2.7. All the results for DS6 presented in this chapter include these measurements. Notice that even with these precise measurements of ϕ, θ, and h, our ability to localize and map this setting suffered due to the large range between the shoreline and the stereo camera. In fact, when the distance between the stereo camera and shoreline is too large (approximatley 12 meters for our stereo camera

Table 4.3: Algorithm Comparison for DS3

DS3 was collected near sunset under mostly sunny conditions.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	151867	348 control points	495874
Distance Traveled: 115 Meters			
$\tilde{m}_t, \tilde{m}_o$	4.79 m, 6.49°	2.08 m, 1.68°	1.66 m, 1.19°
e_{t05}, e_{o05}	3.00 m, 4.25°	1.11 m, 0.80°	0.90 m, 0.46°
e_{t95}, e_{o95}	7.03 m, 8.82°	2.90 m, 2.87°	2.86 m, 1.98°
Max error (trans., att.)	22.27 m, 29.44°	3.19 m, 3.47°	6.57 m, 4.02°
$\tilde{m}_t/d$	4.17 %	1.81 %	1.44 %
Distance Traveled: 235 Meters			
$\tilde{m}_t, \tilde{m}_o$	5.91 m, 8.06°	2.44 m, 1.94°	2.40 m, 1.44°
e_{t05}, e_{o05}	3.19 m, 4.47°	1.21 m, 0.98°	1.03 m, 0.52°
e_{t95}, e_{o95}	24.23 m, 13.99°	5.11 m, 3.17°	6.59 m, 2.82°
Max error (trans., att.)	49.77 m, 33.60°	5.63 m, 3.87°	12.62 m, 4.12°
$\tilde{m}_t/d$	2.52 %	1.04 %	1.02 %
Distance Traveled: 355 Meters			
$\tilde{m}_t, \tilde{m}_o$	19.10 m, 11.21°	2.88 m, 2.07°	2.82 m, 1.68°
e_{t05}, e_{o05}	3.31 m, 4.60°	1.28 m, 0.98°	1.09 m, 0.56°
e_{t95}, e_{o95}	48.97 m, 18.49°	5.81 m, 3.31°	8.25 m, 3.01°
Max error (trans., att.)	54.36 m, 33.60°	7.80 m, 3.87°	12.62 m, 4.12°
$\tilde{m}_t/d$	5.38 %	0.81 %	0.80 %
Distance Traveled: 475 Meters			
$\tilde{m}_t, \tilde{m}_o$	20.51 m, 12.25°	2.98 m, 2.15°	3.23 m, 1.81°
e_{t05}, e_{o05}	3.37 m, 4.65°	1.32 m, 0.99°	1.11 m, 0.58°
e_{t95}, e_{o95}	53.66 m, 19.22°	5.77 m, 3.30°	8.20 m, 3.17°
Max error (trans., att.)	55.86 m, 33.60°	7.80 m, 3.87°	12.62 m, 4.12°
$\tilde{m}_t/d$	4.32 %	0.63 %	0.68 %
Distance Traveled: 595 Meters			
$\tilde{m}_t, \tilde{m}_o$	20.56 m, 12.27°	2.99 m, 2.15°	3.25 m, 1.82°
e_{t05}, e_{o05}	3.38 m, 4.65°	1.33 m, 1.00°	1.11 m, 0.59°
e_{t95}, e_{o95}	53.76 m, 19.25°	5.77 m, 3.33°	8.19 m, 3.17°
Max error (trans., att.)	55.90 m, 33.60°	7.80 m, 3.97°	12.62 m, 4.12°
$\tilde{m}_t/d$	3.45 %	0.50 %	0.55 %

configuration), we are forced to rely solely on propagating the IMU which causes large inaccuracies in our SLAM result. This is main reason the localization and mapping results for this setting are worse than the other five datasets. Finally, we also note that during a brief period of operation our shoreline detector was unable to consistently detect the river because our river detection algorithm was unable to find a sufficient number of matching symmetric pairs or because our river detection algorithm could not correctly match dense stereo correspondences due to the stereo pair failing to satisfy the brightness constancy assumption, or occlusions blocking one of the camera's FOV. Consequently, our algorithm was again forced to rely solely on propagating the IMU, which also caused minor localization and mapping errors.

Table 4.4: Algorithm Comparison for DS4

DS4 was collected at 10 AM under mostly cloudy conditions and clear weather.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	55577	92 control points	17805
Distance Traveled: 40 Meters			
$\tilde{m}_t, \tilde{m}_o$	1.47 m, 1.43°	0.90 m, 0.37°	1.08 m, 0.34°
e_{t05}, e_{o05}	0.64 m, 1.21°	0.47 m, 0.19°	0.39 m, 0.13°
e_{t95}, e_{o95}	2.03 m, 1.64°	1.61 m, 0.64°	1.46 m, 0.72°
Max error (trans., att.)	2.09 m, 1.69°	1.74 m, 0.74°	1.55 m, 0.97°
$\tilde{m}_t/d$	3.66 %	2.25 %	2.70 %
Distance Traveled: 80 Meters			
$\tilde{m}_t, \tilde{m}_o$	2.00 m, 1.62°	1.03 m, 0.52°	1.41 m, 0.45°
e_{t05}, e_{o05}	0.75 m, 1.23°	0.51 m, 0.21°	0.43 m, 0.16°
e_{t95}, e_{o95}	4.43 m, 2.97°	1.88 m, 0.90°	2.72 m, 0.81°
Max error (trans., att.)	4.57 m, 3.07°	2.35 m, 1.03°	3.04 m, 1.00°
$\tilde{m}_t/d$	2.50 %	1.29 %	1.77 %
Distance Traveled: 125 Meters			
$\tilde{m}_t, \tilde{m}_o$	3.14 m, 2.73°	1.06 m, 0.59°	1.84 m, 0.58°
e_{t05}, e_{o05}	0.87 m, 1.25°	0.53 m, 0.23°	0.56 m, 0.17°
e_{t95}, e_{o95}	7.38 m, 4.52°	1.81 m, 1.13°	3.84 m, 0.94°
Max error (trans., att.)	7.69 m, 4.61°	2.35 m, 1.30°	4.19 m, 1.39°
$\tilde{m}_t/d$	2.51 %	0.85 %	1.47 %
Distance Traveled: 165 Meters			
$\tilde{m}_t, \tilde{m}_o$	3.45 m, 2.80°	1.14 m, 0.64°	2.26 m, 0.62°
e_{t05}, e_{o05}	0.97 m, 1.26°	0.54 m, 0.24°	0.62 m, 0.18°
e_{t95}, e_{o95}	10.38 m, 5.71°	1.76 m, 1.42°	4.80 m, 0.93°
Max error (trans., att.)	11.23 m, 5.83°	2.35 m, 1.76°	5.27 m, 1.39°
$\tilde{m}_t/d$	2.09 %	0.69 %	1.37 %
Distance Traveled: 210 Meters			
$\tilde{m}_t, \tilde{m}_o$	3.58 m, 2.81°	1.16 m, 0.66°	2.31 m, 0.61°
e_{t05}, e_{o05}	0.98 m, 1.26°	0.55 m, 0.24°	0.62 m, 0.18°
e_{t95}, e_{o95}	11.09 m, 5.76°	1.82 m, 1.58°	4.89 m, 0.93°
Max error (trans., att.)	15.33 m, 7.31°	2.35 m, 1.95°	5.92 m, 1.39°
$\tilde{m}_t/d$	1.70 %	0.55 %	1.10 %

Figures 4.13–4.24 plot the body-frame velocity estimates and the attitude estimates of Curve SLAM, OKVIS, SPTAM, and the corresponding ground truth as function of time. The body-frame velocity estimate for SPTAM was obtained by subtracting two world-frame position estimates between chronologically successive image frames and multiplying by the frame rate to obtain a world-frame velocity estimate, this world-frame velocity estimate was then transformed to the body frame using SPTAM's estimated attitude. In all six datasets, Curve SLAM's attitude is obtained without relying on an external compass. It should also be noted that without the vision-based curve measurements, the IMU alone would produce quickly growing attitude error estimates.

Figures 4.26–4.31 plot the estimated camera trajectory of Curve SLAM, OKVIS, SPTAM, and the corresponding ground-truth trajectory using Google maps for all six datasets (due to their size, we placed Figures 4.26–4.31 at the end of Section 4.4). To overlay the estimated trajectory onto Google maps, we align the frame where the GPS ground-truth coordinates are defined with

Table 4.5: Algorithm Comparison for DS5

DS5 is a sequence obtained from the KITTI dataset [116].
DS5 was collected under sunny conditions.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	NA	80 control points	21402
Distance Traveled: 65 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	3.45 m, 0.21°	1.10 m, 0.46°
e_{t05}, e_{o05}	NA	1.31 m, 0.08°	0.37 m, 0.16°
e_{t95}, e_{o95}	NA	5.61 m, 0.42°	5.24 m, 8.87°
Max error (trans., att.)	NA	6.33 m, 0.45°	10.37 m, 13.36°
$\tilde{m}_t/d$	NA	5.31 %	1.69 %
Distance Traveled: 130 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	4.69 m, 0.27°	1.23 m, 0.66°
e_{t05}, e_{o05}	NA	1.98 m, 0.09°	0.56 m, 0.18°
e_{t95}, e_{o95}	NA	8.91 m, 0.46°	8.57 m, 12.27°
Max error (trans., att.)	NA	10.58 m, 0.54°	15.59 m, 14.21°
$\tilde{m}_t/d$	NA	3.61 %	0.95 %
Distance Traveled: 195 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	5.15 m, 0.29°	1.59 m, 0.79°
e_{t05}, e_{o05}	NA	2.21 m, 0.13°	0.59 m, 0.23°
e_{t95}, e_{o95}	NA	10.50 m, 0.51°	5.48 m, 12.99°
Max error (trans., att.)	NA	13.05 m, 0.55°	15.59 m, 14.61°
$\tilde{m}_t/d$	NA	2.64 %	0.81 %
Distance Traveled: 260 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	6.16 m, 0.30°	1.98 m, 0.90°
e_{t05}, e_{o05}	NA	2.24 m, 0.13°	0.62 m, 0.24°
e_{t95}, e_{o95}	NA	11.96 m, 0.51°	10.50 m, 13.41°
Max error (trans., att.)	NA	14.76 m, 0.55°	15.59 m, 14.72°
$\tilde{m}_t/d$	NA	2.37 %	0.76 %
Distance Traveled: 330 Meters			
$\tilde{m}_t, \tilde{m}_o$	NA	6.96 m, 0.31°	2.14 m, 1.07°
e_{t05}, e_{o05}	NA	2.26 m, 0.13°	0.65 m, 0.24°
e_{t95}, e_{o95}	NA	15.41 m, 0.50°	13.05 m, 13.41°
Max error (trans., att.)	NA	17.60 m, 0.55°	16.79 m, 14.72°
$\tilde{m}_t/d$	NA	2.11 %	0.65 %

the world frame of each of the datasets. To align coordinate frames, we must find a coordinate transformation that maps a point $\mathbf{r}_k^{\mathcal{W}}$ defined in the coordinates of the world frame $\mathcal{W}$ (the world frame represents the coordinate frame in which each dataset is defined) to a point $\mathbf{r}_k^{\mathcal{W}'}$ defined in the coordinates of the GPS frame $\mathcal{W}'$, where $\mathbf{r}_k^{\mathcal{W}}$ represents the estimated location of the sensor platform in the world frame at time step k. The coordinate transformation that aligns the world frame with the GPS frame is expressed as

$$\mathbf{r}_k^{\mathcal{W}'} = \mathbf{R}_{\mathcal{W}}^{\mathcal{W}'} \mathbf{r}_k^{\mathcal{W}} + \mathbf{c}_{\mathcal{W}}^{\mathcal{W}'} \tag{4.2}$$

The matrix $\mathbf{R}_{\mathcal{W}}^{\mathcal{W}'}$ and translation vector $\mathbf{c}_{\mathcal{W}}^{\mathcal{W}'}$ are obtained by minimizing $\sum_k ||\mathbf{r}_k^{\mathcal{W}'} - \mathbf{g}_k^{\mathcal{W}'}||^2$ where $\mathbf{g}_k^{\mathcal{W}'}$ represents ground-truth coordinates.

In Figures 4.26, 4.27, and 4.28 the labeled white curve is the sidewalk path that was used to provide curves in the Curve SLAM algorithm. In Figure 4.29, the center curve is a road and yellow road lanes located on this road were used as curves in the Curve SLAM algorithm. In Figure 4.30,

Table 4.6: Algorithm Comparison for DS6

DS6 is a sequence obtained from the visual-inertial canoe dataset [116]. DS6 was collected under sunny conditions.

Paramater	Curve SLAM
Map Landmarks	8 control points
Distance Traveled: 30 Meters	
$\tilde{m}_t, \tilde{m}_o$	4.33 m, 3.32°
e_{t05}, e_{o05}	2.84 m, 2.37°
e_{t95}, e_{o95}	7.99 m, 4.20°
Max error (trans., att.)	10.43 m, 4.37°
$\tilde{m}_t/d$	14.44 %
Distance Traveled: 65 Meters	
$\tilde{m}_t, \tilde{m}_o$	4.77 m, 4.08°
e_{t05}, e_{o05}	2.98 m, 2.43°
e_{t95}, e_{o95}	11.73 m, 8.13°
Max error (trans., att.)	16.75 m, 8.39°
$\tilde{m}_t/d$	7.33 %
Distance Traveled: 100 Meters	
$\tilde{m}_t, \tilde{m}_o$	4.91 m, 6.10°
e_{t05}, e_{o05}	3.10 m, 2.51°
e_{t95}, e_{o95}	11.01 m, 11.68°
Max error (trans., att.)	16.75 m, 12.09°
$\tilde{m}_t/d$	4.91 %
Distance Traveled: 135 Meters	
$\tilde{m}_t, \tilde{m}_o$	5.38 m, 6.43°
e_{t05}, e_{o05}	3.15 m, 2.55°
e_{t95}, e_{o95}	10.53 m, 14.62°
Max error (trans., att.)	16.75 m, 14.95°
$\tilde{m}_t/d$	3.98 %
Distance Traveled: 170 Meters	
$\tilde{m}_t, \tilde{m}_o$	5.38 m, 6.43°
e_{t05}, e_{o05}	3.15 m, 2.55°
e_{t95}, e_{o95}	10.53 m, 14.62°
Max error (trans., att.)	16.75 m, 14.95°
$\tilde{m}_t/d$	3.16 %

the labeled road provided curves for the Curve SLAM algorithm. In Figure 4.31, the shoreline of the labeled river provided curves for the Curve SLAM algorithm. In DS3, between the start point and end point of the loop, Curve SLAM estimated the magnitude error between the start pose and end pose to be 6.4 meters, OKVIS estimated the magnitude error between the start pose and end pose to be 7.1 meters. By comparision, ground truth recorded a magnitude error of 4.24 meters between the start pose and end pose. At the final pose, the final attitude error between ground truth and Curve SLAM is 3.8 degrees in yaw, 1.61 degrees in pitch and 2.12 degrees in roll. The final attitude error between OKVIS and ground truth is 1.94 degrees in yaw, 0.01 degrees in pitch and 0.04 degrees in roll.

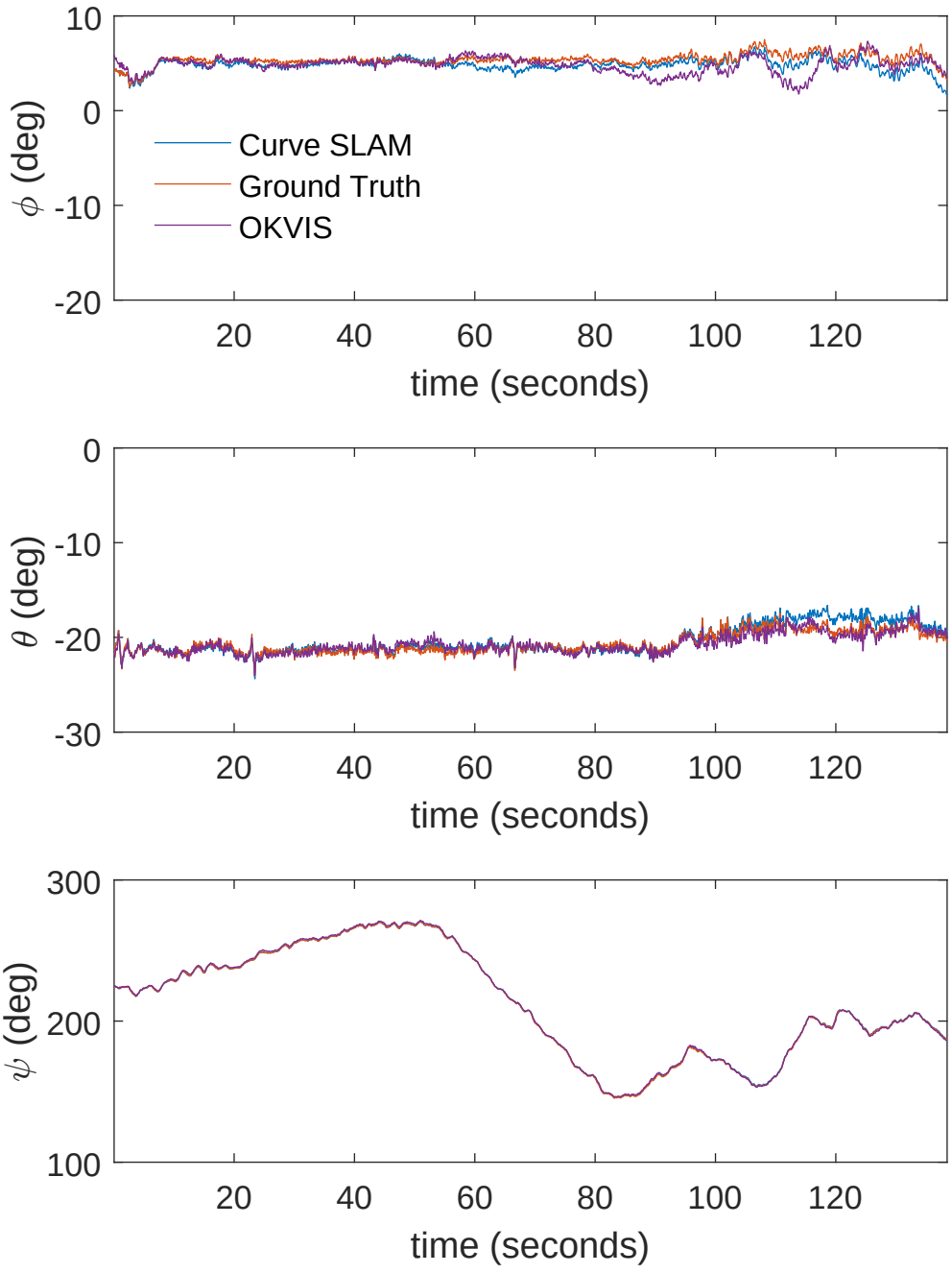


Figure 4.13: Attitude estimates of Curve SLAM and OKVIS along with the corresponding ground truth for DS1.

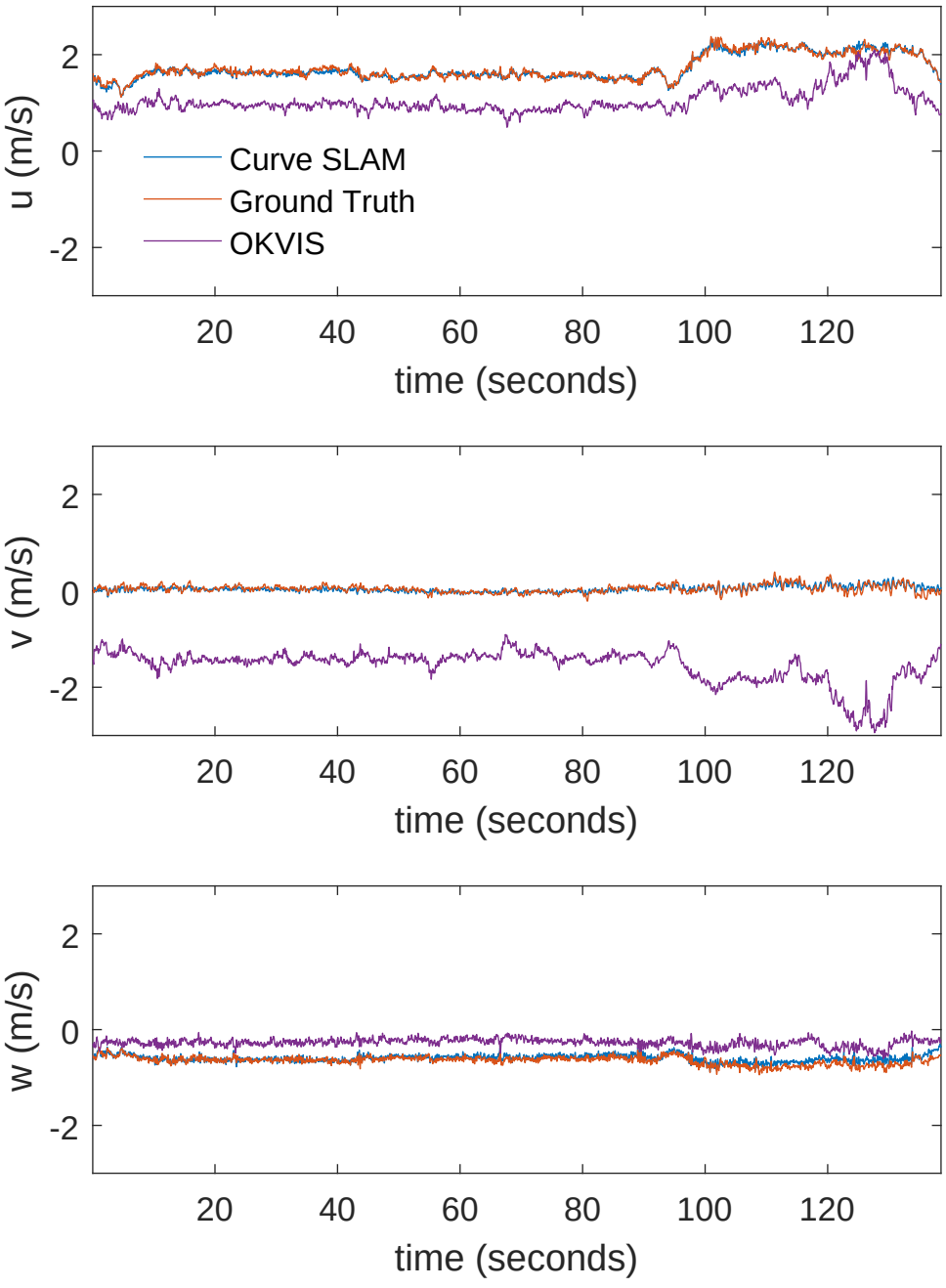


Figure 4.14: Body-frame velocity estimates of Curve SLAM and OKVIS along with the corresponding ground truth for DS1.

