Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2012		**00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Curveslam: Utilizing Higher Level Structure In Stereo Vision-Based Navigation	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Illinois at Urbana-Champaign,Urbana,IL, 61802**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Existing approaches to visual Simultaneous Localization and Mapping (SLAM) typically utilize points as visual feature primitives to represent landmarks in the environment. Since these techniques mostly use image points from a standard feature point detector, they do not explicitly map objects or regions of interest. Further, previous SLAM techniques that propose the use of higher level structures often place constraints on the environment, such as requiring orthogonal lines and planes. Our work is motivated by the need for different SLAM techniques in path and riverine settings, where feature points can be scarce and may not adequately represent the environment. Accordingly, the proposed approach uses B?ezier polynomial curves as stereo vision primitives and offers a novel SLAM formulation to update the curve parameters and vehicle pose. This method eliminates the need for point-based stereo matching, with an optimization procedure to directly extract the curve information in the world frame from noisy edge measurements. Further, the proposed algorithm enables navigation with fewer feature states than most point-based techniques, and is able to produce a map which only provides detail in key areas. Results in simulation and with vision data validate that the proposed method can be effective in estimating the 6DOF pose of the stereo camera and can produce structured, uncluttered maps. Monte Carlo simulations of the algorithm are also provided to analyze its consistency.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**60**	

CURVESLAM: UTILIZING HIGHER LEVEL STRUCTURE IN STEREO VISION-BASED NAVIGATION

BY

DUSHYANT RAO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Advisers:

Assistant Professor Soon-Jo Chung and Professor Seth Hutchinson

ABSTRACT

Existing approaches to visual Simultaneous Localization and Mapping (SLAM) typically utilize points as visual feature primitives to represent landmarks in the environment. Since these techniques mostly use image points from a standard feature point detector, they do not explicitly map objects or regions of interest. Further, previous SLAM techniques that propose the use of higher level structures often place constraints on the environment, such as requiring orthogonal lines and planes. Our work is motivated by the need for different SLAM techniques in path and riverine settings, where feature points can be scarce and may not adequately represent the environment. Accordingly, the proposed approach uses Bézier polynomial curves as stereo vision primitives and offers a novel SLAM formulation to update the curve parameters and vehicle pose. This method eliminates the need for point-based stereo matching, with an optimization procedure to directly extract the curve information in the world frame from noisy edge measurements. Further, the proposed algorithm enables navigation with fewer feature states than most point-based techniques, and is able to produce a map which only provides detail in key areas. Results in simulation and with vision data validate that the proposed method can be effective in estimating the 6DOF pose of the stereo camera, and can produce structured, uncluttered maps. Monte Carlo simulations of the algorithm are also provided to analyze its consistency.

To family, friends, and everyone else who has helped me over the years. To my advisors and peers, who have contributed to every bit of this thesis.

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research (ONR) under Award No. N00014-11-1-0088. This thesis benefitted from stimulating discussions and technical support of Jonathan Yong, Junho Yang, and Ashwin Dani.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

BA	Bundle Adjustment
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
GPS	Global Positioning System
IDP	Inverse Depth Parametrization
KF	Kalman Filter
LIDAR	Light Detection and Ranging
MAV	Micro Aerial Vehicle
RADAR	Radio Detection and Ranging
SFM	Structure From Motion
SLAM	Simultaneous Localization and Mapping
VO	Visual Odometry

CHAPTER 1

INTRODUCTION

A necessary capability for any autonomous vehicle is to progressively construct a map of its surroundings whilst localizing itself within the map, a real-time process known as the SLAM problem [1][2]. GPS has long been used to localize aerial vehicles in various applications, but may not always be available in remote areas, in outdoor environments that have heavy forest canopies, or even in some indoor environments. There exist a variety of alternative sensors that can be used to determine the range and bearing to landmarks in the environment, including laser range finders, RADAR / LIDAR systems, ultrasonic sensors, and stereo and monocular cameras. Nevertheless, favorable size, mass, and power consumption qualities of lightweight cameras make them very attractive for autonomous navigation.

A great deal of past work has focused on visual SLAM. For example, the original MonoSLAM algorithm [3] was able to estimate pose of a monocular camera and map a room, and indeed, our previous work [4][5][6] achieved navigation and mapping using a single camera in an orthogonal indoor environment. However, like most SLAM techniques, such prior works utilize points as landmarks, an approach with a number of drawbacks:

1. The landmarks represent image points, and may have no physical significance in the environment

2. There is no exploitation of higher level structure in the environment

3. Methods that can provide rich maps [7] require a much larger state space, and combinatorial data association is considerably more difficult.

From the perspective of robot motion planning, guidance, and control, it is desirable for the produced map to have some structure and for the landmarks to represent meaningful physical objects. Otherwise, there is no indication of whether landmarks represent obstacles, traversable regions, or points of interest in the environment.

Before presenting the curve-based SLAM algorithm, we begin by examining related work in Visual SLAM.

1.1 Related Work

Monocular vision is a difficult problem, in part because the projective geometry means that depth of a landmark along the axis of the camera (i.e., distance from the camera) cannot be estimated from a single measurement. Early research solved this problem by initializing the landmark after multiple measurements were made [3], while more recent approaches use an Inverse Depth Parametrization (IDP) to initialize the landmark after a single observation [8]. Other monocular approaches utilize the ground planar constraint of different environments to immediately initialize landmarks [4][6], but the methods still require a height measurement from an altimeter sensor. Another technique [9] allows the MonoSLAM method to be applied to larger scale environments, by starting new submaps once the original map grows to an unmanageable size, and then stitching the local maps together into a global map using the Hierarchal SLAM algorithm [10]. Their results show a successful loop closure in a dynamic urban environment, producing an accurate global map.

Stereo vision-based methods allow immediate initialization of landmarks, but can produce erroneous estimates for distant landmarks [11]. This can be improved by modelling the measurements as separate monocular measurements rather than a single stereo frame [12]. Visual odometry techniques have been widely explored, but can suffer from considerable drift, analogous to the drift of standard vehicle odometry. Recent research looks instead to combine visual odometry with monocular SLAM [13], using the "pose prior" obtained from visual odometry to strengthen the Visual SLAM result. Structure from Motion (SFM) techniques such as Bundle Adjustment (BA), can produce a very high level of accuracy, but often at greater computation expense. However, there has been recent research allowing for a real-time implementation using a sliding window bundle adjustment [14], and further work using the random sample consensus (RANSAC) algorithm and and EKF to optimize over a sliding window [15].

However, the algorithms outlined thus far, both mono-based and stereo-based, use feature points as landmarks. Consequently, they are susceptible to the drawbacks outlined earlier, and it is beneficial to consider using an approach that rec-

ognizes structure in the environment.

1.1.1 Higher Level Structure in Vision and SLAM

A number of related works have attempted to discover or utilize higher level structure in SLAM. Some of these assume an orthogonal environment and utilize lines or planes to obtain the SLAM estimate [16][17][18]. However, the work in [16] relies on dense landmark sets (such as a laser scan dataset) and only landmarks that are collinear or coplanar can be used in the SLAM pose estimate, as is the case in [17]. The approach outlined in [18] allows for higher structure in the environment using line segments that can be initialized immediately (analogous to IDP for points), but still requires a larger state space than would be needed for a curve-based approach. Another related work [19] parametrizes 3D lines using Plücker coordinates, suited to a pinhole camera projection model, and uses these in an EKF-based SLAM framework, while the work of [20] utilizes the orthogonality assumption of the indoor environment to map lines on the floor.

Some past works also combine object recognition with visual SLAM [21][22], incorporating objects of interest into the map. However, the work in [21] utilizes an IR scanner as well, while [22] uses orthogonal household objects rather than a more freeform primitive. In both works, the estimation algorithm is still point-based, and the object recognition algorithm is only used to cluster points and improve data association.

The use of curve structures in vision has also been considered in prior research. A number of early works propose the use of algebraic curve primitives in stereo vision rather than points [23][24], deriving closed form implicit expressions for an algebraic curve in Cartesian space given its projection in multiple images. However, while algebraic curves can more diversely describe a range of image shapes, they are difficult to use computationally since only certain polynomial varieties admit a parametric representation. Another work [25] also examines the multi-view relationships for non-algebraic curves, lines and conic sections, while a more recent approach utilizes an iterative technique with a Non-Uniform Rational B-Spline (NURBS) model to perform optimal stereo reconstruction of 3D curves [26]. A similar technique reconstructs smooth space curves with images of objects from different views [27], but assumes that the relative motion between cameras is known, rendering it unsuitable for SLAM. Another SFM-style algo-

rithm performs multi-view reconstruction of a set of curves, and impressive results are presented with a few different image sequences, including top-down imagery from an aerial vehicle [28]. However, as a batch update algorithm which requires the full sequence of images, it cannot be directly applied to real-time robot navigation. More significantly, it only considers curves that are fully observed in several image frames; for SFM this may be reasonable, but in SLAM, partial curves are frequently observed and need to be incorporated as measurements. One approach [29] uses basis curves in the application of road lane detection, but only considers the scenario of a ground vehicle travelling on a road with marked lanes, whereas a more general method in 6DOF is desirable for our application.

The most relevant prior work is that of Pedraza et al. [30][31], in which B-splines are applied to SLAM by using the control points of the curves as a description of the environment. They propose an EKF-SLAM based framework for navigation, with excellent results in multiple indoor environments. However, the algorithm is applied to a laser-based SLAM system, and the observation model utilizes point measurements in 3D position co-ordinates. Further work improves the observation covariance of the algorithm and yields more consistent results with a spline-based observation model [32].

Our proposed algorithm looks to build on their work in a number of ways. Firstly, our algorithm generalizes their approach to 6DOF, allowing for application in non-planar vehicles. Secondly, our application to Visual SLAM means that the algorithm could operate in a range of environments where laser ranging finding returns may not be available, such as in outdoor terrain. In doing so, we also provide a novel measurement parametrization that avoids feature points altogether in visual SLAM: the optimal curve parameters are found from a stereo image pair, and these curve parameters are used as the measurements in SLAM. Lastly, while Pedraza et al. utilize odometry in their approach (a reasonable assumption for a ground vehicle), our proposed technique utilizes vision as the only sensory input, thereby ensuring it can be used onboard any robotic platform, or indeed, a simple stereo camera rig.

1.1.2 Summary and Contributions

A large amount of related work exists in visual navigation and SLAM, but in general, most approaches utilize point features without regard to the structure of the

environment. Clearly, there are drawbacks to utilizing point-based methods in visual SLAM, and these characteristics limit the efficacy of a SLAM algorithm. By parametrizing landmarks in a way that explicitly considers higher level structures, it will be possible to produce structured maps with more meaningful landmarks, which could provide more useful information to the vehicle to aid path planning and control.

A number of past works have looked into utilizing higher level structure for SLAM, but most of these place orthogonality constraints on the environment or still utilize points, clustering them for object classification. A few works have looked into curve structures in vision, but mostly do not consider their application to SLAM. The work of [31] [32] develops a spline-based SLAM framework, but this is only for application to LIDAR-based SLAM for a planar vehicle, and vehicle odometry is also used.

The work presented in this thesis uses a Bézier curve-based landmark parametrization in a stereo vision-based SLAM framework. The proposed algorithm aims to achieve accurate localization and structured mapping by using cubic Bezier polynomial curves [33] as stereo vision primitives instead of feature points. This results in detailed continuous curves mapped through considerably fewer landmark states. Further, instead of explicitly matching points between left and right images, stereo matching is accomplished implicitly through a nonlinear function-fitting process.

To our knowledge, there exists no visual SLAM algorithm utilizing curve primitives. Our algorithm can perform localization and compactly represent the map of the environment as a set of curves. However, in contrast with [31] [29], our algorithm provides 6DOF estimation and can produce results in environments where laser-based techniques would fail. In particular, the novel contributions of the present thesis can be stated as follows:

- We propose a novel method to extract curve parameters, such as control points, from a stereo image pair, without conventional point-based stereo matching. This approach allows us to obtain an absolute measurement of roll, pitch, and height on each stereo frame.

- We propose a novel SLAM formulation utilizing curve structures as landmarks. By using curve structures, we are able to estimate the full 6DOF pose with much fewer landmarks than typical feature point-based methods, and can still produce a structured map, without odometry or use of any sen-

sor other than a camera rig. This is the first such method in existence in visual SLAM.

The remainder of this thesis is organized as follows. Chapter 2 provides a brief overview of the CurveSLAM algorithm; Chapter 3 details the parametrization of cubic Bézier curves and epipolar geometry of curves; Chapter 4 outlines the procedure for extracting curve parameters from stereo images; Chapter 5 derives the EKF-based SLAM formulation and data association method; Chapter 6 presents the simulation and experimental results; and Chapter 7 contains the conclusion and suggestions for ongoing and future work.

CHAPTER 2

CURVESLAM ALGORITHM OVERVIEW

The proposed approach utilizes planar cubic Bézier curves to represent each curve in the world frame. Each observed curve is projected to the left and right images as a series of detected edge points.

Instead of matching feature points between the left and right images, we run a nonlinear curve fitting algorithm to directly determine the Bézier curve parameters (refer to Chapter 3) in the world frame that best correspond to the projected curve point pixels. In this way, the need for stereo matching is eradicated, and the environment can be represented simply by a set of curves.

We also exploit the fact that the curves of interest are constrained to the ground plane. By imposing this planar constraint, it is possible to not only extract the curve parameters, but also a measurement of the height (z) from the camera to the ground plane, and the absolute orientation (pitch and roll) of the camera with respect to the ground plane, as will be explained in more detail in Chapter 4.

