TRAJECTORY SERVOING: IMAGE-BASED TRAJECTORY TRACKING WITHOUT ABSOLUTE POSITIONING

A Dissertation
Presented to
The Academic Faculty

By

Zixuan Wu

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering
Department of Engineering

Georgia Institute of Technology

May 2021

TRAJECTORY SERVOING: IMAGE-BASED TRAJECTORY TRACKING WITHOUT ABSOLUTE POSITIONING

Thesis committee:

Dr. Patricio A. Vela, Advisor
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Matthieu Bloch
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Ghassan AlRegib
Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: March 12, 2021

Vainly facing the hermit in sparkling snow-clad hills;

I forget not the fairy in lone woods beyond the world.

Xueqin Cao

This thesis is dedicated to my friend Nuomin, who encourages me to march forward.

ACKNOWLEDGMENTS

First of all, I would like to express my special gratitude to advisor Patricio A. Vela of the School of Electrical and Computer Engineering at Georgia Institute of Technology. I am deeply impressed by his comprehensive knowledge in this field and rigorous attitude towards scientific research. Without his patient instructions in each group meeting and careful revising of each manuscript writing, it is impossible for me to complete my thesis. He indeed sets an example of being a serious and responsible researcher.

I also need to thank Professor Matthieu Block and Professor Ghassan AlRegib for sharing their precious time as committee members. It is really kind of them to take care of the development of students.

In addition, special thanks are due to the friends and colleagues in Intelligent Vision and Automation Laboratory (IVALAB) who made this work possible. Ph.D. candidate Shiyu Feng, who introduces me into IVALAB, gives me selfless help on learning basic software and robotics programming techniques. Discussions and cooperation with him significantly improve my work efficiency. I should also thank to Justin Smith, Yipu Zhao and other lab members since I cannot make any progress on the research without their code bases.

Moreover, I would thank the developers of the software on which my work is based. This includes but not limited to ROS, Gazebo, and Matlab SLAM Toolbox. I cannot achieve my research ideas without these necessary integrated platforms.

Finally, I must thank my parents who support my academics emotionally and financially. They are undoubtedly the most reliable persons on my road to success.

TABLE OF CONTENTS

Acknowledgments v

List of Tables ix

List of Figures x

Summary xii

Chapter 1: Introduction 1

1.1 Introduction 1

1.2 Contribution 3

Chapter 2: Background 5

2.1 Visual Teach and Repeat 5

2.2 Visual Servoing 6

2.3 Navigation Using Visual SLAM 7

2.4 Noise Effects in SLAM and Visual Servoing 7

2.5 Good Feature Selection 8

2.6 Uncertainty Modeling 9

Chapter 3: Preliminary Knowledge 11

3.1 Simultaneous Localization and Mapping (SLAM) 11

3.1.1 SLAM Basics 11
3.1.2 Code Base 12
3.1.3 Noise Effects in V-SLAM 15
3.2 Visual Servoing (VS) 16
3.2.1 Visual Servoing Basics 16
3.2.2 Image-Based Visual Servoing Rate Equations 17

Chapter 4: Trajectory Servoing System 20
4.1 Trajectory Servoing 20
4.1.1 Trajectory and control signals 21
4.1.2 Features and feature paths 23
4.1.3 Trajectory Servoing Control 23
4.2 Long Distance Trajectory Servoing 25
4.2.1 Feature Replenishment 25
4.3 Trajectory Servoing Benefits in Image Noise 27

Chapter 5: Uncertainty Based Trajectory Servoing 31
5.1 Uncertainty Modeling and Propagation 31
5.1.1 Uncertainty propagation from image to 3D space 32
5.1.2 Uncertainty propagation from 3D space to desired image 33
5.1.3 Uncertainty Model Verification 34
5.2 Generalized least square in controller design 36

Chapter 6: Experimental Results 39

6.1 Short Trajectory Benchmark 39
6.1.1 Experimental Setup 39
6.1.2 Navigation Methods Tested 40
6.1.3 Results and Analysis 42
6.2 Long Trajectory Benchmark 43
6.2.1 Results and Analysis 44
6.3 Navigation with Image Noise 46
6.3.1 Experimental Setups 46
6.3.2 Evaluation Metrics 46
6.3.3 SLAM versus TS 47
6.3.4 TS versus U-TS 51
6.4 Navigation with online path planning 51

Chapter 7: Conclusion & Future Work 53
7.1 Conclusion 53
7.2 Future Work 53

References 55

LIST OF TABLES

6.1 Trajectory Tracking ATE (cm) 41
6.2 Terminal Error (cm) 41
6.3 Control Effort 42
6.4 Estimation ATE (cm) 42
6.5 Tracking ATE (cm) 45
6.6 Terminal Error (cm) 45
6.7 Control Effort 45
6.8 Estimation ATE (cm) 45
6.9 Short Distance: Trajectory Tracking ATE (cm) 49
6.10 Long Distance: Trajectory Tracking ATE (cm) 49
6.11 Short Distance: Other Tracking Errors from Boxplots (cm) 49
6.12 Long Distance: Other Tracking Errors from Boxplots (cm) 50
6.13 Short Distance: Control Effort 50
6.14 Long Distance: Control Effort 50

LIST OF FIGURES

1.1 A trajectory servoing system has two major components. One steers the robot to track short paths, while the other ensures the sufficiency of features to use by querying a SLAM module. 3

3.1 ORB-SLAM System [43] 13

3.2 Good Feature Selection 14

3.3 Gazebo environment top view and robot view with SLAM features. The two red dots are the start poses of the robot. The right figure is the simulated camera image with Gaussian noise and tracked features from (GF) ORB-SLAM. 15

3.4 PBVS Control System 17

3.5 IBVS Control System 17

4.1 Block Diagram for V-SLAM Posed Based Control 21

4.2 Trajectory servoing process. Matches from $\boldsymbol{S}^*$ to $\boldsymbol{S}$ define the control u, where $\boldsymbol{S}^*(t)$ is defined by the desired trajectory. 21

4.3 Feature replenishment process. There are three segments of feature trajectories. Stars are observed point sets at corresponding time. Each circle is the start or end time of next or this segment of feature trajectory. Three feature trajectories are generated by the feature replenishment equation (4.9). 26

4.4 Block Diagram for long-term Trajectory Servoing 27

4.5 (a) Feature distribution with 0.03 noise standard deviation; (b) Feature coordinates standard deviation with different noise levels. Blue line is the unweighted std. Red line is the weighted std. 30

5.1 Robots and Environment in Matlab Simulation: *Rob1* locates at the beginning of the trajectory and *Rob2* locates at the middle of the trajectory. Black '+' are landmarks and those with blue numbers are the landmarks which have been seen by robots. 35

5.2 Desired Image in Random Experiments 36

6.1 Short-distance template trajectories 40

6.2 Short-distance trajectory benchmarking results 41

6.3 Long-distance trajectories 43

6.4 Long trajectory benchmarking results 44

6.5 Box plots of benchmarks. (a) Short distance benchmark between VSLAM pose-based controller (red) and baseline trajectory servoing (green); (b) Long distance benchmark between VSLAM pose-based controller (red) and baseline trajectory servoing (green). (c) Short distance benchmark between baseline (green) and uncertainty-based (blue) trajectory servoing; (d) Long distance benchmark between baseline (green) and uncertainty-based (blue) trajectory servoing. 48

6.6 Navigation with global planning. Blue point is the robot starting position. Orange points are 6 goal points for navigation. In each figure, green is the collision-free global path. Red is the real robot trajectory overlaying on the green trajectory. The figures show successful navigation examples of the same goal point with three different controllers. (a) Pose-based trajectory tracking with perfect odometry. (b) Vision-based trajectory servoing. (c) Pose-based trajectory tracking with estimated poses from V-SLAM. It can be observed that the red trajectory has less deviations from the green trajectory with TS (b) than SLAM (c). 52

SUMMARY

Modern robot navigation systems enable robots to automatically achieve navigation goal in an obstacle filled surroundings by fusing sensor information and estimating robot state. Most of the current researches make use of pose based controllers and intend to localize robots more accurately for less trajectory tracking error in navigation. However, pose estimation in Cartesian space is vulnerable to system latency, IMU drift, GPS error, and image noise. **Therefore, we prefer to redefine and resolve navigation problems in perception space without absolute positioning rather than in world space.**

This thesis first describes an image based visual servoing (IBVS) system for a nonholonomic robot to achieve good trajectory following without real-time robot pose information and without a known visual map of the environment. We call it *trajectory servoing*. The critical component is a feature-based, indirect SLAM method to provide a pool of available features with estimated depth, so that they may be propagated forward in time to generate image feature trajectories for visual servoing.

Subsequently, we investigate the robustness of trajectory servoing against image noise which is commonly researched in prior works. Trajectory servoing is demonstrated to decrease the tracking error compared with SLAM pose based control from both theoretical analysis and experimental benchmark in noise environment. Additionally, a Gaussian uncertainty model will also be proposed to help build a feature covariance weighted least square controller that will improve the trajectory tracking performance. The covariance of desired anchor features could be obtained by uncertainty propagation between perception and world spaces along the trajectory.

Benchmark results via different evaluation metrics show trajectory servoing is more accurate than pose based feedback with or without image noise when both rely on the same underlying SLAM system. Moreover, uncertainty based trajectory servoing further improves the tracking performance when using noisy images.

CHAPTER 1
INTRODUCTION

1.1 Introduction

Navigation systems with real-time needs often employ hierarchical schemes that decompose navigation across multiple spatial and temporal scales. Doing so permits the navigation solution to respond in real-time to novel information gained from sensors, while being guided by the more slowly evolving global path. At the lowest level of the hierarchy lies trajectory tracking to realize the planned paths or synthesized trajectories. In the absence of an absolute reference (such as GPS) and of an accurate map of the environment, there are no external mechanisms to support trajectory tracking. Onboard mechanisms include odometry through *proprioceptive* sensors (wheel encoders, IMUs, etc.) or visual sensors. Pose estimation from proprioceptive sensors is not observable, thus visual sensors provide the best mechanism to anchor the robot's pose estimate to external, static position references.

Indeed visual odometry (VO) or visual SLAM (V-SLAM) solutions are essential in these circumstances. However, they too experience drift, mostly due to the integrated effects of measurement noise and system latency. Specificly, the feedback rate from multiple sensor (IMU, Camera, etc.) and control loops are impossible to be perfectly matched since each latency varies, therefore, raw IMU data uncorrected by VO may be directly sent to controller and cause tracking deviation [1]. Additionally, the accumulation of noise (e.g. IMU bias, camera noise, calibration error, etc.) will cause the VO drift [2] and further undermine the trajectory tracking. From the limitation of cost, cameras that are compatible with small robots are easily be affected by Johnson-Nyquist thermal noise [3]. In the thesis, it is proved that such typical Gaussian additive noise to each pixel will make SLAM pose

based trajectory tracking inaccurate.

The hypothesis explored in this paper is that *performing trajectory tracking in the image domain reduces the sensitivity of trajectory tracking systems reliant on VO or V-SLAM for accuracy*. In essence, the trajectory tracking problem is shifted from feedback in pose space to feedback in perception space. Perception space approaches have several favorable properties when used for navigation [4, 5]. Shifting the representation from being world-centric to being viewer-centric reduces computational demands and improves run-time properties. For trajectory tracking without reliable absolute pose information, simplifying the feedback pathway by skipping processes that are not relevant to–or induce sensitivities to–the local tracking task may have positive benefits. Using imaging sensors to control motion relative to visual landmarks is known as *visual servoing*. Thus, the objective is to explore the use of image-based visual servoing for long-distance trajectory tracking with a stereo camera as the primary sensor. The technique, which we call *trajectory servoing*, will be shown to have improved properties over systems reliant on VO or V-SLAM for pose-based feedback.

Trajectory tracking has been improved by skipping of uncertainty-affected localization using IBVS and reciprocally, IBVS will also be improved by SLAM feature tracking module in trajectory servoing. Gazebo simulations show SLAM tracking will help IBVS to track the features more resistant to image noise. Compared with feature selection methods used in SLAM pose based navigation to achieve more accurate localization, we build an uncertainty model, propagate uncertainty through feature trajectory for all tracked features, and use them in a novel optimization based IBVS controller with a covariance weighting strategy. From the benchmark result, improved trajectory servoing is better than good feature (GF) SLAM pose based tracking system and basic trajectory servoing system with respect to multiple metrics.

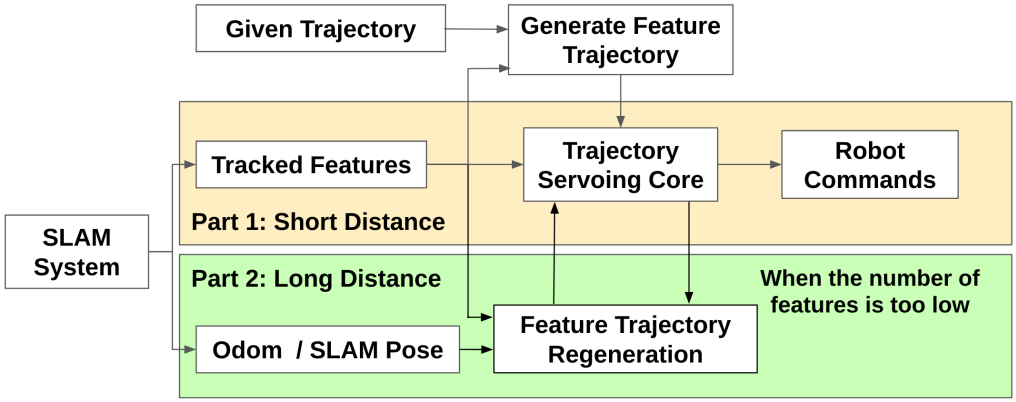

Figure 1.1: A trajectory servoing system has two major components. One steers the robot to track short paths, while the other ensures the sufficiency of features to use by querying a SLAM module.

1.2 Contribution

The system described in the thesis blends visual servoing, SLAM, and basic concepts from visual teach and repeat (VTR) to enable trajectory tracking of feasible paths by a mobile robot in unknown environments with sufficient visual texture. We call this combination of methods *trajectory servoing* because the objective is to perform long-term trajectory tracking using visual servoing techniques. What enables this objective to be met is a stereo visual SLAM system [6], which ties the desired trajectory to the image information.

The algorithmic components and information flow of a trajectory servoing system are depicted in Fig. 1.1, and consist of two major components. The first one, described in §4.1, is a trajectory servoing system for a set of world points and specified trajectory. These points are obtained from the V-SLAM system as well as tracked over time. It is capable of guiding a mobile robot along short paths. The second component, described in §4.2, supervises the core trajectory servoing system and confirms that it always has sufficient features from the feature pool to operate. Should this quantity dip too low, it queries the V-SLAM module for additional features and builds new feature tracks. We also discuss the potential trajectory servoing benefits in §4.3, especially the robustness of tracked feature coordinates to image noise.

Gaussian model is built in §5.1 for each tracked feature and propagated along the feature trajectory to provide additional uncertainty information to trajectory servoing process. §5.2 proposes *uncertainty-based trajectory servoing (U-TS)* to mitigate the negative effects of image noise described in §3.1.3. Based on weighted least square, U-TS applies adaptive covariance related weights for each feature to balance their contributions for a more accurate control variable and less tracking error.