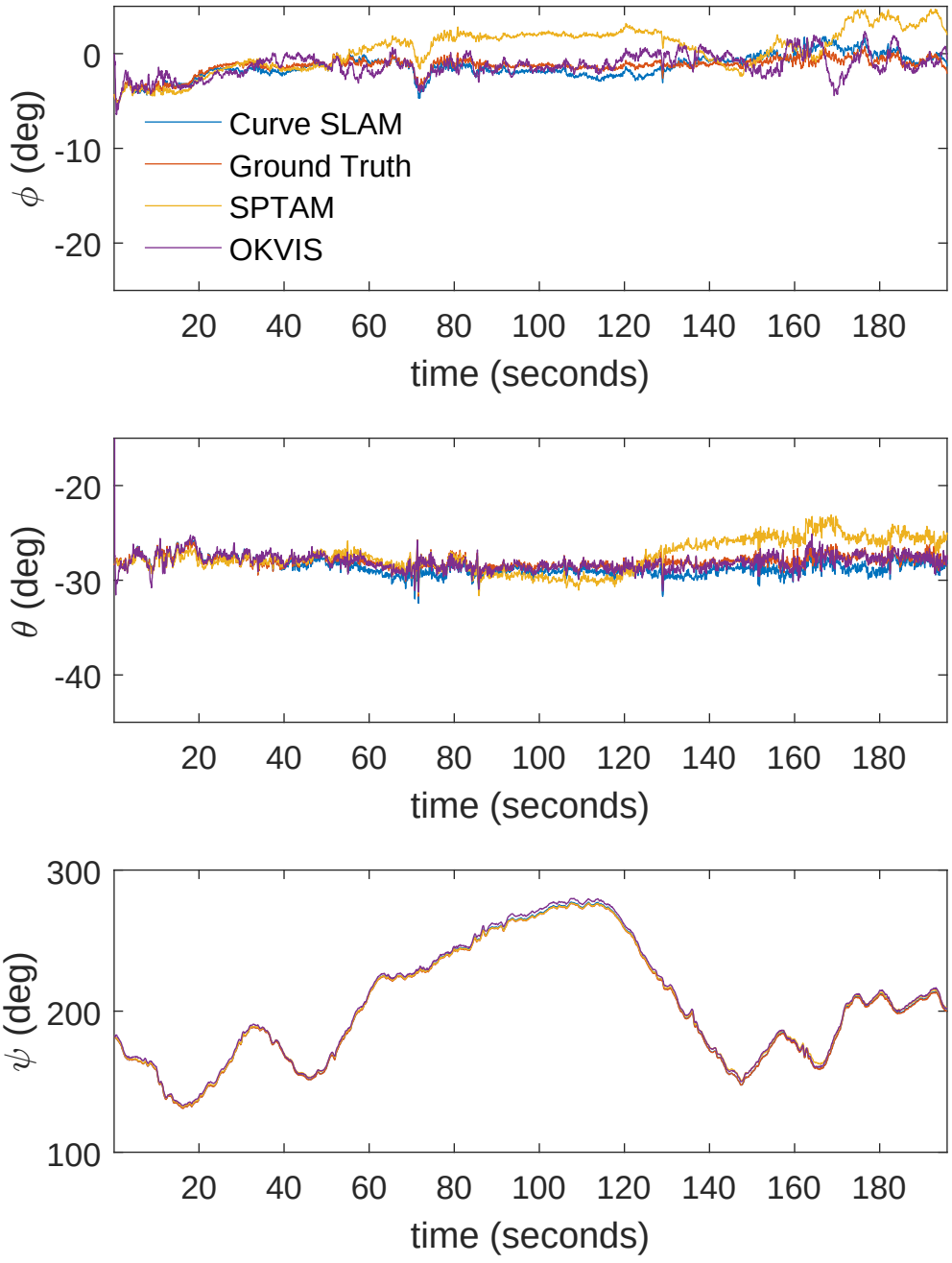


Figure 4.15: Attitude estimates of Curve SLAM, OKVIS, and SPTAM along with the corresponding ground truth for DS2.

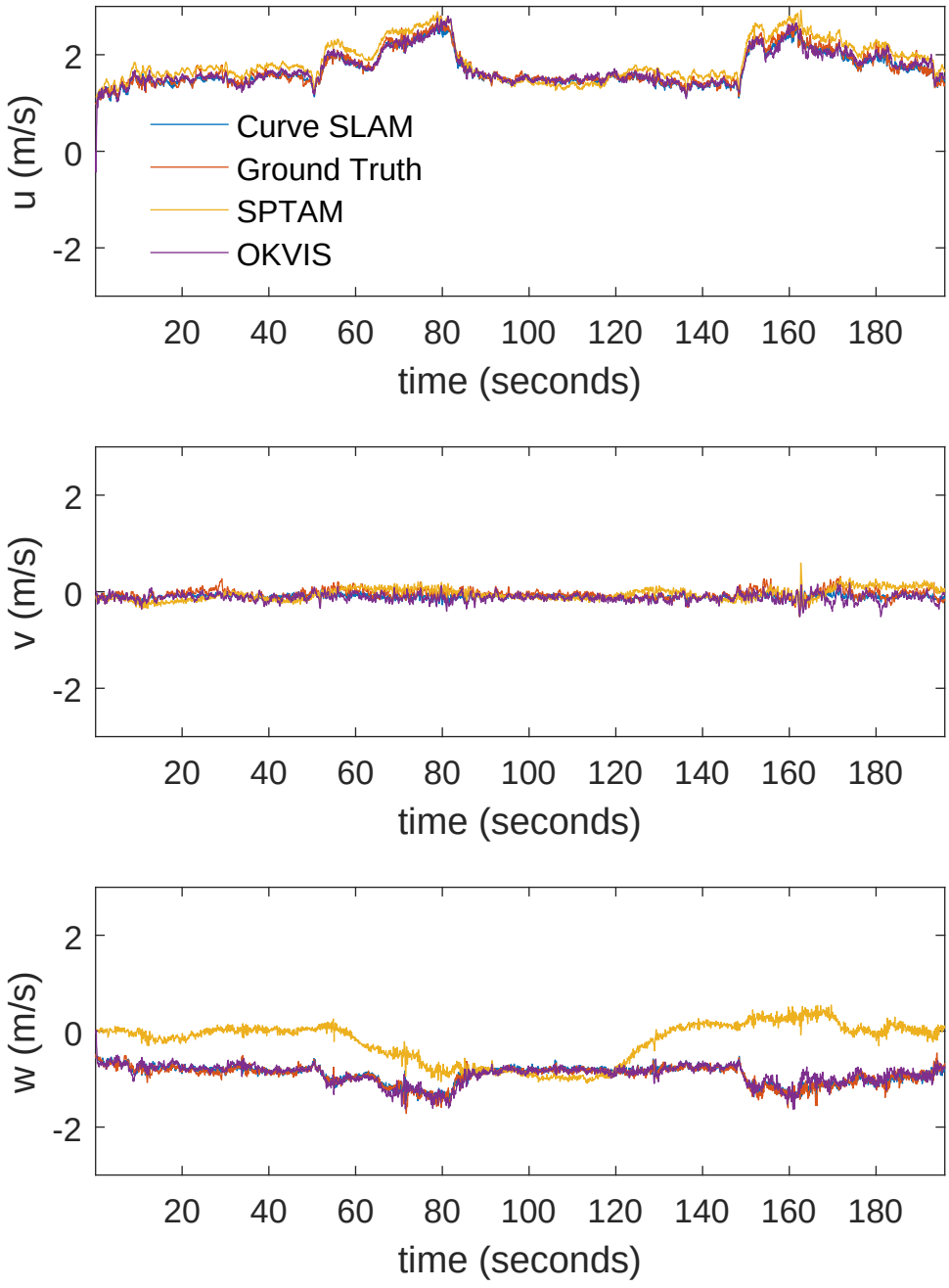


Figure 4.16: Body-frame velocity estimates of Curve SLAM, OKVIS, and SPTAM along with the corresponding ground truth for DS2.

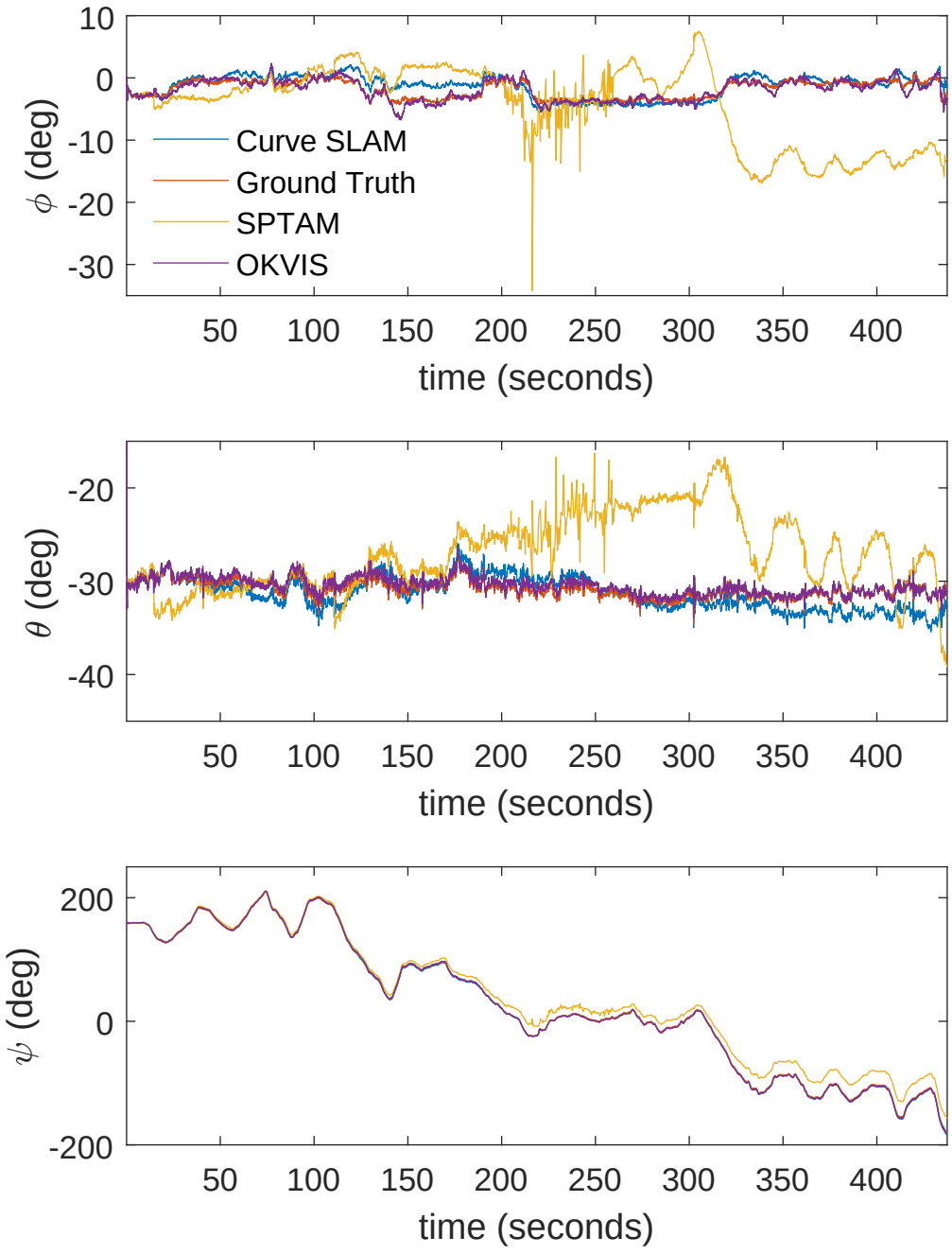


Figure 4.17: Attitude estimates of Curve SLAM, OKVIS, and SPTAM along with the corresponding ground truth for DS3.

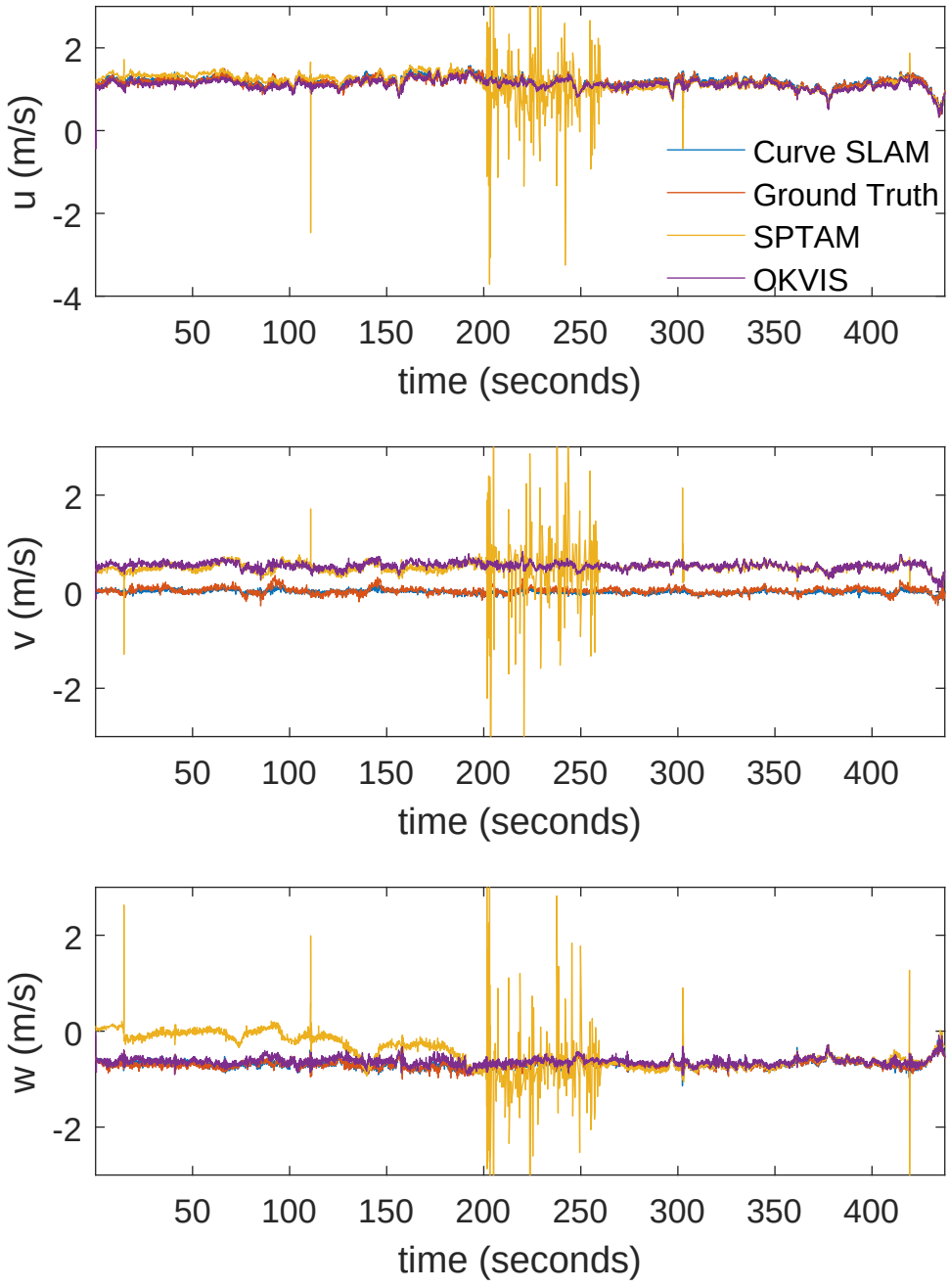


Figure 4.18: Body-frame velocity estimates of Curve SLAM, OKVIS, and SPTAM along with the corresponding ground truth for DS3.

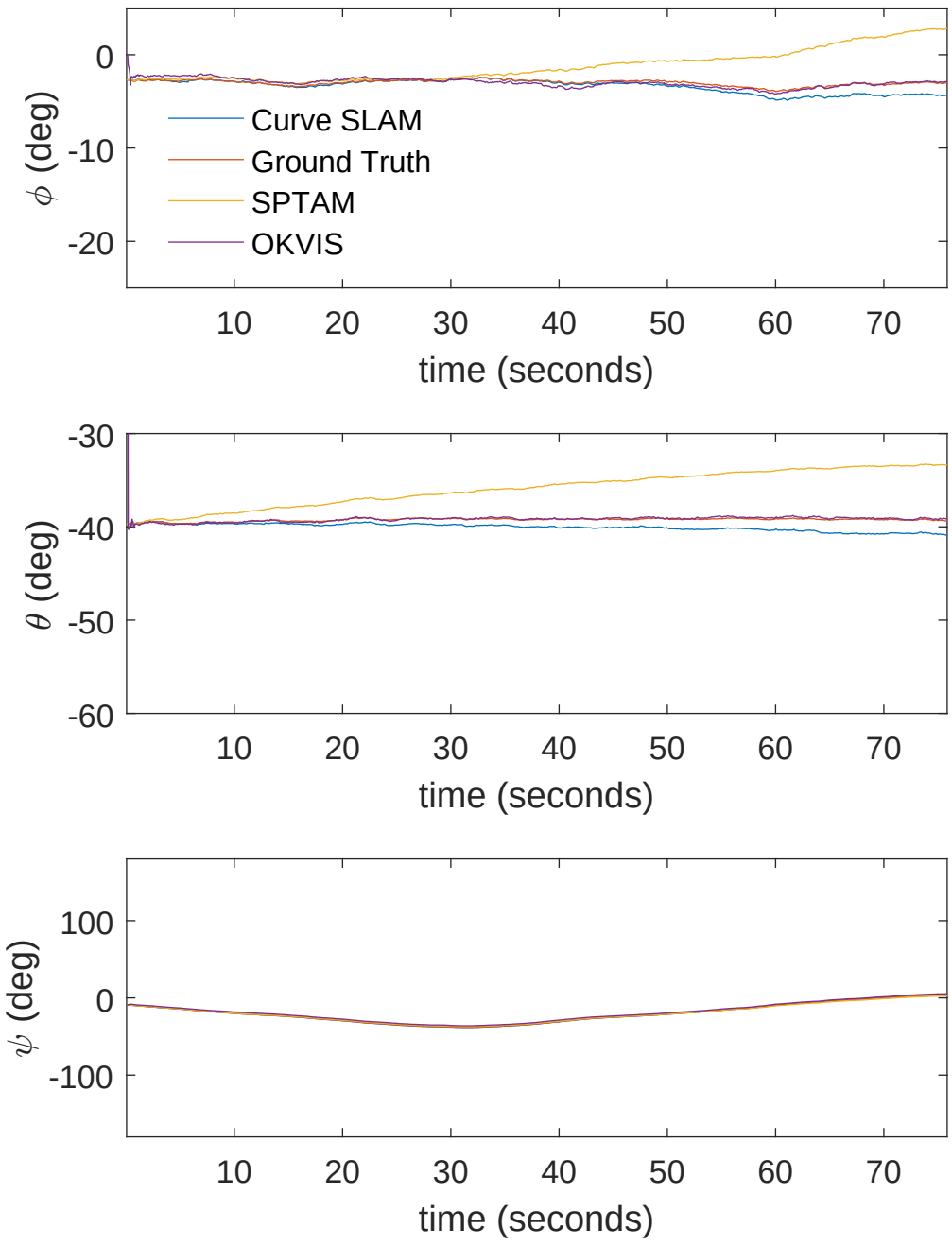


Figure 4.19: Attitude estimates of Curve SLAM, OKVIS, and SPTAM along with the corresponding ground truth for DS4.

