MONOCULAR POSE AND SHAPE ESTIMATION OF MOVING TARGETS, FOR AUTONOMOUS RENDEZVOUS AND DOCKING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Sean Augenstein
June 2011
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Stephen M. Rock)  Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Claire J. Tomlin)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Per K. Enge)

Approved for the University Committee on Graduate Studies.
Abstract

This thesis describes the design and implementation of an algorithm for tracking a moving (e.g., ‘tumbling’) target. No a priori information about the target is assumed, and only a single camera is used. The motivation is to enable autonomous rendezvous, inspection, and docking by robots in remote environments, such as space and underwater. Tracking refers to the simultaneous estimation of both the target’s 6DOF pose and 3D shape (in the form of a point cloud of recognizable features), a problem of the SLAM (‘Simultaneous Localization and Mapping’) and SFM (‘Structure from Motion’) research fields.

This research extends SLAM/SFM to deal with non-communicative moving targets (rigid bodies) with unknown, arbitrary 6DOF motion and no a priori knowledge of mass properties, dynamics, shape, or appearance. Specifically, a hybrid algorithm for real-time frame-to-frame pose estimation and shape reconstruction is presented. The algorithm combines concepts from two existing approaches to pose tracking, Bayesian estimation methods and nonlinear optimization techniques, to achieve a real-time capable, feasible, smooth estimate of the relative pose between a robotic platform and a moving target. The rationale for a hybrid approach is explained, and an algorithm is presented. A specific implementation using a modified Rao-Blackwellized particle filter is described and tested.

Field demonstrations were performed in conjunction with the Monterey Bay Aquarium Research Institute, using the camera-equipped Remotely Operated Vehicle (ROV) Ventana to observe, reconstruct, and track the pose of an underwater tethered target in Monterey Bay. Results are included which demonstrate the performance and viability of the hybrid approach.
Acknowledgments

My heartfelt thanks go to Professor Steve Rock, for his guidance and mentorship in my time working in the Aerospace Robotics Lab. His calm demeanor and good humor contribute to a fun working environment in the ARL. Along with him, I thank the whole ARL for their comradeship and support: Peter Kimball, Stephen Russell, Kiran Murthy, Debbie Meduna, Dan Sheinfeld, Nicolas Lee, Andrew Smith, Eleanor Crane, Shandor Dektor, Sarah Houts, Roland Burton, Jose Padial, Ozhan Turgut, and Marcus Hammond. Also thanks to my predecessors in the ARL who “showed me the ropes”: Alan Chen, Daniel Chavez, Sungmoon Joo, Jinwhan Kim, Aaron Plotnik, Teresa Gadda, Jack Langelaan, and especially Kristof Richmond. Finally, a big thank you to Professor Robert Cannon, the emeritus head of the ARL, and Godwin Zhang, lab technician, both of whom never faltered in their kind words and support.

Outside of the ARL, Professor Claire Tomlin was a fantastic teacher and full of good advice during my early years at Stanford. My time in graduate school was immeasurably enriched through conversations with both her and her students: Gabe Hoffman, Steve Waslander, Haomiao Huang, and most of all, Michael Vitus, my friend and roommate during my first few years at Stanford, and a fellow fan of robotics research and ridiculous action movies.

Professor Per Enge agreed to serve on my reading committee, and contributed many useful comments (in defense and during the reading) that made this thesis turn out a whole lot better. Similarly, Dr. Kurt Konolige at Willow Garage spent an invaluable afternoon with me discussing my research prior to my defense. Professor Stephen Boyd, in addition to teaching my favorite class (Convex Optimization), did me the honor of serving as my defense chair. I am indebted to all of them.

Aside from faculty and students, I owe a debt of gratitude to the staff that guided me through Stanford’s rules and regulations: Sherann Ellsworth and Dana Parga, who kept my funding in order and tolerated (barely) my creative travel plans, and Jayanthi Subramanian, her predecessor Lynn Kaiser, and Robin Murphy in the Aero/Astro office, and Pat Cook in the SGF office, who were always available to answer questions.

A benefit to working in the ARL is the ability to perform hardware experiments,
and I would be remiss if I didn’t thank all those who lent me assistance. At sea, besides my labmate Kiran (a jolly companion/computer genius, in both good times and nauseous), I also give my heartfelt thanks to all the world-class engineers and staff who work at the Monterey Bay Aquarium Research Institute (MBARI). The ROV Ventana pilots (Knute Brekke, Michael Burczynski, Craig Dawe, D.J. Osborne, and Mark Talkovic), Point Lobos crew (notably, Ian Young, Greg Maudlin, and Mark Aiello), as well as Michael Risi all made it possible for me to demonstrate my algorithms in the underwater environment. Back at Stanford, Alex Forman helped me manufacture the syntactic foam block into an underwater target. For the lab experiments, Andrew Smith and Stephen Russell helped me use the lab robotic gantry and the rotational motion demonstrator. Jose Padial helped me gather data for analysis.

During my time at Stanford, I was lucky to have funding from a Stanford Graduate Fellowship, from MBARI, and from a NASA grant through the Institute for Dexterous Space Robotics (IDSR). I thank all of them for their generous financial support.

My time in grad school would not have been nearly as enjoyable without the company of great friends. First and foremost, my many roommates and their significant others: Stephen Pifko, Matthew Simms and Sarah ‘the Dragon’ Garrett, Jeremy and Kara Harris, Dan Berkenstock and Momo Feeny, Lukas Kuhn and Vanessa Cirannek, Michael Vitus, Ronan Flanagan, Debbie Meduna, Joelle Abra, Monika J, and Gwen Yoshinaga (the last four not technically roommates, but eternally over nonetheless). In addition, San Gunawardana, Michael Trela, Alice Ryan, Kelley and Josh Alwood, Eric Doran, David Shoemaker, Gene Lee, and Sean Kamkar were part of an awesome group of friends I made first year who have been close ever since.

I also have to thank my teammates on AC Durand, the greatest soccer club in Palo Alto: Alex Haas, Peter Kimball, Jordan Drewitt, Sarah Houts, Sara Smoot, Alex Forman, Melisa Orta, Stephen Russell, Roland Burton, Ashley Chandler, Kim Shish, Stephen Petsche, Holly Butterfield, Alegra Shum, and Emily Hagan.

Big thanks go to the crew at Skybox Imaging, who have been eternally patient as I’ve worked part-time while wrapping up my Ph.D., and did me the honor of attending my defense (and not heckling): Julian Mann, Ollie Guinan, Dirk Robinson, Ronny Votel, and Hadar Isaac.
My girlfriend Jessie Stock’s love and support has sustained me through the stress of completing graduate school. She has spent many more weekends in the basement of the Durand building than any person should have to who is not a Stanford graduate student! Now that my Ph.D. is complete, I owe her many, many back weekends in San Francisco. Thanks for everything, Jess.

Finally, I have no idea what words of gratitude I can express to my parents, Melvyn and Pauline, who have loved me unconditionally and encouraged my endeavors always. From family trips to Cape Canaveral when I was 5 to serving as a research sounding board during my Ph.D., and despite having absolutely no background in engineering or math, they have always been supportive of my academic interests in spaceflight and robotics. They’ve also been incredible parents. Perhaps the best thing I can tell them is I’m finally done with school!
# Contents

1 Introduction ................................................................. 1  
1.1 Pose & Shape Estimation of Moving Targets ....................... 1  
1.1.1 Motivating Applications ........................................... 1  
1.1.2 Real-time Estimation ............................................. 3  
1.1.3 Monocular Vision Sensing ....................................... 4  
1.2 Related Work .......................................................... 6  
1.3 Contributions .......................................................... 9  
1.4 Variables and Notation ............................................... 9  
1.5 Roadmap ............................................................... 12  

2 Image Processing .......................................................... 14  
2.1 Robust Image Features ................................................ 15  
2.1.1 The Scale Invariant Feature Transform (SIFT) ............... 16  
2.1.2 Benefits of Image Histogram Equalization ................... 17  
2.1.3 Feature Detection via GPUs ...................................... 18  
2.2 Data Structures for Feature Matching .............................. 18  
2.2.1 Classical SIFT Match Algorithm ................................. 21  
2.2.2 Improved SIFT Match Algorithm ............................... 22  
2.3 Using Image Features for Estimation ............................... 22  
2.4 Summary ............................................................... 23  

3 Overview of Online SLAM / SFM ....................................... 24  
3.1 Background of SLAM and SFM ...................................... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Physical Models</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Bayesian Filtering vs. Nonlinear Optimization</td>
<td>30</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Bayesian Filtering</td>
<td>30</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Nonlinear Optimization</td>
<td>39</td>
</tr>
<tr>
<td>3.4</td>
<td>Summary</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Online SLAM/SFM for Moving Targets</td>
<td>46</td>
</tr>
<tr>
<td>4.1</td>
<td>No Motion Model Approaches</td>
<td>48</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Uncertainty in Pose Estimate</td>
<td>49</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Ambiguity in Orientation Estimate</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Motion Model Approaches</td>
<td>56</td>
</tr>
<tr>
<td>4.3</td>
<td>Summary</td>
<td>59</td>
</tr>
<tr>
<td>5</td>
<td>Hybrid Pose Estimation</td>
<td>60</td>
</tr>
<tr>
<td>5.1</td>
<td>The Hybrid Concept</td>
<td>61</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Rotation Estimation</td>
<td>61</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Translation Estimation</td>
<td>62</td>
</tr>
<tr>
<td>5.2</td>
<td>Implementation with Rao-Blackwellized Particle Filters</td>
<td>67</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Representation of States</td>
<td>67</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Algorithm</td>
<td>67</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Feature Initialization</td>
<td>72</td>
</tr>
<tr>
<td>5.3</td>
<td>Summary</td>
<td>75</td>
</tr>
<tr>
<td>6</td>
<td>Simulations and Experiments</td>
<td>76</td>
</tr>
<tr>
<td>6.1</td>
<td>Simulations</td>
<td>77</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Simulation Results</td>
<td>78</td>
</tr>
<tr>
<td>6.2</td>
<td>Laboratory Experiments</td>
<td>82</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Laboratory Hardware</td>
<td>83</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Laboratory Results</td>
<td>86</td>
</tr>
<tr>
<td>6.3</td>
<td>Field Experiments</td>
<td>91</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Field Hardware</td>
<td>91</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Field Results</td>
<td>93</td>
</tr>
</tbody>
</table>
6.4 Summary ................................................. 97

7 Conclusion ............................................. 99

7.1 Review of Contributions ............................... 99

7.2 Future Research ........................................ 100

7.2.1 Real-time Implementation .......................... 101

7.2.2 The Role of Bundle Adjustment .................... 101

7.2.3 Fusion with Range-based Sensors .................. 102

7.2.4 Dual Control ........................................ 104

7.3 Conclusion ............................................. 104

List of References ....................................... 106
List of Figures

1.1 Potential Spacecraft Target: Hubble Space Telescope ............... 2
1.2 Other Potential Space Targets ........................................ 2
1.3 Potential Underwater Target: Midwater Respirameter ............... 3
1.4 Target Properties to be Estimated ................................... 3
1.5 Camera Observation Sequence ....................................... 5
1.6 SLAM/SFM : Camera with Unknown Motion .......................... 8
1.7 SLAM/SFM : Target with Unknown Motion ............................ 8
1.8 Target Pose : $\bar{s}_t$ ................................................. 10
1.9 Target Shape : $X$ .................................................. 11

2.1 Bearings-only, Feature-based Measurements .......................... 15
2.2 Example of SIFT Detection .......................................... 16
2.3 Benefits of CLAHE .................................................. 17
2.4 An NVIDIA GPU .................................................... 18
2.5 Structure of the New Feature Table $J$ ................................ 20
2.6 Structure of the Reference Table $K$ .................................. 20
2.7 Updating Table $K$ with Contents of Table $J$ ......................... 21

3.1 SLAM/SFM : Camera with Unknown Motion .......................... 27
3.2 Bayes Filter - Block Diagram ....................................... 33
3.3 A Gaussian Probability Distribution for $\bar{s}$ .......................... 34
3.4 A Finite Approximation to General Probability Distribution for $\bar{s}$ 35
3.5 Particle Filter - Block Diagram ...................................... 36
3.6 Bimodal Posterior : Target Motion Scenario ......................... 36
Chapter 1

Introduction

1.1 Pose & Shape Estimation of Moving Targets

The ability of autonomous robots to navigate relative to moving rigid bodies is an important technology for expanding humankind’s science and exploration capabilities in remote environments, such as outer space and deep underwater. Broadly speaking, this requires the estimation of two distinct things: the relative pose (i.e., position and orientation) between the target of interest and the robot, and a map/reconstruction of the target of interest.