Relevant benchmark results in §6 show the evidence how (uncertainty-based) trajectory servoing outperform baselines. Both §6.1 and §6.2 include experiment results about the performance of core trajectory servoing system for short paths and the entire system for long paths without image noise. Trajectory servoing is pretty unique in that short-term trajectories can be tracked as well as pose-based feedback control with access to perfect odometry. Though long-term accuracy is undermined by the reliance on SLAM, trajectory servoing minimizes the reliance and exhibits less sensitivity to estimation error than pose feedback methods. These image noise free tests tell us trajectory servoing indeed has architectural advantages by shifting feedback from real time raw pose in Cartesian space to pose triggered feature trajectory in perception space. §6.3 shows the benchmark results in different image noise levels. The performance of trajectory servoing is far better than SLAM pose based controller in each trajectory and U-TS is able to further decrease TS tracking error in noisy environment.

CHAPTER 2

BACKGROUND

2.1 Visual Teach and Repeat

Evidence that visual features can support trajectory tracking or consistent navigation through space lies in the *Visual Teach and Repeat* (VTR) navigation problem in robotics [7, 8]. Given data or recordings of prior paths through an environment, robots can reliably retrace past trajectories. The teaching phase of VTR creates a visual map that contains features associated with robot poses obtained from visual odometry [7, 9, 10, 11, 12]. Extensions include real-time construction of the VTR data structure during the teaching process, and the maintenance and updating of the VTR data during repeat runs [9, 10]. Feature descriptor improvements make the feature matching more robust to the environment changes [12, 13]. Visual data in the form of feature points can have task relevant and irrelevant features, which provide VTR algorithms an opportunity to select a subset that best contributes to the localization or path following task [9, 11]. While visual map construction seems similar to visual SLAM, map construction is usually not dynamic; it is difficult to construct or update visual map in real-time while in motion because of the separation of the teach and repeat phases. In addition, VTR focuses more on local map consistency and does not work toward global pose estimation [11] since the navigation problems it solves are usually defined in the local frame.

Another type of VTR uses the optical flow [8, 14] or feature sequence [15, 16, 17] along the trajectory, which is then encoded into a VTR data structure and control algorithm in the teaching phase. Although this method is similar to visual servoing, the system is largely over-determined. It can tolerate feature tracking failure, compared with traditional visual servo system, but may lead to discontinuities [18]. Though this method handles

long trajectories, and may be be supplemented from new teach recordings, it can only track taught trajectories.

2.2 Visual Servoing

Visual servoing (VS) has a rich history and a diverse set of strategies for stabilizing a camera to a target pose described visually. VS algorithms are classified into one of two categories: image based visual servoing (IBVS) and position based visual servoing (PBVS) [19, 20]. IBVS implementations include both feature stabilization and feature trajectory tracking [20, 21]. As a feedback strategy IBVS regulation does not guarantee what path is taken by the robot since the feature space trajectory has a nonlinear relationship with the Cartesian space trajectory of the robot. Identifying a feature path to track based on a target Cartesian space trajectory requires mapping the robot frame and the target positions into the image frame over time to generate the feature trajectory [21, 22], precisely what is done here.

The target application is trajectory tracking for a mobile robot. Mobile robots have been studied as candidates for visual servoing [7, 8, 23, 24, 25, 26, 27]. Some of them use IBVS but do not use the full IBVS equations involving the image Jacobian. The centroid of the features [7][26], the most frequent horizontal displacement of the matched feature pairs [8] or other qualitative cost functions [25] are used to generate [7] or correct [8] the feedforward angular velocity of mobile robot. These simplifications are reasonable since the lateral displacement of the features reflects task relevant movement to be regulated by the robot. However, they are best suited to circumstances with higher inaccuracy tolerance such as an outdoor, open field navigation. We use more precise velocity relations between the robot and feature motion to generate a feedback control signal for exact tracking similar to [23]. That work studied the path reaching problem with a visible path, which does not hold here.

2.3 Navigation Using Visual SLAM

Visual simultaneous localization and mapping (V-SLAM) systems estimate the robot's trajectory and world structure as the robot moves through space [28]. For some autonomous robots, the SLAM pose estimates provide good signals for using pose-based feedback to track trajectories using standard control policies. The problem associated to reliance on SLAM is the potential to accrue large estimation drift, which degrades trajectory tracking and goal attainment.

Pose estimation accuracy of SLAM is a major area of study [6, 29, 30, 31, 32]. However, most studies only test under open-loop conditions [6, 29, 30, 32], i.e., they only analyze the pose estimation difference with the ground truth trajectory, and do not consider the error induced when the estimated pose informs feedback control. More recently, closed-loop evaluation of V-SLAM algorithms as part of the feedback control and navigation system are tested for individual SLAM systems [33, 34, 35] or across different systems [1]. Closed-loop studies expose the sensitivity of pose-based feedback control and navigation, e.g., sensitivity to V-SLAM estimation drift and latency. Our work builds upon an existing closed-loop benchmarking framework and shows sensitivity reduction over conventional pose based closed-loop navigation solution. The proposed *trajectory servoing (TS)* method is not tied to a specific V-SLAM implementation.

2.4 Noise Effects in SLAM and Visual Servoing

SLAM pose drift (especially IMU drift) could be significantly corrected by visual odometry (VO) whose accuracy is based on feature matching and tracking [36, 37]. However, image noise will negatively affect VO from the following aspects: feature matching [37, 38], feature tracking [37, 39, 40] and pose drifting [41, 38], each of which may cause potential problems. Therefore, many improvements have been applied to mitigate noise effects. For feature matching, FFDNet [42] is used to denoise the image which leads to an increase of

correct matching rate of SIFT, SURF and ORB [38]. For feature tracking, image motion models are usually applied to minimize the regional dissimilarity [39] and set up rejection rules to remove outliers caused by uncertainties like occlusion, scale variation, and wrong association[40].

Unfortunately, VO will still be undermined by drift [36, 38] even we have perfect matching and tracking, since noise could also contribute to VO via inliers and accumulate from the incremental pose estimation in ORB SLAM2 [36, 43, 44] if no global feedback happens (loop closure or global BA). Therefore, some references including prior works of our lab investigate the relations between pose estimation and conditioning of inliers [45, 46, 47, 48, 49, 50]. and only use those well conditioned features to localize.

IBVS has been recognized to be more robust to noise than pose based visual servoing (PBVS) [19, 20], since pose estimation from feature patterns is sensitive to uncertainty [51]. Simulation results also show that IBVS only has a fairly gradual, linear increase of error when the image noise level is increasing [52]. Therefore, we have the reason to hypothesize that trajectory servoing may perform better than SLAM pose based trajectory tracking since it also bypasses localization which will cause drift and amplify the feature noise.

2.5 Good Feature Selection

In V-SLAM or visual navigation systems, localization or navigation is usually over determined by the excess number of tracked features. Sometimes it is beneficial to select part of the most informative or best conditioning feature points, lines or patches to reduce the noise contributions and achieve better system properties (i.e. estimation accuracy, real-time property etc.). In OOMC-SLAM [47, 48], the observability matrix is used to select the most observable triplets and find the maximum consensus set to update the system with Extended Kalman Filter (EKF). It shows higher accuracy than 1-point RANSAC [53] which initializes models through only 1 point. In addition, the non-filter bundle adjustment based ORB

SLAM, good feature selection could also be implemented to improve the performance of least square pose optimization [49, 6]. Similarly the selection of most informative subsegment from each 3D line plays the same role in line-assisted VO/VSLAM [50]. Finally, the most recent research [54] shows that dynamically assigning an appropriate size budget and selecting a condition-maximized subgraph for BA estimation will contribute to better BA-based V-SLAM back-ends, which extends the concept of selection from features to graphs.

Some visual servoing controllers also aim to manage the features they use in order to achieve better control properties. To guarantee a smooth tracking, some visual servoing controllers set different weights to all of the features according to their positions in the image instead of strictly using a feature subset [55, 56]. It inspires us to further improve the accuracy of trajectory servoing using a feature covariance weighting strategy.

2.6 Uncertainty Modeling

The stochastic representation of environment enables robot to understand its observations from a probabilistic aspect [57, 58]. Similar to [57], we derive the depth uncertainty from the first order approximation relation between depth and disparity. Another depth modeling method named inverse depth parametrization could be used to cope with the scenarios where features have infinite distance at the cost of more computational resources for 6D parameters rather than 3D [59], which does not hold true here. As for the pixel coordinate error, we use a common zero mean Gaussian distribution [60] where the variance is extracted from ORB-SLAM2.

In addition to uncertainty modeling, feature covariance propagation is also widely discussed [60, 61, 62, 63, 64, 65]. Uncertainty propagation in stereo reconstruction is initially described in [61, 62] with respect to camera calibration and image noise. Then the covariance propagation in general stereo configurations (not rectified) is proposed in [60, 63, 64]. More detailed and comprehensive analysis including uncertainty propagation between

different camera views could be found in [65, 66]. For those features with long depth, bias introduced in the calculation of depth should also be estimated and compensated [67, 68].

Some researchers have quantitatively investigated how the feature uncertainty affects visual servoing [65, 69, 70]. Even it derives visual servoing tracking error caused by tracked feature uncertainty [69, 70], neither of them consider the desired feature uncertainty or improve the controller with the knowledge of uncertainty information. The uncertainty modeling and propagation in the thesis is similar to [65] which includes the projection and reconstruction process. However, its controller is not compatible with trajectory servoing. Therefore, a new controller is designed considering the uncertainty information.

CHAPTER 3
PRELIMINARY KNOWLEDGE

This chapter will introduce some prerequisite knowledge about the thesis. §3.1.1 describes the SLAM system definitions, classifications, and development. §3.1.2 introduces the good feature (GF) ORB-SLAM system on which we build navigation system and the ROS/Gazebo based benchmark environment. §3.1.3 shows the problems of noise effects in V-SLAM pose estimation. §3.2.1 and §3.2.2 illustrate visual servoing basic concepts and derive the rate equations from robot motion to feature velocity. In general, this chapter will give readers a rough understanding of basic knowledge for the remaining thesis and we recommend to carefully read relevant papers [6, 71, 43, 44] for more details.

3.1 Simultaneous Localization and Mapping (SLAM)

3.1.1 SLAM Basics

Simultaneous localization and mapping (SLAM) comprises the simultaneous estimation of the state of a robot equipped with on-board sensors and the construction of the environment map that the sensors are perceiving [71]. It is widely used in navigation, localization and mapping tasks for indoor mobile robots where there is no GPS absolute positioning.

SLAM is an integrated system containing various subsystems that could be compatible with multiple platforms. Generally it could be divided into *front end*, used to extract and model sensor data, and *back end*, used to infer states from the obtained information. The mainstream types of visual SLAM front end are based on direct methods and feature based methods. The feature based methods use descriptors to match features between two frames and estimate camera relative motion. On the other hand, the direct methods estimate camera motion using optical flow or intensity information without calculating descriptors. As for

the back end, nearly all the pose estimation methods, including filter based, graph based or bundle adjustment (BA) based ones commonly condense down to successive linearizations (e.g. Gauss-Newton, Levenberg-Marquardt).

The development of SLAM is broken into three stages: *classical age*, *algorithmic analysis age* and *robust perception age*. Basic probabilistic approaches and data association methods are proposed in the classical age. The algorithmic analysis age mainly studies the properties of SLAM (e.g. observability, convergence, and consistency). Now we have stepped into the robust perception age where we care about the robustness of SLAM and computation resources. The state-of-art ORB-SLAM [43, 44] system absorbs the advantages of prior works and achieves good real time property with the uniform using of ORB features in all tasks and recovers from tracking failure by real time camera relocalization. Therefore, we select this SLAM system to be the base of trajectory servoing.

3.1.2 Code Base

Good Feature (GF) ORB-SLAM

The code platform used in the thesis is based on ORB-SLAM, a feature-based SLAM system that operates in real time, for small and large indoor and outdoor environments [43]. It is compatible with multiple visual sensors: monocular camera, RGBD camera and stereo camera [44], therefore, it is a commonly used platform in navigation and mapping problems. The use of ORB features guarantees an outstanding balancing between accuracy and efficiency: ORB is not as time-consuming as SIFT and SURF but still remains good invariance to rotation, scale and illumination. Therefore, we are allowed to assign more computational resources to pose optimization, loop closing and camera relocalization from covisibility graphs and essential graphs without compromising much accuracy.

The ORB-SLAM system includes three parallel threads: tracking, local mapping, and loop closing (see Figure 3.1). The tracking module tracks features from the previous frame or from the searching of local map reprojection. Then it uses all the features to update pose

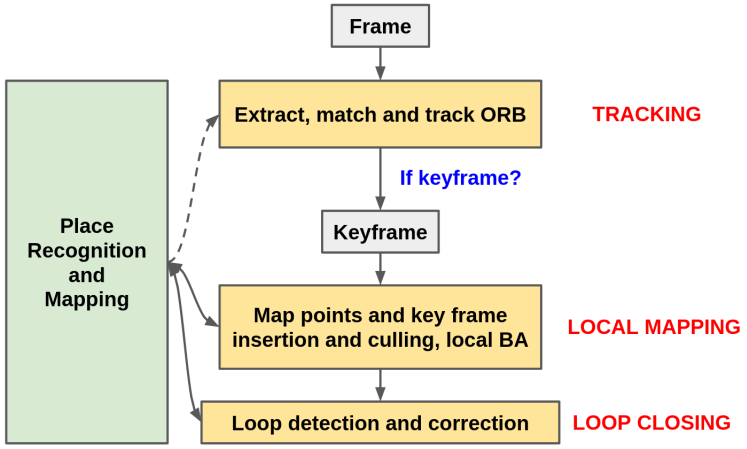

Figure 3.1: ORB-SLAM System [43]

information and decides if a new keyframe should be inserted. Next, the key frames are sent to local mapping module that performs local BA by g2o [72] to obtain an optimal camera pose and feature positions. Selecting for high quality points and culling for redundant key frames are also executed in the local mapping thread. The loop closing first detects the loop from the key frames and then performs pose graph optimization to achieve global consistency. These components coordinate and facilitate with each other towards more accurate robot pose estimation and environment mapping.

Good Feature (GF) ORB-SLAM proposed in [6] modifies the tracking thread of ORB-SLAM by selecting a subset of the matched inliers for more accurate pose estimation. It investigates a common bundle adjustment pose optimization process in feature based VO/VSLAM:

$$\arg\min \|h(x,p) - z\|^2 \tag{3.1}$$

where x is the pose of the camera, p is the 3D feature points and z the is corresponding measurements on 2D image frame. The measurement function, $h(x,p)$, is a combination of world-to-camera transformation and pinhole projection. Based on Gaussian-Newton:

$$x^{(s+1)} = x^{(s)} - H_x^+ \left(z - h\left(x^{(s)}, p\right)\right) \tag{3.2}$$

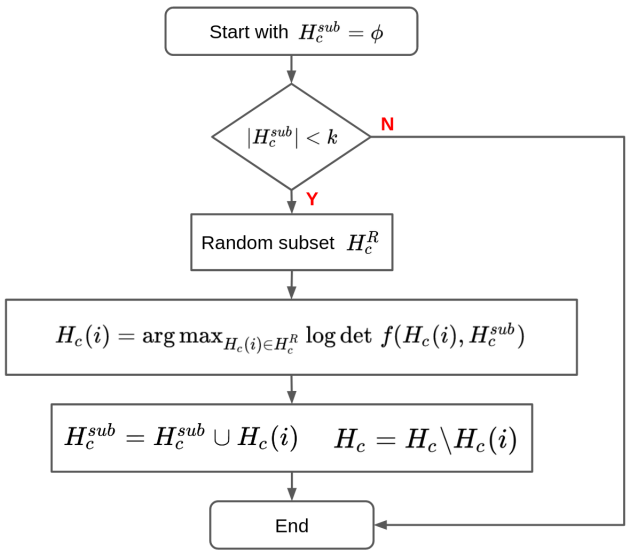

Figure 3.2: Good Feature Selection

where H_x is the Jacobian of $h\left(x^{(s)}, p\right)$. It connects the pose optimization error to measurement and map errors:

$$\epsilon_x = H_x^+ \left(\epsilon_z - H_p \epsilon_p\right) \tag{3.3}$$

H_p is a block diagonal matrix of size $2n \times 3n$, where n is the number of matched features.