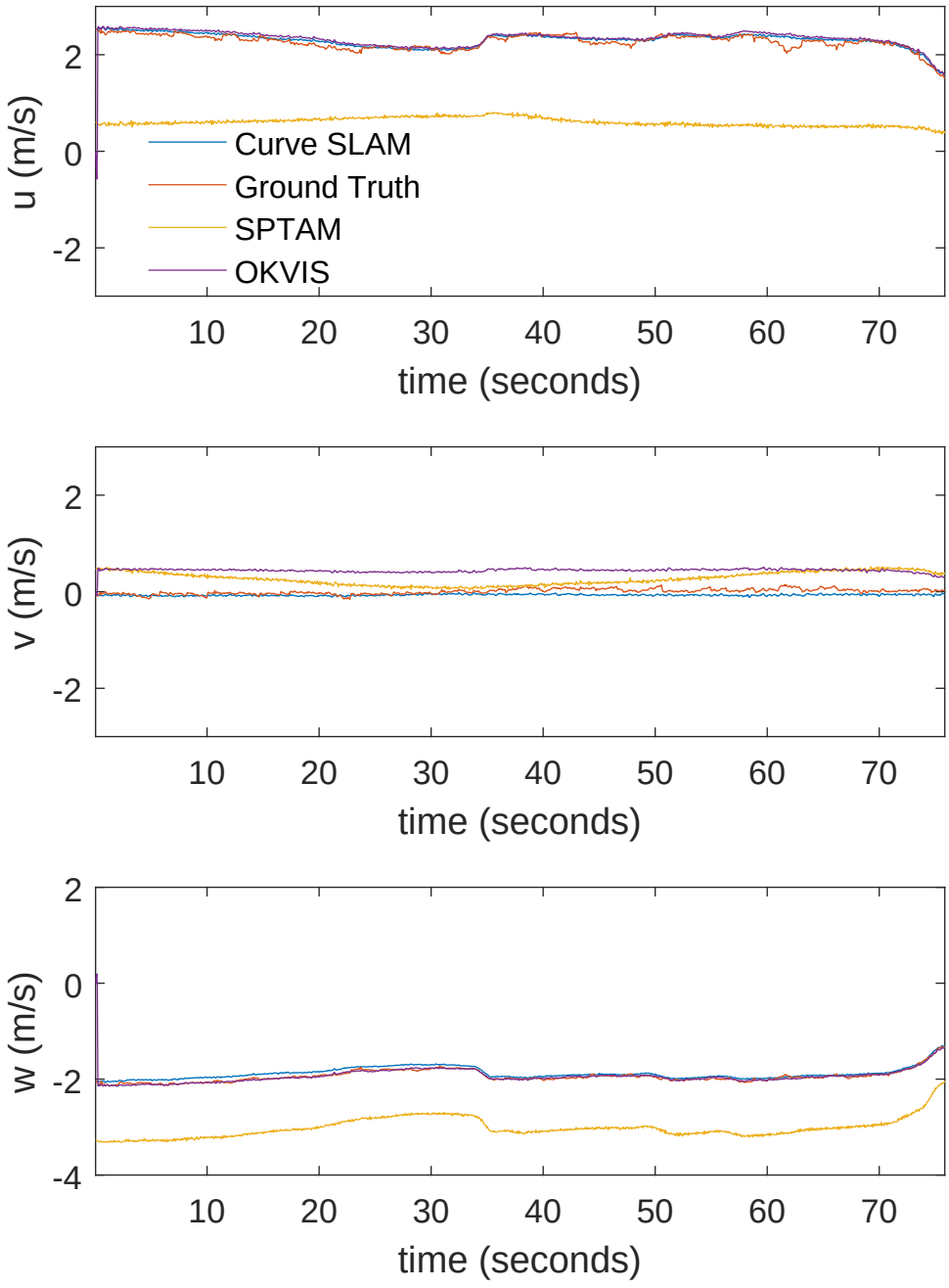


Figure 4.20: Body-frame velocity estimates of Curve SLAM, OKVIS, and SPTAM along with the corresponding ground truth for DS4.

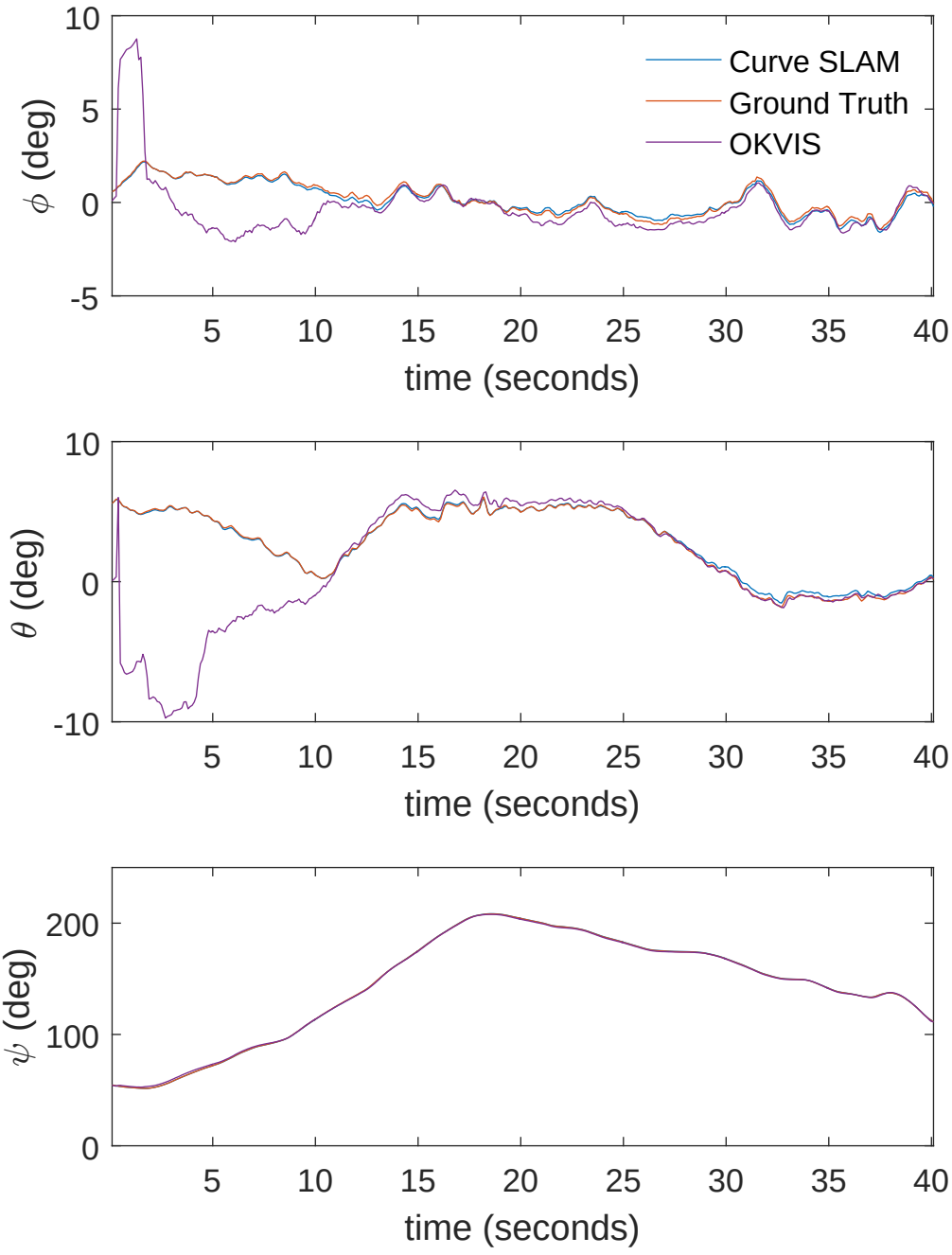


Figure 4.21: Attitude estimates of Curve SLAM and OKVIS along with the corresponding ground truth for DS5.

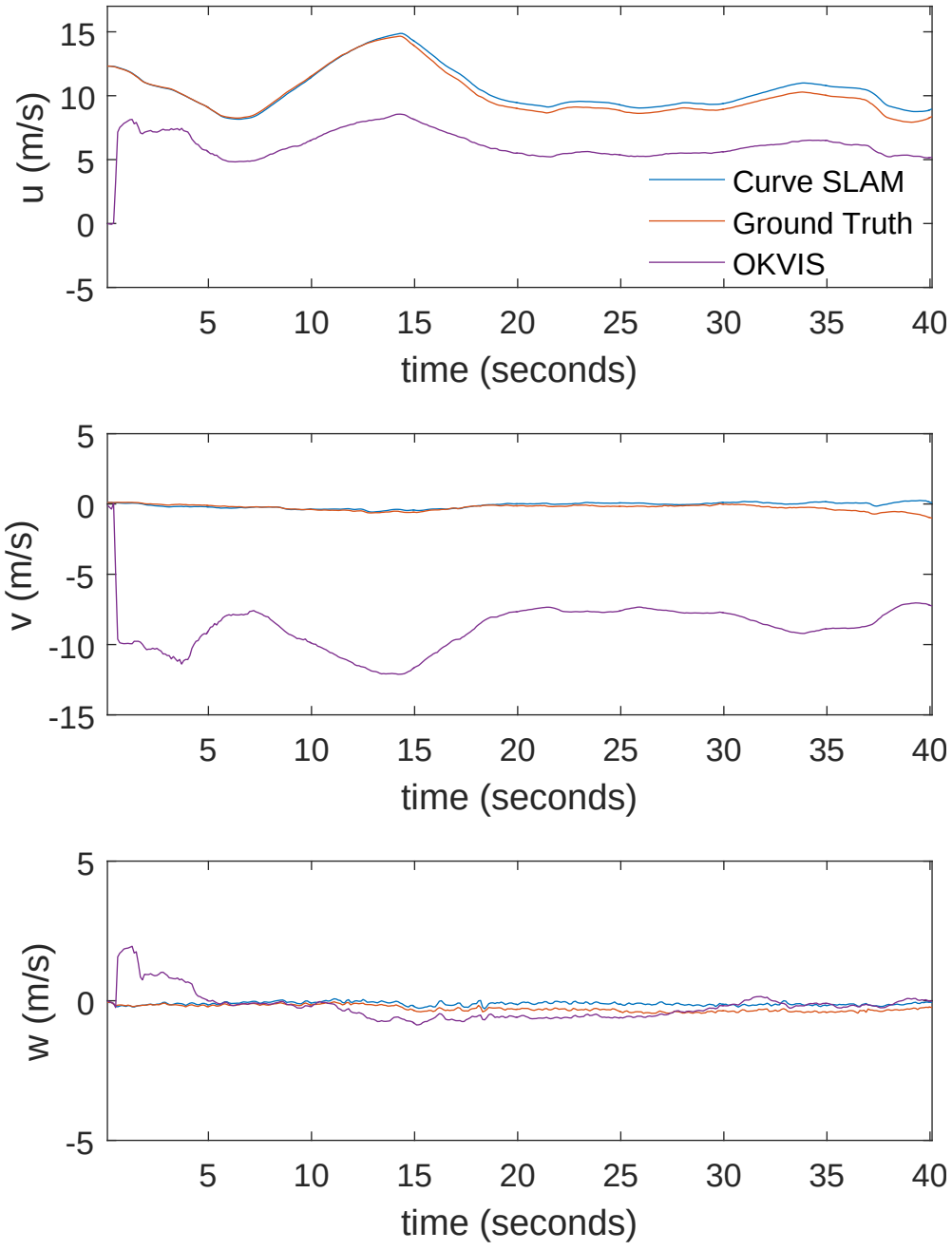


Figure 4.22: Body-frame velocity estimates of Curve SLAM and OKVIS along with the corresponding ground truth for DS5.

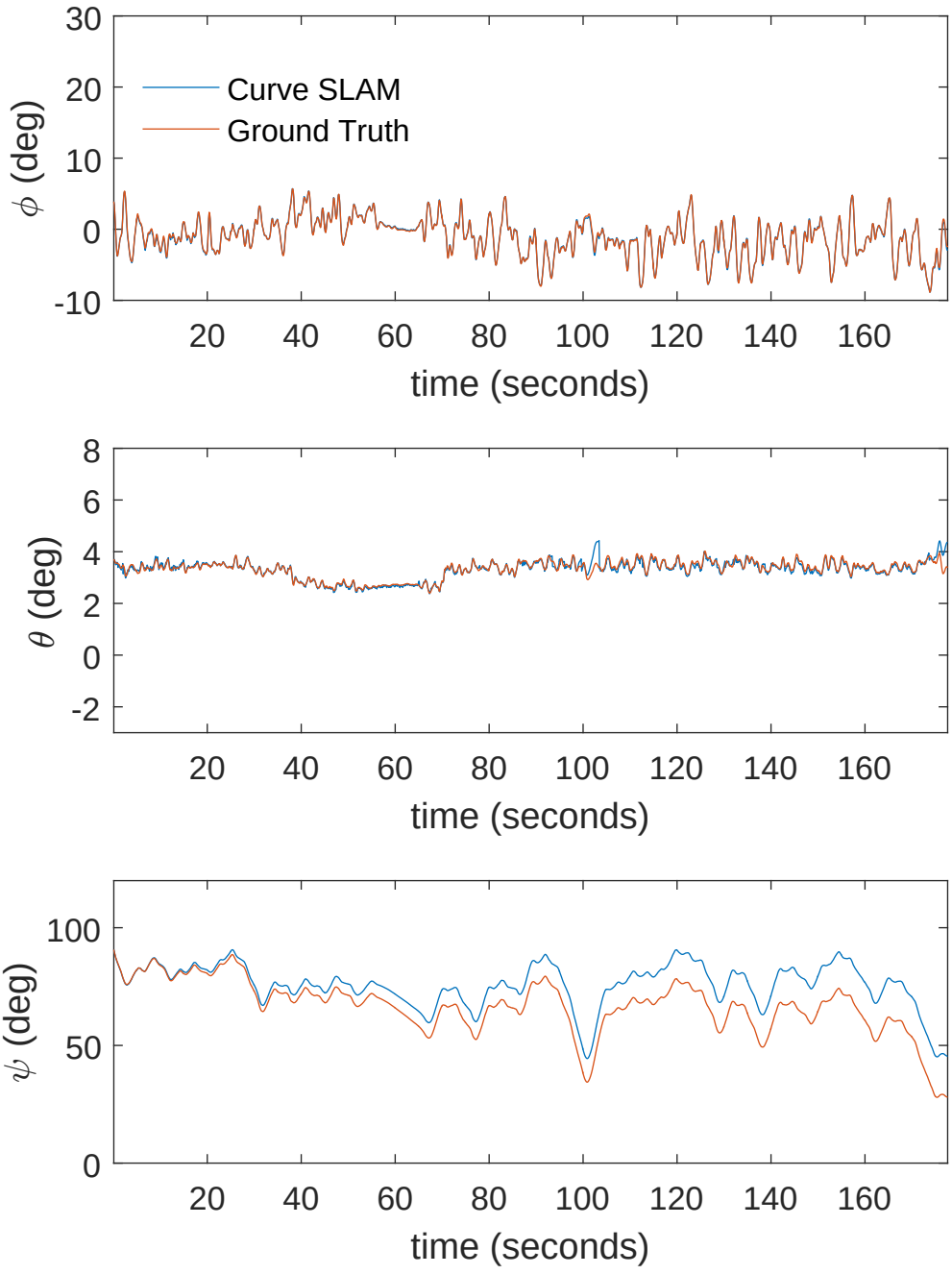


Figure 4.23: Attitude estimates of Curve SLAM along with the corresponding ground truth for DS6.

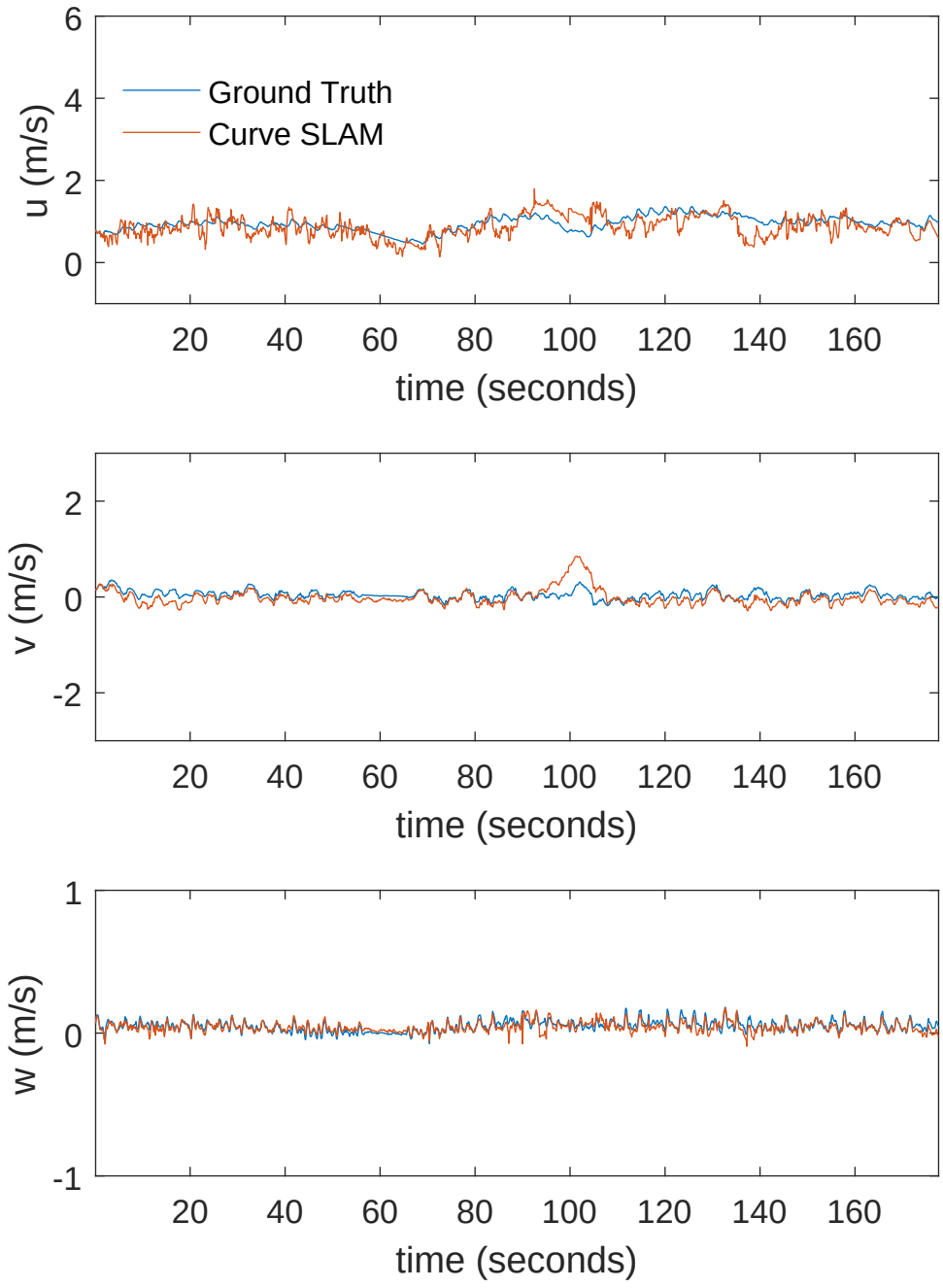


Figure 4.24: Body-frame velocity estimates of Curve SLAM along with the corresponding ground truth for DS6.