This thesis presents a real-time capable monocular vision method for estimating the 6DOF relative pose and a 3D map of a rotating and translating target of interest, for which nothing is known \textit{a priori}. This capability did not previously exist, and it enables a variety of autonomous robotic applications.

1.1.1 Motivating Applications

One motivation for this research is the desire for autonomous rendezvous and docking with non-communicating, damaged space assets (Figure 1.1). Such a capability would both free human astronauts from having to perform this expensive, difficult, and dangerous job in places where human spaceflight currently reaches (i.e., Low Earth Orbit) and enable its use in places where human spaceflight currently does not reach
(such as Geosynchronous Earth Orbit or about other celestial bodies). To be as robust as possible, such a system must be able to handle situations where attitude control has failed and the target is tumbling.

Figure 1.1: Potential Space Target: Hubble Space Telescope (Credit: NASA)

Another space application of this technology is autonomous landing of science probes on small solar system bodies, such as comets and asteroids (Figure 1.2). Comets in particular represent a target undergoing complicated motion, due to outgassing effects imparting forces and moments on the rigid body.

(a) Comet Hartley-2
(b) Asteroid Eros

Figure 1.2: Other Potential Space Targets (Credit: NASA)
A motivation in the underwater environment is enabling remote submersibles to autonomously stationkeep and moor with tethered underwater bodies. An example of such a body is a marine science instrument (Figure 1.3) in need of servicing. These instruments rotate and translate unpredictably due to ocean currents. Also, they often have no *a priori* placed reference markers (or have markers that have been obscured by biogrowth).

### 1.1.2 Real-time Estimation

The research in this thesis focuses on the *estimation* part of the motivating problems described above. Specifically, using a set of sensor measurements, how can a robot estimate in real-time the 3D shape and the 6DOF pose (Figure 1.4) of an unknown, unpredictably moving rigid body target?
Regarding this estimation process, several properties are essential. They are:

- real-time capability, for use in closed-loop control of the robot with respect to its target of interest
- no required \textit{a priori} knowledge about the target of interest, to allow for interaction with any and all rigid bodies. This means no:
  - map or appearance information
  - knowledge of mass properties (inertia tensor or center of mass)
  - fiducials (pre-placed, recognizable reference markers)
  - knowledge of external forces and torques
- no reliance on information from sensors mounted on the target of interest (i.e., target is incommunicative)

1.1.3 Monocular Vision Sensing

Monocular vision is the sensor used in this research. Monocular vision sensing works by repeatedly taking images of the rigid body target with a single camera, and using the changing observations to discern the states of the target (pose and shape). Figure 1.5 shows the concept of how the camera observation process works over time, both in space and underwater. The observing robot (either a satellite or an underwater vehicle), equipped with a camera and with knowledge of its own pose based on onboard sensors, visually observes a 6DOF moving target (either a damaged satellite or an underwater mooring), taking images at each time-step. Based on these images, the target’s pose and shape are estimated.

One reason for this sensing choice is that a monocular camera is standard equipment on most exploration robots, as a part of the science payload. Designing a pose and shape estimation scheme around monocular vision adds a navigation capability without requiring additional hardware. For example, the images in Figures 1.2a, 1.2b, and 1.3 were all acquired by a science camera carried onboard a robotic vehicle.
CHAPTER 1. INTRODUCTION

A second reason is the small form factor and low power budget of a monocular camera make it amenable to integration even on small robotic platforms. This is important particularly in the space environment, where an emerging trend is the use of small satellites built with commercial-off-the-shelf components (i.e., ‘nanosatellites’) to reduce the cost of performing tasks in space.

Monocular vision is an interesting sensor to study in that it represents a ‘limiting case’; it is bearings-only (i.e., at a given time-step the depth of an image feature is unobservable) and using it for estimation yields shape and relative position estimates known only up to a scale factor. Stereo vision, by contrast, can estimate instantaneously the depths of features and determine scale; however, the accuracy is inversely proportional to the baseline between the cameras. In the limiting case of a target that is far away, stereo vision becomes essentially a monocular vision sensor. Therefore, the successful development of a monocular vision estimation algorithm should also be beneficial when designing algorithms around other, richer sensing modalities.

In order to use a camera for sensing, image processing technologies (from the computer vision field) are leveraged at the front end to convert the camera’s image

Figure 1.5: Camera Observation Sequence

(respectively, the Deep Impact probe, the NEAR Shoemaker probe, and the ROV Ventana).
into usable, feature-based measurements for the estimation algorithm. This process-
ing involves the repeated detection and matching of distinctive point image features
(described in Chapter 2).

1.2 Related Work

Relative pose estimation has been an area of extensive study. Absent the addi-
tional problem of dealing with a target with unknown shape/appearance, algorithms
have been designed for tracking target pose in space [30, 21, 45] and underwater
[44, 57, 31]. Impressively, some of these algorithms have been field-deployed success-
fully on underwater[57] and in-space[30] robotic vehicles. However, these algorithms
typically rely on known, distinctive markers (‘fiducials’) [30, 44, 57], human-selected
natural image features [31], shape knowledge [21], or information from an additional,
more complex sensor (e.g., LIDAR) [45]. They do not address the problem posed in
Section 1.1, where no a priori shape/appearance information is known and where no
additional sensors (other than a monocular camera) are available.

Considering the more complex problem of estimating relative pose and
shape, related research exists in the SLAM and SFM fields.

Simultaneous Localization and Mapping (‘SLAM’) involves the concurrent esti-
mation of the relative pose between a robot and its environment and a map of that
environment. It represents the fusion of two distinct mobile robotics problems: (a) robot mapping, where the robot’s pose (location and orientation) is known but its
surrounding environment is unknown, and (b) robot localization, where the robot has
a map of the surrounding environment but doesn’t know its pose (location and ori-
entation) in that map. In SLAM, neither the relative pose nor the map is known,
neither can be estimated on its own without knowing the other, so the two are es-
timated simultaneously. The SLAM field encompasses both offline (or ‘batch’) and
online (or ‘recursive’) forms of the problem, and includes solutions based on various
sensor types.

Structure from Motion (‘SFM’) involves taking a set of overlapping camera images
of a common scene, and based on those images reconstructing the scene being viewed,
as well as the locations of the camera at the time the images were taken. As with SLAM, it encompasses both offline (batch) and online (recursive) forms. In general in this thesis, online monocular SLAM and online SFM will be considered as solving the same problem, and as such will be referred to as a single entity: online SLAM/SFM. A more detailed background on SLAM and SFM is presented in Chapter 3.

Online SLAM/SFM has been the subject of extensive research. Examples of online monocular SLAM/SFM include [65, 3, 2, 16, 19, 20, 43, 35, 38, 40, 41, 48, 55, 58, 13, 12, 5, 9]. These algorithms reflect the two dominant approaches to performing pose estimation during online SLAM/SFM. One approach is to use nonlinear optimization to deterministically find the pose that best minimizes the feature reprojection error. The alternative approach is to use recursive Bayesian filtering to blend information from probabilistic models of (a) how the states evolve over time and (b) how the measurements are formed. A detailed explanation of these existing online SLAM/SFM techniques is given in Chapter 3.

Static Target, Moving Camera vs. Moving Target, Static Camera

The existing body of online SLAM/SFM has in general focused on the problem of a static environment to be mapped, where the camera-equipped robot is the moving entity whose pose is to be tracked. This can be considered a case of colocated pose estimation, where the sensor being used (the camera) is located on the rigid body whose pose is to be tracked (the robot). Knowledge of the unknown relative motion can be improved by adding inertial sensors to the robot, as it is the robot’s motion that is inducing uncertainty in relative pose. See Figure 1.6.

The problem under concern in this thesis, as presented in Section 1.1, differs from the above. The object whose pose is to be tracked and the entity to be mapped are one and the same, external and physically separate from the robot. This can be considered a case of non-colocated pose estimation. In contrast to the problem focused on by existing SLAM/SFM methods, there is no sensor that can be added to the robot to improve knowledge of the unknown relative motion, as it is the target’s motion that is inducing uncertainty in relative pose. See Figure 1.7. As a result, changes in pose can have greater unpredictability. Also, the signals received
by the camera are potentially harder to separate from the sensor noise, resulting in less estimation confidence. These issues are explained in detail in Chapter 4.

For these reasons, applying existing techniques from the SLAM/SFM field to the moving unknown target case (i.e., unknown shape, unknown mass properties, non-communicative, arbitrary motion) will have serious drawbacks to achieving real-time, accurate pose tracking and reconstruction (as explained in Chapter 4). A new method is needed for solving the problem outlined in Section 1.1.
1.3 Contributions

The main contribution of this research is a real-time capable technique for monocular vision-only pose estimation and shape reconstruction of a rotating and translating rigid body. Using this algorithm, a camera can determine the relative pose and shape of an observed target, without any \textit{a priori} knowledge or restrictions on the target’s motion.

The algorithm itself is novel in that it is a unique hybridization of the two dominant approaches to online SLAM/SFM, recursive Bayesian filtering and nonlinear optimization. This hybrid algorithm for moving target online SLAM/SFM is presented in Chapter 5.

In addition, three other contributions are presented, which aid the monocular online SLAM/SFM field in general. They are:

- A detailed description of how this algorithm fits in the context of existing online SLAM/SFM methods (presented over Chapters 3 - 5)
- An improved matching algorithm for SIFT, the specific type of robust image feature used during vision processing (presented in Chapter 2, Section 2.2.2)
- A new, linear cost function for fast, optimization-based determination of target position (presented in Chapter 5, Section 5.1.2)

1.4 Variables and Notation

This section establishes a consistent set of notation used throughout the remainder of this thesis. It also defines a few variables which are widespread throughout the thesis. Other variables will be defined in the text as they are discussed.

**Vectors and Matrices**

Vectors are denoted with an overbar (\(^\overline{}\)). Matrices are denoted with a boldface font (\(\mathbf{M}\)).
Variables

The pose vector is $\bar{s}$. It consists of position ($\bar{\rho} = [\rho_x \rho_y \rho_z]^T$) and orientation, expressed as Euler angles ($\bar{\theta} = [\psi \theta \phi]^T$). Figure 1.8 illustrates the pose variables to be estimated, for the case of an unknown moving target. Additionally, depending on context the pose vector can also contain the velocity ($\dot{\bar{\rho}} = \bar{v} = [v_x \ v_y \ v_z]^T$) and/or the angular rate ($\bar{\omega} = [\omega_x \ \omega_y \ \omega_z]^T$).

A single feature’s position in the map is $\bar{x}$. The group of features’ positions is expressed as matrix $X$ (see Figure 1.9). Appropriate superscripts and subscripts are used to specify feature, frame, and frame origin (for example, in Figure 1.9, the map frame and map origin).

A single measurement is $\bar{z}$. A group of measurements is expressed as matrix $Z$.

Gaussian random noise is denoted by $\nu$ and covariances are given by $\Sigma$ (both usually with a descriptive subscript).

The variable $f_c$ is the focal length of the camera, a parameter used in the measurement model of image formation.
Superscripts and Subscripts

Capital superscripts usually refer to the coordinate frame of a vector: $C$ denotes the camera frame, $M$ the map frame, and $I$ the inertial frame. Rotation matrices have a superscript ratio, where the denominator is the initial frame and the numerator is the final frame. A superscript of $T$ denotes the transpose of a vector or matrix. For example:

$$ (R^{C/M}) = (R^{M/C})^T $$

A superscript in brackets indicates a particle, when particle filters are discussed. For example, $\tilde{s}^{[i]}$ is the $i$th particle describing the distribution of the relative pose vector $\tilde{s}$.

Subscripts are usually used for specifying a component of a spatial vector or time sequence. For example, $\rho_x$ is the $x$-component of the $\tilde{\rho}$ vector, and $\tilde{s}_t$ is the $t$-th element of a time sequence of the $\tilde{s}$ vector. Subscripts are also used for descriptive information.