The discussion about the aspects of map and measurement variance and bias concludes that optimizing the spectral properties of H_c, the combination of H_x and H_p, could improve least squares pose optimization accuracy. Therefore, the stochastic-greedy-based MaxlogDet feature selection algorithm is used to meet real-time demand of ORB-SLAM. It first selects a subset of all the observations, then finds the maximizer observation of log-det and combines it into the set of selected features until a predefined number of features has been selected (see Figure 3.2).

Codebase Implementation

We need a code base on which we could build our trajectory servoing system. For quantifiable and reproducible outcomes, the ROS/Gazebo SLAM evaluation environment from

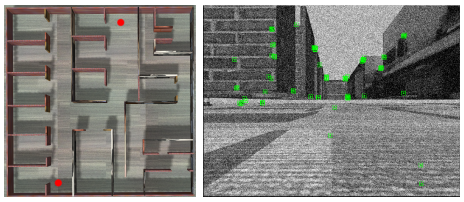

Figure 3.3: Gazebo environment top view and robot view with SLAM features. The two red dots are the start poses of the robot. The right figure is the simulated camera image with Gaussian noise and tracked features from (GF) ORB-SLAM.

[1] is used for the tests. The simulated robot is a Turtlebot. To benchmark the trajectory servoing performance with camera noise, zero-mean Gaussian noise is added to its stereo camera. Figure 3.3 shows a top down view of the world plus robot views without and with image noise. Part of the robot's software stack includes the Good Feature (GF) ORB-SLAM system [6] for estimating camera poses. It is configured to work with a stereo camera and integrated into a loosely coupled, visual-inertial (VI) system [1, 73] based on a multi-rate filter to form a VI-SLAM system. The trajectory servoing system will interface with the GF-ORB-SLAM system to have access to tracked features for IBVS.

We mainly use four ROS packages to achieve the system. The package of gf_orb_slam_2 includes GF ORB-SLAM2 functions that solve and publish visual odometry (VO) for robot [1]. Then VO is fused with IMU pose estimation in ethzasl_msf package, which is an open source library to implement MultiSensor-fusion extended kalman filter (MSF-EKF) [73]. The trajectory_servoing package contains trajectory planning, feature trajectory generating and IBVS controller functions. Finally trajectory_servoing_benchmark package organizes the functions above, repeats trajectory servoing processes records robot poses and analyze tracking error. Other related auxiliary ROS packages for ROS messages defining, odometry converting etc. are not introduced here due to space limitations.

3.1.3 Noise Effects in V-SLAM

Image noise will significantly affect the matching, tracking and localization in V-SLAM system thus the accuracy of visual odometry. The odometry dramatically drifts since un-

certainty of bundle adjustment (BA) and robot motion will locally accumulate when no global feedback (e.g. global BA, loop closure) is activated, especially for an ORB-SLAM system that uses only relative motion without GPS absolute positioning. More specificly, (3.4) gives the covariance of camera pose error [41]:

$$\Sigma_{G_t} = J\Sigma_{G_{t-1}}J^T + \Sigma_{M_{t-1,t}} \tag{3.4}$$

where Σ_{G_t} and Σ_{M_t} are the covariance matrices of camera pose and relative motion respectively. J is the Jacobian matrix relating robot states during consecutive time steps. It is clear that the pose estimation covariance is propagated along the trajectory and relative motion uncertainty is continuously added up.

Experimentally, benchmarking results on ORB-SLAM shows mild image noise will contribute to significant pose estimation error [38]. When the standard deviation (STD) of irradiance-independent image noise σ_c increases from $\sigma_c = 0.005$ to $\sigma_c = 0.05$, the RMSE of ATE increases from 6.7 cm to 26.2 cm, a non-negligible increase of 19.5 cm.

Therefore, many improvements (e.g. feature filtering, feature selection, image denoising etc.) are designed to increase V-SLAM localization accuracy in noisy environment (see §2.4 and §2.5). Such improvement will improve the performance of SLAM pose based trajectory tracking, however, trajectory servoing that uses IBVS controller is possible to be even more robust to noise compared with these advanced V-SLAM pose based tracking system. In §6, the good feature (GF) ORB-SLAM [49] will be the strong baseline methods in which localization accuracy has been significantly improved from basic ORB-SLAM.

3.2 Visual Servoing (VS)

3.2.1 Visual Servoing Basics

Visual servoing employs vision information in close loop systems to control robot motion. It mainly includes two categories: position based visual servoing and image based visual

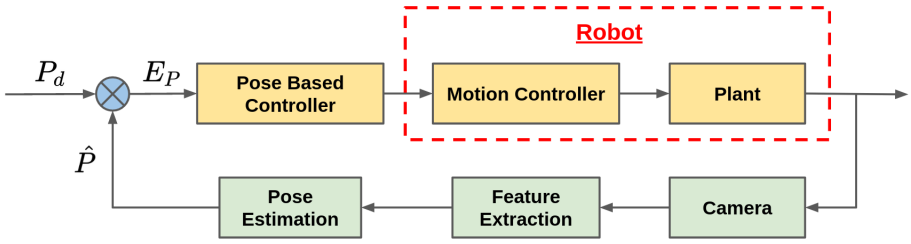

Figure 3.4: PBVS Control System

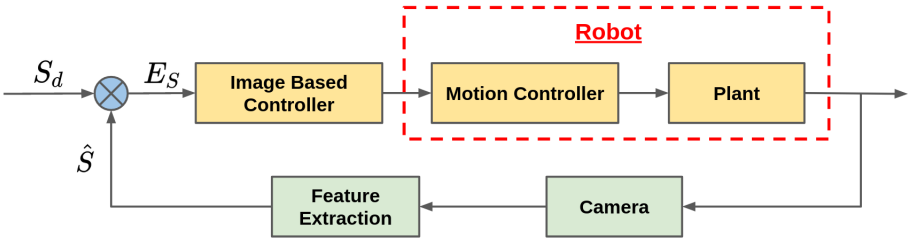

Figure 3.5: IBVS Control System

servoing. The former one builds the mapping between image signals and robot pose and compares the desired and current pose to form close loop control. The latter one directly designs a control law in image space to make current features converge to desired ones and relates feature velocity to robot velocity with the image Jacobian.

The block diagrams of the two visual servoing control system are shown at Figure 3.4 and Figure 3.5. PBVS has an extra pose observer in the feedback loop and feeds pose differences to its pose based controller. IBVS directly uses the image Jacobian to convert image feature error to control variable. For both systems, linear and angular velocities are usually used as control commands that send to robot low-level servo driver (red dashed blocks represent robot).

3.2.2 Image-Based Visual Servoing Rate Equations

The core algorithm builds on IBVS [74], thus this section covers IBVS with an emphasis on how it directly relates the velocity of image features to the robot velocities via the image Jacobian [19, 20]. These equations will inform the trajectory tracking problem under non-holonomic robot motion. We use more modern notation from geometric mechanics

[75] since it provides equations that better connect to contemporary geometric control and SLAM formulations for moving rigid bodies.

Non-Holonomic Robot and Camera Kinematic Models

Let the motion model of the robot be a kinematic Hilare robot model where the pose state $g_{\mathcal{R}}^{\mathcal{W}} \in SE(2)$ evolves under the control $u = [\nu, \omega]^T$ as

$$\dot{g}_{\mathcal{R}}^{\mathcal{W}} = g_{\mathcal{R}}^{\mathcal{W}} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix} = g_{\mathcal{R}}^{\mathcal{W}} \cdot \xi_u, \tag{3.5}$$

for ν a forward linear velocity and ω an angular velocity, and $\xi_u \in \mathfrak{se}(2)$. The state is the robot frame $\mathcal{R}$ relative to the world frame $\mathcal{W}$. The camera frame $\mathcal{C}$ is presumed to be described as $h_{\mathcal{C}}^{\mathcal{R}}$ relative to the robot frame. Consequently, camera kinematics relative to the world frame are

$$\dot{h}_{\mathcal{C}}^{\mathcal{W}} = g_{\mathcal{R}}^{\mathcal{W}} \cdot h_{\mathcal{C}}^{\mathcal{R}} \cdot \mathrm{Ad}_{h_{\mathcal{C}}^{\mathcal{R}}}^{-1} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix} . = g_{\mathcal{R}}^{\mathcal{W}} \cdot h_{\mathcal{C}}^{\mathcal{R}} \cdot \zeta_u \tag{3.6}$$

with $\zeta_u \in \mathfrak{se}(2)$. Now, let the camera projection equations be given by the function $H : \mathbb{R}^3 \to \mathbb{R}^2$ such that a point $q^{\mathcal{W}}$ projects to the camera point $r = H \circ h_{\mathcal{W}}^{\mathcal{C}}(q^{\mathcal{W}})$. Under camera motion, the differential equation relating the projected point to the camera velocity is

$$\dot{r} = \mathrm{D}H(q^{\mathcal{C}}) \cdot \left(\zeta_u \cdot q^{\mathcal{C}}\right), \tag{3.7}$$

where the point q is presumed to be static, i.e., $\dot{q} = 0$, and $q^{\mathcal{C}} = h_{\mathcal{W}}^{\mathcal{C}} q^{\mathcal{W}}$, and D is the differential operator. Since the operation $\xi \cdot q$ is linear for $\xi \in \mathfrak{se}(2)$, $q \in \mathbb{R}^3$, it can be

written as a matrix-vector product $M(q)\xi$ leading to,

$$\dot{r} = \mathbf{D}H(q^{\mathcal{C}}) \cdot M(q^{\mathcal{C}})\zeta_u = L(q^{\mathcal{C}})\zeta_u. \tag{3.8}$$

where $L : \mathbb{R}^3 \times \mathfrak{se}(2)$ is the Image Jacobian. Given the point and projection pair $(q, r) \in \mathbb{R}^3 \times \mathbb{R}^2$, L works out to be

$$L(q) = L(q, r) = \begin{bmatrix} -\frac{f}{q^3} & 0 & r^2 \\ 0 & -\frac{f}{q^3} & -r^1 \end{bmatrix}, \tag{3.9}$$

where f is the focal length. Recall that $r = H(q)$. Re-expressing it as a function of (q, r) simplifies its written form, and exposes what information is available from the image directly $r \in \mathbb{R}^2$ and what additional information must be known to compute it: coordinate q^3 from $q^{\mathcal{C}} \in \mathbb{R}^3$ in the camera frame, which is also called depth. With a stereo camera, the depth value is triangulated. The next section will use these equations for image-based trajectory tracking.

CHAPTER 4
TRAJECTORY SERVOING SYSTEM

4.1 Trajectory Servoing

Pose-based feedback control is generally used in trajectory tracking. In the absence of global position knowledge, V-SLAM system provides the pose estimates needed for feedback. Tracking performance will largely depend on localization accuracy since the V-SLAM system serves as an observer of robot pose in the feedback loop. Figure 4.1 shows the block diagram of the typical V-SLAM pose-based control architecture. Pose estimation is sensitive to image measurements. Any uncertainty (e.g. IMU bias, camera noise, calibration error, etc.) will accumulate [2], immediately affect the feedback loop and further affect the control result. In addition, high visual processing latency will cause lower estimation accuracy due to late correction of raw IMU data [1]. To overcome these shortcomings, we design a mobile robot tracking method that bypasses pose feedback and provides feedback control directly through the image space.

The standard IBVS equations presented in §3.2.2 typically apply to tracked features with known static positions in the world (relative to some frame attached to these positions). As described in §2.1 and §2.2, a visual map or visible targets are usually necessary. The prerequisites needed for stable tracking and depth recovery of features are major challenges regarding the use of visual servoing in unknown environments. Fortunately, they are all possible to obtain based on information and modules available within the software stack of an autonomous mobile robot. This section describes the basic *trajectory servoing* implementation and describes how to build a solution that satisfies the prerequisites. It focuses on short-term navigation where sufficient image features remain within the field of view for the entire trajectory.

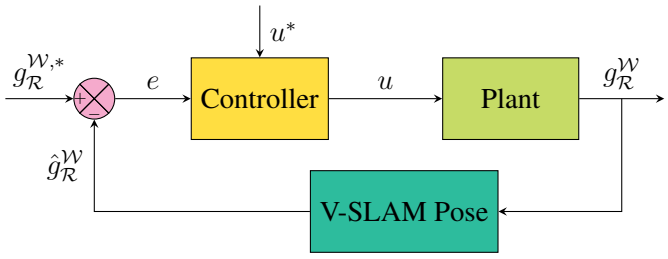

Figure 4.1: Block Diagram for V-SLAM Posed Based Control

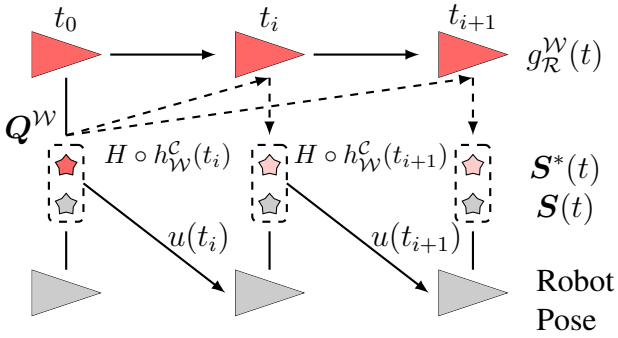

Figure 4.2: Trajectory servoing process. Matches from $\boldsymbol{S}^*$ to $\boldsymbol{S}$ define the control u, where $\boldsymbol{S}^*(t)$ is defined by the desired trajectory.

Trajectory servoing requirements condense down to the following: 1. A set of image points, $\boldsymbol{S}^*(t_0)$, with known (relative) positions; 2. A given trajectory and control signal for the robot starting at the robot's current pose or nearby, $g_{\mathcal{R}}^{\mathcal{W}}(t_0)$; and 3. The ability to index and associate the image points across future image measurements, $\boldsymbol{S}^*(t) \leftrightarrow \boldsymbol{S}(t)$, when tracking the trajectory. The trajectory servoing process and variables are depicted in Figure 4.2. The autonomy modules contributing this information are the navigation and V-SLAM stacks. The navigation stack generates a trajectory to follow. An indirect, feature-based V-SLAM stack will keep track of points in the local environment and link them to previously observed visual features while also estimating their actual position relative to the robot.