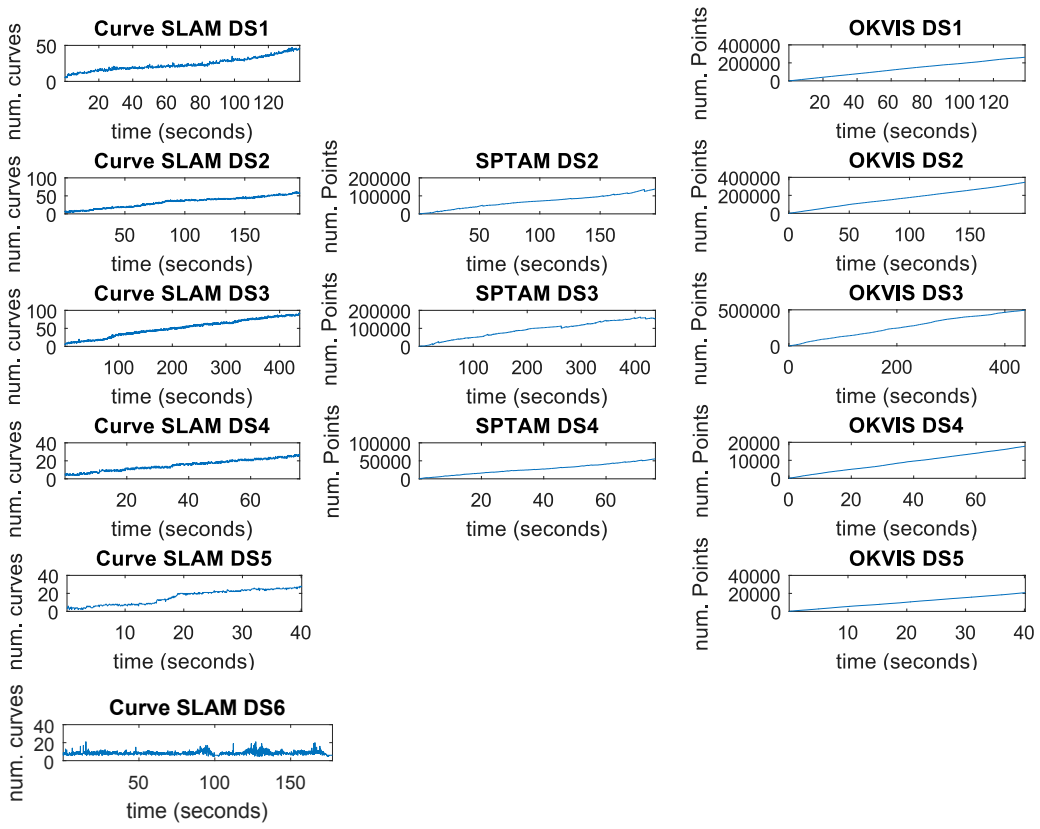


Figure 4.25: The number of curves (Curve SLAM) and points (OKVIS or SPTAM) required to represent the map as a function of time for the six datasets.

4.4 Mapping Results

For mapping purposes, we further reduce the number of points required to represent the path using the method described in Section 3.3.8. A plot of these results is shown in Figure 4.32 for DS1, Figure 4.33 for DS2, Figure 4.34 for DS3, Figure 4.35 for DS4, Figure 4.36 for DS5, and Figure 4.37 for DS6. The mapping results are obtained by transforming the map curves from the world frame where each dataset is defined to the coordinate frame where GPS is defined. The transformation used in this process is given by Equation (4.2). Figure 4.25 displays the number of landmarks required to map the six datasets for Curve SLAM, OKVIS, and SPTAM. In all the applicable datasets, Curve SLAM requires roughly three orders of magnitude fewer

landmarks to represent the map (see Tables 4.1-4.6 for the exact number). Indeed, in DS1 only 160 control points are used to represent 252 meters of a path, in DS2 only 220 control points are used to represent 375 meters of a path, in DS3 only 348 control points are used to represent 603 meters of a path, in DS4 only 92 control points are used to represent 230 meters of a path, in DS5 only 80 control points are used to represent 435 meters of a path, and in DS6 only eight control points are used to represent 170 meters of the shoreline of a river.

An interesting comparison is to evaluate how this result compares with the minimum number of cubic Bézier curves that can be interpolated to the boundary of the path. We answer this question for the curved path in Figure 4.32 by obtaining an appropriately zoomed satellite image of this region with Google maps. We find the boundary of the sidewalk in the satellite image by threshholding and applying a contour detector [103] to the boundary of the sidewalk. For comparison purposes, we match the start point and end point of the GPS-INS ground-truth path to the sidewalk in the satellite image, and convert pixel distance in the satellite image to meters. We find the minimum number of cubic bezier curves that can be interpolated to the path in the satellite image with a greedy algorithm. Our approach is similar to that described in Section 3.3.8. Assuming the spatially ordered pixels $p_i \dots p_m$ belong to the path boundary in the satellite image, and these pixels comprise a growing curve B_s, we attempt to combine the next spatially ordered pixel p_{m+1} along the path boundary to the curve B_s. To check if we can, we 1) interpolate a single Bézier curve to p_{m+1} and $p_i \dots p_m$ 2) check if the residuals of the fit satisfy a threshold d (we use the same value of d as in Section 3.3.8). If they do, p_{m+1} is added to B_s, and we continue to the next spatially ordered pixel coordinate along the boundary. If not, $p_i \dots p_m$ becomes a single curve, and we start a new curve, with p_{m+1} as the start point. This process starts by interpolating a single bezier curve to the first four spatially ordered pixels $p_1 \dots p_4$ of the satellite image and repeats until we reach the end point of the path. We start with four pixels because four is the minimum number of points required to interpolate a single cubic Bézier curve. Implementing this method, we find that the minimum number of cubic Bézier curves required to fit the path in the satellite image is 17, see Figure 4.38.

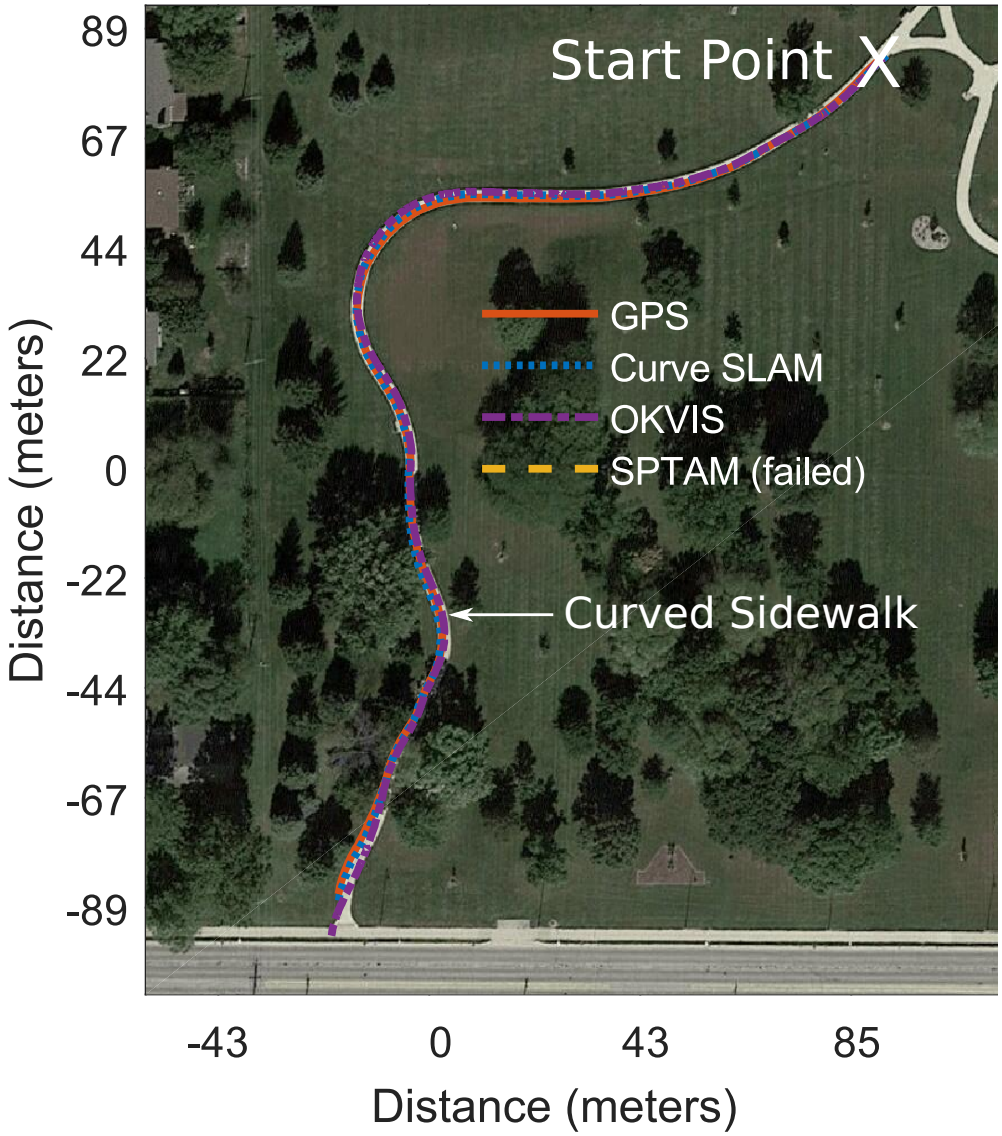


Figure 4.26: Localization results plotted in Google maps for DS1. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, and the ground-truth trajectory of the stereo camera is plotted in red. The estimated trajectory of SPTAM is not visible because the setting lacks distinguishable feature points, and SPTAM failed to track a sufficient number of feature points in this environment. The start point of the GPS track is marked with an X. DS1 was collected at dawn under clear weather conditions. The experiment lasted roughly 138.5 seconds. During this time, our sensors traveled approximately 252.2 meters.

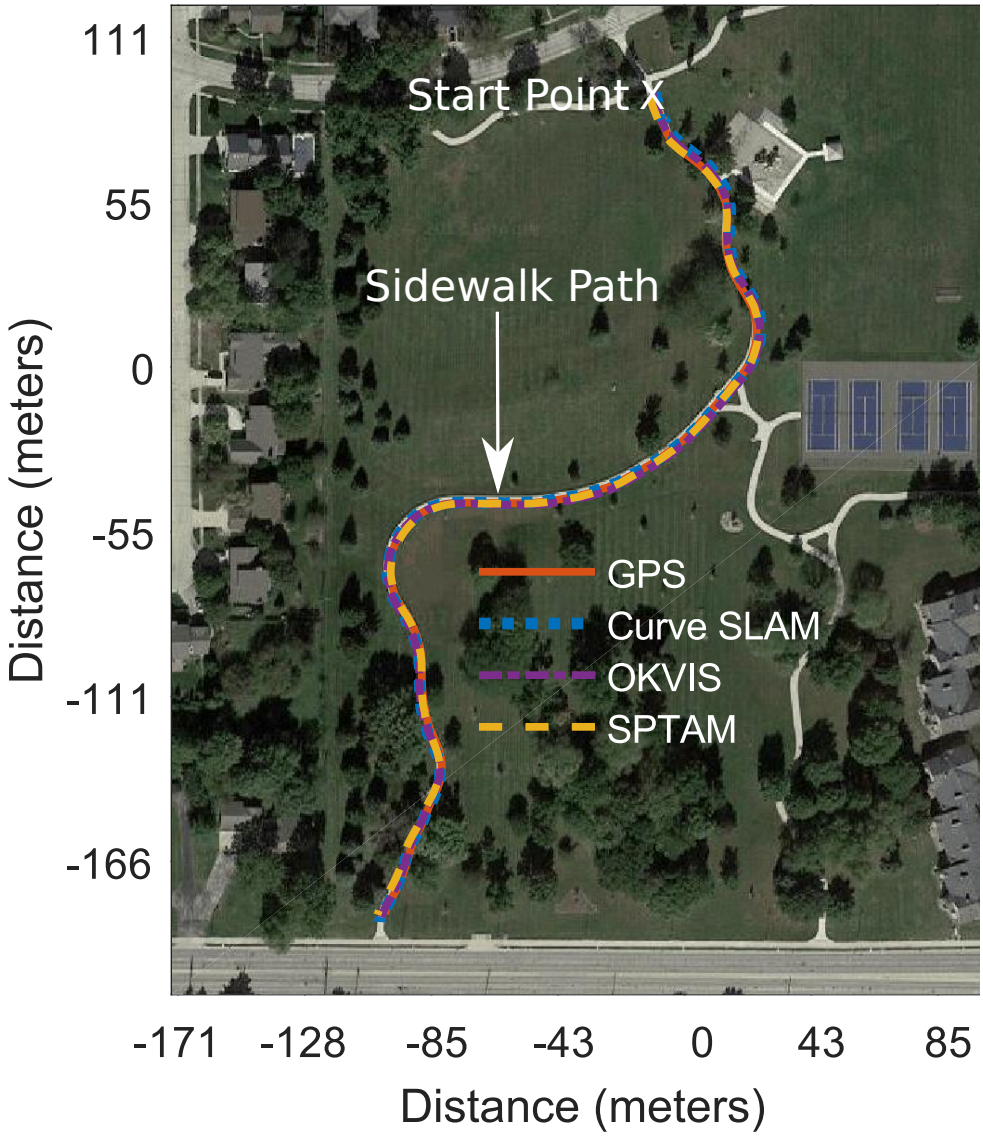


Figure 4.27: Localization results plotted in Google maps for DS2. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, the estimated trajectory of SPTAM is plotted in dashed yellow, and the ground-truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS2 was collected in the mid-afternoon, roughly 2 PM. Part of DS2 was collected during a light rainstorm under cloudy conditions, while the other part of DS2 was collected immediately following this light rainstorm under mostly sunny conditions. DS2 lasted roughly 195.85 seconds. During this time, our sensors traveled approximately 375 meters.

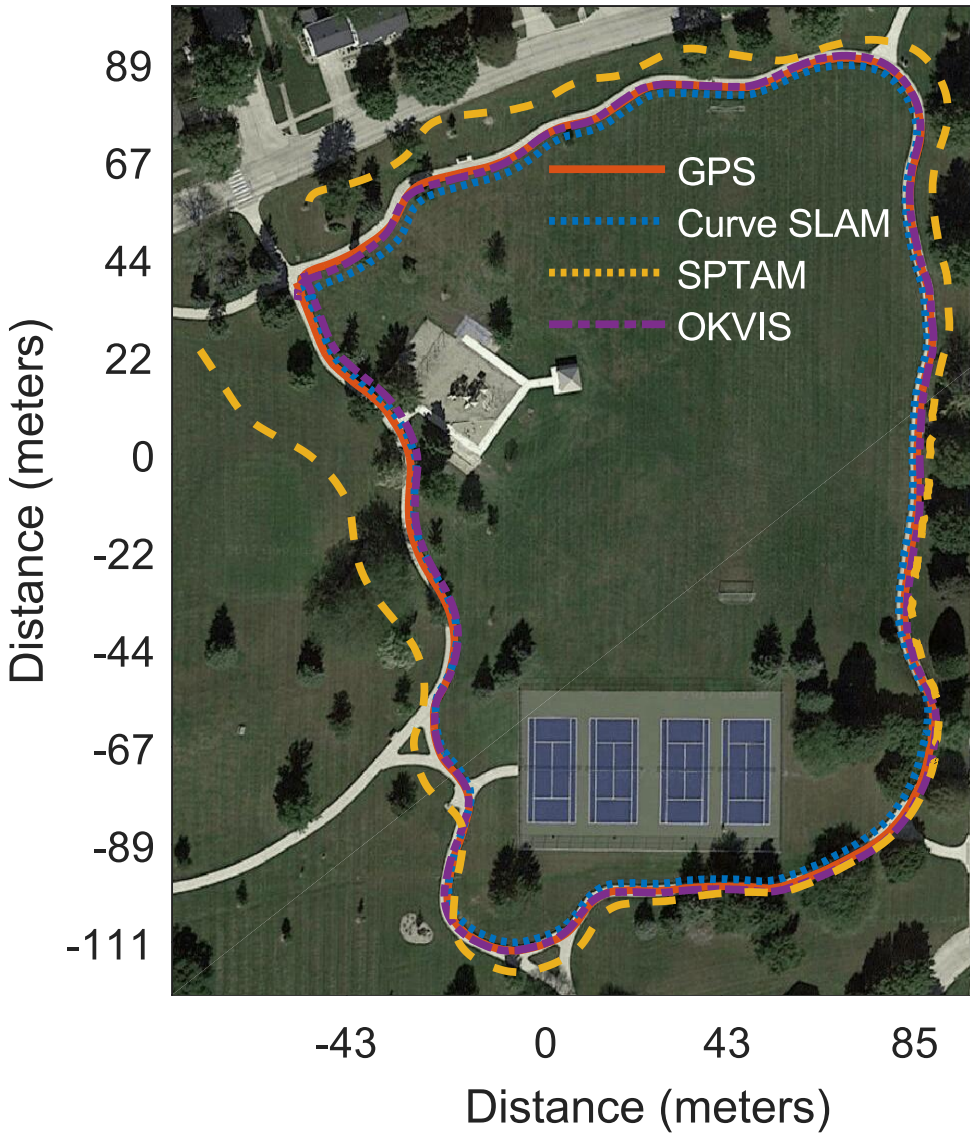


Figure 4.28: Localization results plotted in Google maps for DS3. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, the estimated trajectory of SPTAM is plotted in dashed yellow, and the ground-truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS3 was collected about an hour prior to sunset on a mostly sunny day with clear weather conditions. DS3 lasted roughly 437.55 seconds. During this time, our sensors traveled approximately 603.4 meters.

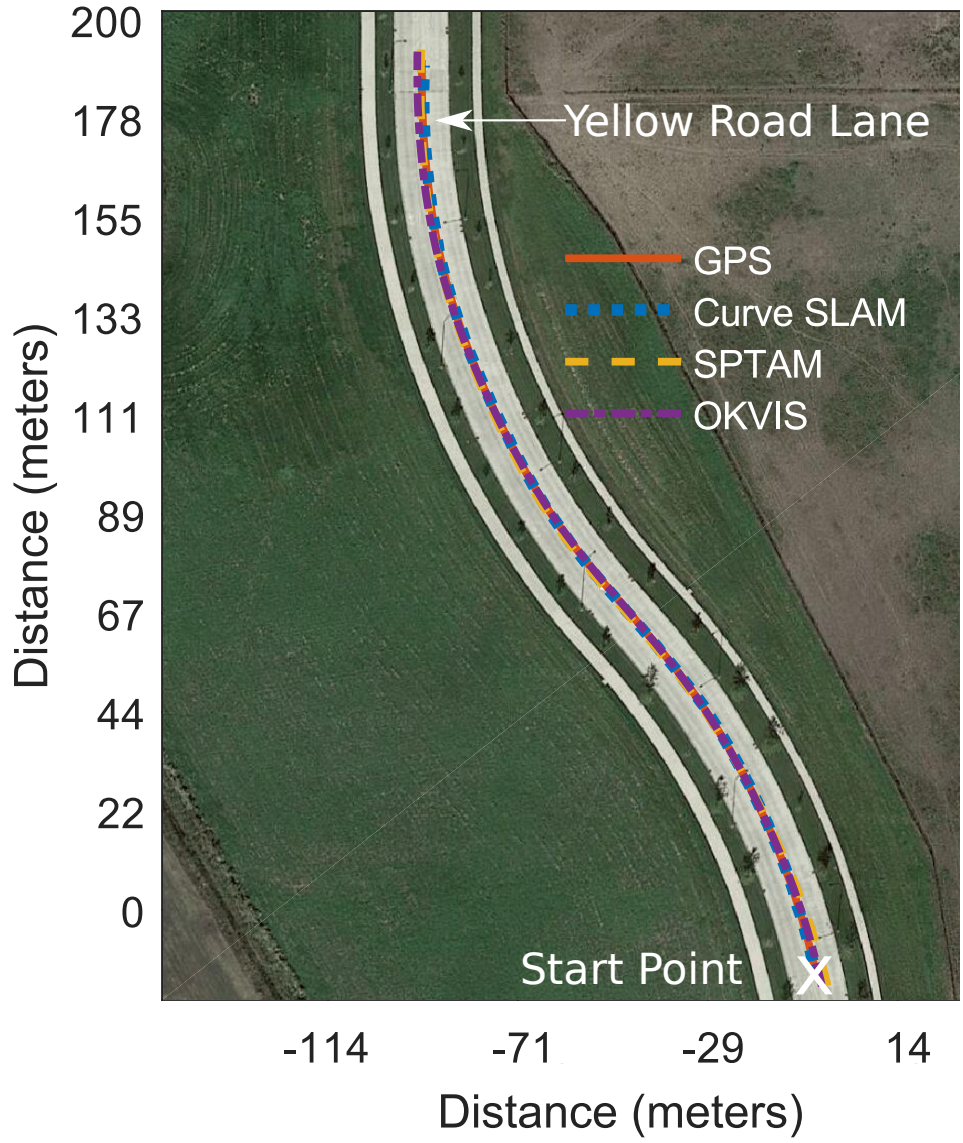


Figure 4.29: Localization results plotted in Google maps for DS4. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, the estimated trajectory of SPTAM is plotted in dashed yellow, and the ground-truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS4 was collected in the morning hours under mostly cloudy conditions with clear weather. DS4 lasted roughly 70 seconds. During this time, our sensors traveled approximately 230 meters.