Ratio subscripts are used with position vectors, to express the head (numerator)
and tail (denominator) position of the vector. For example, \( \bar{x}_j^M / \text{map} \) refers to a position \((j)\) relative to the origin of the map \((\text{map})\), expressed in the map’s coordinate frame \((M)\).

To keep things concise, if the denominator of the subscript is omitted, it is assumed that the position is relative to the origin corresponding to the specified reference frame. That is \( \bar{x}_j^M \equiv \bar{x}_j^M / \text{map} \) and \( \bar{x}_j^C \equiv \bar{x}_j^C / \text{cam} \).

**Predictions**

For greater clarity when discussing estimation techniques, predictions of variables are denoted with a hat (\(^\hat{\cdot}\)). For example, \( \mathbf{Z}_t \) would be the actual set of measurements observed at time \( t \), and \( \mathbf{\hat{Z}}_t \) would be the prediction for those measurements, i.e., the best estimate of what the measurements should be based on the information available.

**Indexing**

The \( t \) index refers to time. When referencing time, a subscript colon (as in \( \mathbf{Z}_{0:t} \)) represents a range of discrete time-steps.

The \( j \) index (and also sometimes the \( k \) index) is used to refer to a specific feature, as both a 2D image measurement and a 3D position in the map or in space. The total number of features in the map \( \mathbf{X} \) is denoted by \( N \).

The \( i \) index is used to refer to a specific particle, when discussing particle filters. The total number of particles is denoted by \( M \).

**1.5 Roadmap**

This chapter has given an overview of the current state of target-relative navigation, and the contributions to the field that are made in this thesis.

Chapter 2 discusses robust image features, and their utilization in this research.

Chapter 3 gives a background on the related SLAM and SFM research fields, and the different types of existing algorithms for solving online SLAM/SFM, which focus on the static target problem.
Chapter 4 focuses on a previously unsolved problem, online monocular vision-only SLAM/SFM of an unknown moving target, and shows that existing algorithms can encounter significant difficulties when applied to this problem.

Chapter 5 outlines a novel hybrid approach to solving this problem, and describes a specific implementation using Rao-Blackwellized Particle Filters (RBPFs). This is the main contribution of the research presented in this thesis.

Chapter 6 describes simulations and laboratory and field experiments conducted to show the efficacy of the new algorithm.

Chapter 7 concludes the thesis and offers recommendations for future research.
Chapter 2

Image Processing

The vision approach taken in this research is feature-based; the measurements taken by the sensor relate to the position of reference points (the ‘features’) consistently observed over multiple time-steps. The map constructed is a sparse point cloud of the locations of these features. With monocular vision, the observations taken are ‘bearings-only’, meaning a single observation by the camera of a feature does not give any information on the range to that feature.

‘Image features’ are general points of interest detected in the 2D intensity map that is a digital black-and-white image. To be useful in robotic applications, an image feature needs to be a repeatable measure of a distinct point in the 3D map. For this reason, it is critical to establish what is known as ‘correspondence’, that is, the matching of an image feature in one image to an image feature observed at a succeeding time-step, where both image features represent the same point in the 3D map. Figure 2.1 is a schematic of the mapping of 3D points (black crosses) to 2D image features, via bearings-only feature-based measurements (dotted lines).

This chapter describes the use of 2D ‘robust’ image features (Section 2.1) for repeatedly detecting recognizable locations in the camera’s field of view, and some tools that aid detection. The data structures required to keep track of features over time (i.e., perform correspondence) are described in Section 2.2. Finally, Section 2.3 describes the physical model of how the camera maps 3D locations to 2D robust image features, a lead-in to the SLAM/SFM algorithms of Chapters 3-5.
2.1 Robust Image Features

The rise of robust image point features has been a key enabler of visual robotic algorithms such as visual SLAM and SFM. Robust image features encode distinctive information, apart from their pixel location, which can be used to establish a feature’s correspondence between different time-steps. Robust features can be thought of as ad-hoc fiducials; instead of using purposely-placed, easily-recognizable patterns, the camera tracks naturally occurring, easily-recognizable patterns. Notable examples are Scale Invariant Feature Transforms (SIFTs) [46], Speeded Up Robust Features (SURFs) [7], Features from Accelerated Segment Test (FASTs) [59], Center Surround Extrema (CenSurEs) [1], and Binary Robust Independent Elementary Features (BRIEFs) [11].

The estimation algorithm presented later in this thesis works for any type of point image features, given that tracking/correspondence is being properly addressed. SIFTs were chosen for this research, as they have been successfully adopted for previous vision-based robotics applications in both air (such as terrestrial SLAM [6, 35, 63]) and underwater (for SLAM[62] and AUV navigation[22]).
2.1.1 The Scale Invariant Feature Transform (SIFT)

The SIFT algorithm, developed by Lowe[46], revolutionized the field of image feature detection and remains one of the most popular robust image features used in computer vision applications. Quoting from the original 2004 paper:

The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images.

Figure 2.2 shows an example of a raw image, with the SIFTs detected visible in red.

![Figure 2.2: Example of SIFT Detection](image)

The 2D pixel locations of the SIFTs are called the ‘keypoints’, and are positioned at the local intensity maxima and minima in the image (exceeding some threshold), at varying scales. These 2D keypoints, in general, map to distinct 3D points in the scene being observed by the camera. Tracking of the keypoints through images in a video stream represents the temporal tracking of these 3D points as they move with respect to the camera.
Each keypoint has an associated ‘descriptor,’ which is an 128-element vector that encodes a histogram of the intensity gradients in the pixels that are in the vicinity of the keypoint. Specifically, the descriptor histogram is of intensity gradients at 8 different orientations for a given patch of pixels, where the keypoint is surrounded by a 4 by 4 array of such patches. The descriptor is encoded in a way that makes it invariant to rotations in the image plane and scale changes. It is this descriptor which acts as a unique signature, allowing recognition of the SIFT in future images of the scene, by matching of descriptors in different images. For a thorough description of SIFT, the process of localizing the keypoints, and the process of constructing the keypoint descriptors, the reader is referred to [46].

### 2.1.2 Benefits of Image Histogram Equalization

In difficult lighting environments, the low intensity range of light in the collected images can cause fewer SIFTs to be detected (due to smaller intensity maxima and minima). For this reason, an algorithm known as Contrast Limited Adaptive Histogram Equalization (CLAHE) [72] was implemented in software, to boost the intensity range of an image, before passing it to the SIFT detection step. This results in more features being detected and available for use by robotics algorithms.

Figure 2.3 shows an example raw image from an underwater camera and the same image after being CLAHE-enhanced.

![Figure 2.3: Benefits of CLAHE](image-url)
2.1.3 Feature Detection via GPUs

An emerging technology for general purpose computing is graphical processing units (GPUs). These single instruction, multiple data (SIMD) processors allow for massively parallel computation (an example is shown in Figure 2.4). For computational tasks which require the same set of instructions to be performed many times over (such as detection of image features and calculation of their associated descriptors), GPUs can provide a massive speed-up in performance.

For this research, a shareware C++ software package [71] was used to detect SIFTs with a GPU.

![Figure 2.4: An NVIDIA GPU (Credit: NVIDIA)](image)

2.2 Data Structures for Feature Matching

To explain how 2D features are tracked consistently over time, it is necessary to introduce a few data structures which are used for storing and processing the raw features described in the previous section.

When a new image is taken, the latest SIFTs are extracted into a feature table $J$. Each newly observed feature $j$ in the table has an associated 2D pixel location (the
‘keypoint’) $\tilde{z}_j$ and an associated 128-vector descriptor $\tilde{n}_j$. Figure 2.5 shows a diagram of table $J$.

To track SIFTs efficiently over time, a second feature table is required: the reference table $K$, storing the reference SIFT features $k$ that have been viewed previous to the current time-step. This reference table $K$ also contains a vector of counters. For each feature $k$, a counter keeps track of the number of time-steps since that feature was observed last. Figure 2.6 shows a diagram of table $K$.

At initialization, all the SIFTs in $J$ are copied into $K$, and the counters for these features are set to 0.

At each successive time-step, the counters are incremented. Then, each of the SIFTs in $J$ are compared against the reference table $K$ to find a match $k_1$, via the methods described below in Sections 2.2.1 and 2.2.2. If a match is found, the new SIFT’s keypoint ($\tilde{z}_j$) and descriptor ($\tilde{n}_j$) vector overwrite those of the match in the reference table (i.e., overwrite $\tilde{z}_{k_1}$ and $\tilde{n}_{k_1}$), so that the reference table contains the latest information on that particular SIFT. Additionally, the counter of $k_1$ is set to 0, and the new pixel coordinates of $k_1$ are passed to the pose estimation algorithm, for use as a measurement of $k_1$ at the current time-step (as described in further chapters).

The counter is used for pruning the library. If every SIFT were stored in $K$ for comparison with future SIFTs, the reference table would grow unmanageably large. Instead, a SIFT is removed from the library if it has not been observed in the last 8 time-steps (a time length chosen empirically). Note that this is a relatively simplistic method for library pruning; more advanced techniques, which take into account the target’s velocity and rotation rate to predict when features leave the field of view or are occluded, might prove more effective and are worth examining.

Figure 2.7 shows an example of the reference table $K$ being updated at the most recent time-step, as SIFTs from the new table $J$ are either matched or inserted as new distinct features.
Figure 2.5: Structure of the New Feature Table $J$

Figure 2.6: Structure of the Reference Table $K$
2.2.1 Classical SIFT Match Algorithm

The classic SIFT correspondence algorithm [46] works as follows. For each newly observed feature \( j \), the Euclidean distance (in the 128-dimensional descriptor space) is computed between it and a set of SIFTs from the reference library \( K \). These reference SIFTs each correspond to an already observed feature. The Euclidean distances to the closest reference feature \( k_1 \) and to the second closest feature \( k_2 \) are stored as defined in Equation 2.1.

\[
\begin{align*}
  k_1 &= \arg\min_{k \in K} ||\vec{n}_j - \vec{n}_k||_2 \\
  k_2 &= \arg\min_{k \in K, k \notin \{k_1\}} ||\vec{n}_j - \vec{n}_k||_2
\end{align*}
\]  

(2.1)

If the ratio of the Euclidean distance between \( j \) and \( k_1 \) to the Euclidean distance between \( j \) to \( k_2 \) is less than a threshold \( m_{\text{dist}} \) (Equation 2.2), then the newly observed
feature \( j \) is declared to be a match to reference feature \( k_1 \).

\[
\frac{||\vec{n}_j - \vec{n}_{k_1}||_2}{||\vec{n}_j - \vec{n}_{k_2}||_2} \leq m_{dist} \tag{2.2}
\]

The Euclidean distance ratio threshold \( m_{dist} \) was set to 0.6 (determined empirically).

Only reference SIFTs within a given pixel distance of a newly observed SIFT are considered as candidates for potential correspondence. The pixel distance cutoff was set to 15 pixels (determined empirically).

### 2.2.2 Improved SIFT Match Algorithm

In this research, an additional logical check is incorporated (Equation 2.3) to improve match performance and eliminate false positives. To be considered a match, the dot product between the new SIFT vector and reference SIFT vector must be greater than a threshold \( m_{dotProd} \). This ensures that, in regions of only a few reference SIFTs, a match is not selected when none of the reference SIFTs is at all similar to the new SIFT.

\[
\vec{n}_j^T \vec{n}_{k_1} \geq m_{dotProd} \tag{2.3}
\]

The value of \( m_{dotProd} \) was set to 0.8 (determined empirically).

### 2.3 Using Image Features for Estimation

The previous two sections have described the detection of robust point image features and their matching over time. These matched image features get passed to the actual SLAM/SFM algorithm (discussed in the next several chapters), to be used as a source of information for reconstructing the relative pose \( \vec{s} \) and the map \( X \) of the target of interest.