This assumption is not unrealistic; most environments have a ground that is close to planar, and results from this proposed algorithm as well as our previous work [6] demonstrate that variations in this planarity have a minimal impact on the effectiveness of the algorithm.

The process can be split into Curve Fitting (Chapter 4) and SLAM (Chapter 5) and is shown in Figure 2.1. The steps are summarized in pseudocode form in Algorithm 1, and are as follows:

1. We begin with a time-synchronized stereo image frame, perform edge detection to extract the curve edge points, and group the edge points into different curves.

2. The grouped edge points are then passed into a nonlinear Levenberg-Marquardt model fitting algorithm, which outputs the curve control points as well as pitch, roll, and height. The algorithm determines the set of parameters that

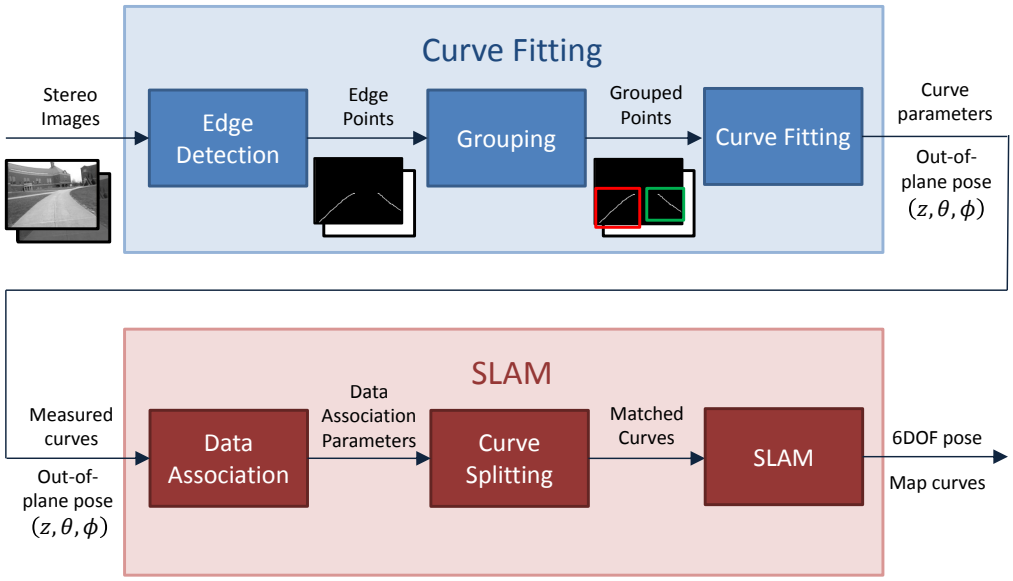

Figure 2.1: Flow diagram of the proposed CurveSLAM algorithm

minimize the pixel reprojection error when compared to the edge point measurements.

3. The curve parameters are considered as measurements for the SLAM procedure. First, data association is performed, determining the correspondence between the measured curves and the existing map curves.

4. Next, a curve splitting algorithm is applied to match the measured curves with segments of existing map curves.

5. The map is updated and the remaining pose variables (yaw, x, and y) are determined by using an EKF-based SLAM formulation.

This process constitutes one iterative loop of the proposed novel CurveSLAM algorithm, and is repeated on every successive stereo image frame. From each of these iterations, we can maintain the full 6DOF pose of the vehicle and the map curves representing the environment.

Algorithm 1 CurveSLAM

while New stereo image frame available **do**
  Run edge detection and group edge points into M different curves
  $\mathbf{y}$:= grouped edge points
  $\mathbf{p}_i$:= control points of i^{th} measured curve
  $\{z,\ \phi,\ \theta\}$:= {height, roll, pitch}
  $\boldsymbol{\beta} := \left[\mathbf{p}_1^T,\ ...,\ \mathbf{p}_M^T,\ z,\ \phi,\ \theta\right]^T$
  while $||\boldsymbol{\delta}|| > \epsilon$ **do**
    predict edge points $\mathbf{y}$ from parameters $\boldsymbol{\beta}$ with $\mathbf{y} = \mathbf{f}(\boldsymbol{\beta})$
    $\mathbf{J} := \frac{\partial \mathbf{f}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$
    $\boldsymbol{\delta} := \left(\mathbf{J}^T\mathbf{J} + \lambda\ \text{diag}\left(\mathbf{J}^T\mathbf{J}\right)\right)^{-1} \mathbf{J}^T\left(\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right)$
    $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \boldsymbol{\delta}$
  end while
  Perform EKF prediction for robot pose $\mathbf{x}_r$
  Find correspondences between M measured curves $\mathbf{p}_i$, and N existing map curves $\mathbf{x}_j$, with $i \in \{0, 1, ..., M\}$ and $j \in \{0, 1, ..., N\}$
  Split curves into segments for direct correspondence
  Add new states and apply EKF update to state vector $\mathbf{x} = \left[\mathbf{x}_r,\ \mathbf{x}_1,\ ...,\ \mathbf{x}_N\right]^T$
end while

CHAPTER 3

CURVE PARAMETRIZATION AND GEOMETRY

This chapter outlines the epipolar geometry of curves and introduces the Bézier curve parametrization used.

For this research, we utilize a calibrated stereo camera rig, and assume that both cameras have the same orientation (i.e., zero relative rotation), separated by a fixed baseline d.

3.1 Epipolar Geometry of Curves

For a calibrated stereo camera pair, a matched stereo measurement of a single point is enough to fully define the 3D point in the body frame of the camera. Three stereo matched points are enough to fully define a plane, with the assumption that the points are not collinear.

Similarly, a single planar curve measured in both left and right images is fully defined with respect to the body frame of the stereo camera rig. Consider the example of a planar polynomial curve $\mathbf{C}$ in 3D space, which projects to two different curves $\mathbf{C_L}$ and $\mathbf{C_R}$ in each of the stereo images (see Figure 3.1). In the inverse projective mapping, a curve in a single image represents a ruled surface projected out from the optical center $\mathbf{O}$ of the camera. If we have a second image of the same curve, the curve in 3D space is the intersection of the ruled surfaces projected from both images (Figure 3.1). Therefore, if the curve is observed in both images, we can uniquely determine the parameters of the curve, as well as the perpendicular distance to the plane containing the curve and the relative orientation between the camera and the plane. Since the curves considered in this thesis are all planar (on the ground plane), each frame gives us a measurement of height, pitch and roll, in addition to the control parameters defining the curve. Naturally, stereo vision is especially susceptible to error in the case of distant objects, so care must be taken to avoid curve features at a large distance.

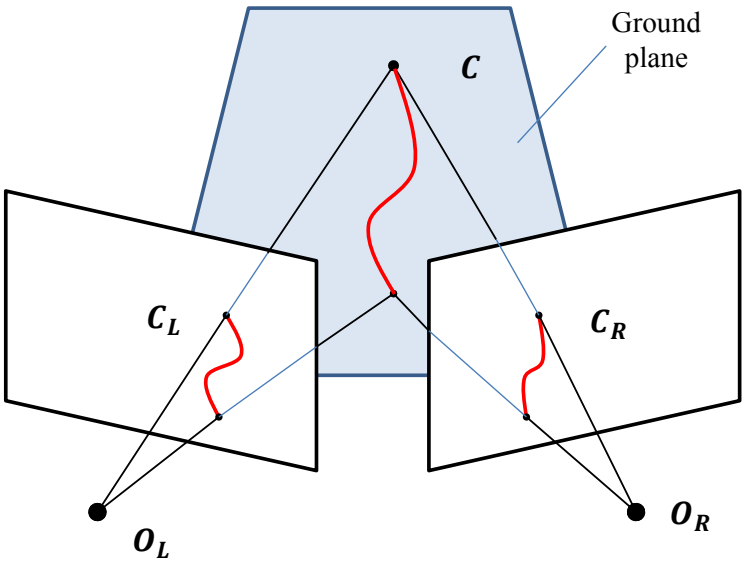

Figure 3.1: Planar Polynomial Curve projected to Stereo Images

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.

3.1.1 Proof of Curve Reconstruction

For this proof, we make the assumption that for a specific curve in $\Re^3$, the map $f : \Re^3 \to \Re^2$ projecting the curve to an image is an isomorphism. Practically, this assumption is not restrictive; it 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). In any case, such an assumption is necessary for visual curve-based SLAM.

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: 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 $f_l(x/z, y/z) = 0$, $f_l : \Re^3 \to \Re^1$.

Then, the inverse image of 0 is given by:

$$M_l = \{(x, y, z) \in \Re^3 \mid f_l(x/z, y/z) = 0\}$$

and using the Preimage Theorem, M_l is a manifold in $\Re^3$ with codimension 1. Thus, M_l is a 2-manifold, which can be represented in implicit form by the set:

$$M_l = \{(x, y, z) \in \Re^3 \mid F_l(x, y, z) = 0\}$$

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:

$$M_r = \{(x, y, z) \in \Re^3 \mid F_r(x, y, z) = 0\}$$

Consider now the function $F : \Re^3 \to \Re^2$ 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 $F_l = F_r = 0$:

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

Since in this case, $F : \Re^3 \to \Re^2$, 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 $\Re^3$ (i.e., a 1-manifold, or a curve).

3.2 Bézier Curve Characteristics

In the proposed algorithm, planar cubic Bézier curves are used to represent the ground planar curves. However, the algorithm itself is scalable to Bézier curves of any order.

The following properties of Bézier curves [33] make them favorable for this application:

1. A cubic Bézier curve can be represented by 4 control points or in parametric form with the position $b_x(t)$ and $b_y(t)$ on the ground plane defined on the interval $t \in [0, 1]$. The control points define the shape of the curve, which loosely follows the path between them, as shown in Figure 3.2.

2. Any affine transformation on a Bézier curve is equivalent to an affine transformation on the control points. Perspective projection is not affine, but for small distances from the camera, it can be approximated as such; this means that projecting a world curve to the left or right images is approximately equivalent to mapping the control points to the image planes.

3. The start and end points of the curve are well-defined; that is, unlike an implicit polynomial representation, the curve is only defined on an interval of the parameter t. This is a useful attribute, since only finite segments of a real world curve are observed.

4. The control points of any partial segment of the curve can be recovered by applying a linear transformation to the control points of the full curve, as will be shown in Section 5.1.1.

In fact, many of these properties are general to all Non-Uniform Rational B-Spline (NURBS) curves, but for the curve fitting method, specifying the parametrization is essential. A specific Bézier curve parametrization also means that the curve is fully specified with the control points (i.e., without a knot vector or weights) and the transformation between control points and polynomial coefficients (derived in the next section) is straightforward.

3.2.1 Notation and Definitions

In the general case of M existing Bézier curves, we will adopt the notation $\mathbf{p}_\ell^{(i)}$ with $\ell \in \{1, 2, ..., M\}$ and $i \in \{0, 1, 2, 3\}$ to refer to the i^{th} control point of

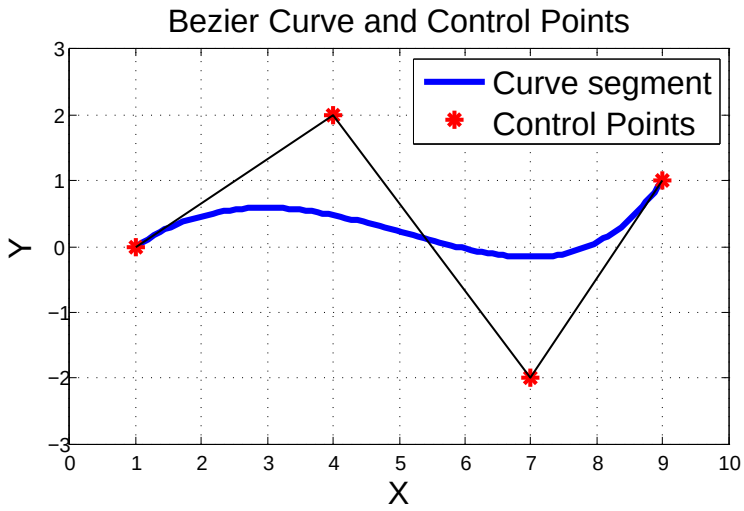

Figure 3.2: A sample cubic Bézier curve and its associated control points

the ℓ^{th} curve. We also use the absence of superscript (i) to represent the 4 control points of the ℓ^{th} curve with the *control point vector*:

$$\mathbf{p}_\ell = \left[p_x^{(0)}, p_x^{(1)}, p_x^{(2)}, p_x^{(3)}, p_y^{(0)}, p_y^{(1)}, p_y^{(2)}, p_y^{(3)}\right]_\ell^T \tag{3.1}$$

Then, a planar Bézier spline function $\mathbf{b}(t)$ which is cubic $(n = 3)$ with a control point vector $\mathbf{p}_\ell$ can be expressed as:

$$\mathbf{b}(t) = \begin{bmatrix} b_x(t) \\ b_y(t) \end{bmatrix} = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{p}_\ell^{(i)}, \quad t \in [0,1] \tag{3.2}$$

By varying t between 0 and 1, any point on the curve can easily be generated or rendered.