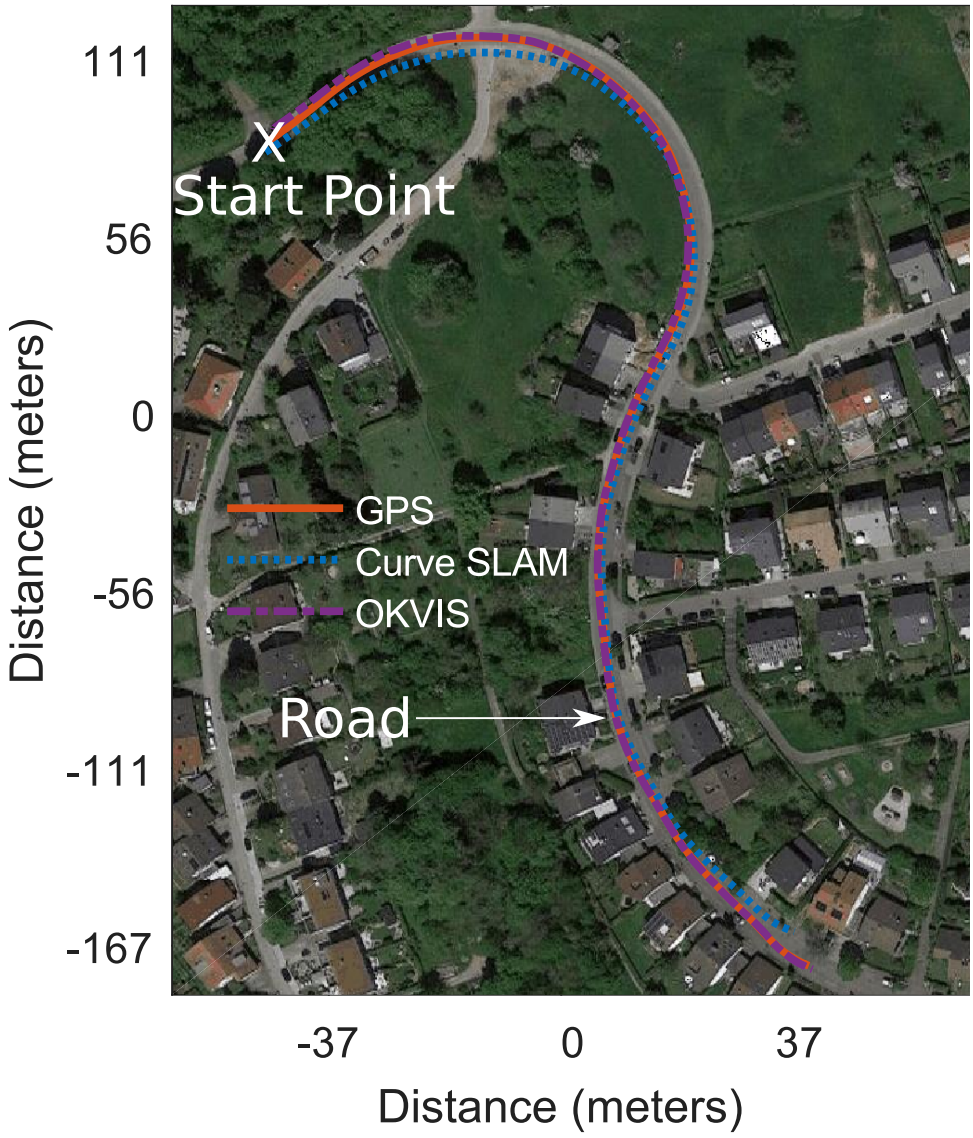


Figure 4.30: Localization results plotted in Google maps for DS5. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, and the ground-truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS5 contains a sequence of data obtained from the KITTI dataset, and was collected under sunny conditions. This sequence was selected due to the presence of curves in the environment and the number of occlusions covering the road.

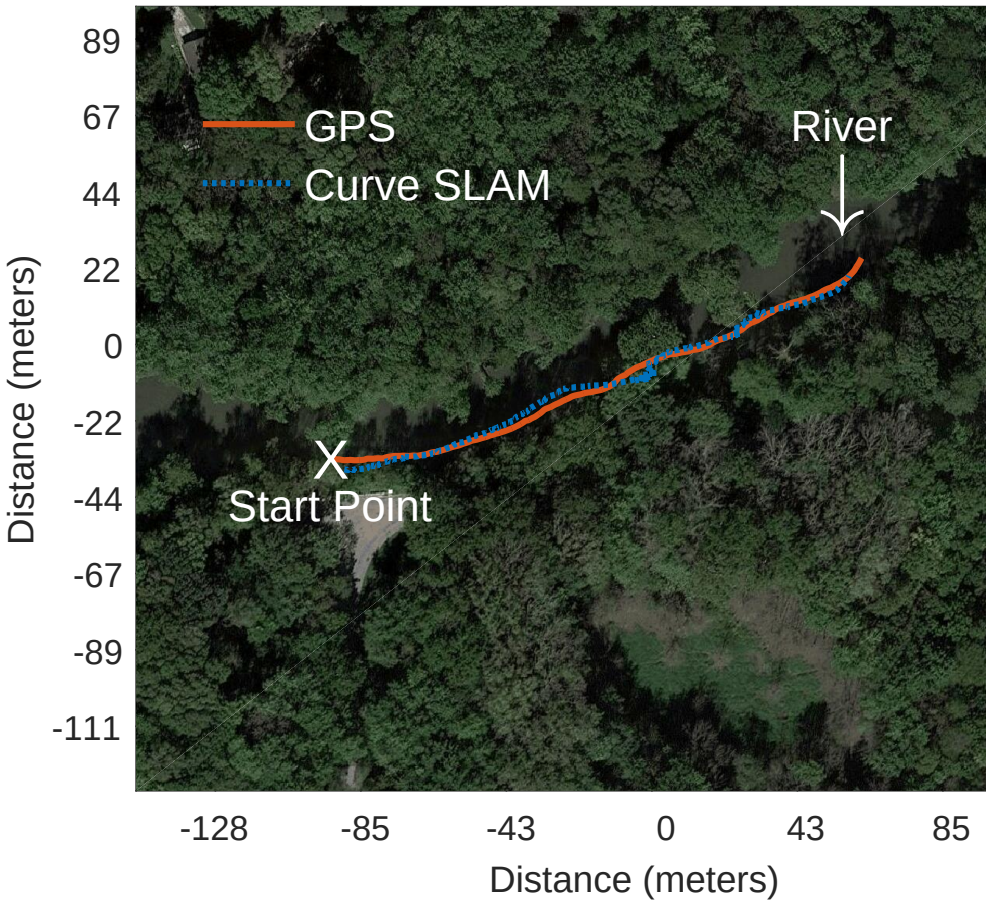


Figure 4.31: Localization results plotted in Google maps for DS6. The estimated trajectory of Curve SLAM is plotted in dashed blue, and the ground-truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS6 contains a sequence of data obtained from the visual-inertial canoe dataset, and was collected under sunny conditions. The experiment lasted roughly 177 seconds. During this time, our sensors traveled approximately 170 meters.

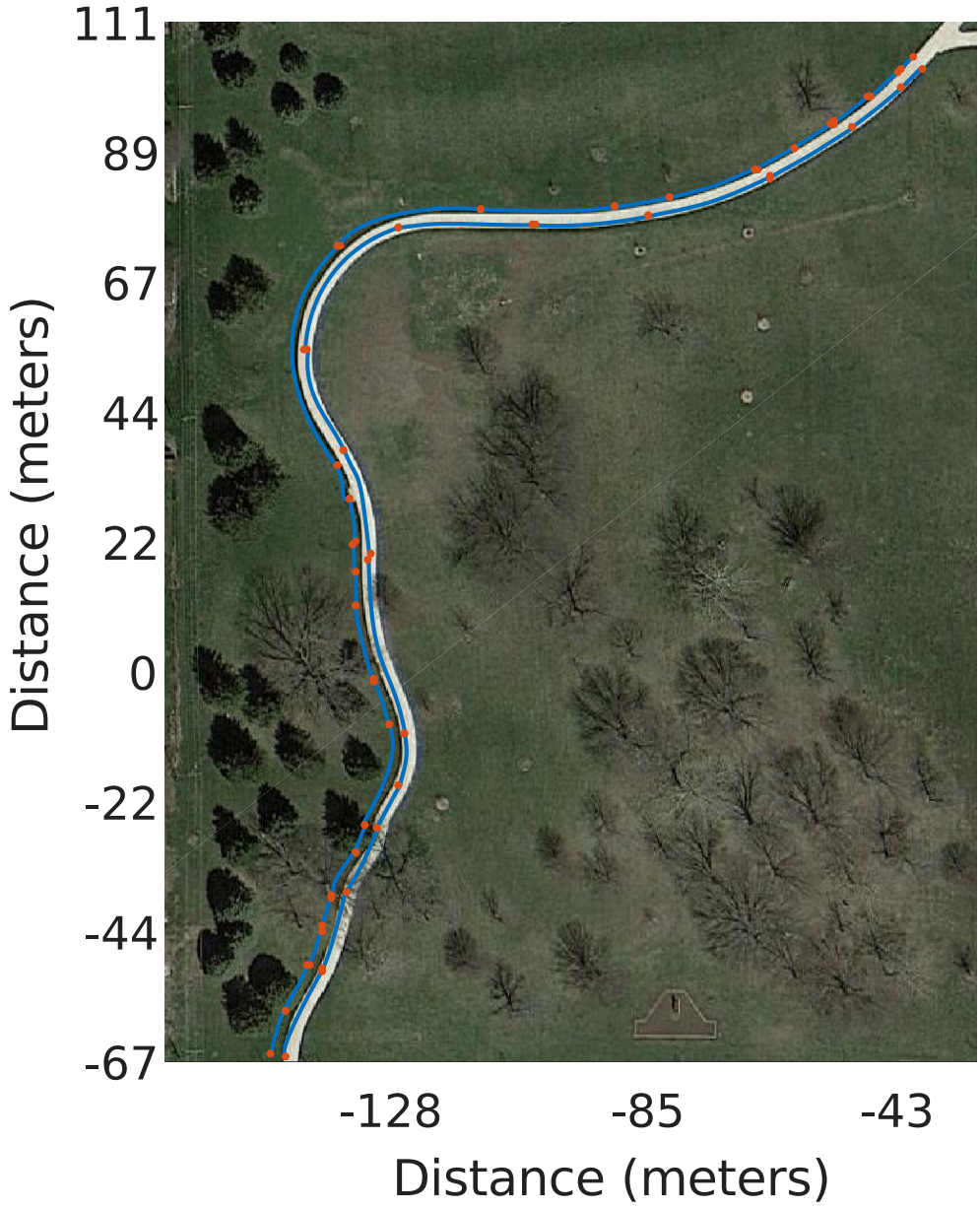


Figure 4.32: Mapping results plotted in Google maps for DS1. The mapping results display the reduced number of curves required to estimate the sidewalk. Red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

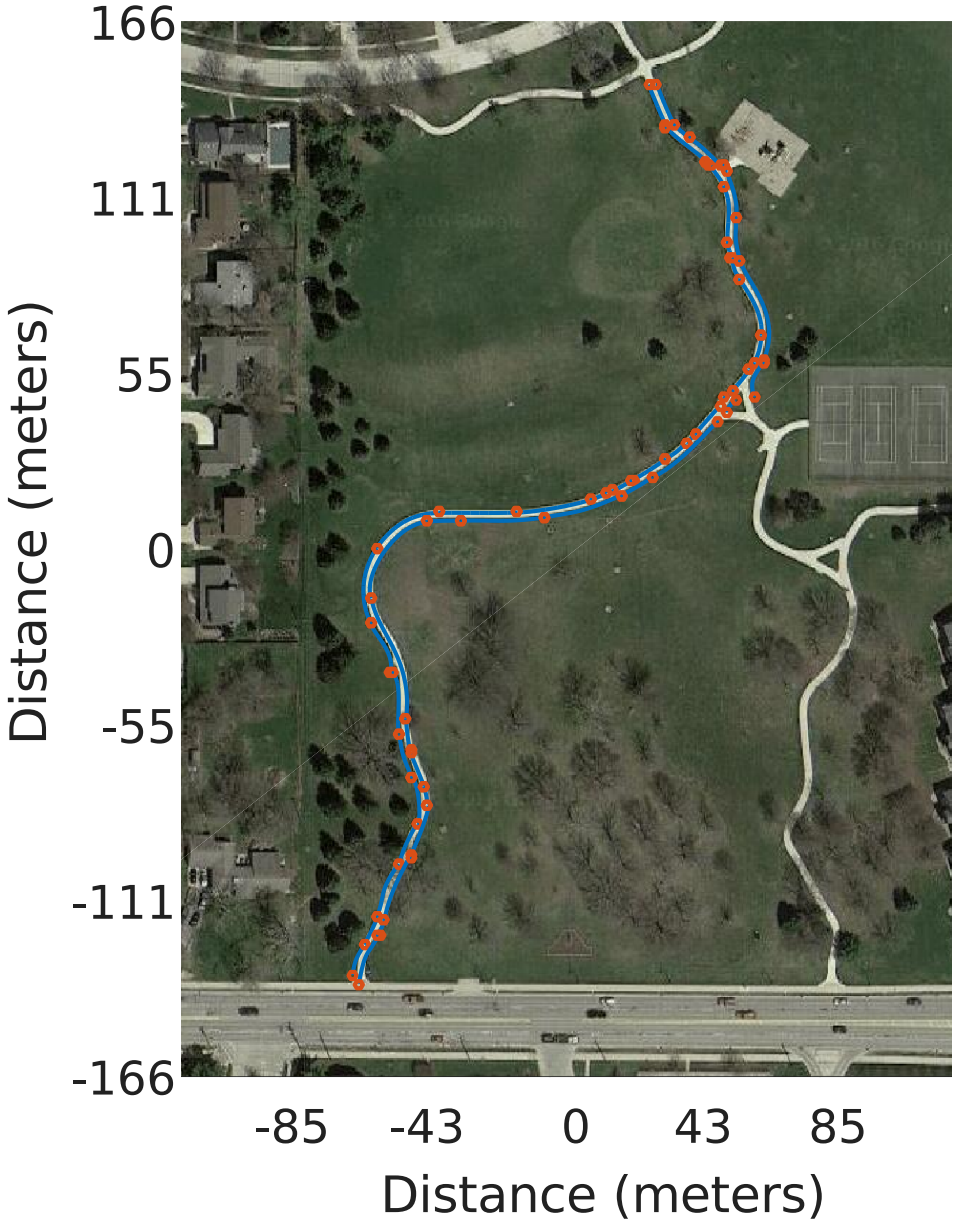


Figure 4.33: Mapping results plotted in Google maps for DS2. The mapping results display the reduced number of curves required to estimate the sidewalk. Red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

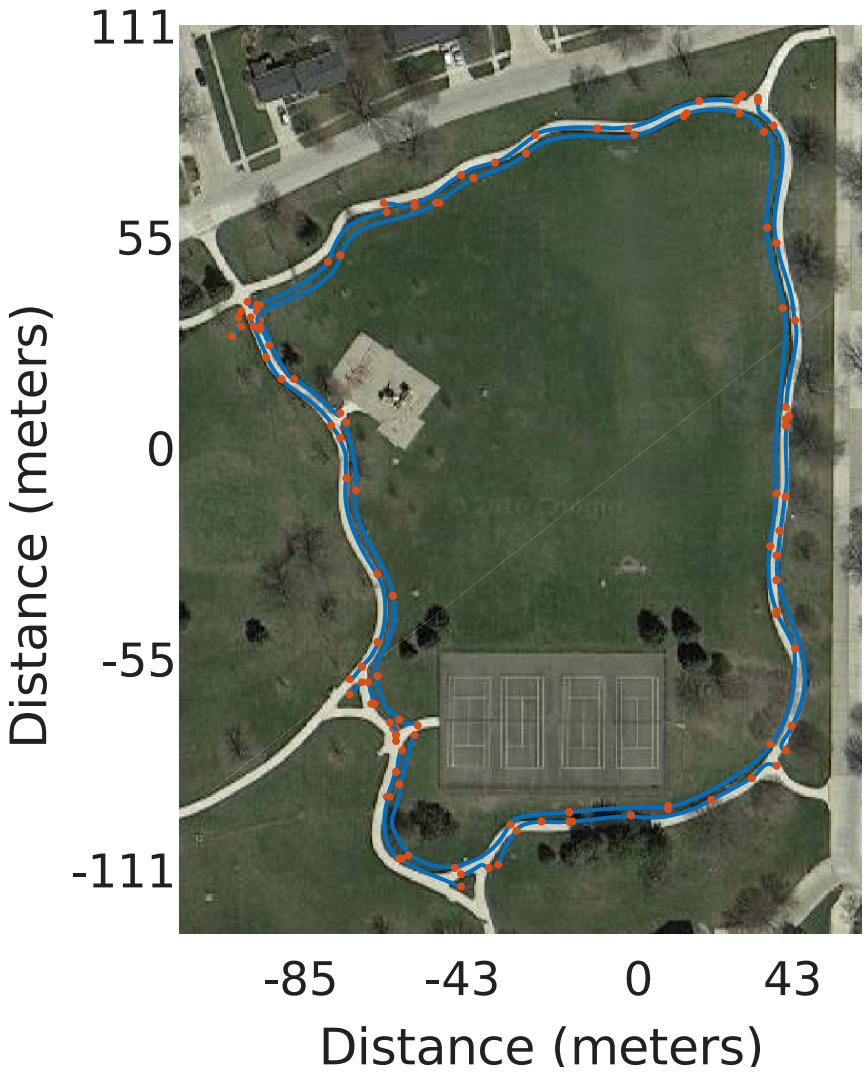


Figure 4.34: Mapping results plotted in Google maps for DS3. The mapping results display the reduced number of curves required to estimate the sidewalk. Red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

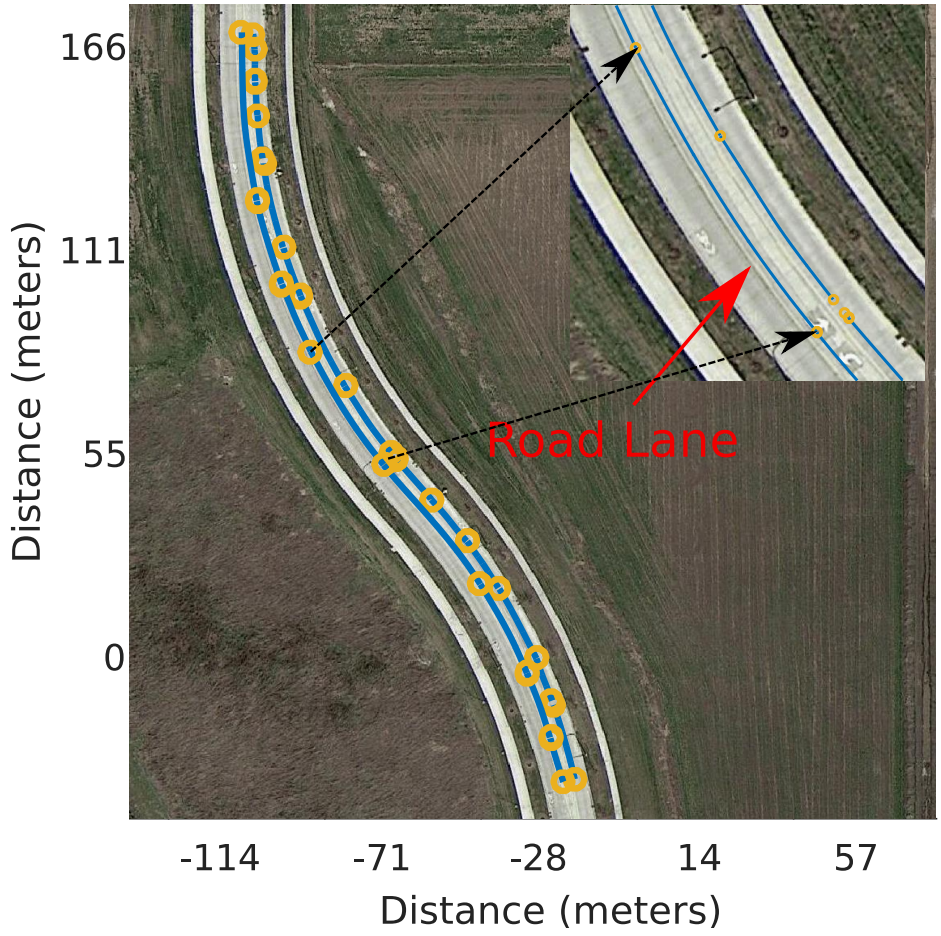


Figure 4.35: Mapping results plotted in Google maps for DS4. The mapping results display the reduced number of curves required to estimate the sidewalk. Yellow points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