To be useful in the context of SLAM/SFM, the algorithm will need a model (denoted \( \mathcal{g}() \)) representing how the camera physically maps 3D locations into 2D
CHAPTER 2. IMAGE PROCESSING

pixel measurements $Z_t$:

$$Z_t = g(s_t, X)$$ (2.4)

For the research in this thesis, the camera is assumed to form 2D images via the perspective projection model:

$$\begin{bmatrix} u_j \\ v_j \end{bmatrix} = \frac{f_c}{z_j/cam} \begin{bmatrix} x_j/cam \\ y_j/cam \end{bmatrix}$$ (2.5)

The position of the feature in the camera frame ($x_j/cam = [x_j/cam, y_j/cam, z_j/cam]^T$) is related to the position of the feature in the map frame ($x_j/map \in X$) via the following transform:

$$x_j/cam = R^{C/I} \left( R^{I/M} x_j/map + t_I/map - t_I/cam \right)$$ (2.6)

Equations 2.4/2.5 are referred to as the ‘measurement model’. The use of this physical model in SLAM/SFM (along with the transform from map frame to camera frame, Equation 2.6) will be explained thoroughly in the next several chapters.

2.4 Summary

This chapter explained the details of the vision processing algorithms which run as the front end to the estimation process. These algorithms detect robust image features and track and match them over time, passing on their 2D pixel locations to the estimation process. Software tools (CLAHE) and hardware tools (GPUs) were used to improve feature detection, while an improved SIFT feature match algorithm was used to improve feature matching.
Chapter 3

Overview of Online SLAM / SFM

The contribution of this thesis, an algorithm for real-time moving target SLAM/SFM, is a novel hybridization of existing techniques in the online SLAM/SFM field. Prior to presenting the new hybrid algorithm (in detail in Chapter 5), it is useful to discuss these existing techniques, which have in general focused on the different problem of a static target and moving camera.

This chapter presents the SLAM/SFM problem and describes the existing approaches to online SLAM/SFM in the context of static targets. The next chapter will demonstrate that these existing SLAM/SFM approaches can have drawbacks when applied to the *moving* target problem. This is the motivation for requiring the new hybrid SLAM/SFM algorithm of Chapter 5.

Section 3.1 below gives a background on the SLAM, SFM, and online SLAM/SFM research fields. Section 3.2 discusses the physical models of measurement formation \( g() \) and motion propagation \( f() \), and their use as part of an online SLAM/SFM estimation process. Section 3.3 describes the two dominant online SLAM/SFM approaches to date: recursive Bayesian estimation (Section 3.3.1) and nonlinear optimization techniques (Section 3.3.2). These methods differ significantly in how they estimate the pose at each time-step.
CHAPTER 3. OVERVIEW OF ONLINE SLAM / SFM

3.1 Background of SLAM and SFM

In the robotic navigation field, the research in this thesis can be considered an example of Simultaneous Localization and Mapping or ‘SLAM’ (and even more specifically, online monocular SLAM). The wider SLAM field includes research utilizing a broad variety of primary sensors (other examples besides monocular vision include stereo vision [62, 64, 63, 6, 56] or LIDAR/sonar [25, 45]). A good general discussion of SLAM is given in [68].

In the computer vision field, the research in this thesis can be considered an example of Structure from Motion or ‘SFM’ (and even more specifically, online SFM). In contrast to SLAM, SFM research is always based around visual sensing (generally monocular vision). As Ma et al. define the SFM problem, “... [given] a number of 2-D images ... to what extent can we estimate the 3-D shape of [an] object ... [and] to what extent can we recover the motion of [the] object relative to the camera?” [47]

It is sometimes assumed that the term ‘SLAM’ by default means an online process (unless specified otherwise), and that the term ‘SFM’ by default means an offline process (unless specified otherwise). For example, Davison et al. draw a distinction between “... Simultaneous Localization and Mapping (SLAM) ...,” which involves “... a moving sensor platform constructing a representation of its environment on the fly while concurrently estimating its ego-motion,” and “... Structure from Motion ... algorithms ...” which are “... fundamentally offline in nature, analyzing a complete image sequence to produce a reconstruction of the camera trajectory and scene structure observed.” [16]

However, in truth both the SLAM and SFM fields involve research into both recursive and batch methods, and whether ‘SLAM’ or ‘SFM’ is the shorthand term used to describe the research often depends on whether it is being connected to either the larger robotics community or the larger computer vision community. For example, in putting his online research in context, Nistér states in one paper that “... Structure from motion has been a highly active research area of computer vision,” [54] while in another paper Nistér et al. state their research is “... [c]losely related [to] what is known in the robotics community as simultaneous localization and mapping (SLAM)”
Strasdat et al., in a recent paper [66], state essentially the same thing when talking about “... real-time SFM – also called monocular SLAM (Simultaneous Localisation and Mapping).” They refer to the “... unfortunate dual terminology ...” of “... the Structure from Motion (SFM) research area in computer vision, whose principles were derived from photogrammetry, and the Simultaneous Localisation and Mapping (SLAM) sub-field of mobile robotics research ...”

While there is considerable research into offline SLAM and SFM, where a batch process is used to estimate the map and a set of camera poses [47, 69], it is not in general applicable to real-time robotics applications, as for large numbers of timesteps and measurements it cannot be solved fast enough. A caveat to this is the periodic use of ‘Bundle Adjustment’ for a posteriori smoothing (explained in detail in Section 3.3.2).

In this thesis online SLAM/SFM will be the focus, as the end-goal is a process with a pose estimation step which can run in real-time.

### Static Target, Moving Camera

In general, the focus of the research community has been on solving the case of SLAM/SFM with a static target to be mapped/reconstructed. In this situation, it is the camera’s pose that is being estimated, as it is the camera which is undergoing unknown motion (Figure 3.1).

Recall the mapping of positions from map frame to camera frame (Equation 2.6):

$$\bar{x}_{j/cam}^C = R^C/I \left( R^I/M \bar{x}_{j/map}^M + \bar{\rho}_{map/o}^I - \bar{\rho}_{o/cam}^I \right) \quad (3.1)$$

For a static target and moving camera, Equation 3.1 can be simplified as follows:

$$\bar{x}_{j/cam}^C = R^C/M \left( \bar{x}_{j/map}^M - \bar{\rho}_{map/cam}^M \right) \quad (3.2)$$

Comparing Equation 3.2 to Equation 3.1, the map frame ($M$) has been taken to be identical to the inertial frame ($I$), with the map origin ($map$) collocated with the reference origin ($o$), so that $\bar{\rho}_{o/cam}^I \equiv \bar{\rho}_{map/cam}^M$ and $R^C/I \equiv R^C/M$ (expanded in
CHAPTER 3. OVERVIEW OF ONLINE SLAM / SFM

Equation 3.3).

\[
R^{C/M} = \begin{bmatrix}
c\psi c\theta & s\psi c\theta & -s\theta \\
c\psi s\theta s\phi - s\psi c\phi & s\psi s\theta s\phi + c\psi c\phi & c\theta s\phi \\
c\psi s\theta c\phi + s\psi s\phi & s\psi s\theta c\phi - c\psi s\phi & c\theta c\phi
\end{bmatrix}^{C/M}
\] (3.3)

The relative pose vector \( \bar{s} \) consists of the position of the camera relative to the reference/map origin expressed in the inertial/map frame (\( \bar{\rho}_{I/o} \), or \( \bar{\rho}_{cam} \) for simplicity) and the orientation Euler angles (\( \bar{\theta}_{cam} = [\psi \ \theta \ \phi]^T \)) expressing the rotation from the inertial/map frame to the camera frame (i.e., forming \( R^{C/I} \)).

\[
\bar{s}_t = \begin{bmatrix} \bar{\theta}_{cam} \\ \bar{\rho}_{cam} \end{bmatrix}_t
\] (3.4)

It is this camera pose vector \( \bar{s}_t \), along with the map of the static environment \( X \), which is to be estimated at each time-step, based primarily on incoming camera measurements \( Z_t \). These variables are related by the physical models of image formation and camera motion.
3.2 Physical Models

At the heart of the online estimation process are the physical models that relate the estimators’ states and measurements. In the case of the SLAM/SFM problem, there are two such physical models.

One is the measurement model described in the last chapter (Equations 2.4/2.5), restated below in functional form. At every time-step, this function maps the feature positions (in the map frame) $X$ and the current pose $\bar{s}_t$ to the measurements observed $Z_t$ on the imaging plane.

\[ Z_t = g(\bar{s}_t, X) \]  

As will be described further in Section 3.3.2, it is possible to estimate the pose using only the measurement model[20, 55]. These algorithms determine a single pose that best aligns the current measurements to the 3D map. Applications include visual odometry for ground vehicle navigation ([55]). Such algorithms can be considered the inverse of Equation 3.5:

\[ \bar{s}_t = h(Z_t, X) = g^{-1}_{\|X\|}(Z_t) \]  

That is, if the measurement model is the function that maps pose to pixel projections, given the 3D feature locations $X$, then Equation 3.6 is the function that maps pixel projections to pose, given the same.

The other physical model that can be utilized is a function $f()$ representing how the states being estimated are expected to evolve over time, known as a ‘motion model’ (or ‘process model’). The alternative approach to the ‘measurement model-only’ scheme of Equation 3.6 is to use both the measurement model and the motion model in the online estimation (either via Bayesian filtering or nonlinear optimization, as described in the next section).

In the SLAM/SFM problem, a motion model is used to predict the current pose
\( \vec{s}_t \) between the camera and the target, given the best estimate of previous pose \( \vec{s}_{t-1} \).

\[
\vec{s}_t = f(\vec{s}_{t-1}) \tag{3.7}
\]

As the reconstruction is of a rigid body, the 3D map \( \mathbf{X} \) should not change with time.

Using a motion model \( f() \) enforces physically realistic relative motion between the camera and the map. It is used for all Bayesian filtering methods (3.3.1), such as [19, 16, 35, 22, 24, 23], as it serves as the time update step of a Bayesian recursive process (see Figure 3.2). It is also used in some nonlinear optimization approaches [38, 40, 41] as a prediction of the relative pose to initialize the iterative refinement (as explained in Section 3.3.2).

In the static target SLAM/SFM problem (Figure 3.1), it is the camera that moves, usually as a result of being mounted to a mobile robotic platform. In this case, the platform is often equipped with navigation sensors which provide additional information \( \vec{u} \) to aid the propagation of the camera’s/robot’s pose:

\[
\vec{s}_t = f(\vec{s}_{t-1}, \vec{u}_{t-1}) \tag{3.8}
\]

Examples of additional sensors that have been used during monocular SLAM/SFM are IMUs [41, 40, 43], wheel encoders [35], gyros [22], compasses and inclinometers [22], and acoustic sensors such as a Doppler Velocity Log (DVL) [22].

Alternatively, the motion model can be a function of the previous pose only:

\[
\vec{s}_t = f(\vec{s}_{t-1}) \tag{3.9}
\]

Algorithms which use such a motion model are sometimes referred to as solving the ‘vision-only’ online SLAM/SFM problem [16], to reference the fact that no other sensor data, apart from the camera’s measurements, are used. Examples of algorithms which use this simple motion model are [15, 19, 16, 38, 29].
3.3 Bayesian Filtering vs. Nonlinear Optimization

Within the online monocular SLAM/SFM field there are two distinct methods\cite{41} of estimating pose at each time-step.

The stochastic Bayesian filtering approach uses a recursion of time updates and measurement updates to probabilistically combine information from the system’s motion model and measurement model, taking into account the noise/uncertainty associated with each. Another common name for this type of system is ‘Causal system’. Independent of online SLAM/SFM, a very large body of research exists on nonlinear filtering (using tools such as extended Kalman filters, particle filters, information filters, etc.).

The alternative approach is to use nonlinear optimization, minimizing an error function to get the best agreement between predicted camera measurements and actual camera measurements. As this approach is a variation on typical solution methods for the general (offline) SFM problem, another common name for this type of system is ‘SFM-based system’.

Quoting from Konolige et al.\cite{41}:

“Causal systems associate 3D landmarks with features, and track the camera frame with respect to these features, typically in a Kalman filter framework. The filter is composed of the set of landmarks and the last pose of the camera. . .

“SfM systems are a type of feature-tracking method that use structure-from-motion methods to estimate the relative position of two or more camera frames, based on matching features.”

These two estimation approaches are now described in turn.

3.3.1 Bayesian Filtering

A Bayesian filtering process is a common tool in the recursive estimation field to fuse models of a system’s dynamics (the ‘time update’) with (noisy) measurements of the
system (the ‘measurement update’) to form a probabilistic estimate of the system’s states.