Further, the curve control points can be transformed to a set of parametric polynomial coefficients to express the planar Bézier curve in the form:

$$\mathbf{b}(t) = \begin{bmatrix} b_x(t) \\ b_y(t) \end{bmatrix} = \begin{bmatrix} \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3 \\ \alpha_4 + \alpha_5 t + \alpha_6 t^2 + \alpha_7 t^3 \end{bmatrix}, \quad t \in [0,1] \tag{3.3}$$

Rearranging the Bézier expressions, we can find the direct relationship for coefficients α_j as:

$$\alpha_j = \begin{cases} \dfrac{n!}{(n-j)!} \displaystyle\sum_{i=0}^{j} \dfrac{(-1)^{j+i} p_x^{(i)}}{i!(j-i)!} & \text{if } i < 4 \\ \dfrac{n!}{(n-j)!} \displaystyle\sum_{i=0}^{j} \dfrac{(-1)^{j+i} p_y^{(i)}}{i!(j-i)!} & \text{if } i \geq 4 \end{cases} \tag{3.4}$$

Expressing the coefficients as a vector $\boldsymbol{\alpha}_\ell = [\alpha_0,\, \alpha_1,\, \alpha_2,\, \alpha_3,\, \alpha_4,\, \alpha_5,\, \alpha_6,\, \alpha_7]_\ell^{\,T}$, the transformation between $\boldsymbol{\alpha}_\ell$ and $\mathbf{p}_\ell$ is linear, with Jacobian expressed using matrix $\mathbf{A} \in \Re^{4\times 4}$:

$$\mathbf{A} \quad : a_{ij} = \begin{cases} \dfrac{n!}{(n-i)!} \dfrac{(-1)^{i+j}}{j!(i-j)!} & \text{if } j \leq i \\ 0 & \text{if } j > i \end{cases} \tag{3.5}$$

$$\frac{d\boldsymbol{\alpha}_\ell}{d\mathbf{p}_\ell} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \tag{3.6}$$

The matrix $\mathbf{A}$ is repeated because the same transformation has to be applied separately to both the x and y coordinates of the control points to obtain the independent parametric polynomial coefficients in x and y. This form will arise frequently throughout the remainder of this thesis.

3.3 Camera Projection Model

For a Bézier curve being viewed by two cameras, we can easily approximate the equivalent projected curves in the left and right images by applying the camera projective transformation to the Bézier control points.

First, we define the frames of reference as follows (see Figure 3.3). The Body frame $\mathbf{f}_b$ is used to denote the frame fixed at the center of the stereo head, with the $\vec{x}, \vec{y}, \vec{z}$ axes following a Forward-Right-Down convention. The Local Ground frame $\mathbf{f}_g$ is the projection of the Body frame onto the ground plane. Finally, the Earth frame $\mathbf{f}_e$ is fixed as the inertial frame.

In other words, ϕ, θ and z (the "out-of-plane degrees of freedom") transform

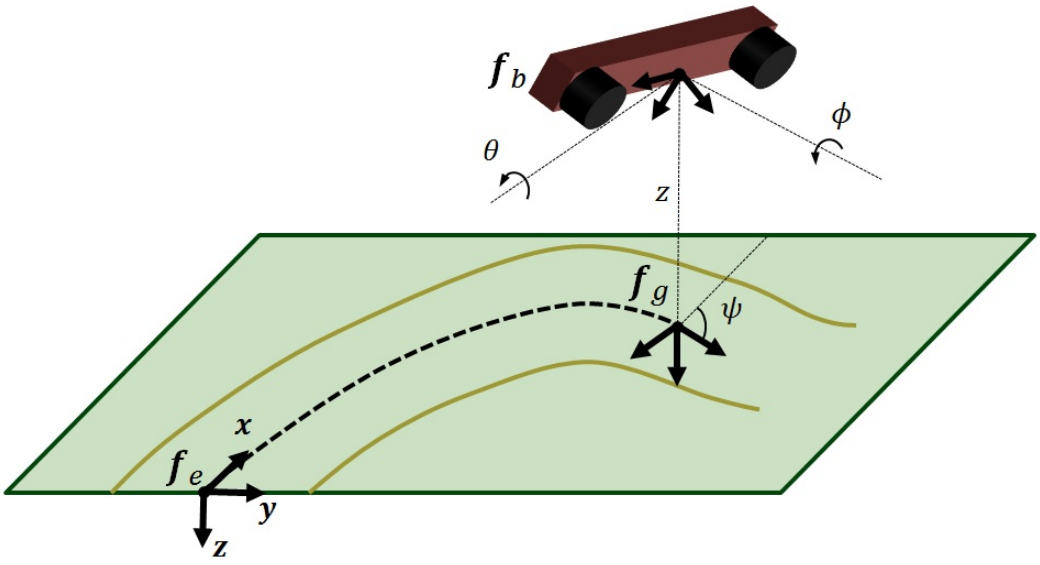

Figure 3.3: Graphical depiction of stereo rig and ground plane, with frames of reference shown

between the Body and Local ground frames, while x, y and ψ (the "planar degrees of freedom") transform between the Local Ground and Earth frames.

This selection of reference frames is natural for this application, because, as mentioned earlier, each stereo image frame only provides the out-of-plane degrees of freedom; the in-plane motion cannot be determined from a single frame.

In the defined frames, a point on the ground plane with Local Ground frame co-ordinates $\mathbf{x}_g = [x_g, y_g, 0]^T$ can then be mapped to a point in both the left and right image (u_l, v_l) and (u_r, v_r) using the simple transformation:

$$\mathbf{x}_b = \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \mathbf{R}_{gb}(\phi, \theta)^{-1}(\mathbf{x}_g - \mathbf{t}_{gb}(z)) \tag{3.7}$$

$$\begin{aligned} u_l &= \frac{f_u}{x_b} z_b + c_y & u_r &= \frac{f_u}{x_b} z_b + c_y \\ v_l &= \frac{f_v}{x_b}(y_b + \frac{d}{2}) + c_x & v_r &= \frac{f_v}{x_b}(y_b - \frac{d}{2}) + c_x \end{aligned} \tag{3.8}$$

where d is the baseline, f_x and f_y are the focal lengths in the x and y directions (in pixels), and c_x and c_y are the image x and y coordinates of the principal point (in pixels). $\mathbf{R}_{gb}(\phi, \theta)$ and $\mathbf{t}_{gb}(z)$ represent the rotation matrix and translation vector to transform a vector between the Body and Local Ground frames:

$$\begin{aligned} \mathbf{R}_{gb}(\phi,\theta) &= \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ \sin\theta\sin\phi & \cos\phi & \cos\theta\sin\phi \\ \sin\theta\cos\phi & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \\ \mathbf{t}_{gb}(z) &= \begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix} \end{aligned} \tag{3.9}$$

Using the second property in Section 3.2, a planar Bézier curve in the ground frame maps to another Bézier curve in both the left and right images, and the control points of the projected curves in the left and right images can be found by applying (3.7) and (3.8) to the control points of the curve in the Local Ground frame.

CHAPTER 4

CURVE FITTING

This chapter outlines the process of extracting the curve control points and out-of-plane pose parameters from each stereo image pair.

4.1 Edge Detection and Grouping

The first step for each stereo frame is to obtain edge points in both images corresponding to the curves on the ground plane. Then, once an edge detector has been applied, it is necessary to minimize the amount of noise pixels (corresponding to irrelevant edges) and group edge points into different curves.

For the purposes of this thesis, we are interested in mapping path environments, with map curves corresponding to the left and right edges of the path. Accordingly, we extract the edge points as follows. First, a Gaussian smoothing filter [34] is applied to the image to minimize noise in edge detection. Then, we exploit the lack of color saturation of the path itself, switching to HSV space and extracting the saturation image. Finally, we apply the Canny edge detector [35] on the saturation image, with a very high hysteresis / stitching threshold and a very low detection threshold, such that only a few significant edges are detected, but contain a large number of pixel points. With the edge points corresponding to either the left or right edges of the path, grouping is straightforward. Further, the edge points are pruned as necessary to ensure that the start and end points of each curve are approximately equivalent in the left and right images.

By applying this procedure, we obtain a series of edge points, grouped into different curves, in both the left and right images. Typical results of this method are shown in Figure 4.1.

It is important to note that the edge grouping requirements may vary depending on the robot's environment and the quantity of curve structures that need to be mapped. For instance, in the case of this thesis, we are most interested in map-

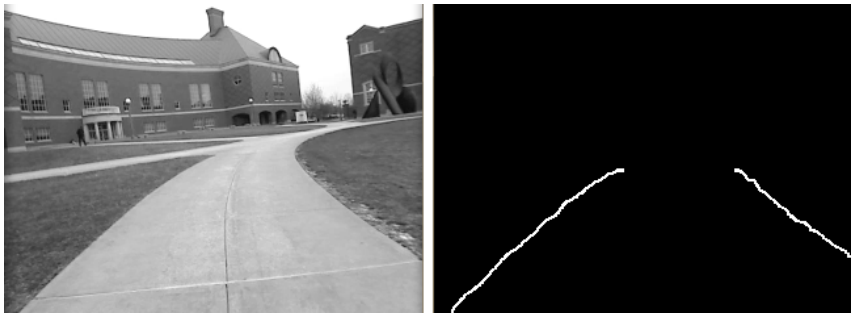

Figure 4.1: Result of Canny detection and edge grouping to extract path edge curves

ping path and riverine environments, where the edge points only correspond to a few different curves (the edges of the river or path). In contrast, in an indoor environment cluttered with objects, it may be necessary to group edge points into a large number of different structures.

Thus, depending on the requirement, this particular step of the algorithm may be altered, to use in conjunction with the proposed curve fitting and SLAM algorithms outlined in this chapter and the following chapters.

4.2 Nonlinear model fitting

Once these grouped edge points are obtained, the Levenberg-Marquardt algorithm [36] is applied to fit the camera projection model to these edge point measurements. With the model described by (3.2) - (3.8), the image measurements are purely a function of the parameters we attempt to estimate (the planar curve control points and the relative orientation between the stereo camera pair and the ground plane). More specifically, we have measurement vector $\mathbf{y}$ comprising n curve points in the left image and m curve points in the right image (grouped into M different curves), and the parameter vector $\boldsymbol{\beta}$:

$$\mathbf{y} = \left[u_{l_1}, v_{l_1}, ..., u_{l_n}, v_{l_n}, u_{r_1}, v_{r_1}, ..., u_{r_m}, v_{r_m}\right]^T$$
$$\boldsymbol{\beta} = \left[\mathbf{p}_1{}^T, \quad ..., \quad \mathbf{p}_M{}^T, \quad z_r, \quad \theta_r, \quad \phi_r\right]^T \tag{4.1}$$

Here, $\mathbf{p}_1$, ..., $\mathbf{p}_M$ represent the control point vectors of each of the M Bézier curves observed in the image, with co-ordinates defined in the Local Ground frame. The parameter variable z_r is the height of the robot above the ground

plane, and θ_r and ϕ_r are the pitch and roll, defined as Euler angles. Then, we attempt to fit the model $\mathbf{y} = \mathbf{f}(\boldsymbol{\beta})$ described by (3.2) - (3.8).

On each iteration, the planar orientation and the current curve parameters are used to determine the projected Bézier curves in the left and right images, using (3.8). The reprojection error for each measured edge pixel is computed as the distance between the pixel and the nearest point on the projected Bézier curve. The nearest point can be determined algebraically by setting the derivative of the distance to zero (see chapter 4.3.1). Then, each iteration k aims to minimize the total reprojection error, adding update $\boldsymbol{\delta}_k$ to the parameter vector $\boldsymbol{\beta}_k$:

$$\begin{aligned} \boldsymbol{\delta}_k &= \left(\mathbf{J}^T\mathbf{J} + \lambda \operatorname{diag}\left(\mathbf{J}^T\mathbf{J}\right)\right)^{-1} \mathbf{J}^T \left(\mathbf{y} - \mathbf{f}(\boldsymbol{\beta}_k)\right) \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \boldsymbol{\delta}_k \end{aligned} \tag{4.2}$$

The Jacobian $\mathbf{J} = \frac{\partial \mathbf{f}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ is the matrix of first-order partial derivatives of the measurements with respect to the parameters, and will be derived in the next section. The parameter λ is a damping parameter that was tuned to optimize the convergence characteristics of the Levenberg-Marquardt algorithm.

4.3 Jacobian Derivation

Given the measurement vector, $\mathbf{y} = [u_{l_1},\, v_{l_1},\, ...u_{l_n},\, v_{l_n},\, u_{r_1},\, v_{r_1},\, ...u_{r_m},\, v_{r_m}]^T$, and the parameter vector $\boldsymbol{\beta} = \left[\mathbf{p}_1{}^T,\, ...,\, \mathbf{p}_M{}^T,\, z,\, \theta,\, \phi\right]^T$, with model $\mathbf{y} = \mathbf{f}(\boldsymbol{\beta})$, the Jacobian to be determined is $\mathbf{J} = \frac{\partial \mathbf{f}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$.

Every measured point corresponds to a point in the Local Ground frame, which corresponds to a point on a curve. Suppose a measured point corresponds to curve γ, with control point vector $\mathbf{p}_\gamma$ and coefficient vector $\boldsymbol{\alpha}_\gamma$. Then, recalling the stereo camera projection equations defined in (3.8) and the parametrization for

cubic Bézier curves, we have the following relations:

$$\begin{aligned}
\mathbf{x}_g &= \sum_{i=0}^{3} \binom{3}{i} (1-t)^{3-i} t^i \begin{bmatrix} p_{ix} \\ p_{iy} \\ 0 \end{bmatrix}_\gamma, \quad t \in [0,1] \\
\mathbf{x}_b &= \mathbf{R}_{gb}(\phi,\theta)^{-1}(\mathbf{x}_g - \mathbf{t}_{gb}(z)) \\
u_l &= \frac{f_u}{x_b} z_b + c_y \qquad\qquad u_r = \frac{f_u}{x_b} z_b + c_y \\
v_l &= \frac{f_v}{x_b}(y_b + \frac{d}{2}) + c_x \qquad\qquad v_r = \frac{f_v}{x_b}(y_b - \frac{d}{2}) + c_x
\end{aligned} \tag{4.3}$$

Here, $\mathbf{x}_g$ is a point which, from the edge grouping stage, is known to lie on curve γ. For explanation of these terms, refer to (3.8).