4.1.1 Trajectory and control signals

Define $\boldsymbol{S} = \{ r_i \}_1^{n_{\mathrm{F}}} \subset \mathbb{R}^2$ as a set of image points in the current camera image, sourced from the set $\boldsymbol{Q} = \left\{ q_i^{\mathcal{W}} \right\}_1^{n_{\mathrm{F}}} \subset \mathbb{R}^3$ of points in the world frame. Suppose that the robot

should attain a future pose given by g^*, for which the points in $\boldsymbol{Q}$ will project to the image coordinates $\boldsymbol{S}^* = H \circ (g^* h_{\mathcal{C}}^{\mathcal{R}})^{-1}(\boldsymbol{Q})$. For simplicity, ignore field of view issues and occlusions between points. Their effect would be such that only a subset of the points in $\boldsymbol{Q}$ would contribute to visual servoing.

Assume that a specific short-duration path has been established as the one to follow, and has been converted into a path relative to the robot's local frame. It either contains the current robot pose in it, or has a nearby pose. Contemporary navigation stacks have a means to synthesize both a time varying trajectory and an associated control signal from the paths. Here, we apply a standard trajectory tracking controller [76] to generate $\xi_u^*(t)$ and $g_{\mathcal{R}}^{\mathcal{W},*}(t)$ by forward simulating (3.5); note that ξ_u^* contains the linear velocity ν^* and angular velocity ω^*. Some navigation stacks use optimal control synthesis to build the trajectory. Either way, the generated trajectory is achievable by the robot.

In the time-varying trajectory tracking case, we assume that a trajectory reference $h_{\mathcal{C}}^{\mathcal{W}}(t)$ exists along with a control signal $u^*(t)$ satisfying (3.6). It would typically be derived from a robot trajectory reference $g_{\mathcal{R}}^{\mathcal{W}}(t)$ and control signal $u^*(t)$ satisfying (3.5). Using those time-varying functions, the equations in (3.6) are solved to obtain the image coordinate trajectories. Written in short-hand to expose only the main variables, the forward integrated feature trajectory $\boldsymbol{S}^*$ is:

$$\begin{aligned} \dot{\boldsymbol{S}}^* &= L \circ h_{\mathcal{W}}^{\mathcal{C}}(t)(\boldsymbol{Q}^{\mathcal{W}}) \cdot \zeta_{u^*(t)}, \text{ with} \\ \boldsymbol{S}^*(0) &= H \circ h_{\mathcal{W}}^{\mathcal{C}}(0)(\boldsymbol{Q}^{\mathcal{W}}). \end{aligned} \tag{4.1}$$

It will lead to a realizable visual servoing problem where ν^*, ω^*, and $\boldsymbol{S}^*(t)$ are consistent with each other. The equations will require converting the reference robot trajectory to a camera trajectory $h_{\mathcal{C}}^{\mathcal{W},*}(t)$ using $\mathrm{Ad}_{h_{\mathcal{C}}^{\mathcal{R}}}^{-1}$.

4.1.2 Features and feature paths

The V-SLAM module provides a pool of visible features with known relative position for the current stereo frame, plus a means to assess future visibility if desired. Taking this pool to define the feature set $\boldsymbol{S}^*(0)$ gives the final piece of information needed to forward integrate (4.1) and generate feature trajectories $\boldsymbol{S}^*(t)$ in the left camera frame. This process acts like a short-term teach and repeat feature trajectory planner but is *simulate* and repeat, for on-the-fly generation of the repeat data.

A less involved module could be used besides a fully realized V-SLAM system, however doing so would require creating many of the fundamental building blocks of an indirect, feature-based V-SLAM system. Given the availability of strong performing open-source, real-time V-SLAM methods, there is little need to create a custom module. In addition, an extra benefit to tracked features through V-SLAM system is that a feature map is maintained to retrieve same reappeared features. As will be shown, this significantly improves the average lifetime of features, especially compared to a simple frame by frame tracking system without the feature map.

After the V-SLAM feature tracking process, we are already working with this feasible set whereby the indexed elements in $\boldsymbol{S}$ correspond exactly to their counterpart in $\boldsymbol{S}^*$ with the same index, i.e., the sets are *in correspondence*.

4.1.3 Trajectory Servoing Control

Define the error to be $\boldsymbol{E} = \boldsymbol{S} - \boldsymbol{S}^*$ where elements with matched indices are subtracted. The error dynamics of the points are:

$$\boldsymbol{E} = \dot{\boldsymbol{S}} - \dot{\boldsymbol{S}}^* = L_u(h^{\mathcal{C}}_{\mathcal{W}}(\boldsymbol{Q}), \boldsymbol{S}; h^{\mathcal{R}}_{\mathcal{C}}) \cdot u - \dot{\boldsymbol{S}}^* \tag{4.2}$$

where we apply the same argument adjustment as in (3.9) so that dependence is on image features then point coordinates as needed. Further, functions or operations applied to

indexed sets will return an indexed set whose elements correspond to the input elements from the input indexed set. Since the desired image coordinates $\boldsymbol{S}^*$ are not with respect to a static goal pose but a dynamic feature trajectory, $\dot{\boldsymbol{S}}^* \neq 0$, see (4.1). Define $\boldsymbol{e}$, $\boldsymbol{s}$, $\boldsymbol{s}^*$, and $\boldsymbol{L}$ to be the vectorized versions of $\boldsymbol{E}$, $\boldsymbol{S}$, $\boldsymbol{S}^*$ and L. Then,

$$\dot{\boldsymbol{e}} = \boldsymbol{L}(h^{\mathcal{C}}_{\mathcal{W}}(\boldsymbol{q}), \boldsymbol{s}; h^{\mathcal{R}}_{\mathcal{C}}) \cdot u - \dot{\boldsymbol{s}}^* \tag{4.3}$$

is an overdetermined set of equations for u when $n_{\mathrm{F}} > 2$. Removing the functional dependence and breaking apart the different control contributions, the objective is to satisfy,

$$\dot{\boldsymbol{e}} = \boldsymbol{L} \cdot u - \dot{\boldsymbol{s}}^* = \boldsymbol{L}^1\nu + \boldsymbol{L}^2\omega - \dot{\boldsymbol{s}}^* = -\lambda\boldsymbol{e}. \tag{4.4}$$

A solution is to define,

$$\omega = \left(\boldsymbol{L}^2\right)^{\dagger}\left(-\boldsymbol{L}^1\nu - \lambda\boldsymbol{e} + \dot{\boldsymbol{s}}^*\right), \tag{4.5}$$

so that

$$\dot{\boldsymbol{e}} = -\lambda\boldsymbol{e} + \Delta\boldsymbol{e}, \tag{4.6}$$

where $\Delta\boldsymbol{e}$ is mismatch between the true solution and the computed pseudo-inverse solution. If the problem is realizable, then $\Delta\boldsymbol{e}$ will vanish and the robot will achieve the target pose. If $\Delta\boldsymbol{e}$ does not vanish, then there will be an error (usually some fixed point $\boldsymbol{e}_{ss} \neq 0$). It is common for the robot's forward velocity to be set to a reasonable constant $\nu = \bar{\nu}$ in the angular control solution (4.4). This law drives the camera frame to the target pose (relative to $\mathcal{W}$).

Since the desired feature set is time varying from $\boldsymbol{S}^*(t)$ in equation (4.1):

$$\dot{\boldsymbol{s}}^* = \boldsymbol{L}^1(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\nu^* + \boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\omega^*. \tag{4.7}$$

The vectorized steering equations (4.5) lead to

$$\omega = \left(\boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}}, \boldsymbol{s})\right)^{\dagger}\left(\boldsymbol{L}^1(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\nu^* - \boldsymbol{L}^1(\boldsymbol{q}^{\mathcal{C}}, \boldsymbol{s})\nu + \boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\omega^* - \lambda \boldsymbol{e}\right). \tag{4.8}$$

They consist of feedforward terms derived from the desired trajectory and feedback terms to drive the error to zero. The feedforward terms should cancel out the $\dot{\boldsymbol{S}}^*$ term in (4.2), or equivalently the now non-vanishing $\dot{\boldsymbol{s}}^*$ term in (4.3). When traveling along the feature trajectory $\boldsymbol{S}^*(t)$, the angular velocity ω is computed from (4.8), where starred terms and ν are known quantities. To the best of our knowledge, no general IBVS tracking equations have been derived that combine feed-forward and feedback control elements.

4.2 Long Distance Trajectory Servoing

Short-term trajectory servoing cannot extend to long trajectories due to feature impoverishment. When moving beyond the initially visible scene, a more comprehensive trajectory servoing system would augment the feature pool $\boldsymbol{S}^*$ with new features. Likewise, if navigation consists of multiple short distance trajectories, then the system must have a regeneration mechanism for synthesizing entirely new desired feature tracks for the new segment. The overlapping needs for these two events inform the creation of a module for feature replenishment and trajectory extension.

4.2.1 Feature Replenishment

The number of feature correspondences n_{F} in $\boldsymbol{S}$ and $\boldsymbol{S}^*$ indicates whether trajectory servoing can be performed without concern. Let the threshold τ_{fr} determine when feature replenishment should be triggered. Define $\boldsymbol{S}_i^*(t)|_{t_{i,s}}^{t_{i,e}}$ as the i^{th} feature trajectory starting from $t_{i,s}$ and ending at $t_{i,e}$. The case $i = 0$ represents the first feature trajectory segment generated by (4.1) for $t_{i,s} = 0$, integrated up to the maximum time t_{end} of the given tra-

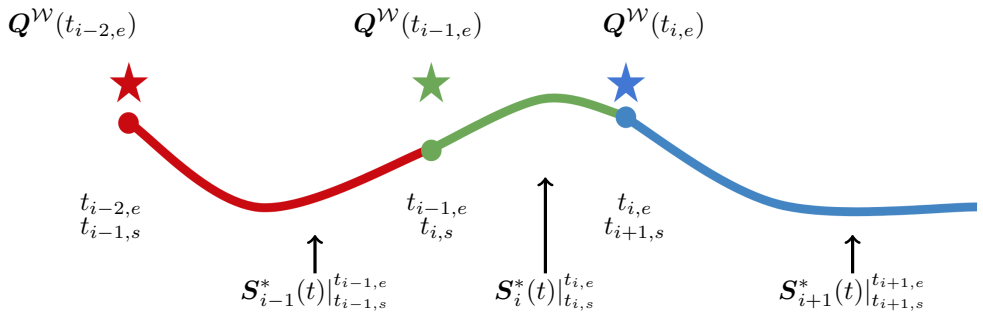

Figure 4.3: Feature replenishment process. There are three segments of feature trajectories. Stars are observed point sets at corresponding time. Each circle is the start or end time of next or this segment of feature trajectory. Three feature trajectories are generated by the feature replenishment equation (4.9).

jectory. The time varying function $n_{\mathrm{F}}(t)$ is the actual number of feature correspondences between $\boldsymbol{S}(t)$ and $\boldsymbol{S}^*_i(t)$ as the robot proceeds.

When $n_{\mathrm{F}}(t) \geq \tau_{fr}$, the feature trajectory $\boldsymbol{S}^*_i(t)$ may be used for trajectory servoing. When $n_{\mathrm{F}}(t) < \tau_{fr}$, the feature replenishment process will be triggered at the current time and noted as $t_{i,e}$. The old feature trajectory $\boldsymbol{S}^*_i(t)|^{t_{i,e}}_{t_{i,s}}$ is finished. A new feature trajectory is generated with

$$\boldsymbol{S}^*_{i+1}(t)|^{t_{i+1,e}}_{t_{i+1,s}} = H \circ (g^*(t_{i,e}, t) h^{\mathcal{R}}_{\mathcal{C}})^{-1}(\boldsymbol{Q}^{\mathcal{W}}(t_{i,e})), \tag{4.9}$$

where $g^*(t_{i,e}, t)$ is the transformation between the current robot pose and a future desired pose ($t > t_{i,e}$) on the trajectory. The poses behind the robot are not included. The set $\boldsymbol{Q}^{\mathcal{W}}(t_{i,e})$ consists of observed points at the current time $t_{i,e}$. The feature pool is augmented by these current features. When this regeneration step is finished, the exact time will be assigned as the $t_{i+1,s}$. Trajectory servoing is performed on this new feature trajectory until the regeneration is triggered again or the arriving at the end of the trajectory. The process of regenerating new feature tracks is equivalent to dividing a long trajectory into a set of shorter segments pertaining to the generated feature trajectory segments. An example of this feature replenishment is shown in Figure 4.3.

During navigation, (4.9) requires the current robot pose relative to the initial pose to

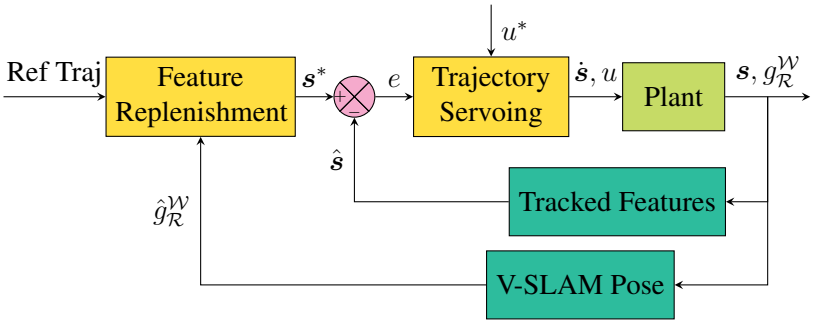

Figure 4.4: Block Diagram for long-term Trajectory Servoing

be known. In the absence of an absolute reference or position measurement system, the only option available is to use the estimated robot pose from V-SLAM, or some equivalent process. Although there are some drawbacks to relying on V-SLAM, it attempts to keep pose estimation as accurate as possible over long periods through feature mapping, bundle adjustment, loop closure, etc.. To further couple V-SLAM and trajectory servoing, we design a multi-loop scheme, see Figure 4.4. The inner loop is governed by trajectory servoing with V-SLAM tracked features. The V-SLAM estimated pose will only be explicitly used in the outer loop when performing feature replenishment. In this way, the inner visual feedback loop will not be affected by the uncertainty from pose estimation. Plus the outer loop will only be activated when starting a new feature replenishment; and since the visual information is received, fused, and optimized with an IMU, it generates a robot pose that is more reliable than the raw poses used in the inner loop of V-SLAM pose based control (shown in Figure 4.1). However, relying on V-SLAM still introduces measurement error or drift, which means that long-term trajectory servoing along an absolute, desired trajectory will accrue error.

4.3 Trajectory Servoing Benefits in Image Noise

Simply speaking, trajectory servoing is composed of IBVS with V-SLAM feature tracking and replenishing. Therefore, it not only inherits all the IBVS benefits but also enjoys a stable feature tracking from V-SLAM.

IBVS has the property to be resistant to noise [19, 20]. Relevant experiments [52]

have shown it will only have a fairly gradual, linear increase of error under increasing image noise level. It also shows there appears to be no significant benefit by increasing the number of feature points in IBVS. These discoveries support the potential use of trajectory servoing when navigating in a noisy environment with a time-variant feature pool.

In addition, V-SLAM feature tracking process will automatically track those low uncertainty features with high probability and vice versa. It guarantees trajectory servoing primarily to navigate with those low uncertainty features and could be verified by Monte Carlo simulation in Gazebo environment. The base configurations of trajectory servoing has been described in §3.1.2. We repeat the Monte Carlo simulations with the images contaminated by additive Gaussian noise at different standard deviations (STD) σ_I from 0.02 to 0.2 with the step of 0.01, i.e.

$$\sigma_I = 0.02, 0.03, 0.04, ..., 0.2 \tag{4.10}$$

Given Gazebo simulation platform, we are able to publish zero control command to Turtlebot to maintain its initial position and record consecutive R frames with tracked points from the left camera.