If \( t \) is the current time-step, the probability distribution that is desired is:

\[
p(\bar{s}_t, \mathbf{X} | \mathbf{Z}_{0:t}) \tag{3.10}
\]

Equation 3.10 represents the best information possible about the states \( \bar{s}_t \) and \( \mathbf{X} \), given all the measurements from initialization up to the current time-step, \( \mathbf{Z}_{0:t} \).

In order to estimate this distribution, two tools are available. One is a probability distribution of how the dynamic states (in this case, the pose \( \bar{s} \)) evolve with time:

\[
p(\bar{s}_t | \bar{s}_{t-1}) \tag{3.11}
\]

Typically, this distribution is based around the motion model (Equation 3.7). For example, Equation 3.11 is often expressed as a Gaussian distribution around the motion model’s prediction:

\[
p(\bar{s}_t | \bar{s}_{t-1}) = \mathcal{N}(f(\bar{s}_{t-1}), \Sigma_{\text{mot}}) \tag{3.12}
\]

The terms ‘motion model’ or ‘process model’ may be used interchangeably between the deterministic (3.7) and probabilistic (3.11) contexts.

The second tool available is a probability distribution of how the measurements \( \mathbf{Z} \) are formed:

\[
p(\mathbf{Z}_t | \bar{s}_t, \mathbf{X}) \tag{3.13}
\]

This distribution is based around the measurement model (Equation 3.5). For example, Equation 3.13 is often expressed as a Gaussian distribution about the measurements predicted:

\[
p(\mathbf{Z}_t | \bar{s}_t, \mathbf{X}) = \mathcal{N}(g(\bar{s}_t, \mathbf{X}), \Sigma_{\text{meas}}) \tag{3.14}
\]

The term ‘measurement model’ may be used interchangeably between the deterministic (3.5) and probabilistic (3.13) contexts.
At the beginning of a time-step, the prior probability distribution of the states is available:

\[ p(\bar{s}_{t-1}, X|Z_{0:t-1}) \] (3.15)

Bayesian filtering is a method for calculating Equation 3.10 from 3.15, 3.11, and 3.13. First, the motion model is used to predict the pose at the current time-step:

\[ p(\bar{s}_t, X|Z_{0:t-1}) = \int p(\bar{s}_t|\bar{s}_{t-1}) \times p(\bar{s}_{t-1}, X|Z_{0:t-1}) \, d\bar{s}_{t-1} \] (3.16)

The calculation performed in Equation 3.16 is referred to as the ‘time update’ or ‘prediction’ step of the Bayes filter.

Second, Bayes’ theorem is applied. Using the measurement model, the estimates of pose \( \bar{s}_t \) and map \( X \) are corrected:

\[ p(\bar{s}_t, X|Z_{0:t}) = \frac{p(Z_t|\bar{s}_t, X) \times p(\bar{s}_t, X|Z_{0:t-1})}{p(Z_t)} \] (3.17)

The calculation performed in Equation 3.17 is referred to as the ‘measurement update’ or ‘correction’ step of the Bayes filter. In practice, the probability distribution of the measurements, \( p(Z_t) \), is assumed uniform, and Equation 3.17 is expressed as:

\[ p(\bar{s}_t, X|Z_{0:t}) = \eta \times p(Z_t|\bar{s}_t, X) \times p(\bar{s}_t, X|Z_{0:t-1}) \] (3.18)

The Bayes filter recursively performs the time update calculations (3.16) and measurement update calculations (3.18) at each time-step, as shown in the block diagram in Figure 3.2.

In general, the marginalization and conditioning of probability distributions that occurs (in Equations 3.16 and 3.17, respectively) is burdensome computationally, so in practice some assumptions are made about (or simplifications made to) the probability models. Different types of Bayes filters can be distinguished by what assumptions and simplifications they make.

For example, assuming Gaussian noise on the motion and measurement models
makes marginalization and conditioning linear matrix calculations. For linear dynamics and measurements, with Gaussian noise, the Bayes filter is called a ‘Kalman filter’ (KF), and is an optimal method for calculating the mean and covariance of the state estimate. In the online SLAM/SFM problem, the perspective projection model of a monocular camera (Equation 2.5) is nonlinear. It is also likely that the system dynamics \( f() \) are nonlinear. For these cases, a non-optimal, nonlinear filter must be used.

Extended Kalman Filters

One choice is the extended Kalman filter (EKF), which is formulated the same as the standard linear KF, with the filter linearized about the predicted state mean at each time step. Examples of EKF-based algorithms for monocular SLAM/SFM are [65, 22, 16, 13, 5, 9].

In a KF/EKF, the state’s probability distribution is assumed to be Gaussian (Figure 3.3), and is parameterized with a mean vector \( \bar{\mu}_{s_t,x} \) and a covariance matrix \( \Sigma_{s_t,x} \).

\[
p(\bar{s}_t, X|Z_{0:t}) : \bar{\mu}_{s_t,x}, \Sigma_{s_t,x} \tag{3.19}
\]
Extended Information Filters

The information filter (IF) is also a Gaussian noise-based filter; it is the dual of the Kalman filter. For nonlinear estimation, an extended information filter (EIF) can be formulated by linearizing the nonlinear system around a reference point, in the same manner as an EKF. IFs/EIFs can be advantageous in that the information matrix (the inverse of the covariance matrix) is sparse, making it faster computationally to update. EIFs are also used for monocular SLAM/SFM [24, 23].

In an IF/EIF, the Gaussian probability distribution is parameterized with an information vector $\eta_{s_t, x}$ and an information matrix $\Lambda_{s_t, x}$.

$$p (s_t, X | Z_{0:t}) : \eta_{s_t, x}, \Lambda_{s_t, x}$$ (3.20)

The information form is related to the covariance form as follows:

$$\bar{\eta} = \Lambda \bar{\mu}$$
$$\Lambda = \Sigma^{-1}$$ (3.21)

Particle Filters

An alternative nonlinear Bayes filter is a particle filter. Particle filters are based on sequential Monte Carlo sampling [18]; instead of representing the state estimate with a mean and covariance, the state is represented with a (weighted) finite set of state
hypotheses (see Equation 3.22 and Figure 3.4).

\[ p(\bar{s}_t, X|Z_{0:t}) : \{\{w^1, \bar{s}^1_t, X^1\} \ldots \{w^i, \bar{s}^i_t, X^i\} \ldots \{w^M, \bar{s}^M_t, X^M\}\} \]

(3.22)

Figure 3.4: A Finite Approximation to General Probability Distribution for \( \bar{s} \)

In Equation 3.22, \( w^i \) is the relative weight of particle \( i \). The particle filter keeps the finite set of particles ‘focused’ about the high likelihood regions of the state space by periodically resampling the particles, based on each particle’s weight \( w^i \). This action removes particles with low weights (in low probability regions of the distribution), and replaces them with more particles near the particles with high weights (in high probability regions of the distribution). Figure 3.5 shows the calculations performed at each time-step in a particle filter.

Examples of particle filter-based algorithms for monocular SLAM/SFM are [19, 35, 48, 58, 12].

When estimating the target’s pose \( \bar{s} \), the strength of a particle filter over an EKF is that the probability distribution is not restricted to being unimodal, quasi-Gaussian. This is important in the context of the moving target SLAM/SFM problem. The posterior is evolving based on bearings-only measurements, and these can give rise to multiple distinct regions of high likelihood in the distribution.

For example, consider the scenario illustrated in Figure 3.6. This shows a slowly rotating target with 4 features at equal depth from the camera at time 0.

Figure 3.7 shows the particle cloud evolving over time; the dark dots are the prior particles, and the light dots are the re-sampled particles. As can be seen, two
distinct groups emerge from the initial distribution, each representing a distinct, high likelihood hypothesis of the angle of the target. Figure 3.8 shows the same results, as plots of particle weights $w_i$ versus hypothesis at images 6, 9, 12, and 15. This illustrates the fact that while the posterior probability is high the target has rotated to some angle about either $+Y$ or $-Y$, the probability is low the target has not rotated at all. A unimodal (e.g., Gaussian) model would be a poor fit for this bimodal probability distribution of the target’s orientation.

Figure 3.5: Particle Filter - Block Diagram

Figure 3.6: Bimodal Posterior : Target Motion Scenario
CHAPTER 3. OVERVIEW OF ONLINE SLAM / SFM

Figure 3.7: Bimodal Posterior : Evolution over Time

Figure 3.8: Bimodal Posterior : Distributions
Rao-Blackwellized Particle Filters

If the pose vector $\bar{s}$ consists of 6 degrees of freedom, and there are $N$ 3D features in the reconstruction $X$, then Equation 3.22 is a probability distribution over $6 + 3N$ states, approximated with a finite distribution of $M$ hypotheses for the states. Such a filter will be intractable, as the number of filter dimensions the particles need to span will necessitate an excessively large number of particles.

In practice, methods are used for reducing the number of dimensions spanned by the particles. One such method is ‘Rao-Blackwellization’[52, 18], and takes advantage of the conditional independence of some of the state variables. Rao-Blackwellized particle filters (or RBPFs) allow for efficient tracking of a large set of target features $X$. A key idea from the FastSLAM literature and earlier work is that, conditioned on the target’s kinematic state $\bar{s}$, the features’ position uncertainties are independent of each other[52, 49]. That is:

$$p(\bar{x}_1, \bar{x}_2, \bar{x}_3 | \bar{s}) = p(\bar{x}_1 | \bar{s})p(\bar{x}_2 | \bar{s})p(\bar{x}_3 | \bar{s})$$

$$\bar{x}_1, \bar{x}_2, \bar{x}_3 \in X$$

(3.23)

Since each particle $i$ contains its own hypothesis of the kinematic state $\bar{s}^{[i]}$, it implicitly conditions its estimates of $X$ on $\bar{s}^{[i]}$. Thus it is possible, within a given particle, to track each feature $\bar{x}_j$ independent of the other features.

RBPFs exploit this partitioning of the state-space[52, 18]. Instead of maintaining a very high-dimensional particle filter over the entire state-space, the particles only sample over the pose state $\bar{s}$; the distributions on the feature locations $\bar{x}_j \in X$ are conditional on $\bar{s}$ and maintained analytically. In FastSLAM, the analytic method for maintaining $p(X | \bar{s}^{[i]})$ is via individual EKFs for each feature $j$. This yields a significant computational benefit when the number of features $N$ gets large. Instead of having to maintain a single large $3N \times 3N$ covariance matrix for the feature position estimates, $MN$ $3 \times 3$ covariance matrices (each associated to a specific particle and
feature) are maintained (where $M$ is the number of particles).

\[
p(s_t, X | Z_{0:t}) = \begin{cases} 
\{ w^{[1]}_t, \bar{s}^{[1]}_t, \bar{\mu}^{[1]}_{\bar{x}_1}, \ldots, \bar{\mu}^{[1]}_{\bar{x}_j}, \ldots, \bar{\mu}^{[1]}_{\bar{x}_N}, \Sigma^{[1]}_{\bar{x}_1}, \ldots, \Sigma^{[1]}_{\bar{x}_j}, \ldots, \Sigma^{[1]}_{\bar{x}_N} \}, \\
\ldots, \\
\{ w^{[i]}_t, \bar{s}^{[i]}_t, \bar{\mu}^{[i]}_{\bar{x}_1}, \ldots, \bar{\mu}^{[i]}_{\bar{x}_j}, \ldots, \bar{\mu}^{[i]}_{\bar{x}_N}, \Sigma^{[i]}_{\bar{x}_1}, \ldots, \Sigma^{[i]}_{\bar{x}_j}, \ldots, \Sigma^{[i]}_{\bar{x}_N} \}, \\
\ldots, \\
\{ w^{[M]}_t, \bar{s}^{[M]}_t, \bar{\mu}^{[M]}_{\bar{x}_1}, \ldots, \bar{\mu}^{[M]}_{\bar{x}_j}, \ldots, \bar{\mu}^{[M]}_{\bar{x}_N}, \Sigma^{[M]}_{\bar{x}_1}, \ldots, \Sigma^{[M]}_{\bar{x}_j}, \ldots, \Sigma^{[M]}_{\bar{x}_N} \} 
\end{cases}
\] (3.24)

For each particle, after a pose $s_t^{[i]}$ prediction step, the Rao-Blackwellized particle filter proceeds to estimate the 3D feature map $X$ (parameterized with $N$ position means $\bar{\mu}_{\bar{x}_j}^{[i]}$ and covariances $\Sigma^{[i]}_{\bar{x}_j}$). The particles are weighted $w^{[i]}$ and then resampled, where likely particles (those with relatively larger weights) are maintained and unlikely particles (those with relatively smaller weights) are discarded.