For any curve k, we can easily transform the control point vector $\mathbf{p}_k$ to parametric polynomial form with coefficient vector $\boldsymbol{\alpha}_k$, and know the Jacobian $\frac{d\boldsymbol{\alpha}_k}{d\mathbf{p}_k}$ (Section 3.2). Then, the Jacobian of $\mathbf{x}_g$ with respect to each of the control point vectors $\mathbf{p}_k$ can be calculated as follows:

$$\frac{d\mathbf{x}_g}{d\mathbf{p}_k} = \frac{d\mathbf{x}_g}{d\boldsymbol{\alpha}_k}\frac{d\boldsymbol{\alpha}_k}{d\mathbf{p}_k} \tag{4.4}$$

$$\frac{d\mathbf{x}_g}{d\boldsymbol{\alpha}_k} = \begin{cases} \begin{bmatrix} 1 & t & t^2 & t^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & t & t^2 & t^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \text{if } k = \gamma \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{4.5}$$

In other words, the rows of the Jacobian corresponding to a measured point are only nonzero for the corresponding curve. The value of t used for each measurement will be explained later in this section.

Next, the Jacobians of $\mathbf{x}_b$ with respect to the parameters $\boldsymbol{\beta}$ are as follows:

$$\frac{d\mathbf{x}_b}{d\mathbf{p}_k} = \mathbf{R}_{gb}(\phi,\theta)^{-1}\frac{d\mathbf{x}_g}{d\mathbf{p}_k} \tag{4.6}$$

$$\frac{d\mathbf{x}_b}{d\theta} = \begin{bmatrix} -\sin\theta & 0 & -\cos\theta \\ \cos\theta\sin\phi & 0 & -\sin\theta\sin\phi \\ \cos\theta\cos\phi & 0 & -\sin\theta\cos\phi \end{bmatrix}\Big[\mathbf{x}_g - \mathbf{t}_{gb}(z)\Big] \tag{4.7}$$

$$\frac{d\mathbf{x}_b}{d\phi} = \begin{bmatrix} 0 & 0 & 0 \\ \sin\theta\sin\phi & -\sin\phi & -\cos\theta\cos\phi \\ -\sin\theta\sin\phi & -\cos\phi & -\cos\theta\sin\phi \end{bmatrix}\Big[\mathbf{x}_g - \mathbf{t}_{gb}(z)\Big] \tag{4.8}$$

$$\frac{d\mathbf{x}_b}{dz} = -\mathbf{R}_{gb}(\phi,\theta)^{-1}\begin{bmatrix}0\\0\\1\end{bmatrix} \tag{4.9}$$

$$\frac{d\mathbf{x}_b}{d\boldsymbol{\beta}} = \begin{bmatrix}\frac{d\mathbf{x}_b}{d\mathbf{p}_1} & \frac{d\mathbf{x}_b}{d\mathbf{p}_2} & \cdots & \frac{d\mathbf{x}_b}{d\mathbf{p}_q} & \frac{d\mathbf{x}_b}{d\theta} & \frac{d\mathbf{x}_b}{d\phi} & \frac{d\mathbf{x}_b}{dz}\end{bmatrix} \tag{4.10}$$

The full Jacobian $\mathbf{J} = \frac{\partial \mathbf{f}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ can then be computed by using the stereo projection equations (3.8). The rows of the Jacobian vary depending on whether the edge point corresponds to the left or right image. For a left image edge point $\mathbf{u}_l = [u_l,\, v_l]^T$ and right image edge point $\mathbf{u}_r = [u_r,\, v_r]^T$, the corresponding rows of the Jacobian can be found with the following relations:

$$\frac{d\mathbf{u}_l}{d\boldsymbol{\beta}} = \frac{d\mathbf{u}_l}{d\mathbf{x}_b}\frac{d\mathbf{x}_b}{d\boldsymbol{\beta}}, \quad \frac{d\mathbf{u}_l}{d\mathbf{x}_b} = \begin{bmatrix} -\dfrac{f_u z_b}{x_b^2} & 0 & \dfrac{f_u}{x_b} \\ -\dfrac{f_v(y_b+\frac{d}{2})}{x_b^2} & \dfrac{f_v}{x_b} & 0 \end{bmatrix} \tag{4.11}$$

$$\frac{d\mathbf{u}_r}{d\mathbf{p}} = \frac{d\mathbf{u}_r}{d\mathbf{x}_b}\frac{d\mathbf{x}_b}{d\boldsymbol{\beta}}, \quad \frac{d\mathbf{u}_r}{d\mathbf{x}_b} = \begin{bmatrix} -\dfrac{f_u z_b}{x_b^2} & 0 & \dfrac{f_u}{x_b} \\ -\dfrac{f_v(y_b-\frac{d}{2})}{x_b^2} & \dfrac{f_v}{x_b} & 0 \end{bmatrix} \tag{4.12}$$

One important thing to note is that this Jacobian still depends on the unknown parameter t through the term $\frac{d\mathbf{x}_g}{d\mathbf{p}_j}$. To determine this t-value for each row of the Jacobian (i.e., for each measurement), we need to determine the point on the projected image curve that is closest to the measured point. In other words, we need to use the parameter estimates to project the world curve to the left or right image (depending on the location of the measured point), and the t-value of the measured

point is the same as that of the closest curve point.

The process of finding the nearest curve point to an external (measured) point is detailed in the next section.

4.3.1 Nearest point Calculation for Cubic Bézier Curves

Assume (all in image co-ordinates) that the measured edge point is given by (u_m, v_m), the closest curve point occurs at $t = t^*$, and the projected image curve has control points (u_0, v_0) ... (u_3, v_3):

$$\begin{bmatrix} u_c(t) = \sum_{i=0}^{3} \binom{n}{i}(1-t)^{n-i}t^i u_i \\ v_c(t) = \sum_{i=0}^{3} \binom{n}{i}(1-t)^{n-i}t^i v_i \end{bmatrix}, \quad t \in [0,1] \tag{4.13}$$

The distance between the measured point and the point on a curve is:

$$s = \sqrt{(u_c(t) - u_m)^2 + (v_c(t) - v_m)^2} \tag{4.14}$$

To find the nearest point, we set the derivative of the distance with respect to t to zero. Taking the derivative and simplifying the expression, we have the following polynomial equation in t^*:

$$\sum_{i=0}^{5} c_i (t^*)^i = 0$$

$$\begin{aligned} c_0 &= u_2(u_3 - u_m) + v_2(v_3 - v_m) \\ c_1 &= {u_2}^2 + 2u_1(u_3 - u_m) + {v_2}^2 + 2v_1(v_3 - v_m) \\ c_2 &= 3(u_2u_1 + u_0(u_3 - u_m) + v_2v_1 + v_0(v_3 - v_m)) \\ c_3 &= 2(2u_2u_0 + {u_1}^2 + 2v_2v_0 + {v_1}^2) \\ c_4 &= 5(u_1u_0 + v_1v_0) \\ c_5 &= 3\left({u_0}^2 + {v_0}^2\right) \end{aligned} \tag{4.15}$$

This can be solved using a numerical root finding technique such as Newton's method. However, this process needs to be performed for every measured edge point on every iteration of the curve fitting algorithm. As a result, computational expense is a necessary consideration. To minimize the computational cost, we approximate this process as follows. Given a sequence of edge point measurements

corresponding to a measured curve, we can "assign" a t-value to each measurement based on its position in the sequence. For example, the first point is assigned $t = 0$, the last point has $t = 1$, and the t-values for all points in between are distributed evenly. Then, this iterative technique is avoided, and the closest curve point can be computed directly from the t-value associated with the measured edge point.

This approximation is accurate under the assumption that the edge points are evenly distributed (i.e., there are no large gaps), which is already a requirement for good curve fitting results.

4.3.2 Initial Guess of Parameters

The Levenberg-Marquardt approach, like most other iterative techniques, is sensitive to the initial parameter estimates, so an intelligent starting point must be selected. Here, our initial guess is obtained by looking at the start and end points of the measured curves. Since we already ensure that the start and end points are approximately equivalent in both images, we have an approximate stereo correspondence. Thus, we can triangulate their positions in the world frame using (3.8), obtaining $2M$ points on the ground plane (recall that M is the number of observed curves). The plane of best fit for these points provides an initial estimate of the roll, pitch and height, and by transforming the points using the determined angles, we also obtain estimates for the start and end control points of the curve in the world frame. Linearly interpolating between these points gives us estimates of the remaining curve control points that are practical for most operating scenarios.

4.3.3 Outlier Rejection

While experiments with the curve fitting approach demonstrate it to be feasible in determining the camera's orientation with respect to the ground plane, it is possible on some frames for the nonlinear optimization to diverge or converge to an incorrect equilibrium, since the problem is not convex. To ensure that outlying curve fitting results do not adversely affect the SLAM results, a simple outlier rejection technique has been used. At each stereo frame, if the measured parameters (output of the curve fit) lie more than 3 standard deviations away (in terms of the measurement covariance) from the current estimate of roll, pitch and height, the

frame is discarded and the SLAM algorithm moves to the next frame. Similarly, if the curve fit has a fitting error over a predefined threshold, the frame is also discarded.

4.3.4 Summary

By applying this technique on raw edge point measurements (grouped into different curves), the algorithm obtains the optimal estimates for the curve control points and the out-of-plane pose parameters, minimizing the reprojection error between the curve model and the measured edge pixels.

The final result is that each stereo frame gives us an absolute measurement of three of the degrees of freedom and an expression for the curves in the ground plane.

CHAPTER 5

SLAM FORMULATION

This chapter details the SLAM formulation for the proposed algorithm. In the following sections, the data association technique is first discussed, a necessary curve splitting algorithm is explained, and then the SLAM equations are derived.

5.1 Curve Data Association

Before deriving the SLAM equations, we first need to consider data association between measured curves (control points obtained in the curve fitting procedure) and map curves (landmarks existing in the map).

Specifically, we need to consider that measured curves do not have a one-to-one correspondence with the map curves. Indeed, the measured curve may correspond to part of a single map curve, or parts of multiple connected map curves, etc. This is illustrated in the example of Figure 5.1(a), where the measured curve $\mathbf{z}$ matches two map curves $\mathbf{x}_i$ and $\mathbf{x}_j$. Thus, we need to determine a) the map curve(s) to which the measured curve corresponds, and b) the start and end points of the map curve segments to which the measured curve corresponds. In the example of Figure 5.1(a), this means we need to determine the parameter values t_i, t_j and t_z. Then, the algorithm can determine whether to add curves, update curves, or perform both.

To determine these parameter values, we track the endpoints of the existing map curves (all in the left image frame) using the Lucas-Kanade tracking algorithm [34]. It is important to stress that the algorithm only needs to track the endpoints of the map curves (as shown in Figure 5.2); all structural information of the curves is maintained through the curve parameters, thereby avoiding the drawbacks of point-based methods outlined earlier.

The procedure for each curve is as follows:

1. The endpoints of the map curve are continuously tracked in all image frames

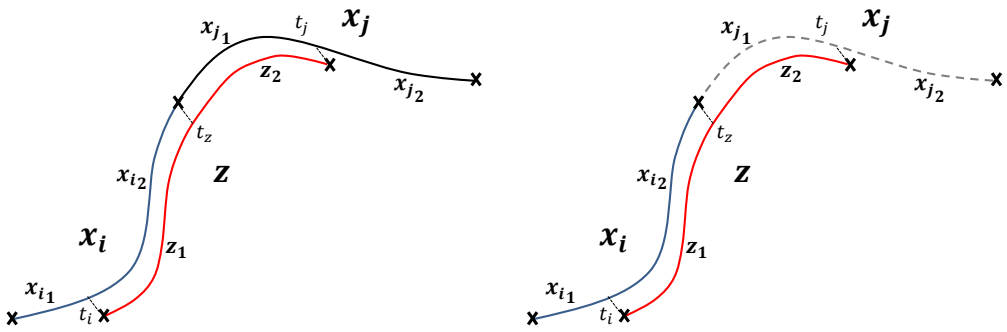

Figure 5.1: Updating map curves (left), and adding new curves to the map (right) based on a measured curve at one timestep. The measured curve is shown in red, and the map curves are in blue and black.

(from the point it is first initialized).

2. The measured curve (i.e., the output of the curve fit) is projected back to the image.

3. Then, t_z is the t-value of the point on the projected curve closest to the tracked endpoint, and can be found using the procedure outlined in chapter 4.3.1.

4. We then specify $t_i = t_j = 1 - t_z$.

While this last step may seem imprecise, it is effective in practice, and, in fact, naturally regulates the length of each introduced map curve. If we ensure that the number of edge pixels in each measured curve is reasonably constant (i.e., same "size" in successive images) then as the map endpoint moves along the measured curve (Figure 5.2), we observe more of the "new" map curve and less of the previous map curve. In Figure 5.1, this is equivalent to the measured curve $\mathbf{z}$ corresponding entirely with $\mathbf{x}_i$ initially, and then moving gradually to $\mathbf{x}_j$. Thus, we ensure that the t-value changes uniformly as motion occurs, and that new map curves have approximately similar lengths.