To avoid potential confusions, we name the map points as *points* and their observations as *features* in this section. Tracking thread of ORB-SLAM (see Figure 3.1) will assign a unique and permanent ID to all the observed features associated to the same map point for consistent tracking. Without the loss of generality, we assume the allocated ID numbers $i = 1, 2, ..., Q$, where Q is the total number of visible points in R frames. And i^{th} point could be observed at frames $F_i = \{f_j | j = 1, 2, ..., N_i\}$ with the coordinates $X_i = \{x_j = (u_j, v_j) \in \mathbb{R}^2\}_{j=1}^{N_i}$, where $|F_i| = N_i$ is the number of frames i^{th} point appears in. Under these assumptions and definitions, we could easily obtain and represent the numerical characteristics of i^{th} feature.

Define the observability score P_i of i^{th} point as the ratio between the number of frames

where i^{th} point could be observed N_i and the total frame number R, i.e.

$$P_i = \frac{N_i}{R} \in [0, 1] \tag{4.11}$$

The sample mean $\bar{X}_i$ and variance D_i are defined as:

$$\bar{X}_i = \frac{1}{N_i} \sum_{j=1}^{N} x_j \tag{4.12}$$

$$D_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \left(X_j - \bar{X}_j \right)^2 \tag{4.13}$$

Figure 4.5 (a) shows the distributions of tracked features when $\sigma_I = 0.03$ and each observed feature has one vote from its observing score and variance. X axis shows the variance of the tracked features, Y axis shows observability scores (i.e. the observation probability of each point in one frame which is obtained from Equation 4.11), and Z axis shows the frequency of tracked features fallen in each bin.

The STDs of feature coordinate observations corresponding to i^{th} point implies their accuracy and reliability, hence, the means of all visible point STDs reflects the general uncertainty at each noise level. Therefore, this mean could also be weighted by observability score and compared with the raw one to detect if those points with lower STDs will have higher observability scores. Suppose $\bar{D}$ to be mean of STDs and $\bar{D}_w$ to be weighted one, we obtain:

$$\bar{D} = \frac{1}{Q} \sum_{i=1}^{Q} D_i \tag{4.14}$$

$$\bar{D}_w = \sum_{i=1}^{Q} P_i D_i \tag{4.15}$$

Figure 4.5 (b) presents the weighted and unweighted results at each noise level. It is observed that when the image is exposed to noise, the weighted means of STD is lower than

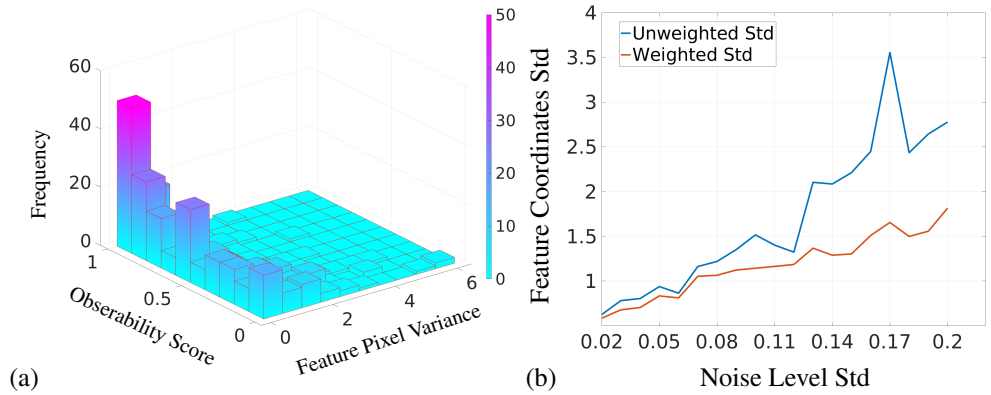

Figure 4.5: (a) Feature distribution with 0.03 noise standard deviation; (b) Feature coordinates standard deviation with different noise levels. Blue line is the unweighted std. Red line is the weighted std.

the unweighted ones.

These simulations show that most of the low variance features could be observed in a large number of frames through the 200 frame and vice versa. It supports the claim that those noise resistant features are more likely to be tracked as a subset of original ORB features that are spatial evenly distributed. Such prerequisite of noise robustness is beneficial to trajectory servoing since it is highly possible that those features vulnerable to noise will not appear in the first frame used for generating the feature trajectory thus not served as desired features in trajectory servoing process.

CHAPTER 5

UNCERTAINTY BASED TRAJECTORY SERVOING

From the aforementioned analysis, we have reasons to think trajectory servoing can perform better with feature noise. Furthermore, unlike pure IBVS where tracking tasks are defined on the image space, trajectory servoing tracks a feature trajectory in the image space which projects features seen from initial pose forward along the desired poses of the planned trajectory in Cartesian space. Therefore, the uncertainty of features in the initial image will propagate along feature trajectories and influence the accuracy of trajectory servoing. In the following part we will reveal that the uncertainty modeling and propagation are helpful to improve the design of IBVS controller.

5.1 Uncertainty Modeling and Propagation

We first specify the notations used here. Define the j^{th} desired features in the desired image at time step t_i as $\boldsymbol{S}_i^* = \{\boldsymbol{s}_j^{i*} = (s_j^{i1,*}, s_j^{i2,*}) \in \mathbb{R}^2\}_{j=1}^{n_i}$ sourced from $\boldsymbol{Q}_i^* = \{\boldsymbol{q}_j^{i*} = (q_j^{i1,*}, q_j^{i2,*}, q_j^{i3,*}) \in \mathbb{R}^3\}_{j=1}^{n_i}$ in the camera frame at t_i, where n_i is the number of desired features in the frame at t_i. Also define the *observation* of $\boldsymbol{S}_i^*$ as $\hat{\boldsymbol{S}}_i^* = \{\hat{\boldsymbol{s}}_j^{i*} = (\hat{s}_j^{i1,*}, \hat{s}_j^{i2,*})\}_{j=1}^{n_i}$ sourced from $\hat{\boldsymbol{Q}}^* = \{\hat{\boldsymbol{q}}_j^{i*} = (\hat{q}_j^{i1,*}, \hat{q}_j^{i2,*}, \hat{q}_j^{i3,*}) \in \mathbb{R}^3\}_{j=1}^{n_i}$ (*observation* here means the desired feature coordinated calculated from noised initial features, namely what we are able to obtain in reality compared with precise values without noise effect). Features seen by camera are defined in a similar way with notation $\boldsymbol{S}, \boldsymbol{Q}, \hat{\boldsymbol{S}}$. The camera observation variance along the image width and height directions are defined as σ_u^2, σ_v^2. We assume the trajectory servoing process starts at time step t_0. $h_{\mathcal{C}}(t_i, t_d) \in SE(2)$ is the camera pose transformation from time step t_i to time step t_d. Note the homogeneous transform of camera pose

from t_0 to t_d as:

$$h_{\mathcal{C}}(t_0, t_d) = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \tag{5.1}$$

In the following parts, $\boldsymbol{K}$ is the camera matrix, f and b are the focal length and baseline of the stereo camera, and d_j^i and $\hat{d}_j^i$ are the disparities and its measurement of the j^{th} feature at time step t_i . To simplify the notations, we will sometimes ignore the indicator of time i or the feature index j in the remaining part of this chapter.

5.1.1 Uncertainty propagation from image to 3D space

In this section we discuss how to propagate the Gaussian uncertainty from image to 3D world. The basic assumption here is that different features and left and right stereo camera are observed independently.

The following discussion is about j^{th} feature pair. If no subscript or superscript is applied, we indicate the j^{th} feature in i^{th} frame by default. First estimate the distribution of the measurement of disparity $\hat{d}$:

$$\begin{aligned} \hat{d} &= \hat{s}_r^1 - \hat{s}_l^1 \\ \hat{s}_r^1 \sim \mathcal{N}(s_r^1, \sigma_u^2), \hat{s}_l^1 &\sim \mathcal{N}(s_l^1, \sigma_u^2) \end{aligned} \tag{5.2}$$

The distribution of $\hat{d}$ is:

$$\hat{d} \sim \mathcal{N}(s_r^1 - s_l^1, \sigma_u^2 + \sigma_u^2 = 2\sigma_u^2) \triangleq \mathcal{N}(\mu_d, \sigma_d^2) \tag{5.3}$$

The next step is to estimate the distribution of the measured feature depth $\hat{q}^3$. Considering

the first order linear approximation of the depth equation of stereo cameras:

$$\hat{q}^3 = \frac{fb}{\hat{d}} \approx \frac{fb}{d} - \frac{fb}{d^2}(\hat{d} - d) \tag{5.4}$$

And calculate the distribution of $\hat{q}^3$ as:

$$\hat{q}^3 \sim \mathcal{N}\left(\frac{fb}{d}, \left(\frac{fb}{d^2}\right)^2 \sigma_d^2\right) \tag{5.5}$$

This linear approximation holds true if the error of disparity measurement is small. When the depth observation error is small enough, the (inverse) projection model of the camera is nearly locally linear. Therefore, linear approximation still applies when propagating uncertainty from image space to 3D Cartesian space. Here is the pinhole inverse projection model:

$$\hat{\boldsymbol{q}} = \hat{q}^3 \boldsymbol{K}^{-1}\, {}_h\hat{\boldsymbol{s}} \approx \underbrace{\boldsymbol{K}^{-1} \begin{bmatrix} \hat{q}^3 & 0 & 0 \\ 0 & \hat{q}^3 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{J}} \underbrace{\begin{bmatrix} \hat{s}^1 \\ \hat{s}^2 \\ \hat{q}^3 \end{bmatrix}}_{\hat{\boldsymbol{X}}_H} \tag{5.6}$$

where ${}_h\hat{\boldsymbol{s}}$ is the homogeneous coordinates of $\hat{\boldsymbol{s}}$. The covariance matrix is $\mathrm{Cov}(\hat{\boldsymbol{X}}_H) = \mathrm{diag}(\sigma_u^2, \sigma_v^2, \sigma_{qz}^2)$, therefore:

$$\hat{\boldsymbol{q}} \sim \mathcal{N}(\hat{\boldsymbol{X}}_H, \boldsymbol{J}\mathrm{diag}(\sigma_u^2, \sigma_v^2, \sigma_{qz}^2)\boldsymbol{J}^T) \tag{5.7}$$

5.1.2 Uncertainty propagation from 3D space to desired image

The distribution of feature 3D coordinates in the current camera frame is obtained in the previous section. However, the needed uncertainty in trajectory servoing should be those of desired features which are directly used in the IBVS controller. To obtain the distribution of desired features on 2D image space, the distribution in the current camera frame should

be transformed to the desired frame and projected back from 3D space to the desired image space. In this section, we suppose the current time is $t = t_0$ and the desired time we expect to transform to is $t = t_d$. The j^{th} 3D coordinate transform from t_0 to t_d is:

$$\hat{\boldsymbol{q}}_j^d = \boldsymbol{R}\hat{\boldsymbol{q}} + \boldsymbol{t} \tag{5.8}$$

So the distribution of $\hat{\boldsymbol{q}}_j^d$ could be calculated:

$$\hat{\boldsymbol{q}}_j^d \sim \mathcal{N}(\boldsymbol{R}\hat{\boldsymbol{q}} + \boldsymbol{t}, \mathrm{Cov}(\hat{\boldsymbol{q}}_j^d) = \boldsymbol{R}\mathrm{Cov}(\hat{\boldsymbol{q}})\boldsymbol{R}^T) \tag{5.9}$$

Then we will calculate the projection of such Gaussian uncertainty to the desired image. The projection model is reused:

$$\hat{\boldsymbol{s}}_j^d = \frac{1}{\hat{q}_j^{d3}}\boldsymbol{K}_{[1:2,:]}\left(\boldsymbol{R}\hat{\boldsymbol{q}} + \boldsymbol{t}\right) \triangleq \frac{1}{\hat{q}_j^{d3}}\boldsymbol{K}_p\left(\boldsymbol{R}\hat{\boldsymbol{q}} + \boldsymbol{t}\right) \tag{5.10}$$

Finally,

$$\hat{\boldsymbol{s}}_j^d \sim \mathcal{N}\left(\frac{1}{\hat{q}_j^{d3}}K_p\left(\boldsymbol{R}\hat{\boldsymbol{q}} + \boldsymbol{t}\right), \left(\frac{1}{\hat{q}_j^{d3}}\right)^2 \boldsymbol{K}_p\mathrm{Cov}(\hat{\boldsymbol{q}}_j^d)\boldsymbol{K}_p^T\right) \tag{5.11}$$

5.1.3 Uncertainty Model Verification

Matlab SLAM Toolbox [77] is used to verify previous mathematics derivation of uncertainty model with Monte Carlo method. There are two robots located on the curve trajectory with zero control input and landmarks for on-board cameras are randomly generated on the map (see Figure 5.1). We name the two robots as *Rob1* and *Rob2*, and the cameras on the two robots as *Cam1* and *Cam2*. Similar to the simulation in §4.3, we also add zero mean Gaussian noise with standard deviation $\sigma_I = 4$ to *Cam1*. However, the Gaussian noise is directly added to feature coordinate positions, not the intensities of each pixel for more persuasive evidence to support our model. Define the images captured by the two

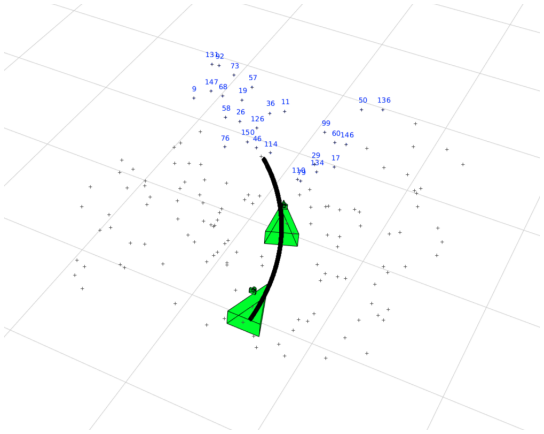

Figure 5.1: Robots and Environment in Matlab Simulation: *Rob1* locates at the beginning of the trajectory and *Rob2* locates at the middle of the trajectory. Black '+' are landmarks and those with blue numbers are the landmarks which have been seen by robots.

cameras *Cam1* and *Cam2* as:

$$F_i = \{f_i^j | j = 1, 2, ..., R\}, i = 1, 2 \tag{5.12}$$

where f_i^j is the i^{th} camera image in j^{th} independent random experiment. R is the total number of experiments and $i = 1, 2$ indicates the camera index. Also define the features in frame f_i^j as:

$$f_i^j = \{x_i^{j,k} = (u_i^{j,k}, v_i^{j,k}) \in \mathbb{R}^2\}_{k=1}^N \tag{5.13}$$

Where N is the number of commonly visible features in *Cam1* and *Cam2*. Define the operator $\mathbf{\Pi}$ as the transform from the feature with the same ID in *Cam1* to *Cam2*, i.e.

$$f_2^j = \mathbf{\Pi} \circ f_1^j \tag{5.14}$$

We calculate F_2 from all $R = 388$ repeated runs of independent random experiment and compare it with the uncertainty model from Equation 5.11.