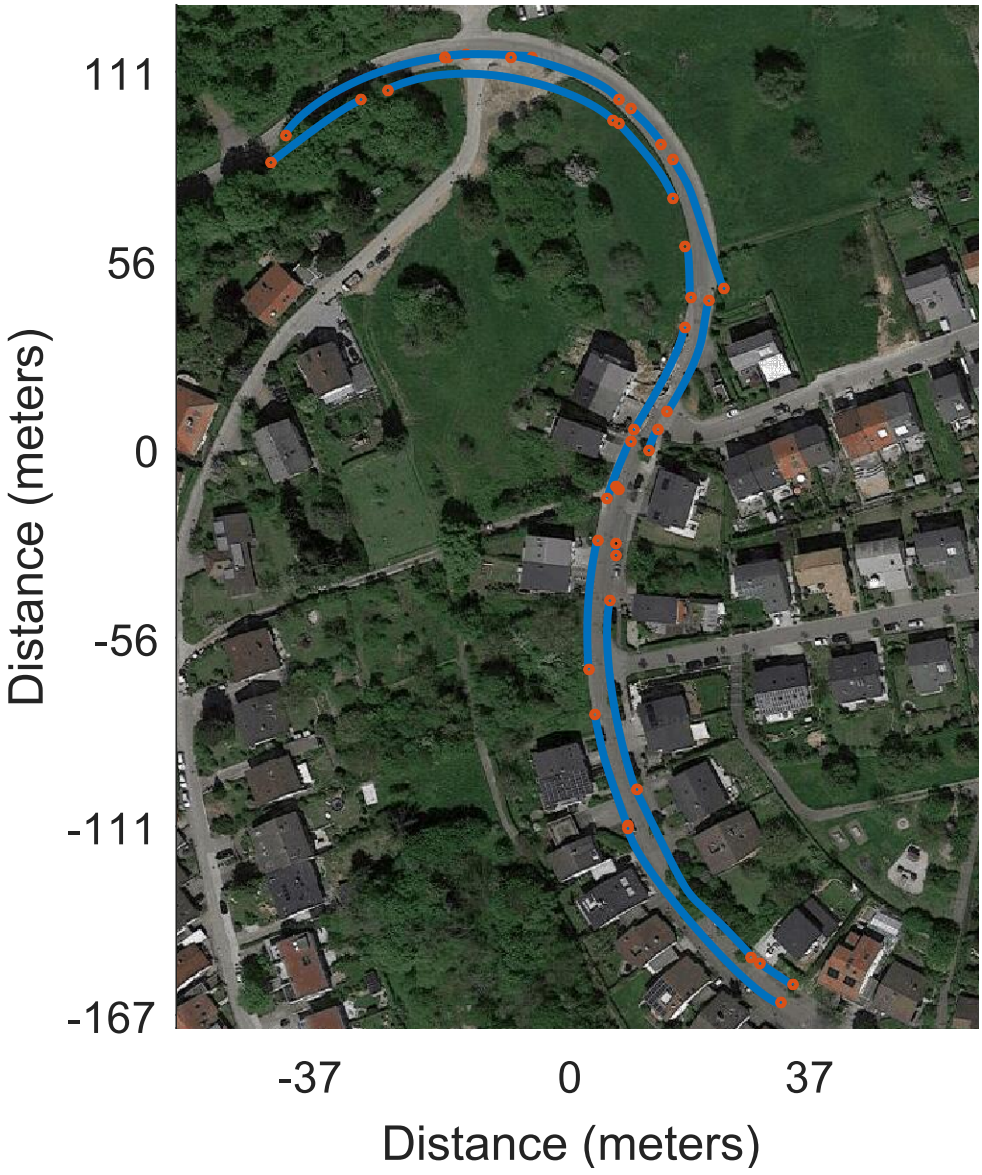


Figure 4.36: Mapping results plotted in Google maps for DS5. The mapping results display the reduced number of curves required to estimate the road. red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

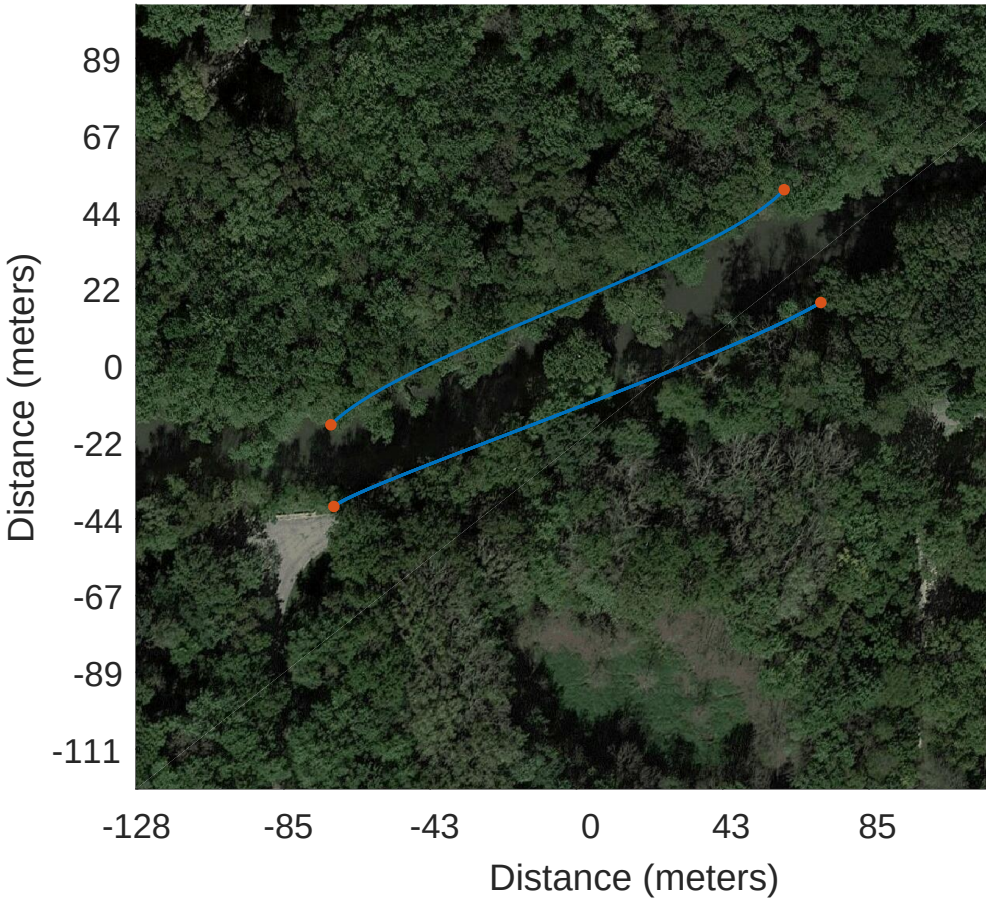


Figure 4.37: Mapping results plotted in Google maps for DS6. The mapping results display the reduced number of curves required to estimate the shoreline of a river. Red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points. In this plot, only eight curve control points are required to map the shoreline of a river over a distance of approximately 170 meters.

Figure 4.38: The theoretical minimum number of cubic Bézier curves required to represent the boundary of the path. This map should be used as a comparision with the map created by Curve SLAM in Figure 4.32. This map was obtained directly from Google maps, and is not a SLAM result. We find the boundary of the sidewalk in the satellite image by threshholding, and applying a contour detector [103] to the boundary of the sidewalk. Then we use a greedy algorihtm to find the minimum number of cubic Bézier curves that can be interpolated to the path.

4.5 Calibration and Parameter Selection

While obtaining the experimental data in this dissertation, we found the stereo camera to be incredibly sensitive to small disturbances, mostly due to vibrations. Thus, we find it helpful to emphasize that our stereo camera and IMU had a precision mounted case that was specifically created to prevent disturbances from disrupting the calibration. Additionally, stereo images on our sensor platform were time-synchronized within nanoseconds of each other using an external hardware trigger. Furthermore, to enhance the accuracy of the sensor system, the stereo calibration, and the camera to IMU calibration were performed immediately following data collection. For further information about our sensor platform, see Appendix B.

Throughout this dissertation, we described various parameters. We now describe how to select these parameters. To calculate $\mathbf{V}_r$, we apply the method in [107]. To do so, we first estimate the error variance σ^2:

$$\sigma^2 = \frac{\sum_{o\in\{L,R\}} \sum_{j=1}^{m} \sum_{i=1}^{n} ||\mathbf{y}_i - \hat{\mathbf{y}}(\boldsymbol{\beta}))||^2}{d_{\boldsymbol{\beta}}}$$

where $d_{\boldsymbol{\beta}}$ is the number of degrees of freedom for error in the parameter vector $\boldsymbol{\beta}$. An estimate of the parameter covariance $\mathbf{V}_r = \mathrm{Var}(\boldsymbol{\beta})$ is given by

$$\mathbf{V}_r = \sigma^2(\mathbf{J}^\top\mathbf{J})^{-1}$$

where $\mathbf{J}$ is the Jacobian of $\hat{\mathbf{y}}$ with respect to $\boldsymbol{\beta}$, i.e., $\mathbf{J} = \frac{\partial\hat{\mathbf{y}}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}}$.

The process noise covariance matrix $\mathbf{W}$ is calculated with the method in [114], where the noise characteristics of the IMU are obtained directly from the manufacturer's data sheet [117]

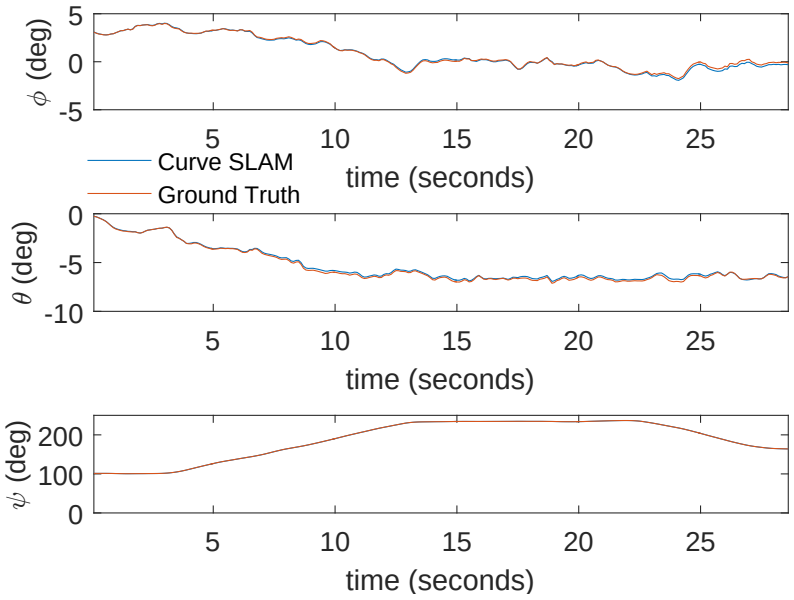
The above calculations for $\mathbf{W}$ and $\mathbf{V}_r$ provide a good starting point for tuning $\mathbf{W}$ and $\mathbf{V}_r$. It is important to note that minor hand-tuning of $\mathbf{W}$ and $\mathbf{V}_r$ was required offline to obtain the results in this chapter. However, this hand-tuning was only required for DS1 and DS6. The filter gains for DS6 are different because DS6 included extra measurements of ϕ, θ, and h that were obtained from the normal and height of the river (see Section 2.2.7). The filter gains for DS2, DS3, and DS4 are the same as DS1. Additionally, the filter gains for DS5 are different from the filter gains of the other datasets because DS5 used a completely different sensor platform (DS5 is a

sequence taken from the KITTI dataset). We obtained the filter gains for DS5 by hand-tuning $\mathbf{W}$ and $\mathbf{V}_r$ on a ground-truth sequence, taken from the KITTI dataset, that is different from DS5. This ground-truth sequence is approximately 307 meters in length. To obtain the filter gains, and all the other parameters required to run Curve SLAM in DS5, we ran Curve SLAM multiple times on this ground-truth sequence offline. After each run, we compared Curve SLAM's estimated attitude, body-frame velocities, and location to ground truth, and tuned all the parameters required to run Curve SLAM until Curve SLAM's estimates matched closely with ground truth. The filter gains, and parameters obtained in this process were then used in DS5. The final attitude, velocity, and localization results obtained with the tuned parameters are shown in Figures 4.39-4.40.

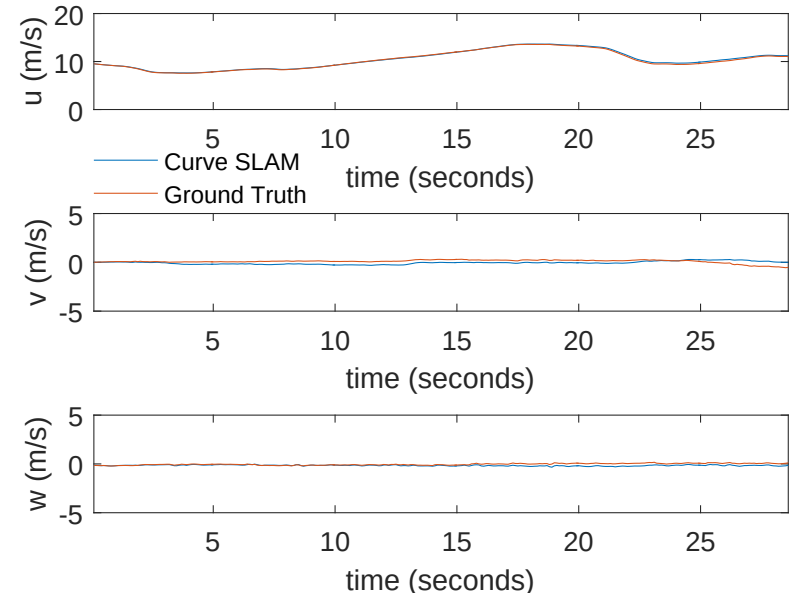
For the initial error covariance matrix $\boldsymbol{P}$, we initialize the robot pose block as $\boldsymbol{P}_{r_p,r_p} = \mathbf{0}^{9\times 9}$. Likewise, the initial error covariance between a control point and robot pose is initialized to zero, i.e., $\boldsymbol{P}_{r_p,j_p} = \mathbf{0}^{9\times 3}$. The error covariance between curve control points are initialized to the initial value of the measurement covariance $\mathbf{V}_0$.

In the initial frame, we obtain a parameter estimate of $\boldsymbol{\beta}$ for use in the Levenberg-Marquardt algorithm in four steps. First, we extract the path boundary. Second, we add curves in the image plane as described in Section 3.3.2. Third, we match curves between the stereo pair as described in Section 3.2.2. Fourth, our initial guess for $\boldsymbol{\beta}$ is obtained by triangulating the control points between matched curves.

In Section 3.2.1, the size ρ_n of the average filter window is $\rho_n = 5 \times 5$. To extract the boundary of the path, the image was processed in the hue, saturation, value (HSV) color space. Due to the stark contrast in hue between the concrete sidewalk and green grass, or the concrete road and yellow road lane (see Figures 4.1–4.4), there is a range of valid threshold values. Assuming a normalized image, we set the thresholds to find the sidewalk or yellow road lane to be between the following values: $\mathrm{H_{min}} = 0.09$, $\mathrm{H_{max}} = 0.5$, $\mathrm{S_{min}} = 0.15$, $\mathrm{S_{max}} = 1.0$, $\mathrm{V_{min}} = 0$, and $\mathrm{V_{max}} = 1$ for the hue, saturation, and value channels respectively in DS1-DS3. For DS4, we set the thresholds as $\mathrm{H_{min}} = 0.04, \mathrm{H_{max}} = 0.26$, $\mathrm{S_{min}} = 0.1$, $\mathrm{S_{max}} = 1.0$, $\mathrm{V_{min}} = 0$, and $\mathrm{V_{max}} = 1$. The size ρ_t of the template used to match and refine the location of curves in Section 3.2.2 is set as $\rho_t = 15 \times 15$ for the left image. In the right image, the size of the image patch is 17×20. We set ρ_r in Section 3.2.3 as $\rho_r = 5$



(a) Attitude estimates



(b) Body-frame velocity estimates

Figure 4.39: A ground-truth sequence used to obtain all the parameters required to run Curve SLAM in DS5. We ran Curve SLAM multiple times on this ground-truth sequence offline. After each run, we compared Curve SLAM's estimated attitude, body-frame velocities, and location to ground truth, and tuned all the parameters required to run Curve SLAM until Curve SLAM's estimates matched closely with ground truth. This figure shows Curve SLAM's final attitude and velocity estimates with all the parameters tuned. These tuned parameters were used in DS5.

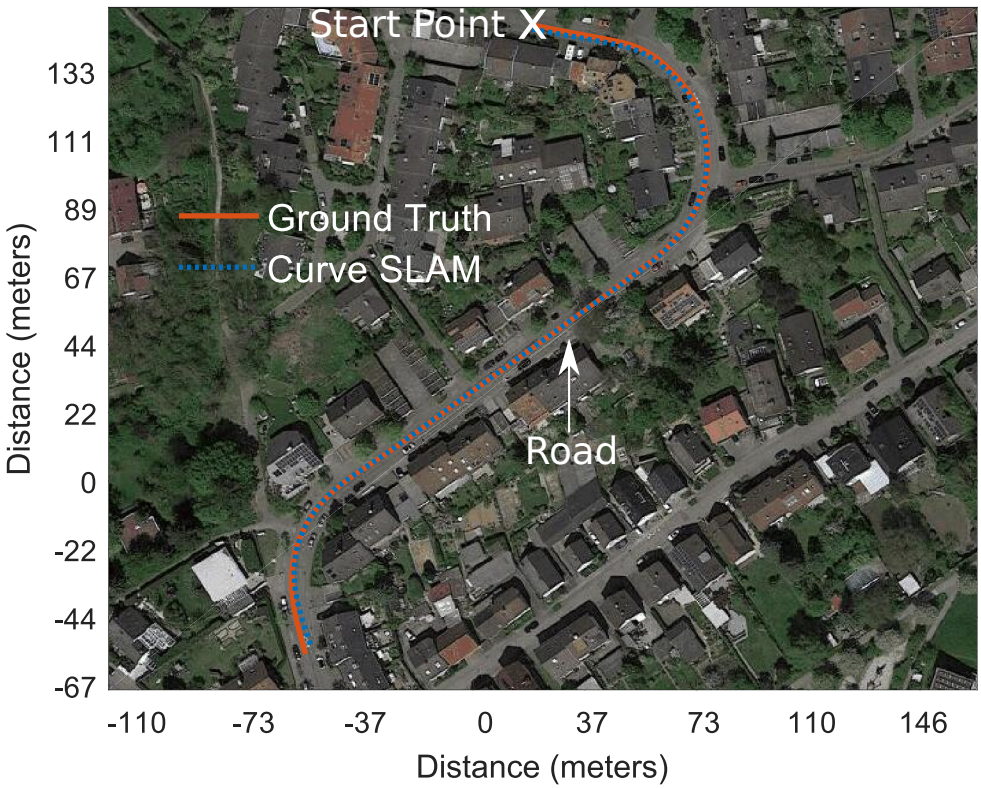


Figure 4.40: A ground-truth sequence used to obtain all the parameters required to run Curve SLAM in DS5. We ran Curve SLAM multiple times on this ground-truth sequence offline. After each run, we compared Curve SLAM's estimated attitude, body-frame velocities, and location to ground truth, and tuned all the parameters required to run Curve SLAM until Curve SLAM's estimates matched closely with ground truth. This figure shows Curve SLAM's final location estimate with all the parameters tuned. These tuned parameters were used in DS5.

pixels. We set ρ_s in Section 3.3.1 as $\rho_s = 100$ millimeters for DS1, DS2, DS3, and DS4, $\rho_s = 2.4$ meters in DS5, and $\rho_s = 1.3$ meters in DS6. To calculate The thresholds for the Mahalanobis distance tests in Section 3.3.1 are set at 2.5 standard deviations for DS1-DS5, and 6 standard deviations for DS6. In Section 3.3.2, we set the size of the image window ρ_w as $\rho_w = 16 \times 16$. Initially, we set ρ_k to 1.5 pixels. If no corner points are within 1.5 pixels of the path boundary, we progressively increase ρ_k from 2.5 pixels to 3.5 pixels until a corner point is found. If no corner points are in this range, we select the initial boundary point as the desired control point. The Shapiro-Wilk test in Section 3.3.3 is a parametric hypothesis test of normality. The null hypotheses is that a parameter is normal with unspecified mean and

variance. The significance level we implemented for the Shapiro-Wilk test is set at 0.05. When the path is not sufficiently smooth, the Shapiro-Wilk test had a tendency to over split the boundary. As described in Section 3.3.3, we incorporated a threshold parameter ρ_p to avoid over splitting the path. While we do not provide a systematic way of selecting ρ_p, we tested a range of values for ρ_p, between two and fifteen pixels, and observed no noticeable effects on the localization accuracy, and very minimal effects on the mapping results. We set ρ_p as $\rho_p = 10$ pixels. We set d in Section 3.3.8 as $d = 1$ meter.

4.6 Conclusion

In this chapter, we presented experimental results of the Curve SLAM algorithm, and compared our algorithm against SPTAM and OKVIS. We applied Curve SLAM to six environments. In the first three environments, a long winding sidewalk provided curve landmarks. In the fourth environment, road lanes provided curve landmarks. In the fifth environment, a road provided curve landmarks. In the sixth environment, the shoreline of a river provided curve landmarks. In the first, third, and fourth locations, Curve SLAM was more accurate than SPTAM and OKVIS because it was difficult to track feature points in these environments. In the second environment, SPTAM and OKVIS were slightly more accurate than Curve SLAM. This result is expected because point-based feature detector/extractor tracking algorithms will likely provide a more robust motion estimate than our tracking algorithm when feature points are readily available. In this regard, point-based approaches are not inferior. Indeed, recent point-based SLAM approaches provide precise motion estimates that run in real-time over long trajectories. Curve SLAM can be modified rather easily to include feature points so that curves and feature points can be used simultaneously to solve the SLAM problem. However, alternative approaches are required to localize and map settings that lack distinguishable feature points and to provide compact, structured maps of the environment. DS6 was a particularly hostile environment for SLAM. In DS6, the river shoreline was out of range of our stereo camera's depth sensing capabilities and our ability to localize and map this setting suffered accordingly. To mitigate this problem, we added absolute measurements of roll ϕ, pitch θ, and height h to our SLAM estimator's mea-

surement update step. We obtained these measurements directly from the stereo camera, and the normal and height of the water's surface (see Section 2.2.7). In all six environments, Curve SLAM required a small number of landmarks to map these settings. In fact, in our experimental evaluations, we observed that Curve SLAM was able to reduce the required number of landmark features by several orders of magnitude relative to SPTAM and OKVIS.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this chapter, we provide a summary and list of major contributions of this dissertation. Additionally, we recommend future work in the area of SLAM and river detection.