To mitigate the effect of tracking errors, the algorithm compares the motion of each tracked point with the motion of two other proximal edge points (both lying on the measured curve). If the difference exceeds a threshold, the data association is considered a failure, and the measured curve is independently added to the map.

Figure 5.2: Early frame (left) and current frame (right), with measured curve endpoints for each frame in red and map curve endpoints in blue

5.1.1 Curve Splitting Algorithm

Next, before we can derive the observation model and the SLAM algorithm, we need to develop a technique for updating an entire map curve based on a partial measurement.

This can be achieved by taking advantage of the De Casteljau algorithm for recursively evaluating Bézier curves [33]. For any n-degree Bézier curve, this approach provides a linear relation that can be used to split the curve at any specific parameter value t_s into two n-degree Bézier curves. For a cubic curve, the algorithm is defined as follows:

1. Start with the 4 Bézier control points $\{\mathbf{a}_0^{(0)}, \mathbf{a}_0^{(1)}, \mathbf{a}_0^{(2)}, \mathbf{a}_0^{(3)}\}$, and choose the split parameter value t_s.

2. Divide each segment of the control polygon in the ratio of t_s to $1 - t_s$ to construct the 3 "intermediate" control points $\{\mathbf{a}_1^{(0)}, \mathbf{a}_1^{(1)}, \mathbf{a}_1^{(2)}\}$.

3. Recurse until the result is a single point; $\mathbf{a}_3^{(0)}$.

Figure 5.3 shows this process, and the two dimensional intermediate control points are denoted as $\mathbf{a}_j^{(i)}$, where j is the level of recursion and i is the number of the control point for the intermediate curve.

Thus, by this recursive definition, each control point of the n-degree intermediate curve is defined as a convex combination of two control points of the $(n+1)$-degree intermediate curve (Figure 5.3), with the recurrence relation:

$$\mathbf{a}_j^{(i)} = (1 - t_s)\mathbf{a}_{j-1}^{(i)} + t_s\mathbf{a}_{j-1}^{(i+1)} \tag{5.1}$$

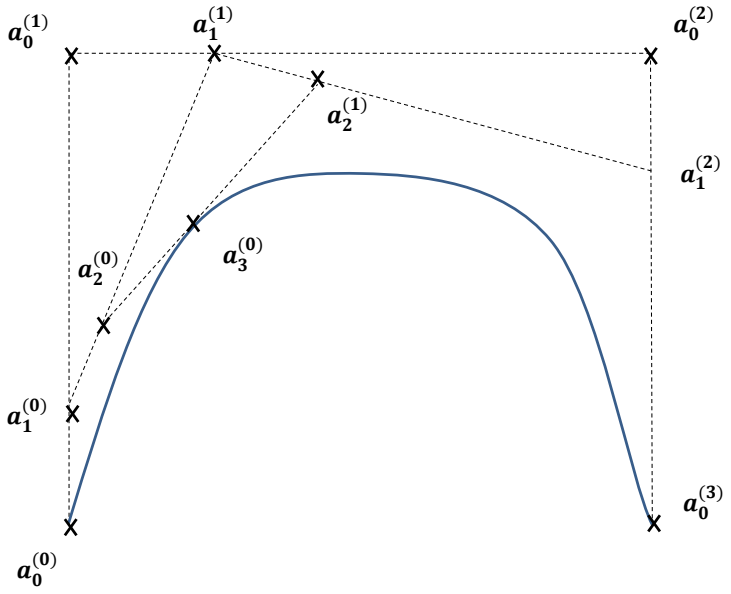

Figure 5.3: Recursive definition of Bézier curves

Noticing that the intermediate points also act as control points for specific segments of the curve, we can use this formulation to split the original Bézier curve into two curves of the same order, at the point $\mathbf{a}_3^{(0)}$. Referring to Figure 5.3, the control points of the original curve are $\{\mathbf{a}_0^{(0)}, \mathbf{a}_0^{(1)}, \mathbf{a}_0^{(2)}, \mathbf{a}_0^{(3)}\}$ and the control points of the split curves are given by $\{\mathbf{a}_0^{(0)}, \mathbf{a}_1^{(0)}, \mathbf{a}_2^{(0)}, \mathbf{a}_3^{(0)}\}$ and $\{\mathbf{a}_3^{(0)}, \mathbf{a}_2^{(1)}, \mathbf{a}_1^{(2)}, \mathbf{a}_0^{(3)}\}$.

Using the notation introduced in chapter 3.2, we can define the control point vector of the original curve as $\mathbf{p}_\ell = \left[p_x^{(0)}, p_x^{(1)}, p_x^{(2)}, p_x^{(3)}, p_y^{(0)}, p_y^{(1)}, p_y^{(2)}, p_y^{(3)}\right]_\ell^T$, and the control point vectors of the split curves in the same form as $\mathbf{p}_{\ell_1}$ and $\mathbf{p}_{\ell_2}$ to refer to the first and second segments respectively. Again following the previously defined notation, we can refer to each control point as $\mathbf{p}_\ell{}^{(i)}$ for the original curve, and $\mathbf{p}_{\ell_1}^{(i)}$ and $\mathbf{p}_{\ell_2}^{(i)}$ for the split curves.

Then, using (5.1), we get the following relationships:

$$\begin{aligned} \mathbf{p}_{\ell_1}^{(i)} &= \sum_{j=0}^{i} \binom{i}{j} {t_s}^j (1-t_s)^{i-j} \mathbf{p}_\ell^{(j)} \\ \mathbf{p}_{\ell_2}^{(i)} &= \sum_{j=i}^{n} \binom{n-i}{j-i} {t_s}^{j-i} (1-t_s)^{n-j} \mathbf{p}_\ell^{(j)} \end{aligned} \tag{5.2}$$

We can write this in matrix form using lower triangular matrix $\mathbf{S}_1$ and upper

triangular matrix $\mathbf{S}_2$:

$$
\begin{aligned}
\mathbf{S_1} \quad &: s_{ij} = \begin{cases} \binom{i}{j} {t_s}^j (1-t_s)^{i-j} & \text{if } j \leq i \\ 0 & \text{if } j > i \end{cases} \\
\mathbf{S_2} \quad &: s_{ij} = \begin{cases} \binom{n-i}{j-i} {t_s}^{j-i} (1-t_s)^{n-j} & \text{if } j \geq i \\ 0 & \text{if} j < i \end{cases} \\
\mathbf{p}_{\ell 1} \quad &= \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_1 \end{bmatrix} \mathbf{p}_\ell \\
\mathbf{p}_{\ell 2} \quad &= \begin{bmatrix} \mathbf{S}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \end{bmatrix} \mathbf{p}_\ell
\end{aligned} \tag{5.3}
$$

Thus, importantly, any segment of a Bézier curve can be obtained from the whole curve by applying a linear transformation to the original control points. Since the transformation matrices are triangular with non-zero diagonal elements, they are invertible. Further, they are independent of the control points and are only a function of the parameter value at the split point, t_s.

Given this method for splitting curves, we can easily equate a measured curve to any combination of existing map curves in order to a) update the existing map, and b) add new curves to the map. Based on the data association parameters determined using the procedure in chapter 5.1, it is easy to determine which matrix to use.

5.2 State Model

The state model is similar to a point-based EKF-SLAM formulation, with the state vector $\mathbf{x}$ containing the robot pose $\mathbf{x}_r$ and N landmark states $\mathbf{x}_i$ with $i \in \{1, 2, ..., N\}$.

$$
\mathbf{x} = \left[{\mathbf{x}_r}^T, \ {\mathbf{x}_1}^T, \ ..., \ {\mathbf{x}_N}^T \right]^T \tag{5.4}
$$

The robot pose state consists of the vehicle position vector $\mathbf{r}_r = [x_r, \, y_r, \, z_r]^T$, orientation in terms of Euler angles $\mathbf{\Psi}_r = [\phi_r, \, \theta_r, \, \psi_r]^T$, the linear velocity $\mathbf{v}_r$ and angular velocity $\boldsymbol{\omega}_r$, all defined with respect to the Earth frame. By using this parametrization, the process model is purely linear.

$$\mathbf{x}_r = \left[{\mathbf{r}_r}^T,\ {\mathbf{\Psi}_r}^T,\ {\mathbf{v}_r}^T,\ {\boldsymbol{\omega}_r}^T \right]^T \tag{5.5}$$

The landmark states $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ are the control point vectors of each of the N planar curves in the Earth frame, with the j^{th} landmark given by control point vector $\mathbf{x}_j = \left[x^{(0)},\ x^{(1)},\ x^{(2)},\ x^{(3)},\ y^{(0)},\ y^{(1)},\ y^{(2)},\ y^{(3)} \right]_j^T$, where $\left(x^{(i)},\ y^{(i)} \right)$ are the coordinates of the i^{th} control point.

As in [31], it is assumed that the state is normally distributed, with an estimate at timestep k of the mean

$$\hat{\mathbf{x}}(k|k) = \left[\hat{\mathbf{x}}_r(k|k)^T,\ \hat{\mathbf{x}}_1(k|k)^T,\ ...,\ \hat{\mathbf{x}}_N(k|k)^T \right]^T \tag{5.6}$$

and covariance matrix

$$\mathbf{P}(k|k) = \begin{bmatrix} \mathbf{P}_{r,r}(k|k) & \mathbf{P}_{r,1}(k|k) & \cdots & \mathbf{P}_{r,N}(k|k) \\ \mathbf{P}_{1,r}(k|k) & \mathbf{P}_{1,1}(k|k) & \cdots & \mathbf{P}_{1,N}(k|k) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{N,r}(k|k) & \mathbf{P}_{N,1}(k|k) & \cdots & \mathbf{P}_{N,N}(k|k) \end{bmatrix} \tag{5.7}$$

Here, the submatrices $\mathbf{P}_{r,r}(k|k)$, $\mathbf{P}_{r,i}(k|k)$, and $\mathbf{P}_{i,i}(k|k)$ (with $i \in \{1,\ 2,\ ...,\ N\}$) are the cross-covariances between the robot pose (subscript r) and the N curves (subscripts $1, 2, ..., N$).

5.3 Vehicle Process Model

For the prediction step, a kinematic motion model is used. We model the unknown linear and angular accelerations as zero-mean Gaussian vectors $\mathbf{a} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_a)$ and $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\alpha)$ respectively, both in the Earth frame, such that at timestep k, the linear and angular velocities experience a change of approximately $\mathbf{V}(k) = \mathbf{a}\Delta t$ and $\mathbf{\Omega}(k) = \boldsymbol{\alpha}\Delta t$. We get the following vehicle process model:

$$\begin{aligned}\mathbf{x}_r(k+1) &= \begin{bmatrix}\mathbf{r}_r(k+1)\\ \boldsymbol{\Psi}_r(k+1)\\ \mathbf{v}_r(k+1)\\ \boldsymbol{\omega}_r(k+1)\end{bmatrix}\\ &= \mathbf{x}_r(k) + \begin{bmatrix}(\mathbf{v}_r(k)+\mathbf{V}(k))\,\Delta t\\ (\boldsymbol{\omega}_r(k)+\boldsymbol{\Omega}(k))\,\Delta t\\ \mathbf{V}(k)\\ \boldsymbol{\Omega}(k)\end{bmatrix}\end{aligned} \tag{5.8}$$

The tradeoff for linearity in the process model is the fact that the process noise covariance matrices $\boldsymbol{\Sigma}_a$ and $\boldsymbol{\Sigma}_\alpha$ are not diagonal (indeed, they are diagonal in the body frame). In other words, the angular and linear accelerations are only independent in the body frame, and the associated diagonal covariance matrix will require a basis change from the Body frame to the Earth frame to obtain $\boldsymbol{\Sigma}_a$ and $\boldsymbol{\Sigma}_\alpha$.

5.4 Observation Model

The state vector, comprising the pose state and all of the landmark states, is:

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_r^T\ \boldsymbol{\Psi}_r^T\ {\mathbf{v}_r}^T\ {\boldsymbol{\omega}_r}^T\ {\mathbf{x}_1}^T\ ...\ {\mathbf{x}_N}^T\end{bmatrix}^T \tag{5.9}$$

The measurement vector, containing the out-of-plane pose measurement $\{\phi_r,\ \theta_r,\ \psi_r\}$ and the M curve measurements in the Local ground frame, is:

$$\mathbf{z} = \begin{bmatrix}\phi_r\ \theta_r\ z_r\ {\mathbf{z}_1}^T\ ...\ {\mathbf{z}_M}^T\end{bmatrix}^T \tag{5.10}$$

These are measured from the curve fitting process, outlined in Chapter 4.

Since the out-of-plane pose is measured directly, the update step for these variables is straightforward. Nonetheless, the relationship between the existing curve states and the measurements needs to be determined.

5.4.1 Updating Existing Curve States

Consider the case in Figure 5.1(a) where a measured curve $\mathbf{z}$ corresponds as shown to two existing map curves $\mathbf{x}_i$ and $\mathbf{x}_j$.

With the t_i, t_j and t_z values representing the correspondences (obtained from the data association step), we can directly derive the observation model by splitting the curves with de Casteljau's algorithm (Section 5.1.1). First, we split the original measurement into two ($\mathbf{z}_1$ and $\mathbf{z}_2$), and then equate these two with the appropriate segments of the map curves:

$$\begin{aligned} \mathbf{z}_1 &= \mathbf{x}_{i_2} = \begin{bmatrix} \mathbf{S}_2(t_i) & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2(t_i) \end{bmatrix} \mathbf{x}_i \\ \mathbf{z}_2 &= \mathbf{x}_{j_1} = \begin{bmatrix} \mathbf{S}_1(t_j) & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_1(t_j) \end{bmatrix} \mathbf{x}_j \end{aligned} \tag{5.11}$$

Here, $\mathbf{S}_2(t_i)$ and $\mathbf{S}_1(t_j)$ are the linear transformations required to split the curves at parameter values t_i and t_j (defined in Section 5.1.1), and need to be applied separately to both the x and the y co-ordinates of the map curves.