Figure 5.2 shows the uncertainty ellipses generated by the uncertainty model. The 3

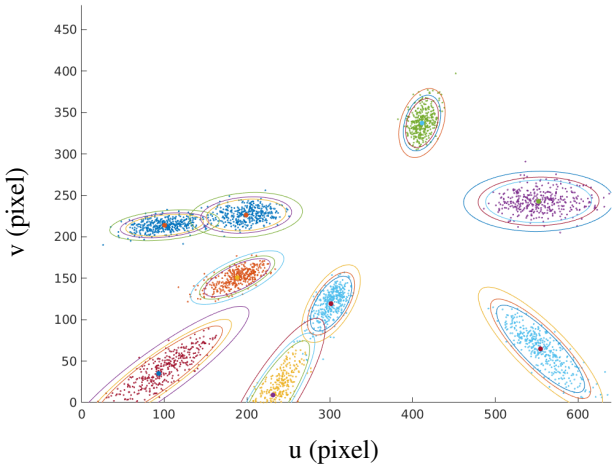

Figure 5.2: Desired Image in Random Experiments

ellipses around each set of points represent 90%, 95% and 99% confidence interval. The distribution of each group of points obeys the uncertainty model. Therefore, it validates our uncertainty derivations.

5.2 Generalized least square in controller design

After the uncertainty modeling of the desired and current features, we have more information about the features than the traditional visual servoing controllers. In the baseline controller Equation 4.8, we find the angular velocities that minimize the error on the image through the pseudo inverse of partial image Jacobian. The reason we include linear velocity in the feedforward loop rather than feedback loop is that we have finely discretized the feature trajectory by way-points. The difference between the current and desired feature coordinates (especially the low depth features contributing most to translation observation) are small and nearly not observable in the sense of translation estimation. The smallest singular value of the full image Jacobian is sufficiently small for the observation noise to dominate the reconstruction of linear and angular velocities in the least square problem.

To improve the basic trajectory servoing controller, we turn it into an optimization problem to incorporate uncertainty information into the algorithm. The cost function of the

controller has the form of generalized least square which is weighted by the inverse covariance of features. In this way, we give more weight to the features with less uncertainty in the error terms in least square. So we turn the original trajectory servoing controller (see Equation 4.8)

$$\omega = \left(\boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}}, \boldsymbol{s})\right)^{\dagger} \boldsymbol{P} \quad \text{where}$$
$$\boldsymbol{P} = \boldsymbol{L}^1(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\nu^* - \boldsymbol{L}^1(\boldsymbol{q}^{\mathcal{C}}, \boldsymbol{s})\nu + \boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}^*}(t), \boldsymbol{s}^*(t))\omega^* - \lambda \boldsymbol{e} \tag{5.15}$$

to an equivalent least square problem:

$$\hat{\omega} = \arg\min_{\omega} \| \underbrace{\overbrace{\boldsymbol{L}^2(\boldsymbol{q}^{\mathcal{C}}, \boldsymbol{s})\omega}^{\mathbf{C}_1(\omega)} - \boldsymbol{P}}_{\mathbf{C}(\omega)} \|_2^2 \tag{5.16}$$

The redefinition of the calculation of ω as an optimization problem helps us understand the actual meaning of ω from the overdetermined system: a minimizer of the difference between feature motion on the image plane and that required by the control law. Simultaneously, since we have the distribution of desired features and observed features, it is reasonable to apply higher weight to the error terms associated with low variance features when calculating cost and vice versa.

Calculate the covariance of $\mathbf{C}(\omega)$:

$$\begin{aligned} \boldsymbol{\Omega} \triangleq \mathrm{Cov}\left(\mathbf{C}(\omega)\right) &= \lambda^2 \mathrm{Cov}(\hat{\boldsymbol{s}} - \hat{\boldsymbol{s}}^*) \\ &= \lambda^2 \mathrm{Cov}\left([\boldsymbol{I}, -\boldsymbol{I}][\hat{\boldsymbol{s}}, \hat{\boldsymbol{s}}^*]^T\right) \\ &= \lambda^2 (\mathrm{Cov}(\hat{\boldsymbol{s}}) + \mathrm{Cov}(\hat{\boldsymbol{s}}^*)) \end{aligned} \tag{5.17}$$

Generalized least square means that we optimize the following objective function:

$$\hat{\omega} = \arg\min_{\omega} \mathbf{C}(\omega)^T \left(\mathrm{Cov}\left(\mathbf{C}(\omega)\right)\right)^{-1} \left(\mathbf{C}(\omega)\right) \tag{5.18}$$

The difference between cost function Equation 5.18 and the ordinary least square one Equation 5.16 is that the contribution of error terms are weighted by the inverse of the covariance matrix. Since low variance means low observation error, the selection of ω should primarily decrease cost terms associated with these accurate features.

The analytical solution of generalized least square optimization has the form:

$$\hat{\omega} = \left(\mathbf{C}_1(\omega)^T \mathbf{\Omega}^{-1} \mathbf{C}_1(\omega)\right)^{-1} \mathbf{C}_1(\omega)^T \mathbf{\Omega}^{-1} \boldsymbol{P} \tag{5.19}$$

CHAPTER 6
EXPERIMENTAL RESULTS

This chapter focus on the evidence that supports the conclusions summarized from previous chapters: 1. trajectory servoing requires SLAM to obtain stably tracked features and it is more accurate than SLAM pose based navigation. 2. trajectory servoing is more robust to image noise compared with SLAM pose based navigation. 3. uncertainty based trajectory servoing (U-TS) improves the performance of baseline trajectory servoing (TS) using noisy images. §6.1 and §6.2 shows short and long trajectories benchmark results without image noise and compares the results with respect to tracking error, terminal error, control effort and pose estimation error. Its conclusion supports claims 1. With similar environment and task settings, §6.3 adds noise to image and use a diversity of statistics to prove claims 2 and 3. Finally, §6.4 defines a real navigation task completed by trajectory servoing.

6.1 Short Trajectory Benchmark

This section runs several short distance trajectory servoing experiments to evaluate the accuracy of the image based feedback strategy supplemented by stereo V-SLAM. The hypothesis is that mapping trajectory tracking to image space will improve short term trajectory tracking by mitigating the impact of SLAM estimation drift from the feedback loop.

6.1.1 Experimental Setup

Some basic configurations have been stated in §3.1.2. Turtlebot is tasked to follow a specific short-distance trajectory. A total of five paths were designed, loosely based on Dubins paths. They are denoted as straight (S), weak turn (WT), straight+turn (ST), turn+straight (TS), turn+turn (TT), and are depicted in Fig. 6.1. Average length of all trajectories are $\sim$4m. Longer paths would consist of multiple short segment reflecting variations on this

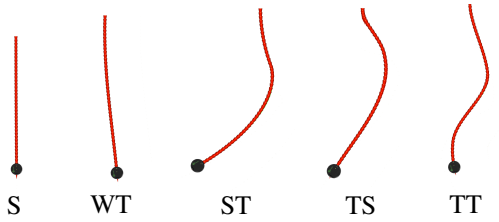

Figure 6.1: Short-distance template trajectories

path set. They are designed to ensure that sufficient feature points visible in the first frame remain visible along the entirety of the path. Five trials per trajectory are run. The desired and actual robot poses are recorded for performance scoring. Performance metrics computed are average translation error (ATE).

6.1.2 Navigation Methods Tested

In addition to the trajectory servoing algorithm, several baseline methods are implemented. The first is a pose feedback strategy using *perfect odometry* (PO) as obtained from the actual robot pose in the Gazebo simulator. The second replaced PO with the SLAM estimated pose, which are delayed and have uncertainty. The third is a trajectory servoing method without the V-SLAM system, which would be a naive implementation of visual servoing based on (4.8). It is called VS+ to differentiate from trajectory servoing, and uses a frame-by-frame stereo feature tracking system [78]. Pose-based control [1] uses a geometric trajectory tracking controller with feedforward $[\nu^*, \omega^*]^T$ and feedback

$$\begin{aligned} \nu_{cmd} &= k_x * \widetilde{x} + \nu^* \\ w_{cmd} &= k_\theta * \widetilde{\theta} + k_y * \widetilde{y} + \omega^* \end{aligned} \tag{6.1}$$

where,

$$[\widetilde{x}, \widetilde{y}, \widetilde{\theta}]^T \simeq \widetilde{g} = g^{-1} g^* = \left(g_{\mathcal{R}}^{\mathcal{W}}\right)^{-1}(t) \left(g_{\mathcal{R}}^{\mathcal{W},*}\right)(t). \tag{6.2}$$

The controller gains have been empirically tuned to give good performance for the perfect odometry case, and have been extensively used in prior work [1, 4, 5].

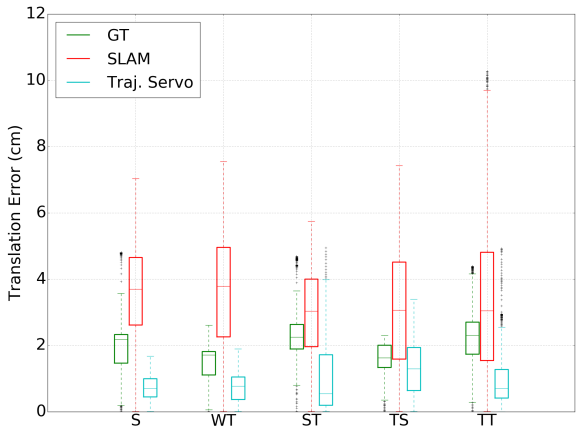

Figure 6.2: Short-distance trajectory benchmarking results

Table 6.1: Trajectory Tracking ATE (cm)

Seq.	PO	SLAM	TS	VS+
S	2.27	3.67	0.76	x
WT	1.62	3.62	0.76	x
ST	2.50	2.96	1.06	x
TS	1.58	3.24	1.33	x
TT	2.35	3.50	1.04	x
Avg. ATE	2.06	3.40	***0.99***	x

Table 6.2: Terminal Error (cm)

Seq.	PO	SLAM	TS
S	2.47	5.44	1.29
WT	1.83	5.93	1.30
ST	2.85	1.50	2.99
TS	1.85	1.76	2.04
TT	2.34	3.07	2.27
Avg.	2.27	3.54	***1.98***

Table 6.3: Control Effort

Seq.	PO		SLAM		TS	
	ν	ω	ν	ω	ν	ω
S	7.88	0.20	7.10	0.19	5.86	1.17
WT	8.14	0.55	7.35	0.48	6.16	1.84
ST	7.58	4.73	6.70	3.47	5.67	3.38
TS	8.18	4.92	7.33	4.27	6.19	4.46
TT	8.47	4.82	7.72	4.24	6.46	4.75
Avg.	8.05	3.04	7.24	**2.53**	***6.07***	*3.12*
	11.09		9.77		***9.19***	

Table 6.4: Estimation ATE (cm)

Seq.	SLAM	TS
S	1.16	1.41
WT	1.63	1.66
ST	2.48	3.00
TS	4.22	3.91
TT	3.75	3.36
Avg.	3.02	***2.67***

6.1.3 Results and Analysis

Tables 6.1 to 6.3 quantify the outcomes of all methods tested. Figure 6.2 consists of box-plots of the trajectory tracking error for the different template trajectories and methods (minus VS+). The first outcome to note is that VS+ fails for all paths. The average length of successful servoing is 0.4m ($\sim$10% of the path length). Inconsistent data association rapidly degrades the feature pool and prevents consistent use of features for servoing feedback. Without the feature map in V-SLAM, a reappeared feature will be treated as a new feature and assigned with a different index, which easily violates the *correspondence* rule from §4.1.2. As noted there, any effort to improve this would increasingly approach the computations found in a V-SLAM method. Maintaining stable feature tracking through V-SLAM is critical to trajectory servoing.

Comparing SLAM and TS, the Table values show lower errors across the board, and lower control cost. Trajectory tracking error is lower by 71% and terminal error is lower by 44%. Interestingly, the pose estimation error is only reduced by around 12%, which

indicates weak improvement in pose estimation from the better tracking via TS. This is most likely a result of the SLAM pose predictions being decoupled from the control system, leading to uninformed motion priors. ORB-SLAM uses a motion prior based on its own internal estimates. Tightly coupling the two parts may improve the SLAM system, but may also introduce new sensitivities. The TS approach reduces control effort by 19%.

Comparing PO with TS shows better performance by TS, which was not expected. This might suggest some additional tuning would be necessary. However, usually there is trade-off between tracking error and control cost. Attempts to improve tracking usually increase the control cost, and vice-versa. Based on the values of the Tables, it is not clear that better would be possible. Tests later in this manuscript will show that the benefit does not persist for longer paths, thus the improvement for short trajectories is of limited use. The finding here is that implementing a purely image-based approach to trajectory tracking through unknown environments is possible, and can work well over short segments in the absence of global positioning information.

6.2 Long Trajectory Benchmark

This section modifies the experiments in §6.1 to involve longer trajectories that will trigger feature replenishment and synthesize new feature trajectory segments. The set of trajectories to track is depicted in Figure 6.3. They are denoted as right u-turn (RU), left u-turn (LU), straight+turn (ST), and zig-zag (ZZ). Each trajectory is longer than 20m. Testing and evaluation follows as before (minus VS+).

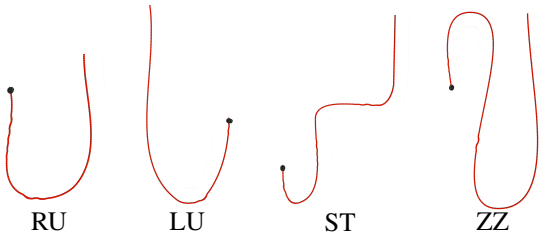

Figure 6.3: Long-distance trajectories

6.2.1 Results and Analysis

Figure 6.4 consists of boxplots of the translation-only trajectory tracking ATE for the template trajectories. As hypothesized, the error over longer trajectories is affected by the need to use SLAM pose estimates for regeneration. The PO method outperforms TS, but TS still operates better than SLAM. However, it appears that in the one case where the SLAM system performed the worst, the TS approach had better performance. The tabulated results in Table 6.5 and Table 6.6 indicates that on average the TS approach improved over SLAM by 39% and 21% in terms of tracking error and terminal error. What is interesting is that the estimation ATE of both systems is comparable (Table 6.8), which indicates that trajectory servoing may have better closed-loop noise rejection properties by working directly in image-space instead of using inferred pose estimates from the SLAM estimator. Table 6.7 suggests that control efforts of both ν and ω are the lowest. The combined control effort of TS is reduced 19% and 10% from PO and SLAM. That TS may change the performance vs. cost trade-off curves should be studied further.