**Noise Issues**

A drawback to nonlinear Bayesian filters is that they operate poorly when noise covariances are large, due to approximations made for handling non-linear models. For example, in an extended Kalman filter formulation, large noise covariance causes slow convergence and increased linearization error. In a particle filter formulation, large noise covariance requires increased numbers of particles, causing non-real-time performance.

### 3.3.2 Nonlinear Optimization

While the goal of Bayesian estimation is a stochastic estimate of the state, an alternative deterministic approach is to formulate the solution for the target’s pose and/or reconstructed shape as a nonlinear optimization problem. The nonlinear optimization approach is often a tool used for solving offline SFM problems, where all the poses are solved for in post-processing after all the measurements have been taken.

Nonlinear optimization can also be used for real-time pose tracking in the context
of online SLAM/SFM, where only the most recent measurements are used to determine the latest pose. Examples of algorithms which use nonlinear optimization for instantaneous pose estimation are [38, 20, 55, 54, 40, 41].

The most common way to perform this optimization is via nonlinear least squares, where a quadratic cost function $u(\bar{s})$ is formed which penalizes deviation from observations according to an error function $e(\bar{s})$. This cost function can be thought of as an inverse representation of the pose likelihood (e.g., minimum cost corresponds to maximum likelihood).

$$u(\bar{s}) = \frac{1}{N} \sum_{j=1}^{N} e(\bar{s})^T e(\bar{s})$$ \hspace{1cm} (3.25)

$$e(\bar{s}) = \bar{z}_{j,t} - g(\bar{s}, \bar{x}_j)$$ \hspace{1cm} (3.26)

Note that the error function is nonlinear, as it involves the nonlinear measurement model.

The best estimate of pose is the argument that minimizes the cost function:

$$\hat{s}_t = \arg \min_{\bar{s}} u(\bar{s})$$ \hspace{1cm} (3.27)

For example, Figure 3.9 graphs the theoretical cost function, that is, the shape of $u(\bar{s})$ if there were no noise in the system. Each plot represents one of the degrees of freedom of camera pose ($\rho_x, \rho_y, \rho_z$ across the top, $\phi, \theta, \psi$ across the bottom); the curve in each plot shows the changing value of the cost function as the hypothesis for that degree of freedom is varied. For each plot, the true pose is indicated with a blue square.

The zero-noise cost function is minimal (with zero error) when the candidate pose is equal to the true pose. The best estimate of this true value is found by driving down the actual cost function curve (i.e., the one that is affected by random measurement noise) to find the argument at the minimum.
Iterative Minimization

In practice, this best estimate is calculated by repeatedly linearizing \( e(\tilde{s}) \) and minimizing the resulting quadratic approximation to \( u(\tilde{s}) \). The argument that minimizes the quadratic approximation to \( u(\tilde{s}) \), is taken as the next reference point for linearizing \( e(\tilde{s}) \). This is done until convergence, i.e., until the change in the latest best estimate \( \hat{s}_t \) falls below some tolerance (i.e., Newton’s method).

Linearizing the error function \( e(\tilde{s}) \) means linearizing the measurement model. The measurement model for a single observation \( j \) is linearized about a reference point \( \tilde{s}_0 \) via the first order Taylor polynomial expansion:

\[
 g(\tilde{s}, \bar{x}_j) \approx g(\tilde{s}_0, \bar{x}_j) + G_j (\tilde{s} - \tilde{s}_0)
\]

\[
 G_j = \left. \frac{\partial g}{\partial \tilde{s}} \right|_j
\]

(3.28)

The matrix \( G_j \) is the Jacobian \( \left. \frac{\partial g}{\partial \tilde{s}} \right|_j \) of the measurement model evaluated at the linearization point.

Expressing the error and cost functions (Equations 3.26 and 3.25) in terms of the
linearized measurement model:

\[
e(\bar{s}) \approx \bar{z}_{j,t} - g(\bar{s}_0, \bar{x}_j) - G_j (\bar{s} - \bar{s}_0)
\]

\[
e(\bar{s}) \approx \bar{b}_j - G_j \bar{s}
\] (3.29)

\[
u(\bar{s}) \approx \frac{1}{N} \sum_{j=1}^{N} (\bar{b}_j - G_j \bar{s})^T (\bar{b}_j - G_j \bar{s})
\]

\[
u(\bar{s}) \approx \frac{1}{N} (\bar{b} - G\bar{s})^T (\bar{b} - G\bar{s})
\]

\[
u(\bar{s}) \approx \frac{1}{N} (\bar{b}^T \bar{b} - 2\bar{s}^T G^T \bar{b} + \bar{s}^T G^T G \bar{s})
\] (3.30)

In Equation 3.29 the known terms are collected into vector \(\bar{b}_j\), and in Equation 3.30 the vectors \(\bar{b}_j\) and matrices \(G_j\) are concatenated into \(\bar{b}\) and \(G\).

The argument that minimizes Equation 3.30 is now solved for by taking the derivative with respect to \(\bar{s}\) and setting it equal to zero.

\[
0 = \frac{\partial \nu}{\partial \bar{s}} \approx \frac{1}{N} \left( -2G^T \bar{b} + 2G^T G \bar{s} \right)
\]

\[
G^T G \bar{s} \approx G^T \bar{b}
\] (3.31)

\[
\hat{\bar{s}}_t \approx (G^T G)^{-1} G^T \bar{b}
\] (3.32)

Effects of Measurement Noise

The iterative minimization of the cost function takes place with measurements corrupted by random noise. Recalling Equation 3.14, this measurement noise is most often assumed to be zero-mean Gaussian, with standard deviation \(\sigma_{meas}\). Under the approximation of linearity for a given iteration of the nonlinear least squares process,
we can map this uncertainty into a Gaussian uncertainty in the pose estimate:

\[
\Sigma_{\hat{s}} = \sigma_{meas}^2 \left( G^T G \right)^{-1} \\
\Sigma_{\hat{s}} = \sigma_{meas}^2 \left( \sum_{j=1}^{N} G_j^T G_j \right)^{-1}
\] (3.33)

Equation 3.33 shows that the uncertainty in pose estimate is inversely proportional to the magnitude of the elements in the Jacobians \( G_j \).

The Jacobian is expanded as follows, where \( \bar{x}_j^{c/cam} \) is the 3D position of a feature in the camera’s reference frame:

\[
G_j = \frac{\partial g}{\partial \bar{x}_j^{c/cam}} \frac{\partial \bar{x}_j^{c/cam}}{\partial \bar{s}}
\] (3.34)

The first term in Equation 3.34 is the Jacobian of the measurement model w.r.t. feature camera frame position \( \frac{\partial g}{\partial \bar{x}_j^{c/cam}} \), calculated via differentiation of Equation 2.5.

\[
\frac{\partial g}{\partial \bar{x}_j^{c/cam}} = \begin{bmatrix}
\frac{f_c}{z_j^{c/cam}} & 0 & \frac{-f_c x_j^{c/cam}}{(z_j^{c/cam})^2} \\
0 & \frac{f_c}{z_j^{c/cam}} & \frac{-f_c y_j^{c/cam}}{(z_j^{c/cam})^2}
\end{bmatrix}
\] (3.35)

The second term in Equation 3.34 is the Jacobian of the feature’s camera frame position w.r.t. the pose \( \frac{\partial \bar{x}_j^{c/cam}}{\partial \bar{s}} \), calculated via differentiation of Equation 3.2 (for a static target SLAM/SFM problem). Assuming a nominal configuration where the Euler angles are approximately zero, the Jacobian is:

\[
\frac{\partial \bar{x}_j^{c/cam}}{\partial \bar{s}} = \begin{bmatrix}
y_j^{c/cam} & -z_j^{c/cam} & 0 & -1 & 0 & 0 \\
-x_j^{c/cam} & 0 & z_j^{c/cam} & 0 & -1 & 0 \\
0 & x_j^{c/cam} & -y_j^{c/cam} & 0 & 0 & -1
\end{bmatrix}
\] (3.36)

Note that the individual matrix terms in Equations 3.35 and 3.36 involve components of the vector distance between the features and the camera. A rough conclusion to make is that the greater this distance (i.e., the farther the camera is from the features observed), the larger the magnitude of the elements in the Jacobians \( G_j \) are,
and thus the smaller the uncertainty in pose estimation $\Sigma_s$.

**Initializing the Minimization**

In general, the cost function $u(\bar{s})$ is non-convex, meaning that multiple local minima of the cost function exist. For this reason, the minimization algorithm needs to be ‘smartly’ initialized with a reasonable initial value $\bar{s}_0$ of the pose estimate, to ensure convergence within the correct local minima.

One way to do this is use a closed-form solution such as the 3-point pose algorithm [27] to generate pose hypotheses from random groups of three image features, and then choose the best hypothesis using RANdom SAmple Consensus (RANSAC) [26, 54]. This best hypothesis is then taken as the starting point $\bar{s}_0$ for nonlinear optimization to find the optimal pose (in this context, it is sometimes referred to as ‘iterative refinement’).

An alternative method of warm-starting the nonlinear optimization is to use a pose prediction calculated from the motion model $f(\cdot)$ and the previous relative pose $\bar{s}_{t-1}$.

The nonlinear optimization approach to real-time pose tracking is also sometimes referred to as an ‘SFM-based’ system, as it uses algorithms derived from the full, offline, batch SFM community (Section 3.1), simplified to run in a recursive, online manner. As such, it also relates to the use of multi-view ‘Bundle Adjustment’.

**Connection to Bundle Adjustment**

Bundle Adjustment generally refers to the case of using nonlinear optimization for batch SLAM/SFM, to correct a set of prior estimates of pose and the map. Papers on Bundle Adjustment are numerous [70, 28, 20, 67, 51, 29, 39, 42]. “Smoothing” is also used as a synonym for Bundle Adjustment [34], to connotate that “… rather than filtering, [the goal is] the maximum a posteriori (MAP) estimate for the entire trajectory … and the map of landmarks …” [33]
Multi-view Bundle Adjustment is solved by formulating the cost function:

\[
\begin{align*}
    u_{\text{batch}}(\vec{s}_{0:t}, X) &= \frac{1}{N(t + 1)} \sum_{\tau=0}^{t} \sum_{j=1}^{N} e_{j,\tau}(\vec{s}_{0:t}, X)^T e_{j,\tau}(\vec{s}_{0:t}, X) \\
    e_{j,\tau}(\vec{s}_{0:t}, X) &= \vec{z}_{j,\tau} - g(\vec{s}_{\tau}, \vec{x}_{j})
\end{align*}
\] (3.37)

This is a non-convex cost function, and the prior estimate of the pose and map is used at initialization, to ensure convergence within the correct local minimum of the cost function. The best estimate of all of the poses \(\vec{s}_{0:t}\) and the map \(X\) is the argument that minimizes the cost function:

\[
\hat{s}_{0:t}, \hat{X} = \underset{\vec{s}_{0:t}, X}{\text{argmin}} \ u_{\text{batch}}(\vec{s}_{0:t}, X)
\] (3.38)

In this context, the process of using nonlinear optimization for real-time, frame-by-frame pose estimation (i.e., as done in Equations 3.25, 3.26, and 3.27) can be considered a simplification of multi-view Bundle Adjustment, where only the single pose at the latest time-step (using only the latest measurements) is estimated. For this reason, sometimes frame-by-frame (i.e., instantaneous) pose estimation is referred to as single-view motion-only Bundle Adjustment (to differentiate it from the multi-view, motion-and-structure case of Equations 3.37 and 3.38).

Apart from the simplified single-view motion-only case, the full multi-view Bundle Adjustment (i.e., in the form of Equations 3.37 and 3.38) is also sometimes utilized during a nonlinear optimization-based online SLAM/SFM process, as a periodic, asynchronous update step to improve accuracy (see Chapter 7 for more).