In the general case, we have a series of split measurements each corresponding to a segment of a different curve. To handle this case, let each split measurement $\mathbf{z}_i$ correspond to existing map curve i with associated split matrix $\mathbf{S}_i$ determined from the t-value at the data association step. Then, we can generalize this to:

$$\mathbf{z}_i = \begin{bmatrix} \mathbf{S}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_i \end{bmatrix} \mathbf{x}_i \tag{5.12}$$

If we consider that the measurement is made in the vehicle body frame, and the vehicle has planar pose of $\{x, y, \psi\}$, then the observation equation in the form $\mathbf{z} = \mathbf{h}(\mathbf{x})$ becomes the following:

$$\mathbf{z}_i = \mathbf{R}_{eg}(\psi_r)^{-1} \left(\begin{bmatrix} \mathbf{S}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_i \end{bmatrix} \mathbf{x}_i - \mathbf{t}_{eg}(x_r, y_r) \right) \tag{5.13}$$

where $\mathbf{R}_{eg}(\psi_r)$ and $\mathbf{t}_{eg}(x_r, y_r)$ are the rotation matrix and translation vector transforming the curve parameters between the Local Ground frame and the Earth

frame.

Since we have 4 different x and y co-ordinates (corresponding to the control points), the expressions for the rotation and translation matrices vary slightly from the well-known matrix forms. If we define the column vector $\mathbf{U} = [1,\ 1,\ 1,\ 1]^T$ and utilize the 4×4 identity matrix $\mathbf{I}_4$, we have:

$$\mathbf{R}_{eg} = \begin{bmatrix} \mathbf{I}_4 \cos\psi_r & -\mathbf{I}_4 \sin\psi_r \\ \mathbf{I}_4 \sin\psi_r & \mathbf{I}_4 \cos\psi_r \end{bmatrix}, \qquad \mathbf{t}_{eg} = \begin{bmatrix} \mathbf{U}x_r \\ \mathbf{U}y_r \end{bmatrix} \tag{5.14}$$

The standard EKF update equations are then applied.

5.4.2 Adding New States

In the case where map curve $\mathbf{x}_j$ does not yet exist (Figure 5.1(b)), we can insert the curve into the state vector and augment the covariance matrix with the necessary cross-covariances $\mathbf{P}_{N+1,N+1}$, $\mathbf{P}_{r,N+1}$, and $\mathbf{P}_{i,N+1}$:

$$\begin{aligned} \mathbf{x}_{N+1} &= \mathbf{g}(\mathbf{x}, \mathbf{z}) \\ &= \begin{bmatrix} \mathbf{S}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{-1} \end{bmatrix} [\mathbf{R}_{eg}\mathbf{z}_2 + \mathbf{t}_{eg}] \\ \mathbf{P}_{N+1,N+1} &= \mathbf{G}_x \mathbf{P} \mathbf{G}_x^T + \mathbf{G}_z \mathbf{R} \mathbf{G}_z^T \\ \mathbf{P}_{r,N+1} &= \mathbf{P}_{N+1,r}^T = \mathbf{P}_{r,r}\mathbf{G}_x^T \\ \mathbf{P}_{i,N+1} &= \mathbf{P}_{N+1,i}^T = \mathbf{P}_{r,i}\mathbf{G}_x^T \end{aligned} \tag{5.15}$$

where $\mathbf{R}$ is the measurement covariance matrix, $\mathbf{S}$ is the split matrix associated with the newly added curve, and $\mathbf{G}_x = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ and $\mathbf{G}_z = \frac{\partial \mathbf{g}}{\partial \mathbf{z}}$ are the Jacobians of $\mathbf{g}(\mathbf{x}, \mathbf{z})$ with respect to the state and measurement respectively.

5.5 Extended Kalman Filtering

Given the process and observation models, the standard EKF equations follow, outlined in detail in the following sections.

5.5.1 Prediction stage

In the prediction stage at timestep $k+1$, the estimated state and covariance matrix are updated using the Jacobian $\mathbf{F}$ of the process model $\mathbf{f}(\mathbf{x})$:

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{f}\left(\hat{\mathbf{x}}(k|k)\right) \tag{5.16}$$

$$\mathbf{P}_{r,r}(k+1|k) = \mathbf{F}\mathbf{P}_{r,r}(k|k)\mathbf{F}^T + \mathbf{Q}(k) \tag{5.17}$$

$$\mathbf{P}_{r,i}(k+1|k) = \mathbf{F}\mathbf{P}_{r,i}(k|k) \tag{5.18}$$

$$\mathbf{P}_{i,r}(k+1|k) = \mathbf{P}_{i,r}(k|k)\mathbf{F}^T \tag{5.19}$$

$$\mathbf{P}_{i,i}(k+1|k) = \mathbf{P}_{i,i}(k|k) \tag{5.20}$$

The Jacobian $\mathbf{F}$ is found by linearizing the process model about the current operating point (i.e., the current pose estimate). Since only the robot pose terms are dynamic (while all landmarks are stationary), $\mathbf{F}$ is defined here as the Jacobian of robot pose states. Specifically:

$$\mathbf{F} = \frac{d\boldsymbol{x}_r(k+1)}{d\boldsymbol{x}_r(k)} \tag{5.21}$$

$$= \begin{bmatrix} \frac{d\mathbf{r}_r(k+1)}{d\mathbf{r}_r(k)} & \frac{d\mathbf{r}_r(k+1)}{d\boldsymbol{\Psi}_r(k)} & \frac{d\mathbf{r}_r(k+1)}{d\mathbf{v}_r(k)} & \frac{d\mathbf{r}_r(k+1)}{d\boldsymbol{\omega}_r(k)} \\ \frac{d\boldsymbol{\Psi}_r(k+1)}{d\mathbf{r}_r(k)} & \frac{d\boldsymbol{\Psi}_r(k+1)}{d\boldsymbol{\Psi}_r(k)} & \frac{d\boldsymbol{\Psi}_r(k+1)}{d\mathbf{v}_r(k)} & \frac{d\boldsymbol{\Psi}_r(k+1)}{d\boldsymbol{\omega}_r(k)} \\ \frac{d\mathbf{v}_r(k+1)}{d\mathbf{v}_r(k)} & \frac{d\mathbf{v}_r(k+1)}{d\boldsymbol{\Psi}_r(k)} & \frac{d\mathbf{v}_r(k+1)}{d\mathbf{v}_r(k)} & \frac{d\mathbf{v}_r(k+1)}{d\boldsymbol{\omega}_r(k)} \\ \frac{d\boldsymbol{\omega}_r(k+1)}{d\mathbf{r}_r(k)} & \frac{d\boldsymbol{\omega}_r(k+1)}{d\boldsymbol{\Psi}_r(k)} & \frac{d\boldsymbol{\omega}_r(k+1)}{d\mathbf{v}_r(k)} & \frac{d\boldsymbol{\omega}_r(k+1)}{d\boldsymbol{\omega}_r(k)} \end{bmatrix} \tag{5.22}$$

$$= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I}\Delta t & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{I}\Delta t \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{5.23}$$

$$\tag{5.24}$$

The matrix $\mathbf{Q}(k)$ is the covariance of the process noise, which, from the process model, has the following form:

$$\mathbf{Q}(k) = \mathbf{F}\mathbf{Q}\mathbf{F}^T\Delta t = \begin{bmatrix} \Delta t^3\boldsymbol{\Sigma}_a & \mathbf{0} & \Delta t^2\boldsymbol{\Sigma}_a & \mathbf{0} \\ \mathbf{0} & \Delta t^3\boldsymbol{\Sigma}_\alpha & \mathbf{0} & \Delta t^2\boldsymbol{\Sigma}_\alpha \\ \Delta t^2\boldsymbol{\Sigma}_a & \mathbf{0} & \Delta t\boldsymbol{\Sigma}_a & \mathbf{0} \\ \mathbf{0} & \Delta t^2\boldsymbol{\Sigma}_\alpha & \mathbf{0} & \Delta t\boldsymbol{\Sigma}_\alpha \end{bmatrix} \tag{5.25}$$

Recall that all pose variables are defined in the Earth frame, while we assume that the error covariances are independent only in the body frame. This means that the covariance matrices $\mathbf{\Sigma}_a$ and $\mathbf{\Sigma}_\alpha$ are not diagonal. Indeed, we define two corresponding diagonal covariance matrices in the Body frame, $\mathbf{\Sigma}_a{}'$ and $\mathbf{\Sigma}_\alpha{}'$, and apply a change of basis between the Body and Earth frames using the rotation matrix $\mathbf{R}_{eb}$:

$$\begin{aligned}\mathbf{\Sigma}_a &= \mathbf{R}_{eb}\mathbf{\Sigma}_a{}'\mathbf{R}_{eb}^T \\ \mathbf{\Sigma}_a &= \mathbf{R}_{eb}\mathbf{\Sigma}_a{}'\mathbf{R}_{eb}^T\end{aligned} \tag{5.26}$$

5.5.2 Update stage

In the update stage at timestep $k+1$, we have the measurement vector $\mathbf{z}(k) = \left[{\mathbf{z}_1}^T, ..., {\mathbf{z}_M}^T\right]^T$ containing the (partial) measurement of M map curves, and the estimated state and covariance matrix are updated using the Jacobian $\mathbf{H}$ of the process model $\mathbf{h}(\mathbf{x})$:

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{K}(k)\left[\mathbf{z}(k) - \mathbf{h}(\hat{\mathbf{x}}(k+1|k))\right] \tag{5.27}$$

$$\mathbf{S}(k) = \mathbf{H}\mathbf{P}(k+1|k)\mathbf{H}^T + \mathbf{R}(k) \tag{5.28}$$

$$\mathbf{K}(k) = \mathbf{P}(k+1|k)\mathbf{H}^T\mathbf{S}(k)^{-1} \tag{5.29}$$

$$\mathbf{P}(k+1|k+1) = \mathbf{P}(k+1|k) - \mathbf{K}(k)\mathbf{S}(k)\mathbf{K}(k) \tag{5.30}$$

Here, $\mathbf{R}(k)$ is the covariance matrix of the measurement noise, which we also assume is zero except for along the diagonals.

We can derive the observation Jacobian as follows:

$$\mathbf{H} = \begin{bmatrix} \frac{d\mathbf{z}_1}{d\mathbf{x}_r} & \frac{d\mathbf{z}_1}{d\mathbf{x}_1} & \cdots & \frac{d\mathbf{z}_1}{d\mathbf{x}_N} \\ \frac{d\mathbf{z}_2}{d\mathbf{x}_r} & \frac{d\mathbf{z}_2}{d\mathbf{x}_1} & \cdots & \frac{d\mathbf{z}_2}{d\mathbf{x}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{d\mathbf{z}_M}{d\mathbf{x}_r} & \frac{d\mathbf{z}_M}{d\mathbf{x}_1} & \cdots & \frac{d\mathbf{z}_M}{d\mathbf{x}_N} \end{bmatrix} \tag{5.31}$$

The rotation and transformation matrices as used in the SLAM observation equations are given by:

$$\mathbf{R}_{ge} = \begin{bmatrix} \mathbf{I}_4 \cos\psi_r & \mathbf{I}_4 \sin\psi_r \\ -\mathbf{I}_4 \sin\psi_r & \mathbf{I}_4 \cos\psi_r \end{bmatrix}, \qquad \mathbf{t}_{ge} = \begin{bmatrix} \mathbf{U}x_r \\ \mathbf{U}y_r \end{bmatrix} \tag{5.32}$$

Since each (split) measurement corresponds one to one with a segment from one single curve, most of the partial derivatives with respect to state curves are zero. Suppose that measurement $\mathbf{z}_i$ corresponds to state curve $\mathbf{x}_k$ with curve splitting matrix $\mathbf{S}$. Then, for each measured curve $\mathbf{z}_i$ we have:

$$\frac{d\mathbf{z}_i}{dx_r} = \begin{bmatrix} -\mathbf{U}\cos\psi \\ \mathbf{U}\sin\psi \end{bmatrix}, \qquad \frac{d\mathbf{z}_i}{dy_r} = \begin{bmatrix} -\mathbf{U}\sin\psi \\ -\mathbf{U}\cos\psi \end{bmatrix} \tag{5.33}$$

$$\frac{d\mathbf{z}_i}{d\psi_r} = \begin{bmatrix} -\mathbf{I}_4 \sin\psi_r & \mathbf{I}_4 \cos\psi_r \\ -\mathbf{I}_4 \cos\psi_r & -\mathbf{I}_4 \sin\psi_r \end{bmatrix} \left[\begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \mathbf{x}_k - \mathbf{x}_r \right] \tag{5.34}$$

$$\frac{d\phi_r}{d\phi_r} = 1, \qquad \frac{d\theta_r}{d\theta_r} = 1, \qquad \frac{d\psi_r}{d\psi_r} = 1 \tag{5.35}$$

$$\frac{d\mathbf{z}_i}{d\mathbf{x}_r} = \begin{bmatrix} \frac{d\mathbf{z}_i}{dx_r} & \frac{d\mathbf{z}_i}{dy_r} & \frac{d\mathbf{z}_i}{d\psi_r} & \frac{d\mathbf{z}_i}{dv_r} & \frac{d\mathbf{z}_i}{d\omega_r} \end{bmatrix} \tag{5.36}$$

$$\frac{d\mathbf{z}_i}{d\mathbf{x}_k} = \mathbf{R}_{ge} \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \tag{5.37}$$

$$\frac{d\mathbf{z}_i}{d\mathbf{x}_j} = \mathbf{0}, \qquad j \neq k \tag{5.38}$$

All other terms in the Jacobian are zero. It is important to note that since the curve measurements are in the Local Ground frame, they are independent of the out-of-plane pose variables.