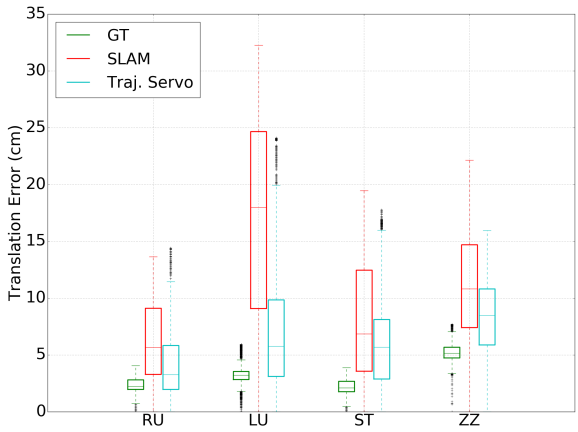

Figure 6.4: Long trajectory benchmarking results

Table 6.5: Tracking ATE (cm)

Seq.	PO	SLAM	TS
RU	2.40	6.19	4.02
LU	3.47	17.22	7.30
ST	2.25	7.95	6.06
ZZ	5.19	11.01	8.30
Avg. ATE	**3.33**	10.59	*6.42*

Table 6.6: Terminal Error (cm)

Seq.	PO	SLAM	TS
RU	2.33	11.61	6.77
LU	3.14	26.06	18.90
ST	2.38	8.61	7.74
ZZ	4.28	5.22	7.18
Avg.	**3.03**	12.88	*10.15*

Table 6.7: Control Effort

Seq.	PO		SLAM		TS	
	ν	ω	ν	ω	ν	ω
RU	15.51	14.95	13.94	12.08	11.94	10.80
LU	17.77	9.53	16.38	9.24	13.67	9.94
ST	17.80	13.87	16.00	12.96	13.74	12.29
ZZ	23.17	11.73	21.00	10.67	17.97	10.89
Avg.	18.56	12.52	16.83	11.24	***14.33***	***10.98***
	31.08		28.07		***25.31***	

Table 6.8: Estimation ATE (cm)

Seq.	SLAM	TS
RU	9.15	8.49
LU	13.68	11.09
ST	7.00	7.96
ZZ	12.13	10.37
Avg.	10.49	***9.48***

6.3 Navigation with Image Noise

6.3.1 Experimental Setups

In order to show the robustness of trajectory servoing to image noise and improvements of uncertainty-based trajectory servoing (U-TS), we have benchmark experiments in the noise standard deviation (STD) from 0.05 to 0.07 with 0.005 as the step (see §3.1.2).

Figure 3.3 shows two robot start poses to simulate different sets of the environment, which stabilizes and generalizes the benchmark results. The same 5 short trajectories as in §6.1 and the first 3 long trajectories in §6.2, RU, LU and ST are used respectively for the two starting points. In order to have representative and persuasive results, 10 runs for short trajectories and 5 runs for long trajectories are repeated for each trajectory.

6.3.2 Evaluation Metrics

There are five different metrics to evaluate the tracking performance. *Average Tracking Error* (ATE) shows the averaged performance along the entire trajectory. *Max Tracking Error* (MTE) represents the extreme error in each tracking task. From the generation of boxplot, outliers of tracking errors are extracted. Therefore, MTE is computed with/without these outliers. Moreover, in boxplots, the value of the upper quartile means the range of major data samples. *Upper Quartile Tracking Error* (UQTE) is derived from this. The last is the *Control Effort* computed as the norm of all control signals, i.e. linear and angular velocities, to finish each tracking task. To be noted, each metric is computed from the combination of varying noise standard deviations, robot start poses and multiple repeated runs. From the comparisons of these metrics, it is reasonable to statistically evaluate different tracking methods: V-SLAM pose based controller (SLAM), baseline trajectory servoing (TS) and uncertainty-based trajectory servoing (U-TS).

6.3.3 SLAM versus TS

The first set of benchmark aims to compare the tracking performance between the SLAM pose based controller and baseline trajectory servoing with image noise. The boxplots of short and long distance trajectory benchmark are shown in Figure 6.5 (a)(b). It could be observed that the baseline trajectory servoing has better tracking performance in both short and long trajectory benchmarks. The SLAM and TS columns in Table 6.9-Table 6.14 show quantitative statistic results.

First, in Table 6.9 and Table 6.10, by adding noise, ATEs of SLAM increase 130% for short trajectories and 252% for long trajectories. However, ATEs of baseline trajectory servoing only increase 34% for short trajectories and 80% for long trajectories. Trajectory servoing has lower ATE increase with image noise. Next looking at the results of SLAM and TS with noise, ATE of trajectory servoing decreases 83% for short trajectories and 72% for long trajectories. In Table 6.11 and 6.12, other tracking error statistics also support the huge improvement from SLAM to TS by: 83% and 62% for MTE with Outliers, 83% and 66% for MTE without Outliers, 84% and 69% for UQTE in short and long trajectories, which shows its consistency among these tracking error metrics.

From Table 6.13 and 6.14, the control effort of linear velocity v decreases nearly 19% in both short and long trajectory benchmarks. The reason is that trajectory servoing is constrained to work under the planned linear velocity, however, SLAM pose-based controller may have over-actuated velocities. The control effort of angular velocity ω for TS increases only 2.8% in short distance benchmark and only 4% in long distance benchmark, which means both are comparable. The improvement of control effort is more observable in long trajectory benchmark since it is easier for a long-term control process to amplify its effort.

These benchmark results demonstrate that trajectory servoing system is more robust to image noise and can achieve much better tracking performance when involving with uncertainty. The improvement is more obvious in the short trajectory benchmark, because

inaccurate SLAM estimated poses used in the feature trajectory regeneration may undermine the long term performance of trajectory servoing.

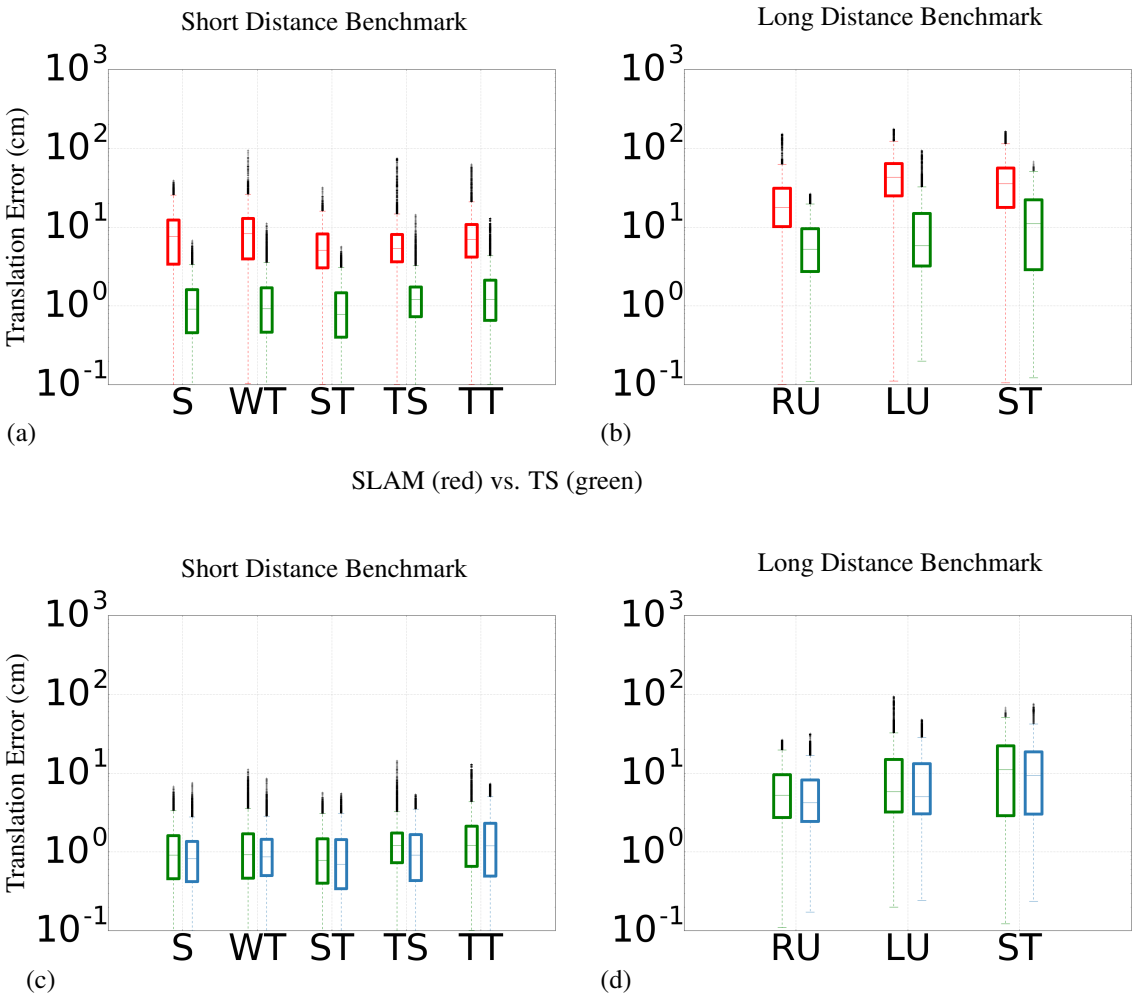

Figure 6.5: Box plots of benchmarks. (a) Short distance benchmark between VSLAM pose-based controller (red) and baseline trajectory servoing (green); (b) Long distance benchmark between VSLAM pose-based controller (red) and baseline trajectory servoing (green). (c) Short distance benchmark between baseline (green) and uncertainty-based (blue) trajectory servoing; (d) Long distance benchmark between baseline (green) and uncertainty-based (blue) trajectory servoing.

Table 6.9: Short Distance: Trajectory Tracking ATE (cm)

Seq.	SLAM		TS		U-TS
	¬noise	noise	¬noise	noise	noise
S	3.67	8.55	0.76	1.13	0.98
WT	3.62	9.63	0.76	1.31	1.08
ST	2.96	5.89	1.06	1.02	0.93
TS	3.24	7.09	1.33	1.42	1.15
TT	3.50	8.18	1.04	1.74	1.49
Avg. ATE	3.40	7.81	0.99	1.32	1.13

Table 6.10: Long Distance: Trajectory Tracking ATE (cm)

Seq.	SLAM		TS		U-TS
	¬noise	noise	¬noise	noise	noise
RU	6.19	23.63	4.02	7.10	6.19
LU	17.22	45.68	7.30	10.42	9.21
ST	7.95	41.19	6.06	13.75	11.82
Avg. ATE	10.45	36.8	5.79	10.42	9.07

Table 6.11: Short Distance: Other Tracking Errors from Boxplots (cm)

Seq.	MTE w/ Outliers			MTE w/o Outliers			Upper Quartile TE		
	SLAM	TS	U-TS	SLAM	TS	U-TS	SLAM	TS	U-TS
S	39.14	6.80	7.51	25.55	3.32	2.74	12.14	1.56	1.30
WT	94.56	11.26	8.56	26.24	3.49	2.85	12.74	1.64	1.40
ST	31.80	5.61	5.54	15.97	3.03	2.95	8.12	1.42	1.33
TS	75.16	14.30	5.38	14.62	3.24	3.45	7.97	1.72	1.61
TT	63.26	12.95	7.38	20.75	4.31	4.89	10.73	2.10	2.21
Avg.	60.78	10.18	**6.87**	20.63	3.48	**3.38**	10.34	1.69	**1.57**

Table 6.12: Long Distance: Other Tracking Errors from Boxplots (cm)

Seq.	MTE w/ Outliers			MTE w/o Outliers			Upper Quartile TE		
	SLAM	TS	U-TS	SLAM	TS	U-TS	SLAM	TS	U-TS
RU	152.46	26.53	31.69	62.50	19.66	16.77	31.09	9.50	8.17
LU	176.31	94.13	47.98	123.23	32.39	28.41	64.20	14.88	13.18
ST	164.46	68.27	75.65	114.31	50.97	41.98	56.29	22.24	18.64
Avg.	164.41	62.98	**51.77**	100.01	34.34	**29.05**	50.53	15.54	**13.33**

Table 6.13: Short Distance: Control Effort

Seq.	SLAM		TS		U-TS	
	v	ω	v	ω	v	ω
S	7.43	0.28	5.97	1.31	5.98	0.98
WT	7.73	0.54	6.27	2.01	6.26	1.62
ST	7.05	3.59	5.62	3.65	5.63	3.62
TS	7.73	4.38	6.32	4.60	6.31	4.50
TT	8.00	4.28	6.78	4.91	6.78	4.73
Avg.	7.59	3.23	6.19	3.32	6.19	3.11
	10.82		9.51		**9.30**	

Table 6.14: Long Distance: Control Effort

Seq.	SLAM		TS		U-TS	
	v	ω	v	ω	v	ω
RU	14.71	11.80	11.97	11.27	11.96	11.39
LU	16.85	9.10	13.67	9.60	13.67	9.97
ST	16.97	12.80	13.65	14.15	13.64	13.89
Avg.	16.18	11.47	13.10	11.67	13.09	11.75
	27.65		24.77		**24.84**	

6.3.4 TS versus U-TS

These experiments aim to evaluate the performance of the uncertainty based trajectory servoing (U-TS) versus the baseline (TS). The same benchmark configurations are performed. Boxplots are shown in Figure 6.5(c)(d). Statistical results are provided in the U-TS columns in Table 6.9-6.14. It shows uncertainty-based trajectory servoing can further improve the performance.

From Table 6.9-6.12, tracking errors of U-TS decrease for both short and long distance benchmarks: 15% (short) and 13% (long) for ATE, 33% (short) and 18% (long) for MTE with outliers. When excluding the outliers, the improvement of short distance decreases a little: 3% (short) and 15% (long) for MTE without outliers, 7% (short) and 14% (long) for UQTE. The statistics are more consistent for long trajectories, since short trajectory percentages are easier to be perturbed by small disturbance of original values.

In Table 6.13 and 6.14, the control effort of linear velocity is not changed since the same sequence of linear velocities are applied. For short distance benchmark, the control effort of angular velocity of U-TS decreases 6% from TS. Therefore, U-TS can reduce the overall control effort compared with TS in short trajectories. Howevce in the long trajectory benchmark, U-TS control effort has a negligible increase (0.66%) that may come from a more intense angular correction. In conclusion, uncertainty based trajectory servoing (U-TS) can achieve better tracking with higher accuracy with a similar control effort.

6.4 Navigation with online path planning

To show how trajectory servoing works for a navigation task, we move the environment down to the ground and create a 2D occupancy map. Given a goal, the global planner identifies a feasible collision-free path from the map. The online global path is used for the robot to apply trajectory servoing. The full system in Figure 1.1 is used. Figure 6.6 shows one successful navigation task for three tracking methods. The robot starts from the

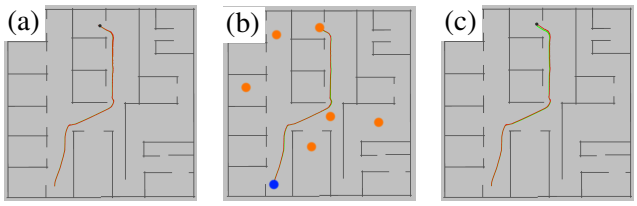

Figure 6.6: Navigation with global planning. Blue point is the robot starting position. Orange points are 6 goal points for navigation. In each figure, green is the collision-free global path. Red is the real robot trajectory overlaying on the green trajectory. The figures show successful navigation examples of the same goal point with three different controllers. (a) Pose-based trajectory tracking with perfect odometry. (b) Vision-based trajectory servoing. (c) Pose-based trajectory tracking with estimated poses from V-SLAM. It can be observed that the red trajectory has less deviations from the green trajectory with TS (b) than SLAM (c).

blue point, and ends at 6 different positions, marked as orange points. The average ATE for PO, SLAM and TS are 1.31cm, 7.39cm and 7.30cm. And the average terminal errors are 1.46cm, 14.49cm and 13.13cm. Trajectory servoing, enabled by V-SLAM, has better performance than using V-SLAM for pose-based feedback. The smaller performance gap between TS and SLAM for these navigation tasks requires further investigation. Per Tables 6.4 and 6.8, it appears to be dependent on SLAM estimation error, as the gap is close to the estimation gap ($\sim$10%). There may be a mechanism to better estimate motion from the initial pose to the terminal pose of a trajectory segment, which might involve more tightly coupling the short distance module with the SLAM module (see Fig. 1.1).