### 3.4 Summary

This chapter explained the background of the static target, moving camera SLAM/SFM problem. The physical models that can be used during SLAM/SFM estimation, namely the measurement model and the motion model, were presented. Finally, the two dominant approaches to SLAM/SFM, Bayesian filtering and nonlinear optimization, were discussed in detail, in the context of static target SLAM/SFM.
Chapter 4

Online SLAM/SFM for Moving Targets

This chapter explores the application of existing SLAM/SFM algorithms (described in the last chapter) to the problem of real-time tracking and reconstruction of an unpredictable moving target. The existing algorithms, including those that don’t use a motion model (i.e., rely on a nonlinear optimization based only on the measurements) and those that do (e.g., Bayesian recursive methods), are shown to have drawbacks when applied to the moving target SLAM/SFM problem. As a result, a new algorithm is required (the topic of Chapter 5).

Two important facts are explained in this chapter. The first is that existing optimization approaches which don’t use a motion model can face large uncertainty and ambiguity/bimodality in the pose likelihood (explained in Section 4.1); this motivates making use of a motion model. The second is that a real-time approach using a motion model to predict 6DOF pose can fail, due to computational intractability caused by large uncertainty in parts of the motion model (explained in Section 4.2).

Moving Target, Static Camera

In contrast to the previous chapter, in this chapter it is the target’s (not the camera’s) non-uniform motion which is inducing the uncertainty in the relative pose between camera and target (Figure 4.1).
Figure 4.1: SLAM/SFM: Target with Unknown Motion

Recall the mapping of positions from the map frame to the camera frame (Equation 2.6), restated here:

\[
\bar{x}_{C\text{/cam}}^j = R_{C\text{/I}} \left( R_{I\text{/map}} \bar{x}_{I\text{/map}}^j + \bar{\rho}_{I\text{/map}} - \bar{\rho}_{I\text{/cam}} \right)
\]  

(4.1)

Equation 4.1 can now be simplified as follows:

\[
\bar{x}_{C\text{/cam}}^j = R_{C\text{/M}} \left( \bar{x}_{M\text{/map}}^j \right) + \bar{\rho}_{C\text{/map}}
\]  

(4.2)

In Equation 4.2, the camera frame (C) has been taken to be identical to the inertial frame (I), with the camera origin (cam) collocated with the reference origin (o), so that \( \bar{\rho}_{I\text{/o}} = \bar{\rho}_{I\text{/map}} \) and \( R_{I\text{/M}} \equiv R_{C\text{/M}} \) (expanded in Equation 4.3).

\[
R_{C\text{/M}} = \begin{bmatrix}
c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\
s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\
-s\theta & c\theta s\phi & c\theta c\phi
\end{bmatrix}_{C\text{/M}}
\]  

(4.3)

The relative pose vector \( \bar{s} \) consists of the position of the map relative to the reference/camera origin expressed in the inertial/camera frame (\( \bar{\rho}_{I\text{/o}} \), or \( \bar{\rho}_{I\text{/map}} \) for simplicity) and the orientation Euler angles (\( \bar{\theta}_{map} = [\psi \theta \phi]^T \)) expressing the rotation
from the map frame to the inertial/camera frame (i.e., forming $R_i^m$).

$$\bar{s}_t = \begin{bmatrix} \bar{\theta}_{\text{map}} \\ \bar{\rho}_{\text{map}} \end{bmatrix}_t$$  \hspace{1cm} (4.4)

In this chapter, it is this target pose vector $\bar{s}_t$, along with a reconstruction of the target $X$, which is to be estimated at each time-step, based primarily on incoming camera measurements $Z_t$. The next two sections describe different approaches to performing this online estimation.

Note the contrast between Equations 4.2-4.4 (for the moving target case) and Equations 3.2-3.4 (for the moving camera case).

### 4.1 No Motion Model Approaches

As explained in Chapter 3, in Sections 3.2 and 3.3.2, there are online SLAM/SFM algorithms that use only the information from the camera measurements $Z_t$. Using $Z_t$, the target pose $\bar{s}_t$ is determined by nonlinear optimization.

Recalling Section 3.3.2 (Equations 3.25, 3.26, and 3.27), a cost function $u(\bar{s})$ is formed, which reflects the error between actual measurements $Z_t$ and predicted measurements $g(\bar{s}, X)$, for a candidate pose $\bar{s}$. It is assumed that a (partially built) map $X$ is available.

$$u(\bar{s}) = \frac{1}{N} \sum_{j=1}^{N} e(\bar{s})^T e(\bar{s})$$  \hspace{1cm} (4.5)

$$e(\bar{s}) = \bar{z}_{j,t} - g(\bar{s}, \bar{x}_j)$$  \hspace{1cm} (4.6)

The best estimate for pose at time-step $t$, $\hat{s}_t$, is the argument that minimizes this cost function, i.e., the target pose that causes the best alignment to the observations.

$$\hat{s}_t = \arg\min_{\bar{s}} u(\bar{s})$$  \hspace{1cm} (4.7)
This minimization is an iterative process, performed by linearizing the measurement model (i.e., replacing it with its first order Taylor polynomial approximation), calculating the best pose via linear least squares, and then repeating. Converging on the true pose is impeded by measurement noise, caused by pixelization and other effects.

Recalling Equations 3.33 and 3.34, a measure of the certainty in the best estimate at the current iteration can be found in the pose covariance \( \Sigma_s \), calculated via the Jacobians \( G_j \) of the measurement model (restated below).

\[
\Sigma_s = \sigma_{\text{meas}}^2 \left( \sum_{j=1}^{N} G_j^T G_j \right)^{-1}
\]  

(4.8)

\[
G_j = \frac{\partial \bar{x}}{\partial s} \frac{\partial \bar{x}_j^{\text{C}}}{\partial \bar{s}}
\]  

(4.9)

In Chapter 3, when discussing the static target SLAM/SFM problem, the Jacobian \( \frac{\partial \bar{x}_j^{\text{C}}}{\partial \bar{s}} \) was calculated via differentiation of Equation 3.2. Now, in the context of the moving target SLAM/SFM problem, this Jacobian is now calculated via differentiation of Equation 4.2. Assuming a nominal configuration where the Euler angles are approximately zero, the Jacobian is:

\[
\frac{\partial \bar{x}_j^{\text{C}}}{\partial \bar{s}} = \begin{bmatrix} 
-y_j^{\text{c}} + \rho_y & z_j^{\text{c}} - \rho_z & 0 & 1 & 0 & 0 \\
-x_j^{\text{c}} - \rho_x & 0 & -z_j^{\text{c}} + \rho_z & 0 & 1 & 0 \\
0 & -x_j^{\text{c}} + \rho_x & y_j^{\text{c}} - \rho_y & 0 & 0 & 1 
\end{bmatrix}
\]  

(4.10)

4.1.1 Uncertainty in Pose Estimate

The difference between Equations 3.36 and 4.10 reflects the difference between mapping camera pose (as defined in Equation 3.4) to feature position in camera frame and mapping target pose (as defined in Equation 4.4) to feature position in camera frame.

The vector \( \bar{\rho} \) is the position of the target’s reference point in the camera’s reference frame. Note that some of the individual matrix terms in Equation 4.10 involve
differences between $\bar{\rho}$ and the positions of the features in the camera’s reference frame, $\bar{x}_{j/cam}$. In the extreme case, if the separation between camera and target gets very large:

$$\bar{x}_{j/cam} \approx \bar{\rho} \quad (4.11)$$

$$\frac{\partial \bar{x}_{j/cam}}{\partial \bar{s}} \approx \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \ 0 & 0 & 0 & 0 & 1 \ 0 & 0 & 0 & 0 & 1 \ \end{bmatrix} \quad (4.12)$$

A rough conclusion to make is that the greater the separation between camera and target (i.e., the farther the camera is from the features observed), the smaller the magnitude of the elements in the Jacobians $G_j$ are, and thus the larger the uncertainty in pose estimation $\Sigma_s$.

Another way to look at this uncertainty is via nonlinear plots of the cost function $u(\bar{s})$ versus candidate pose. Consider an example target observed from a camera, in the configuration as shown in Figure 4.2. The target is the blue wireframe located at upper left. The 3D feature locations ($X$) are the blue dots on the front face of the target. The camera is at lower right, with the image plane shown as a green square. The 2D projected image features ($Z$) are shown as red dots on the image plane.

Figures 4.3, 4.4, 4.5, 4.6, 4.7, and 4.8 each show motion in a particular degree of freedom of the target's pose (respectively, $\rho_x$, $\rho_y$, $\rho_z$, $\phi$, $\theta$, and $\psi$), and the resulting shape of the cost function $u(\cdot)$ in that degree of freedom (the black line) with no measurement noise. The units of the y-axis are focal length-normalized pixels.

For each plot, the true pose is indicated with a blue square. The zero-noise cost function is minimal (with zero RMS error) when the candidate pose is equal to the true pose. The red dotted line is the amount of error due to pixelization effects. This line represents the limit of knowledge in determining the pose; below this line it is impossible to separate signal from noise, so minimization below this line is subject to random error. Therefore, the range between the intersections of the zero-noise cost function and the noise line represent the range of likely values of a particular degree
A reference value of 0.001 focal length-normalized pixels was chosen for the error limit (the red dotted line), based on an assumed quantization error of half a pixel width (0.5 unnormalized, standard pixels) and a camera focal length of 500 standard pixels. This focal length value is representative of the two cameras used in the experiments of Chapter 6, which have focal lengths of 405 and 610 pixels, respectively.

As can be observed in Figures 4.3 through 4.8, the certainty with which pose can be estimated differs by degree of freedom. Certain degrees of freedom (specifically, $\rho_x$, $\rho_y$, and $\psi$) have zero-noise cost functions which are narrow and steep, meaning that the pose estimate on these degrees of freedom has greater certainty (the range between intersections of the noise line is small).

On the other hand, certain degrees of freedom (specifically, $\phi$, $\theta$, and $\rho_z$) have zero-noise cost functions which are wide with less slope, meaning that the pose estimate on these degrees of freedom has less certainty (the range between intersections of the noise line is large).

For example, examining Figure 4.6, the zero-noise cost function when only considering motion in the $\phi$ degree of freedom is less than the noise level between $-0.293$ radians and $0.325$ radians. In the face of the random noise of the camera sensor, the
CHAPTER 4. ONLINE SLAM/SFM FOR MOVING TARGETS

Figure 4.3: Estimating Target Pose: $\rho_x$

Figure 4.4: Estimating Target Pose: $\rho_y$
CHAPTER 4. ONLINE SLAM/SFM FOR MOVING TARGETS

Figure 4.5: Estimating Target Pose : $\rho_z$

Figure 4.6: Estimating Target Pose : $\phi$
CHAPTER 4. ONLINE SLAM/SFM FOR MOVING TARGETS

Figure 4.7: Estimating Target Pose: $\theta$

Figure 4.8: Estimating Target Pose: $\psi$
best estimate of $\phi_t$ can conceivably fall anywhere over a range of .618 radians, or approximately 35 degrees. This represents a large amount of uncertainty for this degree of freedom of orientation.

In summary, using only information from the camera’s measurements of feature projections, estimates of the target pose can lack confidence and vary over a wide range of values on some degrees of freedom. Estimating rotation is more problematic than estimating translation, as two of three degrees of freedom ($\phi$ and $\theta$) tend to be uncertain.

### 4.1.2 Ambiguity in Orientation Estimate

Another problem with estimating target rotation by nonlinear optimization is the fact that the $\phi$ and $\theta$ degrees of freedom can have non-convex cost functions; that is, there are multiple local minima, and a minimization can converge to an incorrect value of pose (Figure 4.9).

The existence of the false local minima represents an ambiguity in rotational direction; that is, the rotation of the target in $\theta$ from 0 radians to 0.2 radians and the rotation in $\theta$ from 0 radians to -0.2 radians looks almost the same, from the perspective of where the features project onto the image plane. So an estimation process based only on the best fit alignment of features on the image plane can be ‘confused’.