CHAPTER 6

RESULTS

This chapter presents the results obtained using the proposed algorithm. Vision-based results are presented in order to demonstrate the effectiveness of the algorithm in real environments, and Monte Carlo simulation results are used to analyze the accuracy and consistency of the SLAM algorithm.

6.1 Vision Results

To demonstrate effectiveness with real data, three paths, with lengths ranging up to 100m, were mapped using the algorithm. Vision data was obtained using a stereo camera rig with a fixed baseline of 55cm. The estimated maps are shown in Figure 6.1 overlaid on Google satellite imagery. Over these distances, the drift of the maps and trajectories are on the order of 2-5m. The second and third paths are particularly challenging due to the lack of clear path edges; nonetheless, the algorithm is able to yield a reasonable map. In the second example, the algorithm also deals with a number of successive frames without a good curve measurement, where the pose is updated based on the prediction alone. Here, the trajectory is discontinuous when it finally receives an update, but the map still remains continuous (Figure 6.1(b)).

Such environments could not be mapped with laser ranging sensors, since there would be few laser returns from path edges. With the path in Figure 6.1(b) partially obstructed by overhanging trees, even satellite imagery cannot produce adequate detail. More significantly, the demonstrated level of accuracy is achieved with few states, only using curve structures that are integral to the mapping requirements. Ultimately, navigation in path environments is possible without utilizing point features, with the edges of the path alone.

The results demonstrate that a) the curve fitting algorithm can extract real world curves from edge points in a stereo image pair, and b) the EKF-based CurveSLAM

(a) Mapping of a path near Talbot Laboratory, UIUC, Urbana, IL (30m)

(b) More challenging path in Crystal Lake Park, Urbana, IL (50m). The algorithm recovers from a series of frames without good path curve measurements (discontinuity shown)

(c) Longer path in Crystal Lake Park, Urbana, IL (100m).

Figure 6.1: Vision results on three paths, with varying length and difficulty

formulation can process these measurements into a cumulative pose and map estimate. The algorithm is fully autonomous, and currently operates at 5-10 Hz on a laptop with 2.3 GHz Pentium Dual-core processor; this could be improved with adequate code optimization.

6.2 Simulation Results

To examine the effectiveness of the CurveSLAM algorithm, simulations were performed using three sample environments. Two main sources of error were incorporated as additive Gaussian noise: error in the detected edge pixels with standard deviation $\sigma_p = 2$ pixels, and error in the estimated data association parameter (t-value) of standard deviation $\sigma_{da} = 0.1$. These are both exaggerated estimates of the error encountered in real environments: the simulated noisy edge points are scattered more than edge points in real images (Figure 6.2), while an error of 0.1 in the matching parameter means the curve matching is incorrect by a tenth of the curve length.

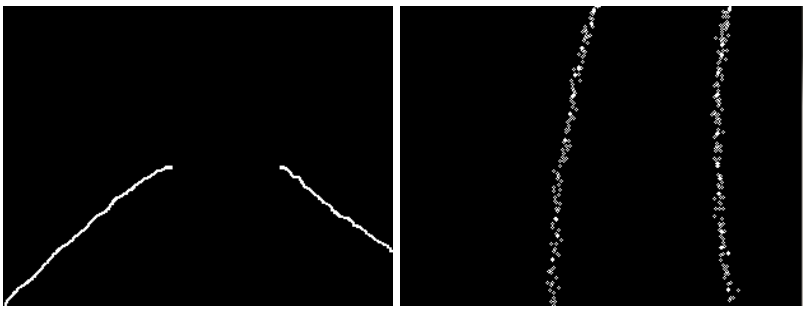

Figure 6.2: Typical edge pixels in real environment (left), and typical edge pixels in simulation (right)

In each environment, the vehicle travelled two loops (for a total distance of 80m, 120m, and 200m respectively), and loop closure was formulated as the solution to the constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{x}_c} \quad & \mathbf{f}(\mathbf{x}_c) &=& \left(\mathbf{x}_c - \mathbf{x}_u\right)^T \mathbf{P}^{-1} \left(\mathbf{x}_c - \mathbf{x}_u\right) \\ \text{subject to} \quad & \mathbf{h}(\mathbf{x}_c) &=& \mathbf{0} \end{aligned} \tag{6.1}$$

Here, the unconstrained state is given by $\mathbf{x}_u$, while the constrained state (following

loop closure) is given by $\mathbf{x}_c$. The constraint equations $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ specify the completion of the loop (i.e., $x_r = 0$, $y_r = 0$, and $\psi_r = 0$). The solution approach for this problem is found in [37].

The constructed map, as well as the vehicle localization errors, are shown in Figures 6.3, 6.4, and 6.5.

The mapping results demonstrate the effectiveness of CurveSLAM. While some drift is to be expected, the resulting estimation of the maps are reasonably accurate. The largest localization errors arise in the planar variables (x, y, and ψ), which can be expected, since the remaining pose variables are measured directly from the curve fitting process. The estimated maps are nearly as accurate as those obtained in simulation by Pedraza et al. [31]. Notably, however, our simulations consider data association error as well, and do not utilize vehicle odometry, with the linear and angular velocities also estimated within the EKF. It must also be noted that while laser range finders are renown for high levels of accuracy, the accuracy of vision data tends to be lower: this is a compromise for the portability and richness of information that a camera can provide, and one of the challenges of Visual SLAM. Nonetheless, the simulation results are comparable to [31].

6.2.1 Consistency Analysis

EKF-SLAM consistency has been studied extensively in the literature [38][39][40], with various suggestions to improve filter consistency.

When the true vehicle state is known (as is the case in simulation), the well known Normalised Estimation Error Squared (NEES) can be used to characterize filter performance and consistency. It is the error squared normalized by the covariance, given by:

$$\epsilon_k = \left(\mathbf{x}(k|k) - \hat{\mathbf{x}}(k|k)\right)^T \mathbf{P}(k|k)^{-1} \left(\mathbf{x}(k|k) - \hat{\mathbf{x}}(k|k)\right) \tag{6.2}$$

Under the hypothesis that the filter is consistent and approximately Linear-Gaussian (an assumption for all EKF-based SLAM algorithms), ϵ_k follows the distribution of the sum-square of $\dim(\mathbf{x}_k)$ standard random variables, which is a χ^2 distribution, with the same number of degrees of freedom as the vehicle, which is, in our case, 6DOF. Then, by running N Monte Carlo runs, and as N approaches infinity, the expected value of the NEES is $E[\epsilon_k] = \dim(\mathbf{x}(k)) = 6$ [38]. With 50 Monte Carlo runs, we have a 95% confidence interval of $[5.08, 7.00]$. That is,

we can be 95% certain that the filter behaves as a consistent Linear-Gaussian estimator if the average value of the NEES over 50 runs remains within this range for the whole simulation duration. If the NEES is below this interval, the estimate of the covariance is conservative, while if the NEES is above this interval, the covariance estimate is optimistic (i.e., underestimated).

Thus, we set up our Monte Carlo runs as follows: the robot travels through the three simulated environments, and the NEES is recorded over 50 Monte Carlo runs. For each run, the algorithm is initialized with a different seed for the Random Number Generator, thereby sampling the entire error space as the number of runs approaches infinity. The plots are shown in Figure 6.6.

The NEES plots in Figure 6.6 illustrate the consistency properties of the CurveSLAM algorithm. A higher NEES value is undesirable, since it indicates an optimistic covariance estimate and filter inconsistency. There are no obvious symptoms of this in the map, such as divergence or "jumps" in the vehicle trajectory estimate, but with a gradually inconsistent filter, this is a possibility over much larger distances, a potential limitation of any EKF-based approach.

As indicated by the initial low NEES values, the covariance estimates begin conservatively, and then remain in the vicinity of the 95% confidence interval. A spike occurs as the vehicle commences its second loop, but the NEES quickly reduces back to reasonable levels. Indeed, the only extended period for which the covariance estimate is optimistic (i.e., the NEES is high) is during the vehicle's second loop, when it revisits areas it has previously observed. These consistency results are an improvement over those of Pedraza et al. [31], in terms of the peak NEES value and the duration of filter inconsistency. Particularly, it takes longer before the NEES remains beyond the 95% confidence interval, and the peak NEES is at least an order of magnitude lower.

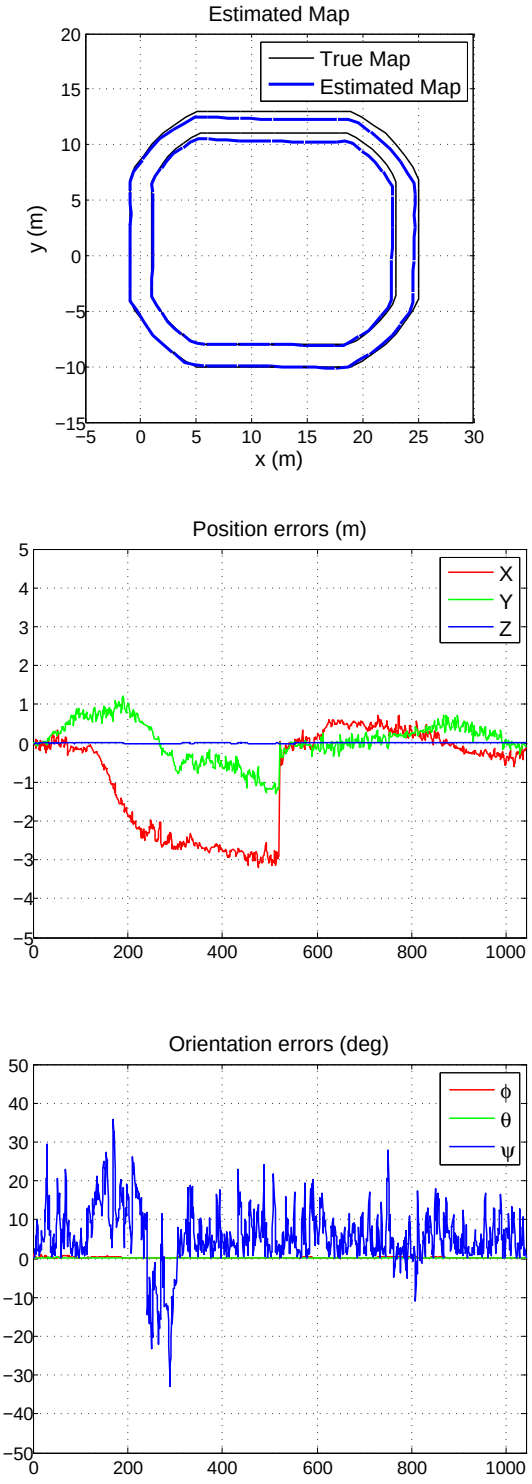

Figure 6.3: Simulation results for map 1: estimated and true map (top), position error (center), and orientation error (bottom).

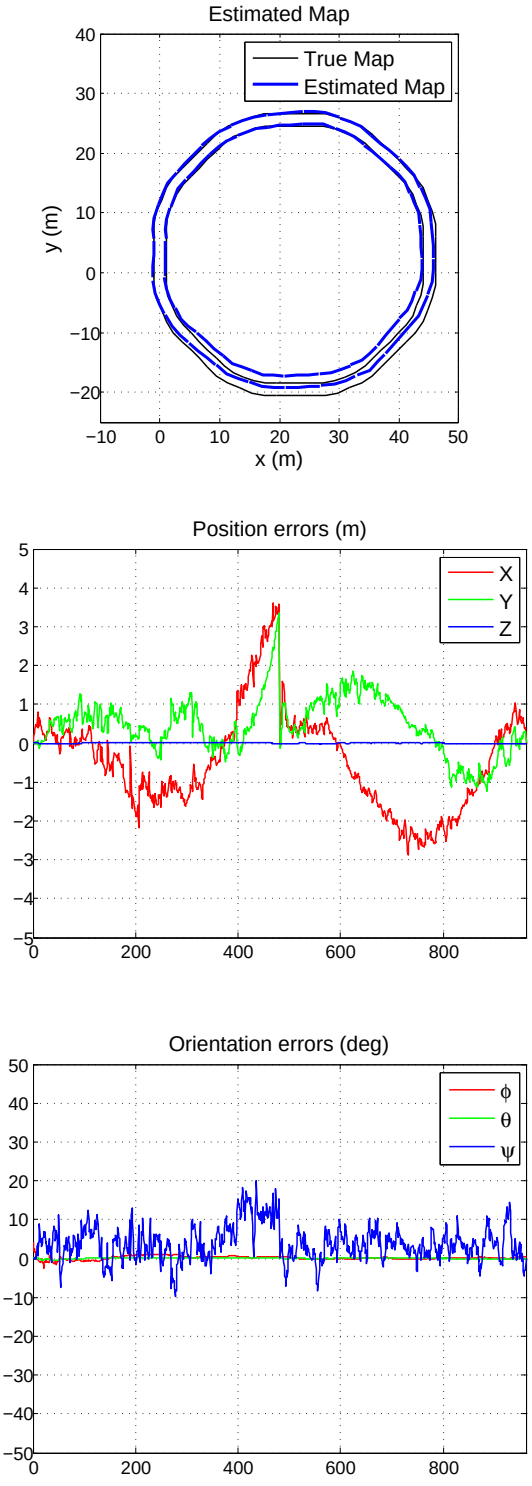

Figure 6.4: Simulation results for map 2: estimated and true map (top), position error (center), and orientation error (bottom).