CHAPTER 7
CONCLUSION & FUTURE WORK

7.1 Conclusion

In the thesis, we presented a vision-based navigation approach for a non-holonomic mobile ground robot called *trajectory servoing*, which combines the IBVS and V-SLAM. This trajectory tracking method can successfully follow a given short trajectory without externally derived robot pose information. By integrating with estimated robot poses from V-SLAM, it can achieve long-term navigation. The V-SLAM system also provides robust feature tracking to support stable trajectory servoing. Experiments demonstrated improved tracking accuracy over pose-based trajectory tracking using estimated SLAM poses.

Similar to related works, we also investigated image noise effects of trajectory servoing based visual navigation. The close loop benchmark shows that trajectory servoing is more robust to the image noise than traditional SLAM pose based controller. In §4.3 we find that the important reason is that trajectory servoing only tracks low uncertainty features consistently. Besides, an improved controller is designed based on weighted least square that balance features contribution to control result ω by their covariance. Experiment results also show the further improvement of tracking performance although it is not that significant. Overall, trajectory servoing is a more preferable trajectory tracking method compared with SLAM pose based one and directly closing loop in the perception space for trajectory tracking is a reasonable solution to have better performance.

7.2 Future Work

In the future work, we will apply this uncertainty-based trajectory servoing on a real Turtlebot platform to test its performance. In addition, we may extract the most informative and

observable features to control linear velocity. Hopefully, we could also investigate a more integrated and tighter coupling of SLAM and visual servoing besides tracking and positioning for a better coordination between SLAM and trajectory tracking.

REFERENCES

[1] Y. Zhao, J. S. Smith, S. H. Karumanchi, and P. A. Vela, "Closed-loop benchmarking of stereo visual-inertial SLAM systems: Understanding the impact of drift and latency on tracking accuracy," *ICRA*, pp. 1105–1112, 2020.

[2] H. Liu, R. Jiang, W. Hu, and S. Wang, "Navigational drift analysis for visual odometry," *Comput. Informatics*, vol. 33, pp. 685–706, 2014.

[3] J. B. Johnson, "Thermal agitation of electricity in conductors," *Phys. Rev.*, vol. 32, pp. 97–109, 1 Jul. 1928.

[4] J. S. Smith and P. A.Vela, "PiPS: Planning in perception space," in *ICRA*, May 2017, pp. 6204–6209.

[5] J. S. Smith, S. Feng, F. Lyu, and P. A. Vela, "Real-time egocentric navigation using 3d sensing," in *Machine Vision and Navigation*, O. Sergiyenko, W. Flores-Fuentes, and P. Mercorelli, Eds. Cham: Springer International Publishing, 2020, pp. 431–484, ISBN: 978-3-030-22587-2.

[6] Y. Zhao and P. A. Vela, "Good feature matching: Toward accurate, robust VO/VSLAM with low latency," *T-RO*, vol. 36, no. 3, pp. 657–675, 2020.

[7] S. Šegvić, A. Remazeilles, A. Diosi, and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation," *CVIU*, vol. 113, no. 2, pp. 172–187, 2009.

[8] T. Krajník, F. Majer, L. Halodová, and T. Vintr, "Navigation without localisation: Reliable teach and repeat based on the convergence theorem," in *IROS*, 2018, pp. 1657–1664.

[9] L. Halodovà, E. Dvořáková, F. Majer, T. Vintr, O. M. Mozos, F. Dayoub, and T. Krajník, "Predictive and adaptive maps for long-term visual navigation in changing environments," in *IROS*, 2019, pp. 7033–7039.

[10] T. Do, L. C. Carrillo-Arce, and S. I. Roumeliotis, "High-speed autonomous quadrotor navigation through visual and inertial paths," *IJRR*, vol. 38, no. 4, pp. 486–504, 2019.

[11] P. Furgale and T. Barfoot, "Visual teach and repeat for long-range rover autonomy," *J. Field Robot.*, vol. 27, pp. 534–560, Sep. 2010.

[12] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, "Image features for visual teach-and-repeat navigation in changing environments," *Robotics and Autonomous Systems*, vol. 88, pp. 127–141, 2017.

[13] N. Zhang, M. Warren, and T. D. Barfoot, "Learning place-and-time-dependent binary descriptors for long-term visual localization," in *ICRA*, 2018, pp. 828–835.

[14] T. Nguyen, G. K. Mann, R. G. Gosine, and A. Vardy, "Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle," *J. Intell. Robotics Syst.*, vol. 84, no. 1-4, pp. 217–240, 2016.

[15] K. Kidono, J. Miura, and Y. Shirai, "Autonomous visual navigation of a mobile robot using a human-guided experience," *Robotics and Autonomous Systems*, vol. 40, no. 2, pp. 121–130, 2002.

[16] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot, "A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera," in *Canadian Conference on Computer and Robot Vision*, 2014, pp. 238–245.

[17] A. Vardy, "Using feature scale change for robot localization along a route," in *IROS*, 2010, pp. 4830–4835.

[18] N. M. Garcia and E. Mails, "Preserving the continuity of visual servoing despite changing image features," in *IROS*, vol. 2, 2004, 1383–1388 vol.2.

[19] F. Chaumette and S. Hutchinson, "Visual servo control. I. basic approaches," *RAM*, vol. 13, no. 4, pp. 82–90, 2006.

[20] ——, "Visual servo control. II. advanced approaches," *RAM*, vol. 14, no. 1, pp. 109–118, 2007.

[21] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 534–549, 2002.

[22] F. Fahimi and K. Thakur, "An alternative closed-loop vision-based control approach for unmanned aircraft systems with application to a quadrotor," in *ICUAS*, 2013, pp. 353–358.

[23] A. Cherubini, F. Chaumette, and G. Oriolo, "Visual servoing for path reaching with nonholonomic robots," *Robotica*, vol. 29, no. 7, pp. 1037–1048, 2011.

[24] A. Ahmadi, L. Nardi, N. Chebrolu, and C. Stachniss, "Visual servoing-based navigation for monitoring row-crop fields," in *ICRA*, May 2020, pp. 4920–4926.

[25] A. Remazeilles, F. Chaumette, and P. Gros, "3d navigation based on a visual memory," in *ICRA*, 2006, pp. 2719–2725.

[26] A. Diosi, S. Segvic, A. Remazeilles, and F. Chaumette, "Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing," *T-ITS*, vol. 12, no. 3, pp. 870–883, 2011.

[27] G. Blanc, Y. Mezouar, and P. Martinet, "Indoor navigation of a wheeled mobile robot along visual routes," in *ICRA*, 2005, pp. 3354–3359.

[28] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *T-RO*, vol. 32, no. 6, pp. 1309–1332, 2016.

[29] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. J. Kelly, A. J. Davison, M. Luján, M. F. P. O'Boyle, G. Riley, N. Topham, and S. Furber, "Introducing slambench, a performance and accuracy benchmarking methodology for slam," in *ICRA*, 2015, pp. 5783–5790.

[30] S. Saeedi, B. Bodin, H. Wagstaff, A. Nisbet, L. Nardi, J. Mawer, N. Melot, O. Palomar, E. Vespa, T. Spink, C. Gorgovan, A. Webb, J. Clarkson, E. Tomusk, T. Debrunner, K. Kaszyk, P. Gonzalez-De-Aledo, A. Rodchenko, G. Riley, C. Kotselidis, B. Franke, M. F. P. O'Boyle, A. J. Davison, P. H. J. Kelly, M. Luján, and S. Furber, "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality," *Proceedings of the IEEE*, vol. 106, no. 11, pp. 2020–2039, 2018.

[31] M. Bujanca, P. Gafton, S. Saeedi, A. Nisbet, B. Bodin, M. F. P. O'Boyle, A. J. Davison, P. H. J. Kelly, G. Riley, B. Lennox, M. Luján, and S. Furber, "Slambench 3.0: Systematic automated reproducible evaluation of slam systems for robot vision challenges and scene understanding," in *ICRA*, 2019, pp. 6351–6358.

[32] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *ICRA*, 2018, pp. 2502–2509.

[33] I. Cvišić, J. Ćesić, I. Marković, and I. Petrović, "Soft-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles," *J. Field Robot.*, vol. 35, no. 4, pp. 578–595, 2018.

[34] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, "Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors," *RA-L*, vol. 3, no. 3, pp. 1801–1807, 2018.

[35] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *J. Field Robot.*, vol. 35, no. 1, pp. 23–51, 2018.

[36] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *RAM*, vol. 18, no. 4, pp. 80–92, 2011.

[37] F. Fraundorfer and D. Scaramuzza, "Visual odometry : Part ii: Matching, robustness, optimization, and applications," *RAM*, vol. 19, no. 2, pp. 78–90, 2012.

[38] L. Cao, J. Ling, and X. Xiao, "Study on the influence of image noise on monocular feature-based visual slam based on ffdnet," *Sensors*, vol. 20, no. 17, 2020.

[39] Jianbo Shi and Tomasi, "Good features to track," in *CVPR*, 1994, pp. 593–600.

[40] S. Šegvić, A. Remazeilles, and F. Chaumette, "Enhancing the point feature tracker by adaptive modelling of the feature support," in *ECCV*, A. Leonardis, H. Bischof, and A. Pinz, Eds., 2006, pp. 112–124, ISBN: 978-3-540-33835-2.

[41] H. Liu, R. Jiang, W. Hu, and S. Wang, "Navigational drift analysis for visual odometry," *Comput. Informatics*, vol. 33, pp. 685–706, 2014.

[42] M. Tassano, J. Delon, and T. Veit, "An Analysis and Implementation of the FFDNet Image Denoising Method," *IPOL*, vol. 9, pp. 1–25, 2019.

[43] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *T-RO*, vol. 31, no. 5, pp. 1147–1163, 2015.

[44] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *T-RO*, vol. 33, no. 5, pp. 1255–1262, 2017.

[45] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, 2012, pp. 524–530.

[46] A. Thyagharajan, O. J. Omer, D. Mandal, and S. Subramoney, "Towards noise resilient SLAM," in *ICRA*, 2020, pp. 72–79.

[47] G. Zhang and P. A. Vela, "Optimally observable and minimal cardinality monocular slam," in *ICRA*, 2015, pp. 5211–5218.

[48] ——, "Good features to track for visual slam," in *CVPR*, Jun. 2015.

[49] Y. Zhao and P. A. Vela, "Good feature selection for least squares pose optimization in vo/vslam," in *IROS*, 2018, pp. 1183–1189.

[50] ——, "Good line cutting: Towards accurate pose tracking of line-assisted VO/VSLAM," in *ECCV*, Sep. 2018.

[51] H. Shi, M. Xu, and K. S. Hwang, "A fuzzy adaptive approach to decoupled visual servoing for a wheeled mobile robot," *IEEE Trans Fuzzy Syst*, vol. 28, no. 12, pp. 3229–3243, 2020.

[52] N. Gans, P. Corke, and S. Hutchinson, "Comparison of robustness and performance of partitionned image based visual servo systems," *ACRA*, Nov. 2001.

[53] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-point RANSAC for extended kalman filtering: Application to real-time structure from motion and visual odometry," *J. Field Robot.*, vol. 27, no. 5, pp. 609–631, 2010.

[54] Y. Zhao, J. S. Smith, and P. A. Vela, *Good graph to optimize: Cost-effective, budget-aware bundle adjustment in visual slam*, 2020. arXiv: 2008.10123.

[55] N. M. Garcia and E. Mails, "Preserving the continuity of visual servoing despite changing image features," in *IROS*, vol. 2, 2004, 1383–1388 vol.2.

[56] N. Garcia-Aracil, E. Malis, R. Aracil-Santonja, and C. Perez-Vidal, "Continuous visual servoing despite the changes of visibility in image features," *T-RO*, vol. 21, no. 6, pp. 1214–1220, 2005.

[57] W. van der Mark, J. C. van den Heuvel, and F. C. A. Groen, "Stereo based obstacle detection with uncertainty in rough terrain," in *IEEE Intelligent Vehicles Symposium*, 2007, pp. 1005–1012.

[58] M. Perrollaz, A. Spalanzani, and D. Aubert, "Probabilistic representation of the uncertainty of stereo-vision and application to obstacle detection," in *IEEE Intelligent Vehicles Symposium*, 2010, pp. 313–318.

[59] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth parametrization for monocular SLAM," *T-RO*, vol. 24, no. 5, pp. 932–945, 2008.

[60] D. Belter, M. Nowicki, and P. Skrzypczyński, "Modeling spatial uncertainty of point features in feature-based rgb-d slam," *Machine Vision and Applications*, vol. 29, no. 5, pp. 827–844, Jul. 2018.

[61] N. Pugeault, S. Kalkan, E. Baseski, F. Wörgötter, and N. Krüger, "Reconstruction uncertainty and 3d relations," in *VISAPP*, 2008, ISBN: 978-989-8111-21-0.

[62] J. Chen, Z. Ding, and F. Yuan, "Theoretical uncertainty evaluation of stereo reconstruction," in *ICBBE*, 2008, pp. 2378–2381.

[63] G. Di Leo, C. Liguori, and A. Paolillo, "Propagation of uncertainty through stereo triangulation," in *IEEE Instrumentation Measurement Technology Conference Proceedings*, 2010, pp. 12–17.

[64] ——, "Covariance propagation for the uncertainty estimation in stereo vision," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 5, pp. 1664–1673, 2011.

[65] A. H. Abdul Hafez, S. Achar, and C. V. Jawahar, "Visual servoing based on gaussian mixture models," in *ICRA*, 2008, pp. 3225–3230.

[66] G. Flandin and F. Chaumette, "Visual Data Fusion : Application to Objects Localization and Exploration," INRIA, Research Report RR-4168, 2001.

[67] G. Sibley, L. Matthies, and G. Sukhatme, "Bias reduction and filter convergence for long range stereo," in *Robotics Research*, 2007, pp. 285–294, ISBN: 978-3-540-48113-3.

[68] A. Donate, X. Liu, and E. G. Collins, "Efficient path-based stereo matching with subpixel accuracy," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 183–195, 2011.

[69] V. Kyrki, "Control uncertainty in image-based visual servoing," in *ICRA*, 2009, pp. 1516–1521.

[70] A. Assa and F. Janabi-Sharifi, "Closed-loop uncertainty modeling for visual servoing," in *ICRA*, 2013, pp. 3089–3094.

[71] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[72] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[73] S. M. Weiss, "Vision based navigation for micro helicopters," PhD thesis, ETH Zurich, Zürich, 2012.

[74] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[75] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[76] R. Olfati-Saber, "Near-identity diffeomorphisms and exponential ϵ-tracking and ϵ-stabilization of first-order nonholonomic SE(2) vehicles," in *ACC*, vol. 6, May 2002, 4690–4695 vol.6.

[77] J. Solà, T. Vidal-Calleja, J. Civera, and J. M. Martinez-Monti, "Impact of landmark parametrization on monocular EKF-SLAM with points and lines," 43 pages. Submitted to IJCV., 2010.

[78] S. Feng. (2020). Frame-by-frame stereo feature tracking system.