5.1 Summary and Contributions

In this dissertation, we presented a SLAM algorithm that uses parameterized Bézier curves as landmark primitives rather than feature points. Our approach allows us to create a map that recovers the outline, shape, and dimensions of unique objects in an environment with a small number of curve control points. We validated the performance of our SLAM algorithm extensively in numerous settings and compared Curve SLAM to two state of the art SLAM algorihtms: the open keyframe-based visual-inertial SLAM algorithm (OKVIS) and a stereo version of the parallel tracking and mapping algorithm (SPTAM). In addition to Curve SLAM, we presented a vision-based algorithm to detect the border of water in a river setting containing specular reflections and distinguishable symmetric feature points. Our water detection algorithm allowed us to use Curve SLAM to create a sparse structured map of a river.

5.1.1 River Boundary Detection

To detect a river, our algorithm relies on the observation that 3-D objects triangulated below the water's surface are reflections, not physically located in the world, while 3-D objects triangulated above the water's surface are the source of these reflections. Therefore, the boundary that separates water from land corresponds to the boundary in which dense 3-D stereo data are

either above or below the water's surface; we detect a river in an image by identifying this boundary. To do so, we presented an algorithm that robustly estimates the normal and height of the surface of water. Then, with the normal and height, we presented an algorithm to identify where dense 3-D stereo data are above or below the surface of water. We demonstrated the robustness of our algorithm by applying it to a video sequence obtained from the visual-inertial canoe dataset that is over 320 seconds long and roughly 319 meters in length. In fact, the robustness of our river detection algorithm allowed us to use it as an object recognition module for Curve SLAM in Chapter 3. Finally, we showed how to obtain a precise measurement of roll, pitch, and height from the water surface normal, and we demonstrated the accuracy of the roll and pitch measurements by comparing them to a precise roll and pitch ground-truth signal. Due to the precision of these measurements, we included them as measurements in the Curve SLAM algorithm in Chapter 4.

5.1.2 The Curve SLAM Algorithm

We presented a SLAM algorithm that uses Bézier curves as landmark primitives rather than feature points. Our approach allows us to create a sparse map that recovers the outline, shape, and dimensions of unique objects in the environment. We showed how to interpolate, split, and match curves in a stereo pair. This allowed us to reconstruct the 3-D location of curves, parameterized by a small number of control points. Then, we showed how to formulate and solve the SLAM problem using Bézier curves and an extended Kalman filter. To solve the SLAM problem, we presented a data association algorithm that compares the physical dimensions of curve landmarks between chronologically sequential stereo image frames to remove curve outliers. Additionally, we showed how to determine the polynomial order of curves, we formulated the prediction and measurement step of the extended Kalman filter, and we showed how to add curves to the SLAM state and inside the image plane. Finally, we presented a curve combining algorithm that further reduces the number of landmark states representing the map.

5.1.3 Summary of Experimental Results

We presented experimental results of the Curve SLAM algorithm, and compared our algorithm to SPTAM and OKVIS. We applied Curve SLAM to six environments. In the first three environments, a long winding sidewalk provided curve landmarks. In the fourth environment, road lanes provided curve landmarks. In the fifth environment, a road provided curve landmarks. In the sixth environment, the shoreline of a river provided curve landmarks. In the first, third, and fourth locations, Curve SLAM was more accurate than SPTAM and OKVIS because it was difficult to track feature points in these environments. In the second environment, SPTAM and OKVIS were slightly more accurate than Curve SLAM. This result is expected because point-based feature detector/extractor tracking algorithms will likely provide a more robust motion estimate than our tracking algorithm when feature points are readily available. In this regard, point-based approaches are not inferior. Indeed, recent point-based SLAM approaches provide precise motion estimates that run in real-time over long trajectories. Curve SLAM can be modified rather easily to include feature points so that curves and feature points can be used simultaneously to solve the SLAM problem. However, alternative approaches are required in order to localize and map settings that lack distinguishable feature points and to provide compact, structured maps of the environment. DS6 was a particularly hostile environment for SLAM. In DS6, the river shoreline was out of range of our stereo camera's depth sensing capabilities and our ability to localize and map this setting suffered accordingly. To mitigate this problem, we added measurements of roll ϕ, pitch θ, and height h to our SLAM estimator's measurement update step. We obtained these measurements directly from the stereo camera, and the normal and height of the water's surface (see Section 2.2.7). In all six environments, Curve SLAM required a small number of landmarks to map these settings. In fact, in our experimental evaluations, we observed that Curve SLAM was able to reduce the required number of landmark features by several orders of magnitude relative to SPTAM and OKVIS.

5.1.4 Summary of Contributions

The contributions of this dissertation can be summarized as follows:

- We presented an algorithm that interpolates, splits, and matches curves in a stereo pair, allowing us to reconstruct the 3-D location of curves that are parameterized by a small number of control points.

- We presented a data association algorithm that compares the physical dimensions of curve landmarks between chronologically sequential stereo image frames to remove curve outliers. Our data association algorithm is designed to find false associations when a small number of landmark curves are tracked between image frames, allowing Curve SLAM to operate in settings that lack distinguishable feature points, and to decrease the size of the SLAM state-space.

- We presented a curve combining algorithm that reduces the number of curve landmarks representing the map, allowing us to construct large maps with fewer landmark states than conventional point-based SLAM algorithms.

- We presented an online algorithm that segments the boundary of water in a river environment containing specular reflections.

- We showed how to formulate and solve an optimization problem that allows us to robustly estimate the normal and height of the water's surface, as well as the 3-D location of symmetric feature points.

- We showed how the normal and height of the water's surface can be used to obtain a precise measurement of roll, pitch, and height at every stereo frame. We showed how to fuse these measurements directly into our SLAM estimator to enhance the localization and mapping accuracy of an autonomous robot in a river setting.

- We showed how to find, match, and remove symmetric feature point outliers in a stereo camera that contains images of a river setting.

- We presented an algorithm to find the boundary separating 3-D data which are either above or below the surface of water, allowing us to extract the border of water from a stereo pair.

- We validated Curve SLAM and our water detection algorithm with experimental hardware in real-world settings. We applied Curve SLAM

to multiple outdoor environments, and we rigorously compared Curve SLAM against the open keyframe-based visual-inertial SLAM (OKVIS) [3] algorithm and a stereo version of the parallel tracking and mapping (SPTAM) algorithm [2, 4].

5.2 Recommendations for Future Work

There are two topics we recommend for future work. The first topic is to further harness the sparse structured nature of curves to solve problems directly related to SLAM. The second topic is to relax assumptions on our water detection algorithm, and increase its computational speed. Our recommendations for future work are as follows:

- Investigate how to use Curve SLAM to solve the long-term autonomous navigation problem. In particular, Curve SLAM can be used to localize a robot within a pre-existing map when the appearance of the environment changes drastically, e.g., at different times of the day, across seasonal changes, or in adverse environmental conditions. Since Curve SLAM recovers an object's shape, if an object's shape does not change between different passes of an environment, its shape can be compared between different passes of an environment to localize a robot within a pre-existing map.

- Use the structure and sparsity of curve control points to solve the multi-view bundle adjustment problem with curves as feature primitives instead of feature points. A small number of control points, rather than hundreds of feature points, can be used to speed up the computation time of a multi-view bundle adjustment problem while providing a motion estimate that is more accurate than the single-view stereo triangulation presented in Chapter 3.

- Increase the computational speed of our water detection algorithm. This can be done by exploiting the near-parallel nature of symmetric pairs. Indeed, instead of performing stereo-depth computations iteratively, i.e., one image patch at a time, the boundary of all image patches can be defined prior to performing stereo computations by exploiting

the near-parallel nature of symmetric pairs. Then, the stereo computations within all the defined image patches can be run in parallel. Another way to increase the computational speed is to use a feature detector/descriptor algorithm that is more efficient than SIFT.

- Relax the assumption that reflections are present.
- Compare our water detection algorithm against a state-of-the-art deep learning algorithm that has been trained to find the border of water.

APPENDIX A

PROOF OF 3-D CURVE RECONSTRUCTION

In this appendix, we prove that with the correct correspondence between a left and right image curve, we are able to estimate the 3-D location of this curve in the body frame.

A.1 Epipolar Geometry of Curves

Consider a 3-D curve $\mathbf{C}$ that projects to $\mathbf{C}_L$ and $\mathbf{C}_R$ in the left and right stereo images, see Figure A.1. From the perspective of the left image, the 3-D location of $\mathbf{C}$ could be located anywhere along the rays that project from the optical center $\mathbf{O}_L$ of the camera passing through the image plane of $\mathbf{C}_L$ to form the points comprising $\mathbf{C}$. With a curve correspondence between the left and right images, the 3-D curve lies at the intersection of the rays that project from the optical center of each camera. Thus, we can estimate the 3-D location of $\mathbf{C}$.

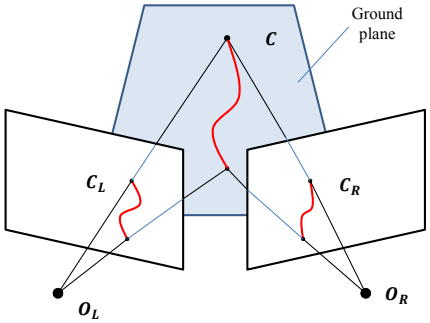


Figure A.1: A polynomial curve projected to a stereo image pair.

We can prove, under certain assumptions, that the preimage of two image curves is itself a curve in world co-ordinates. This proof is outlined below.

A.1.1 Proof of Curve Reconstruction

For this proof, we make the assumption that for a specific curve in $\mathbb{R}^3$, the map $f : \mathbb{R}^3 \to \mathbb{R}^2$ projecting the curve to an image is an isomorphism. This assumption only implies that the curve is fully observed in both images (i.e., a curve should not lose information and appear as a point or line when projected to the image).

Background:

For a smooth map between manifolds given by $f : X \to Y$, $y \in Y$ is a *regular value* of f if $\forall x \in f^{-1}(y)$, $df_x : T_xX \to T_yY$ is surjective. Here, T_xX and T_yY are the tangent spaces of X and Y at points x and y.

The Preimage Theorem: If $f : X \to Y$ is a smooth map, and $y \in Y$ is a regular value of f, then $M = \{x : x \in f^{-1}(y)\}$ is a submanifold of X, and the codimension of M in X is equal to the dimension of Y.

Proposition 1: Given a stereo image frame, and a curve observed in each image, the preimage is itself a curve in the world frame.

Proof:

We can define a curve in the left image as $f_L(u_L, v_L) = 0$. Under normalized perspective projection, $u_L = x/z$ and $v_L = y/z$. So, we can express this curve as $f_L(x/z, y/z) = 0$, $f_L : \mathbb{R}^3 \to \mathbb{R}^1$.

Then, the inverse image of 0 is given by

$$M_L = \{(x, y, z) \in \mathbb{R}^3 \mid f_L(x/z, y/z) = 0\}$$

and using the preimage theorem, M_L is a manifold in $\mathbb{R}^3$ with codimension 1. Thus, M_L is a 2-manifold, which can be represented in implicit form by the set:

$$M_L = \{(x, y, z) \in \mathbb{R}^3 \mid F_L(x, y, z) = 0\}$$

Similarly, if we define the curve in the right image as $f_R(u_r, v_r) = 0$, using a similar argument the inverse image of 0 in the right image is also a 2-manifold given by

$$M_R = \{(x, y, z) \in \mathbb{R}^3 \mid F_R(x, y, z) = 0\}$$

Consider now the function $F : \mathbb{R}^3 \to \mathbb{R}^2$ given by

$$F(x, y, z) = \begin{bmatrix} F_L(x, y, z) \\ F_R(x, y, z) \end{bmatrix}$$

The inverse image M of the stereo image curves is the intersection of the two surfaces M_L and M_R, or the set of points for which $F_L = F_R = 0$:

$$M = \{(x, y, z) \in \mathbb{R}^3 \mid F(x, y, z) = \mathbf{0}]$$

Since in this case, $F : \mathbb{R}^3 \to \mathbb{R}^2$, we can conclude using the preimage theorem that the inverse image of the point $[0, 0]^T$ will be a manifold of codimension 2 in $\mathbb{R}^3$ (i.e., a 1-manifold, or a curve).

APPENDIX B

HARDWARE PLATFORM

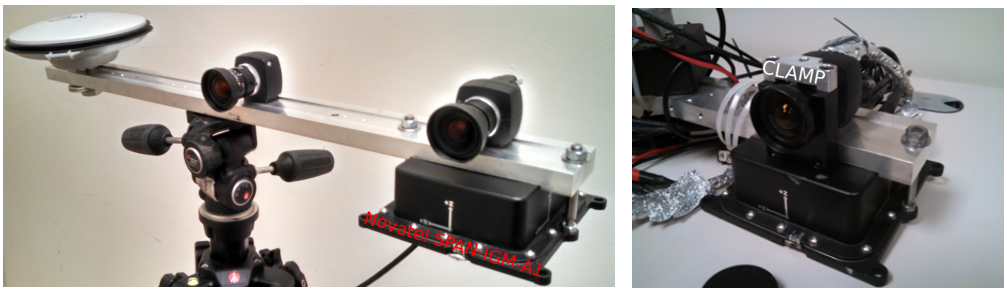
In this appendix, we provide a more thorough description of our sensor package, the calibration procedure, and setbacks that we encountered while collecting the experimental datasets in this dissertation.

B.1 Sensor Package

Our sensor platform consists of a stereo camera with two IDS-UI-3250ML color cameras equipped with an external hardware trigger and a Kowa JCM series 3.5 mm focal length lens, and a Novatel SPAN-IGM-A1 GNSS inertial navigation system equipped with a GPS and an Analog Devices ADIS 16488 IMU (see Figure B.1). The IMU includes a triaxis accelerometer and triaxis gyroscope. Further details about the individual hardware specifications of the cameras and Novatel SPAN-IGM-A1 can be obtained from their manufacturer's website [117, 118].

As shown in Figure B.1, all these sensors are rigidly attached to an aluminum plate to prevent external disturbances from disrupting the calibration. In particular, each of the cameras are fastened to the bottom of the aluminum plate using machine-precision screw slots, and a machine-precision lens-mount. Additionally, the back edge of the aluminum plate has a small lip the cameras rest against to further eliminate camera vibrations and ensure the stereo pair satisfies the fronto-parallel configuration. The bottom edge of the aluminum plate contains a machine-precision slot which, combined with two bolts and screws, are used to rigidly attach the Novatel SPAN-IGM-A1 to the bottom of the aluminum plate. There are various screw slots that allow the stereo pair to have different baselines. The baseline can be set as 12 cm, 24 cm, 36 cm, 48 cm, or 60 cm.

A necessary requirement of the sensors is that the stereo camera must be



(a) Sensor Package.

(b) Zoomed-in view of the camera's lens mount.

Figure B.1: Our sensor platform consists of a stereo camera with two IDS-UI-3250ML color cameras and a Novatel SPAN-IGM-A1 GNSS inertial navigation system equipped with a GPS and an Analog Devices ADIS 16488 IMU. All these sensors are rigidly attached to an aluminum plate with machine precision to prevent external disturbances from disrupting the calibration.

time synchronized by an external hardware trigger, and all measurements must be timestamped on the same clock. We used a single clock located on the Novatel SPAN-IGM-A1 to record all the sensor readings. The Novatel SPAN-IGM-A1 already records the timestamps of the gyroscope, accelerometer, and GPS measurements. The process to record camera timestamps is as follows: the cameras are triggered to capture an image by the Novatel SPAN-IGM-A1 which outputs a square wave signal that triggers the cameras to capture an image. When a stereo image is captured, each camera sends a square wave back to the Novatel SPAN-IGM-A1, which immediately records the timestamp of the acquired images.

B.2 Calibration Procedure

All the sensors must be calibrated so that measurements can be transformed to the same frame of reference; we used the body frame of the left camera as our frame of reference. To calibrate the sensors, we must determine the SE(3) transformation between the stereo pair, and the SE(3) transformation between the left camera and the Novatel SPAN-IGM-A1. To do so, we used Kalibr [111–113], an open-source calibration package. Although, to calibrate just the stereo pair, we experimented with various open-source software packages in ROS, Matlab, or Opencv that work with similar accuracy compared

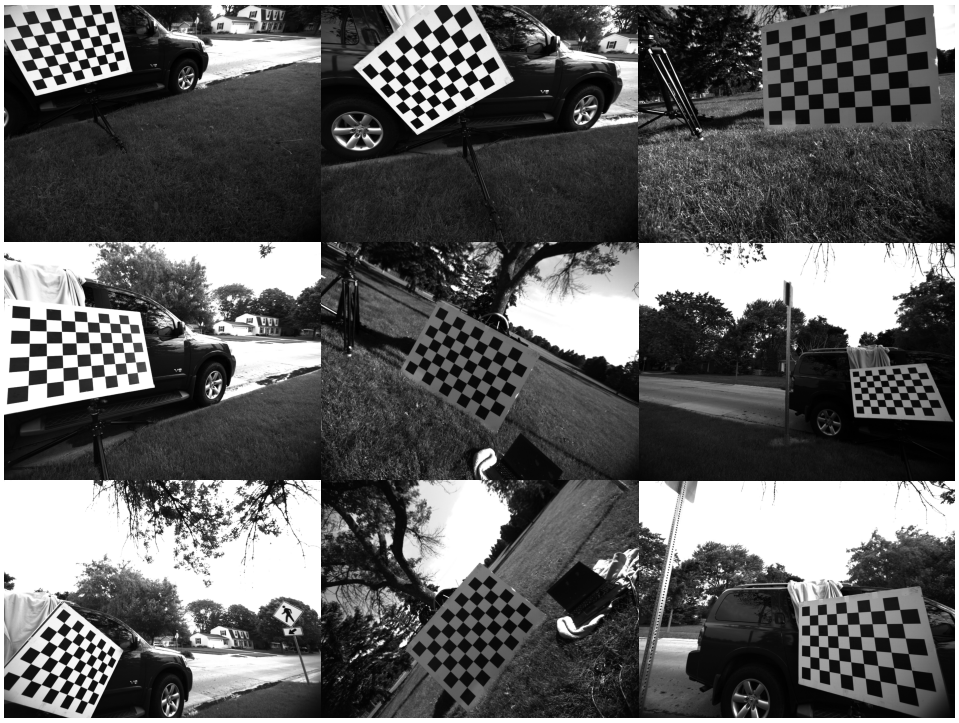


Figure B.2: Sample calibration images.

to Kalibr. Information regarding the algorithm used to calibrate the stereo pair can be found in [119].

The calibration procedure works by first calibrating the stereo pair together, followed by calibrating the stereo pair to the span Novatel SPAN-IGM-A1. The input required to calibrate the stereo pair is roughly 30 pictures of a checkerboard pattern that is visible in both the left and right camera, and the known distance between the corners of the checkerboard (this can be measured by a ruler). To obtain accurate results, it is imperative that the surface of the checkerboard pattern is planar (an assumption of the stereo calibration algorithm), the left and right images are time synchronized, fast movements of the camera are eliminated to avoid blurring the corners of the checkerboard pattern, the checkerboard is captured across the entire FOV, and the checkerboard is captured at varying depths and orientation, see Figure B.2 as an example.

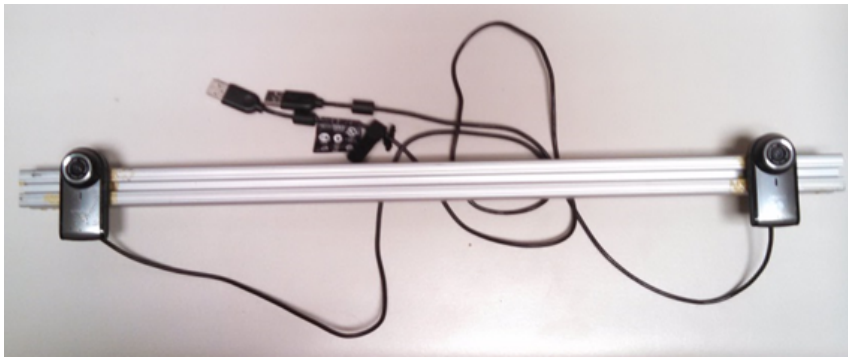
After obtaining the calibration between the stereo pair, we follow a similar procedure to calibrate the stereo pair to the IMU. In particular, the input required to calibrate the stereo pair to the IMU is a video sequence of checkerboard patterns, visible in both the left and right camera, that are

time synchronized to gyroscope and accelerometer measurements that are recorded from the Novatel SPAN-IGM-A1. Additionally, a second required input is the noise characteristics of the gyroscope and accelerometer: the angular random walk, velocity random walk, rate noise density, and acceleration noise density. We obtain these parameters directly from the data sheet of the IMU. Note, for a more thorough mathematical description of the gyroscope and accelerometer noise model see [114, 120, 121]. To obtain accurate results, it is imperative that all 6 degrees of freedom of the gyroscope and accelerometer are sufficiently excited while capturing the video sequence of the checkerboard patterns. Additionally, the suggestions on calibrating the stereo camera described in the previous paragraph should be followed as close as possible.