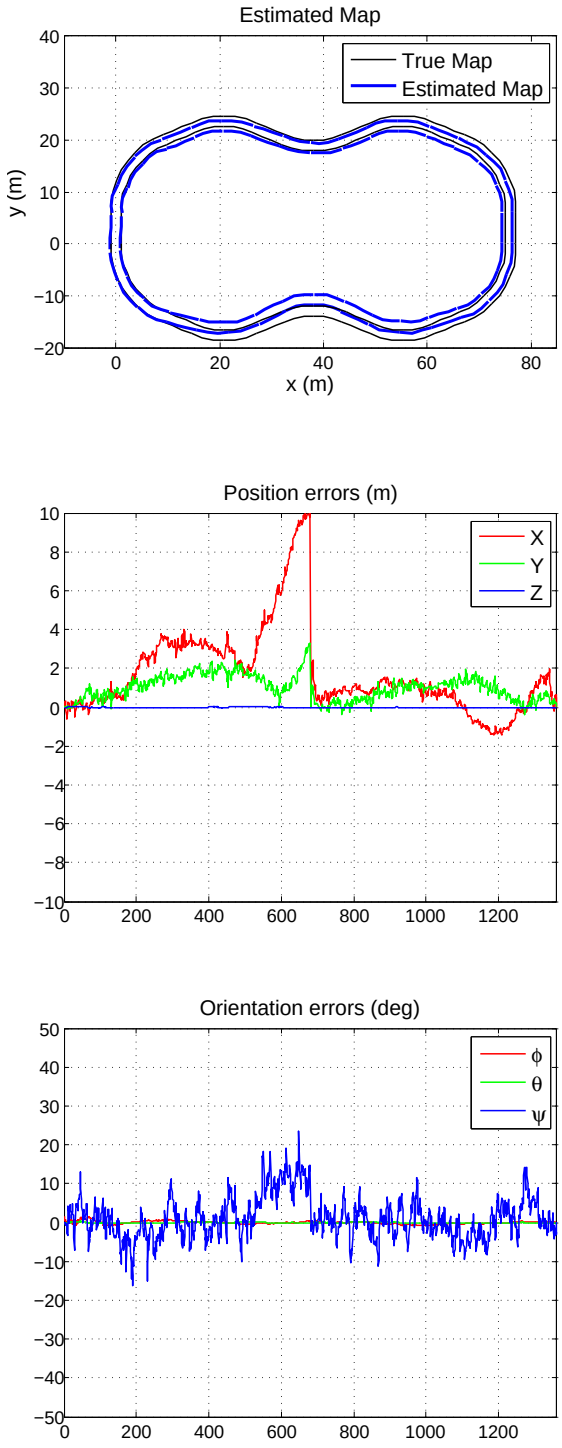

Figure 6.5: Simulation results for map 3: estimated and true map (top), position error (center), and orientation error (bottom).

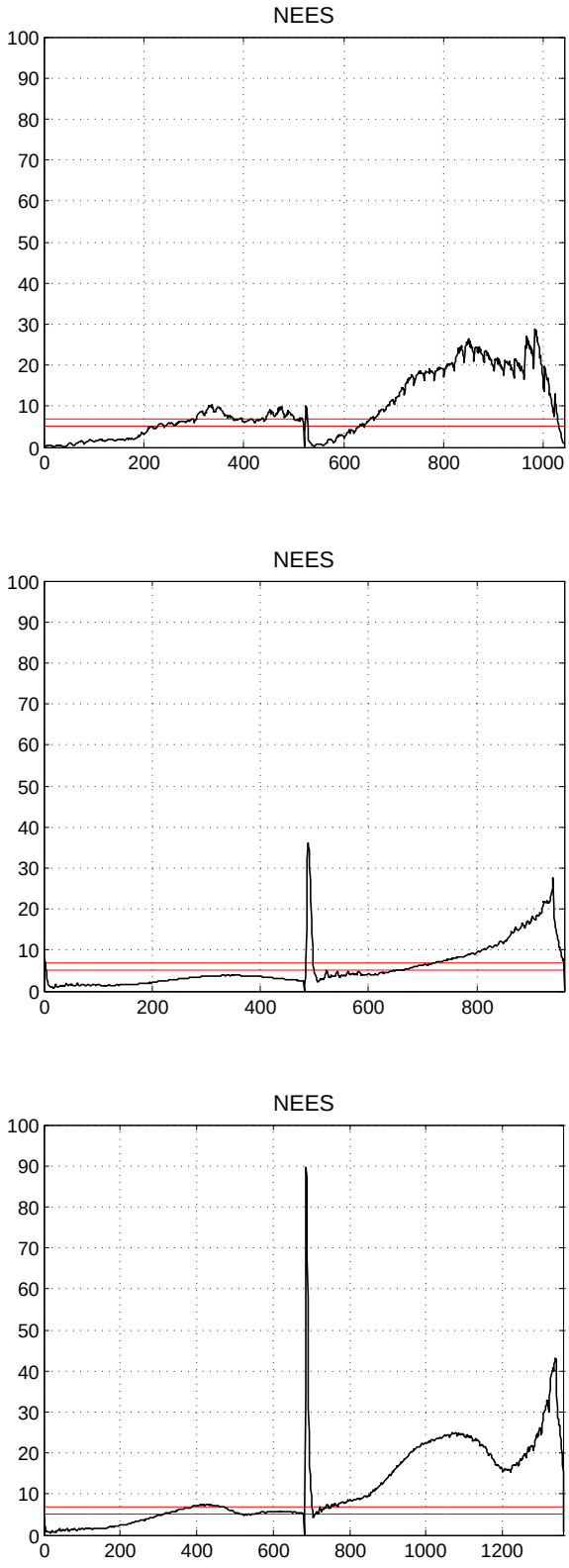

Figure 6.6: Monte Carlo consistency results for maps 1 to 3 (top to bottom), showing the average NEES over 50 runs

CHAPTER 7

CONCLUSIONS

This thesis has presented a technique to incorporate higher level curve structures into visual SLAM. Simulation results suggest that the CurveSLAM formulation can produce accurate mapping results, while experimental results demonstrate the effectiveness of the curve fitting and CurveSLAM algorithms with real data. In both cases, the proposed algorithm can produce structured, uncluttered maps and provide good navigation results with a much smaller state space than most point-based visual SLAM techniques. Further, the algorithm can be useful in areas in which laser range finding techniques will fail (for example, mapping the edges of a path). Monte Carlo simulation results show that the proposed technique can maintain consistency, offering an improvement over previous work.

Nonetheless, further work is needed to ensure that the method is effective in producing accurate and consistent estimates over large distances.

7.1 Future Work

Current work is focused on obtaining consistent mapping results over larger distances and in a range of environments. Future tasks include (but are not limited to) the following:

1. Replacing the iterative curve fitting technique with an analytical approach

   Previous work in curve-based reconstruction offers analytical solutions for algebraic curves, but these can be difficult to utilize computationally in the proposed SLAM framework. To our best knowledge, there is no existing analytical technique for stereo projective reconstruction of Bézier curves, and such a method would be ideal for this algorithm.

2. Extending the algorithm to admit non-planar curves

Our experiments demonstrated that the assumption of ground planarity is reasonable, and that the proposed algorithm is even able to provide mapping and localization accuracy in environments without a precisely planar ground. Nonetheless, there is some benefit to extending this approach to non-planar curves. Firstly, by allowing the use of curves outside the ground region, other significant structures can be mapped (eg. trees). Secondly, this allows for additional observability in the motion of the vehicle, and allow for reasonable estimates even when the ground plane is not in view.

3. Improvement of algorithm consistency

   This may be achieved using submap techniques or utilizing techniques highlighted in the SLAM consistency literature, such as by using the First Ever Jacobians (FEJ) method [39], or by utilizing bearing only measurements [40]. Alternatively, another option would be to modify the SLAM algorithm to better account for the nonlinear observation and process model (such as using an estimator based on the FastSLAM algorithm [41]).

4. Application to Path Planning and Control

   The mapping and navigation results shown would be quite useful when developing motion planning and control algorithms. With the boundaries of the path continually maintained by the SLAM algorithm, a novel planning / control algorithm could be used to take into account this additional useful information.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics & Automation Magazine*, vol. 13, no. 99, pp. 80–88, 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[3] A. J. Davison, "Real-time simultaneous localization and mapping with a single camera," in *Proceedings of the IEEE International Conference on Computer Vision*, 2003, pp. 1403–1410.

[4] K. Celik, S.-J. Chung, and A. Somani, "Mono-vision corner SLAM for indoor navigation," in *Proceedings of the IEEE International Conference on Electro/Information Technology*, 2008, pp. 343–348.

[5] K. Celik, S.-J. Chung, M. Clausman, and A. Somani, "Monocular vision SLAM for indoor aerial vehicles," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1566–1573.

[6] J. Yang, D. Rao, S. Chung, and S. Hutchinson, "Monocular vision based navigation in gps denied riverine environments," in *Proceedings of the AIAA Infotech at Aerospace Conference*. AIAA-2011-1403.

[7] M. Tomono, "Robust 3d SLAM with a stereo camera based on an edge-point icp algorithm," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009, pp. 4306–4311.

[8] J. Civera, A. Davison, and J. Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, 2008.

[9] L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardos, "Mapping large loops with a single handheld camera," in *Proceedings of Robotics: Science and Systems*, 2007.

[10] C. Estrada, J. Neira, and J. D. Tardos, "Hierarchical SLAM: Realtime accurate mapping of large environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[11] L. Paz, P. Piniés, J. Tardós, and J. Neira, "Large-scale 6-DOF SLAM with stereo-in-hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[12] J. Sola, A. Monin, and M. Devy, "BiCamSLAM: Two times mono is more than stereo," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 4795–4800.

[13] F. Alcantarilla, P., M. Bergasa, L., and F. Dellaert, "Visual odometry priors for robust EKF-SLAM," in *Proceedings of the IEEE Conference on Robotics and Automation*, 2010, pp. 3501–3506.

[14] 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, 2008.

[15] J. Civera, O. Grasa, A. Davison, and J. Montiel, "1-point RANSAC for EKF-based structure from motion," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3498–3504.

[16] V. Nguyen, A. Harati, and R. Siegwart, "A lightweight SLAM algorithm using orthogonal planes for indoor mobile robotics," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 658–663.

[17] A. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas, "Discovering higher level structure in visual SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 980–990, 2008.

[18] J. Sola, T. Vidal-Calleja, and M. Devy, "Undelayed initialization of line segments in monocular SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1553–1558.

[19] T. Lemaire and S. Lacroix, "Monocular-vision based SLAM using line segments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 2791–2796.

[20] G. Zhang and H. Suh, I., "SoF-SLAM: Segments-on-floor-based monocular SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2083–2088.

[21] Y. Lee and J. Song, "Visual SLAM in indoor environments using autonomous detection and registration of objects," *Multisensor Fusion and Integration for Intelligent Systems*, pp. 301–314, 2009.

[22] S. Ahn, M. Choi, J. Choi, and W. Chung, "Data association using visual object recognition for EKF-SLAM in home environment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2588–2594.

[23] M. An and C. Lee, "Stereo vision based on algebraic curves," in *Proceedings of the 13th International Conference on Pattern Recognition*, vol. 1, 1996, pp. 476–482.

[24] J. Kaminski and A. Shashua, "Multiple view geometry of general algebraic curves," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 195–219, 2004.

[25] 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.

[26] 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 America A*, vol. 22, no. 9, pp. 1746–1762, 2005.

[27] F. Kahl and J. August, "Multiview reconstruction of space curves," in *Proceedings of the IEEE International Conference on Computer Vision*, 2003, pp. 1017–1024.

[28] R. Fabbri and B. Kimia, "3D curve sketch: Flexible curve-based stereo reconstruction and calibration," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1538–1545.

[29] A. Huang, S. Teller et al., "Probabilistic lane estimation using basis curves," in *Proceedings of Robotics: Science and Systems (RSS)*, 2010.

[30] L. Pedraza, G. Dissanayake, J. Miró, D. Rodriguez-Losada, and F. Matia, "BS-SLAM: Shaping the world," in *Proceedings of Robotics: Science and Systems*, 2007.

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

[32] M. Liu, S. Huang, G. Dissanayake, and S. Kodagoda, "Towards a consistent SLAM algorithm using B-splines to represent environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 2065–2070.

[33] L. Piegl and W. Tiller, *The NURBS Book*. Springer-Verlag, 1997.

[34] D. Forsyth and J. Ponce, *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[35] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[36] W. Press, B. Flannery, S. Teukolsky, W. Vetterling et al., *Numerical recipes*. Cambridge University Press, 2007, vol. 547.

[37] C. Estrada, J. Neira, and J. Tardós, "Hierarchical SLAM: Real-time accurate mapping of large environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[38] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3562–3568.

[39] G. Huang, A. Mourikis, and S. Roumeliotis, "Analysis and improvement of the consistency of extended kalman filter based SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008, pp. 473–479.

[40] A. Tamjidi, H. Taghirad, and A. Aghamohammadi, "On the consistency of EKF-SLAM: Focusing on the observation models," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2083–2088.

[41] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the 18th National Conference on Artificial Intelligence*, 2002, pp. 593–598.