B.2.1 Lessons Learned about the Sensitivity of a Stereo Camera

While collecting datasets, we found the stereo camera to be particularly sensitive to vibrations, and when we first started collecting data for Curve SLAM, we were unaware of this sensitivity. Consequently, many early trips collecting stereo data taught us how to create a robust sensor platform rather then creating a usable dataset for Curve SLAM. Some of the first stereo cameras we used to acquire data are shown in Figure B.3, these cameras are not usable as a stereo camera for two main reasons: they either do not have a hardware trigger to synchronize the left and right image, or they are not resistant to vibration. We checked this with two steps: first, immediately prior to data collection, we calibrated the cameras and ensured the rectified stereo pair was row aligned. Second, we collected data, and checked again if the rectified stereo pair was row aligned after data collection. Oftentimes, the calibration would be off, resulting in erroneous and unusable depth estimates.

The work by Dang et al. [122] suggests that the orientations of a stereo camera must be known with an accuracy of 0.01-0.001 degrees. To reach this conclusion, they [122] did a thorough sensitivity analysis of an ideal stereo camera (by ideal we mean a stereo camera in which both optical-axis are parallel and both images are perfectly row aligned), and they showed that the uncertainty of a depth estimate is functionally dependent on the



(a) A stereo camera that is not time synchronized.

(b) A stereo camera that is sensitive to vibrations.

Figure B.3: The first stereo cameras we used to acquire data. These cameras are not usable as a stereo camera for two main reasons: they either do not have a hardware trigger to time synchronize the left and right image, or they are not resistant to vibration.

uncertainty of the orientation between the stereo pair multiplied by the depth squared. Indeed, we repeat one equation from their work which describes this sensitivity:

$$\Delta Z = -\frac{Z^2}{b}(1 + x_L^2)\Delta\psi \tag{B.1}$$

where ΔZ represents a perturbation from the true depth Z, $\Delta\psi$ represents a perturbation from the true yaw angle ψ between the stereo pair, b is the baseline and $x_L \in [0, 1]$ represents normalized image coordinates. Equation (B.1) demonstrates that the orientations of a stereo camera must be known with an accuracy of 0.01-0.001 degrees. Furthermore, Equation (B.1) shows that if the cameras are not stationary, they must be time synchronized, and minor camera vibrations disrupting the calibration cannot be tolerated. Equation (B.1) also explains why our final sensor package was constructed with a high level of precision, see Section B.1.

APPENDIX C

SUPPLEMENTARY FILES

The supplementary file DissertationCurveSLAM.mp4 is a video recording that demonstrates experimental results of the Curve SLAM algorithm applied to the six datasets described in Chapter 4. The supplementary file DissertationShoreline.mp4 is a video recording that demonstrates experimental results of the shoreline detection algorithm described in Chapter 2.

REFERENCES

[1] A. L. Rankin, L. H. Matthies, and A. Huertas, "Daytime water detection by fusing multiple cues for autonomous off-road navigation," in *24th Army Science Conference, ASC '04*, 2004.

[2] T. Pire, T. Fischer, J. Civera, P. D. Cristóforis, and J. J. Berlles, "Stereo parallel tracking and mapping for robot localization," in *Proceedings of The International Conference on Intelligent Robots and Systems (IROS)*, pp. 1373–1378, October 2015, Hamburg, Germany.

[3] S. Luetenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, March 2015.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, November 2007, Washington D.C.

[5] V. Nguyen, A. Harati, and R. Siegwart, "A lightweight SLAM algorithm using orthogonal planes for indoor mobile robotics," in *International Conference on Intelligent Robots and Systems*, pp. 658–663, October and November 2007.

[6] A. Gee, D. Chekhlov, W. Calway, and W. Mayol-Cuevas, "Discovering higher level structure in visual SLAM," *IEEE Transactions on Robotics*, vol. 24, pp. 980–990, October 2008.

[7] P. Ozog, N. Carlevaris-Bianco, A. Kim, and R. M. Eustice, "Long-term mapping techniques for ship hull inspection and surveillance using an autonomous underwater vehicle," *Journal of Field Robotics*, vol. 33, no. 3, pp. 265–289, January 2015.

[8] A. Dani, G. Panahandeh, S.-J. Chung, and S. Hutchinson, "Image moments for higher-level feature based navigation," in *International Conference on Intelligent Robots and Systems*, November 2013, Tokyo, Japan. pp. 602–609.

[9] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-D line-based map using stereo SLAM," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1364–1377, December 2015.

[10] P. Smith, I. Reid, and A. Davison, "Real-time monocular SLAM with straight lines," in *Proceedings of the British Machine Vision Conference*, pp. 17–26, September 2006.

[11] Y. Lu and D. Song, "Visual navigation using heterogeneous landmarks and unsupervised geometric constraints," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 736–749, June 2015.

[12] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1352–1359, June 2013, Portland, Oregon.

[13] S. Nuske, S. Choudhury, S. Jain, A. Chambers, L. Yoder, S. Scherer, L. Chamberlain, H. Cover, and S. Singh, "Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers," *Journal of Field Robotics*, vol. 32, no. 8, pp. 1141–1162, February 2015.

[14] J. Yang, A. Dani, S.-J. Chung, and S. Hutchinson, "Vision-based localization and robot-centric mapping in riverine environments," *Journal of Field Robotics*, vol. 34, no. 3, pp. 429–450, May 2015.

[15] C. Schmid and A. Zisserman, "The geometry and matching of lines and curves over multiple views," *International Journal of Computer Vision*, vol. 40, no. 3, pp. 199–233, 2000.

[16] K. Kedem and Y. Yarmovski, "Curve based stereo matching using the minimum Hausdorff distance," in *Proeedings of the Twelfth Annual Symposium on Computational Geometry*, pp. 415–418, 1996.

[17] Y. Xiao and Y. Li, "Optimized stereo reconstruction of free-form space curves based on a nonuniform rational B-spline model," *Journal of the Optical Society of America*, vol. 22, no. 9, pp. 1746–62, September 2005.

[18] L. Pedraza, D. Rodriguez-Losada, F. Matia, G. Dissanayake, and J. V. Miro, "Extending the limits of feature-based SLAM with B-splines," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 353–366, 2009.

[19] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: a survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, February 2016.

[20] 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, September/October 2010.

[21] M. Paton, F. Pomerleau, K. MacTavish, C. J. Ostafew, and T. D. Barfoot, "Expanding the limits of vision-based localization for long-term route-following autonomy," *Journal of Field Robotics*, vol. 34, no. 1, pp. 98–122, January 2017.

[22] C. McManus, B. Upcroft, and P. Newman, "Learning place-dependant features for long-term vision-based localisation," *Autonomous Robots, Special Issue on Robotics Science and Systems 2014*, vol. 39, no. 3, pp. 363–387, October 2015.

[23] C. Linegar, W. Churchill, and P. Newman, "Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, Stockholm, Sweden. pp. 787–794.

[24] M. Kaess, H. Johannsson, R. Roberts, V. IIa, J. J. Leonard, and F. Dellaert, "iSAM2: incremental smoothing and mapping using the Bayes tree," *International Journal of Robotics Research (IJRR)*, vol. 31, no. 2, pp. 216–235, February 2012.

[25] N. Keivan and G. Sibley, "Asynchronous adaptive conditioning for visual-inertial SLAM," *International Journal of Robotics Research (IJRR)*, vol. 34, no. 13, pp. 1573–1589, November 2015.

[26] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3565–3572, April 2007, Roma, Italy.

[27] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, May 2013.

[28] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: a scalable real-time causal approach," *The International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011.

[29] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, January 2011.

[30] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1066–1077, October 2008.

[31] J. Kim, K.-J. Yoon, and I. S. Kweon, "Bayesian filter for keyframe-based visual SLAM," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 517–531, April 2015.

[32] A. L. Rankin, L. H. Matthies, and P. Bellutta, "Daytime water detection based on sky reflections," *2011 IEEE International Conference on Robotics and Automation*, pp. 5329–5336, May 2011.

[33] S. Achar, B. Sankaran, S. Nuske, S. Scherer, and S. Singh, "Self-supervised segmentation of river scenes," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, pp. 6227–6232, May 2011.

[34] A. Rankin and L. Matthies, "Daytime water detection based on color variation," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 215–221, October 2010.

[35] G. Loy and J.-O. Eklundh, *Detecting Symmetry and Symmetric Constellations of Features*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 508–521.

[36] K. Köser, C. Zach, and M. Pollefeys, "Dense 3D reconstruction of symmetric scenes from a single image," in *Proceedings of the 33rd International Conference on Pattern Recognition*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 266–275.

[37] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: State estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, no. 1, 2012.

[38] L. Yang, J. Liu, and X. Tang, "Depth from water reflection," *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1235–1243, April 2015.

[39] P. Mettes, R. T. Tan, and R. C. Veltkamp, "Water detection through spatio-temporal invariant descriptors," *Computer Vision and Image Understanding*, vol. 154, pp. 182–191, January 2017.

[40] S. Fazekas, T. Amiaz, D. Chetverikov, and N. Kiryati, "Dynamic texture detection based on motion analysis," *International Journal of Computer Vision*, vol. 82, no. 1, p. 48, 2008.

[41] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.

[42] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909–926, May 2008.

[43] J. Chen, G. Zhao, M. Salo, E. Rahtu, and M. Pietikainen, "Automatic dynamic texture segmentation using local descriptors and optical flow," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 326–339, January 2013.

[44] A. Ravichandran, R. Chaudhry, and R. Vidal, "Categorizing dynamic textures using a bag of dynamical systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 342–353, February 2013.

[45] F. D. Snyder, D. D. Morris, P. H. Haley, R. Collins, and A. M. Okerholm, "Autonomous River Navigation," in *Proceedings of SPIE, Mobile Robots XVII*, D. W. Gage, Ed., vol. 5609, Orlando, FL, March 2004, pp. 221–232.

[46] J. Yao, S. Ramalingam, Y. Taguchi, Y. Miki, and R. Urtasun, "Estimating drivable collision-free space from monocular video," in *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*, January 5-9 2015, Waikoloa, HI, USA.

[47] G. Hitz, F. Pomerleau, M. E. Garneau, C. Pradalier, T. Posch, J. Pernthaler, and R. Y. Siegwart, "Autonomous inland water monitoring: Design and application of a surface vessel," *IEEE Robotics Automation Magazine*, vol. 19, no. 1, pp. 62–72, March 2012.

[48] A. Valada, P. Velagapudi, B. Kannan, C. Tomaszewski, G. Kantor, and P. Scerri, "Development of a low cost multi-robot autonomous marine surface platform," in *Field and Service Robotics*, ser. Springer Tracts in Advanced Robotics, K. Yoshida and S. Tadokoro, Eds., vol. 92. Springer, Berlin, Heidelberg, 2014, pp. 643–658.

[49] M. Dunbabin, A. Grinham, and J. Udy, "An autonomous surface vehicle for water quality monitoring," in *Australian Conference on Robotics and Automation*, December 2009, Sydney, Australia.

[50] J. C. Leedekerken, M. F. Fallon, and J. J. Leonard, *Mapping complex marine environments with autonomous surface craft.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 525–539. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28572-1_36

[51] S. Rathinam, P. Almeida, Z. Kim, S. Jackson, A. Tinka, W. Grossman, and R. Sengupta, "Autonomous searching and tracking of a river using an UAV," in *2007 American Control Conference*, pp. 359–364, July 2007.

[52] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000029664.99615.94

[53] S. Sinha, K. Ramnath, and R. Szeliski, "Detecting and reconstructing 3D mirror symmetric objects," in *Proceedings of the 12th European Conference on Computer Vision (ECCV)*. Springer, October 2012. [Online]. Available: https://www.microsoft.com/en-us/research/publication/detecting-and-reconstructing-3d-mirror-symmetric-objects/

[54] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 1849–1856.

[55] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.

[56] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[57] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org, 2015.

[58] K. Meier, S.-J. Chung, and S. Hutchinson, "Visual-inertial curve simultaneous localization and mapping: Creating a sparse structured world without feature points," *Journal of Field Robotics; 00:1-29*, 2017, https://doi.org/10.1002/rob.21759.

[59] K. Meier, S. J. Chung, and S. Hutchinson, "Visual-inertial curve SLAM," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, October 2016, pp. 1238–1245.

[60] M. J. Hannah, "Computer matching of areas in stereo images," Ph. D. dissertation, Stanford University, 1974.

[61] R. C. Bolles, H. Baker, and M. J. Hannah, "The JISCT stereo evaluation," in *Image Understanding Workshop*, pp. 263–274, April 1993.

[62] H. Hirschmuller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1582–1599, September 2009.

[63] V. Vaish, M. Levoy, R. Szeliski, C. L. Zitnick, and S. B. Kang, "Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, ser. CVPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2331–2338.

[64] X. Hu and P. Mordohai, "A quantitative evaluation of confidence measures for stereo vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2121–2133, November 2012.

[65] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, "Review of stereo vision algorithms and their suitability for resource-limited systems," *Journal of Real-Time Image Processing*, vol. 11, no. 1, 2016.

[66] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1068–1080, June 2008.

[67] D. Scharstein and R. Szelinski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 7–42, April 2002.

[68] T. Kanade, "Development of a video-rate stereo machine," in *Image Understanding Workshop*, pp. 549–557, November 1994, Monterey, CA.

[69] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proceedings of the Third European Conference on Computer Vision (Vol. II)*, ser. ECCV '94. Stockholm, Sweden: Springer-Verlag New York, Inc., 1994, pp. 151–158.

[70] J. Xu, Q. Yang, J. Tang, and Z. Feng, "Linear time illumination invariant stereo matching," *International Journal of Computer Vision*, vol. 119, no. 2, pp. 179–193, 2016.

[71] Y. S. Heo, K. M. Lee, and S. U. Lee, "Robust stereo matching using adaptive normalized cross-correlation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 807–822, 2011.

[72] Y. S. Heo, K. M. Lee, and S. U. Lee, "Joint depth map and color consistency estimation for stereo images with different illuminations and cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1094–1106, May 2013.

[73] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, February 2008.

[74] M. J. Black and P. Anandan, "The robust estimation of multiple motion: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75–104, January 1996.

[75] D. Scharstein and R. Szeliski, "Stereo matching with nonlinear diffusion," *International Journal of Computer Vision*, vol. 28, no. 2, pp. 155–174, June 1998.

[76] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 600–608, 2004.

[77] D. G. Jones and J. Malik, "A computational framework for determining stereo correspondence from a set of linear spatial filters," in *Computer Vision — ECCV'92*, G. Sandini, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 395–410.

[78] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 401–406, April 1998.

[79] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 5695–5703.

[80] E. Tola, V. Lepetit, and P. Fua, "DAISY: An efficient dense descriptor applied to wide baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 815–830, May 2010.

[81] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Computer Vision – ACCV 2010*, R. Kimmel, R. Klette, and A. Sugimoto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 25–38.

[82] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *IEEE Intelligent Vehicles Symposium*, June 2011, Baden Germany.

[83] T. Dekel, S. Oron, M. Rubinstein, S. Avidan, and W. T. Freeman, "Best-buddies similarity for robust template matching," in *Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[84] D. T. Vu, B. Chidester, H. Yang, M. N. Do, and J. Lu, "Efficient hybrid tree-based stereo matching with applications to postcapture image refocusing," *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3428–3442, August 2014.

[85] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1858–1865, October 2008.

[86] M. F. Tappen and W. T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters," in *Proceedings Ninth IEEE International Conference on Computer Vision*, vol. 2, October 2003, pp. 900–906.

[87] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, September 2004.

[88] J. Lu, Y. Li, H. Yang, D. Min, W. Eng, and M. Do, "PatchMatch filter: Edge-aware filtering meets randomized search for visual correspondence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1866–1879, September 2017.

[89] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, December 1959.

[90] M. Miller, S.-J. Chung, and S. Hutchinson, "The visual-inertial canoe dataset," *The International Journal of Robotics Research*, Under Review.

[91] O. Bernard, D. Friboulet, P. Thevenaz, and M. Unser, "Variational B-spline level-set: A linear filtering approach for fast deformable model evolution," *IEEE Transactions on Image Processing*, vol. 18, no. 6, pp. 1179–1191, 2009.

[92] T. Dietenbeck, M. Alessandrini, D. Friboulet, and O. Bernard, "CREASEG: A free software for the evaluation of image segmentation algorithms based on level-set," in *2010 17th IEEE International Conference on Image Processing (ICIP)*, Hong Kong, China, 2010, pp. 665–668.

[93] T. Chan and L. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, February 2001.

[94] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[95] S. Lankton and A. Tannenbaum, "Localizing region-based active contours," *IEEE Transactions on Image Processing*, vol. 17, no. 11, pp. 2029–2039, 2008.

[96] C. Li, C. Kao, J. Gore, and Z. Ding, "Minimization of region-scalable fitting energy for image segmentation," *IEEE Transactions on Image Processing*, vol. 17, no. 10, pp. 1940–1949, 2008.

[97] Y. Shi and W. C. Karl, "A real-time algorithm for the approximation of level-set-based curve evolution," *IEEE Transactions on Image Processing*, vol. 17, no. 5, pp. 645–656, 2008.

[98] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "MultiNet: Real-time joint semantic reasoning for autonomous driving," *arXiv preprint arXiv:1612.07695*, 2016.

[99] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. Springer-Verlag New York, INC. New York, NY, USA, 1997.

[100] D. Salomon, *Curves and Surfaces for Computer Graphics*. Springer Science+Business Media, Inc., New York, 2006.

[101] T. G. Berry and R. R. Patterson, "The uniqueness of Bézier control points," *Computer Aided Geometric Design*, vol. 14, no. 9, pp. 877–879, December 1997.

[102] R. R. Patterson, "Projective transformations of the parameter of a Bernstein-Bézier curve," *ACM Transactions on Graphics*, vol. 4, no. 4, pp. 276–290, October 1985.

[103] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary image by border following," *Computer Vision Graphics and Image Processing*, vol. 30, no. 1, pp. 32–46, April 1985.

[104] M. Khan, "Approximation of data using cubic Bézier curve least square fitting," http://hipp.certec.lth.se/trac/raw-attachment/wiki/BezierPaths/, 2007.

[105] H. Wang, J. Kearney, and K. Atkinson, "Arc-length paramaterized spline curve for real-time simulation," in *Proc. 5th International Conference on Curves and Surfaces*, June 2002, pp. 387–396.

[106] M. Walter and A. Fournier, "Approximate arc length parameterization," in *Proceedings of the 9th Brazilian Symposium on Computer Graphics and Image Processing*, October 1996, pp. 143–150.

[107] R. H. Myers, D. C. Montgomery, G. G. Vining, and T. J. Robinson, *Generalized Linear Models: With Applications in Engineering and the Sciences*, 2nd ed. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2010.

[108] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of the 1981 DARPA Imaging Understanding Workshop*, pp. 121–130, 1981.

[109] J. Shi and C. Tomasi, "Good features to track," in *9th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, June 1994, Seattle, WA.

[110] S. Shapiro and M. Wilk, "An analysis of variance test for normality," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, December 1965.

[111] P. Furgale, J. Rehder, and R. Siegwart, "Unified and spatial calibration for multi-sensor systems," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1280–1286, 3-7 November 2013, Tokyo, Japan.

[112] J. Maye, P. Furgale, and R. Siegwart, "Self-supervised calibration for robotic systems," in *IEEE Intelligent Vehicles Symposium*, pp. 473–480, June 2013.

[113] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions." in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2088–2095, May 14-18 2012, Saint Paul, Minnesota.

[114] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation: A tutorial for quaternion algebra," MARS Lab, University of Minnesota, Technical Report 2005-002, Rev. 57, March 2005.

[115] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, Rhode Island, June 2012, pp. 3354–3361.

[116] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the KITTI dataset," *International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, August 2013.

[117] http:/www.novatel.com/assets/Documents/Papers/SPAN-IGM-A1-PS.pdf.

[118] https://en.ids-imaging.com/store/products/cameras/usb-3-0-cameras/ueye-ml.html.

[119] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., September 2008.

[120] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge, Computer Laboratory, Tech. Rep., 2007.

[121] J. L. Crassidis, "Sigma-point Kalman filtering for integrated GPS and intertial naviation," in *AIAA Guidance, Navigation, and Control Conference*, August 2005, San Fransisco.

[122] T. Dang, C. Hoffman, and C. Stiller, "Continuous stereo self-calibration by camera parameter tracking," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1536–1550, July 2009.