This issue can be addressed by taking into account previous information of the target’s pose. For example, if it was known that the pose is changing in a positive sense on the $\theta$ degree of freedom and that $\theta_{t-1}$ was 0 radians, than the true pose at this time-step ($\theta_t = 0.2$ rad) could be properly disambiguated from the alternative hypothesis at this time-step ($\theta_t = -0.2$ rad). This motivates the need to include motion model information in the estimation process.
4.2 Motion Model Approaches

A large set of existing SLAM/SFM techniques incorporate motion models (as discussed in Chapter 3, Section 3.2). As this thesis focuses on the situation where the non-communicating target, not the exploration robot, is the object moving with uncertainty, it is assumed that no sensor information pertaining to the relative motion is available to help ‘guide’ the motion model. Therefore, to use a motion model means to use a vision-only motion model (as in Equation 3.9).
 CHAPTER 4. ONLINE SLAM/SFM FOR MOVING TARGETS

The motion model for a rigid body is:

\[
\begin{align*}
\bar{\theta}_t &= \bar{\theta}_{t-1} + \Delta t \cdot M(\bar{\theta}_{t-1})\bar{\omega}_{t-1} \\
\bar{\omega}_t &= \bar{\omega}_{t-1} + \Delta t \cdot J^{-1}T_{total} \\
\bar{p}_t &= \bar{p}_{t-1} + \Delta t \cdot \bar{v}_{t-1} \\
\bar{v}_t &= \bar{v}_{t-1} + \Delta t \cdot \left(\frac{1}{m}F_{total} + J^{-1}T_{total} \times \bar{r} + \bar{\omega}_t \times \bar{\omega}_t \times \bar{r}\right)
\end{align*}
\]

(4.13)

In Equation 4.13, a matrix function \(M(\bar{\theta})\) of the Euler angles maps angular rates \(\bar{\omega}\) to Euler angle derivatives \(\dot{\bar{\theta}}\) [10]. Also, \(J\) is the inertia matrix, \(m\) is the mass, \(T_{total}\) is the total external torque, and \(F_{total}\) is the total external force.

Since the external torquing/forcing terms \(T_{total}\) and \(F_{total}\) and mass properties \(J\) and \(m\) are not known, Equation 4.13 can be written as a set of kinematic relationships driven by noise (\(\bar{\nu}_\omega\) and \(\bar{\nu}_v\)):

\[
\begin{align*}
\bar{\theta}_t &= \bar{\theta}_{t-1} + \Delta t \cdot M(\bar{\theta}_{t-1})\bar{\omega}_{t-1} \\
\bar{\omega}_t &= \bar{\omega}_{t-1} + \bar{\nu}_\omega \\
\bar{p}_t &= \bar{p}_{t-1} + \Delta t \cdot \bar{v}_{t-1} \\
\bar{v}_t &= \bar{v}_{t-1} + \Delta t \cdot (\bar{\omega}_t \times \bar{\omega}_t \times \bar{r}(t)) + \bar{\nu}_v
\end{align*}
\]

(4.14) (4.15)

In Equations 4.14 and 4.15, \(\bar{\nu}_\omega\) and \(\bar{\nu}_v\) are zero-mean, Gaussian random forcing terms (drawn from \(\mathcal{N}(0, \Sigma_\omega)\) and \(\mathcal{N}(0, \Sigma_v)\), respectively). Covariances \(\Sigma_\omega\) and \(\Sigma_v\) are chosen to be large enough to encompass the full range of feasible accelerations. The term \(\bar{r}(t)\) is the positional offset of the reference point from the physical center of rotation.

In general the vector \(\bar{r}(t)\), which locates the center of rotation of the target, is not known and changes with time, as forces and moments acting on the rigid body change over time. Before the motion model can be implemented in a real-time pose estimator, the resulting large unknown term in the translational part of the model
(\dot{\omega}_t \times \ddot{\omega}_t \times \ddot{r}(t)) must be addressed.

A potential solution is to increase the translational uncertainty, to account for both zero-mean random accelerations caused by random forces and moments and the effect of not knowing \( \ddot{r}(t) \). In this case, Equation 4.15 is modelled as:

\[
\begin{align*}
\bar{\rho}_t &= \bar{\rho}_{t-1} + \Delta t \cdot \bar{v}_{t-1} \\
\bar{v}_t &= \bar{v}_{t-1} + \bar{\nu}_{\bar{v},Total}
\end{align*}
\]  

(4.16)

In Equation 4.16, the covariance \( \Sigma_{\bar{v},Total} \) associated to \( \bar{\nu}_{\bar{v},Total} \) must be chosen large enough to encompass the full range of feasible accelerations as well as the deterministic effects of the term involving \( \ddot{r}(t) \).

However, the effect of this is to make a Bayesian approach inaccurate and/or intractable for real-time implementation. Even though the covariance associated with the rotational equations of motion (\( \Sigma_{\bar{\omega}} \)) can be kept small, the covariance associated with the translational equations of motion (\( \Sigma_{\bar{v},Total} \)) must be made large. As discussed in Section 3.3.1, this has various detrimental effects, due to the approximations made for nonlinear Bayesian filters. In an extended Kalman filter formulation, large process noise covariance causes slow convergence and increased linearization error. In a particle filter formulation, large process noise requires increased numbers of particles, causing non-real-time performance.

Note that, in the case of more typical SLAM/SFM situations (i.e., moving camera, static target), this issue is typically not a problem since \( \ddot{r}(t) \) is known: it is the constant physical offset on the robotic platform between the camera and the center of mass, which can be measured. For the moving target problem in this thesis, the offset \( \ddot{r}(t) \) is separated physically from the exploration robot (so unmeasurable) and can be changing with time.

As can be seen from the preceding sections, existing methods of solving the online SLAM/SFM problem will encounter substantial difficulties when applied to the moving target case. A new method of performing online SLAM/SFM is needed, to solve the unknown moving target problem accurately and in real-time. This method is presented in the next chapter.
4.3 Summary

This chapter explained the background of moving target SLAM/SFM, the problem under consideration in this thesis. Moving target SLAM/SFM was shown to have issues that could hinder real-time, accurate performance if estimated via the two dominant existing techniques described in Chapter 3 (nonlinear optimization and Bayesian filtering). This motivates the need for a new SLAM/SFM technique capable of addressing the moving target case.
Chapter 5

Hybrid Pose Estimation

As explained in the previous chapter (and shown experimentally in Chapter 6), the existing methods of Chapter 3 are incapable of tracking the relative pose in real-time during moving target SLAM/SFM. Section 4.1 showed that if using only a measurement model, the orientation estimate can be sensitive to noise, and potentially have a multimodal likelihood distribution. Section 4.2 showed that if a motion model is used, the uncertainty in the reference position of the target can be large enough to prevent accurate tracking in real-time.

This chapter presents a novel solution to this dilemma[4]. Section 5.1 outlines the estimation concept, which represents a hybridization of Bayesian filtering and nonlinear optimization. The concept is to separate orientation $\tilde{\theta}_t$ and position $\tilde{\rho}_t$ states, estimate the former based on Bayesian principles involving a motion model, and estimate the latter based on measurement model-only nonlinear optimization principles. Section 5.2 then describes a specific implementation of this hybrid concept, adapted from the existing Bayesian filtering SLAM/SFM algorithm known as ‘FastSLAM’[49, 50]. This algorithm uses Rao-Blackwellized particle filters (described in Chapter 3).
5.1 The Hybrid Concept

Rotation is predicted in a Bayesian filter using the available motion model (Section 5.1.1). Using the rotational motion model preserves a feasible smooth trajectory for relative orientation. Translation is predicted via measurement inversion, to avoid using the part of the motion model with large covariance noise. As is explained in Section 5.1.2, given relative orientation, a translation-only measurement inversion is a fast calculation via minimization of a convex function, i.e., the cost function has only a single local minimum, so there is no risk of convergence to incorrect values of pose. Finally, in the same way as Bayesian filtering, the overall pose estimate \( \bar{s}_t \) is corrected via comparison of the predicted latest measurement \( \hat{Z}_t \) and the actual latest measurement \( Z_t \) (the map estimate \( X \) is also corrected). The process then repeats.

In this way, the hybrid solution yields a real-time capable and physically realistic method for tracking pose. The following sections give the specifics of the rotation and translation estimation processes.

5.1.1 Rotation Estimation

The pose at time-step \( t \) is now defined as:

\[
\bar{s}_t = \begin{bmatrix}
\bar{\theta} \\
\bar{\omega} \\
\bar{\rho}
\end{bmatrix}_t
\]

(5.1)

Rotation is predicted in a Bayesian filter using the available motion model. Recalling Equation 4.14, this is:

\[
\bar{\theta}_t = \bar{\theta}_{t-1} + \Delta t \cdot M(\bar{\theta}_{t-1})\bar{\omega}_{t-1} \\
\bar{\omega}_t = \bar{\omega}_{t-1} + \nu\bar{\omega}
\]

(5.2)

By using a motion model, the filter is ‘guided’ to the correct peak in probability for \( \bar{\theta}_t \), i.e., the one that agrees with the system’s underlying kinematics (avoiding the issues of Section 4.1.2). In contrast to the large uncertainty present in the translation
motion model (explained in the last chapter), the rotational noise term $\bar{\nu}_\omega$ is only due
to zero-mean random angular accelerations.

Equation 5.2 can also be written in functional form as:

$$
\begin{bmatrix}
\bar{\theta} \\
\bar{\omega}
\end{bmatrix}_t = 
\begin{bmatrix}
f_{\bar{\theta}}(\bar{\theta}_{t-1}, \bar{\omega}_{t-1}) \\
f_{\bar{\omega}}(\bar{\theta}_{t-1}, \bar{\omega}_{t-1})
\end{bmatrix}
$$

(5.3)

The prediction of orientation at the upcoming time-step, $\bar{\theta}_t$, is now used as part
of the calculation for predicting translation, described in the next section.

### 5.1.2 Translation Estimation

Translation estimation is based on the fact that, for a given $\bar{\theta}_t$, $X$, and $Z_t$, it is
possible to rearrange the measurement equations so that there is a single close fit when
aligning the map and measurements, i.e., the estimate of $\bar{\rho}_t$ is the maximum likelihood
of a unimodal uncertainty distribution. Specifically, the measurement inversion to
estimate relative position is performed by a fast linear least squares calculation.

An initial estimate of the feature locations in the camera frame, denoted $X_o^C$, is
calculated as follows:

$$
X^C_o = R \left( \bar{\theta}_{t} \right)^C/M X + \bar{\rho}_t e_1_{1 \times N}
$$

$$
X^C_o = 
\begin{bmatrix}
x^C_{1o} \\
y^C_{1o} \\
z^C_{1o} \\
... \\
x^C_{jo} \\
y^C_{jo} \\
z^C_{jo} \\
... \\
x^C_{No} \\
y^C_{No} \\
z^C_{No}
\end{bmatrix}
$$

(5.4)

In Equation 5.4, $e_1_{1 \times N}$ is a $1 \times N$ matrix of all ones.

Recalling Section 2.3, a perspective projection model of the camera is assumed,
so a single feature’s measurement is expressed as:

$$
\begin{bmatrix}
u_j \\
v_j
\end{bmatrix} = \frac{f_c}{z^C_j} \begin{bmatrix} x^C_j \\
y^C_j
\end{bmatrix}
$$

(5.5)

The measurements taken ($u_j$ and $v_j$) are pixel coordinates on the image plane.
Rearranging Equation 5.5 gives:

\[
\begin{bmatrix}
0 \\
0 \\
\end{bmatrix} = \begin{bmatrix}
u_j z_j^C - f_c x_j^C \\
v_j z_j^C - f_c y_j^C \\
\end{bmatrix} \tag{5.6}
\]

Equation 5.6 can be stacked to form one large linear expression involving all the measurements at a given time, \(Z_t\).

\[
\begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
0 \\
\end{bmatrix} = \begin{bmatrix}
u_1(z_1^C + \Delta \rho_z) - f_c(x_1^C + \Delta \rho_x) \\
v_1(z_1^C + \Delta \rho_z) - f_c(y_1^C + \Delta \rho_y) \\
\vdots \\
u_j(z_j^C + \Delta \rho_z) - f_c(x_j^C + \Delta \rho_x) \\
v_j(z_j^C + \Delta \rho_z) - f_c(y_j^C + \Delta \rho_y) \\
\vdots \\
u_N(z_N^C + \Delta \rho_z) - f_c(x_N^C + \Delta \rho_x) \\
v_N(z_N^C + \Delta \rho_z) - f_c(y_N^C + \Delta \rho_y) \\
\end{bmatrix} \tag{5.7}
\]

The initial estimate of feature locations in the camera frame \(X_o^C\) is to be corrected by translational offsets \(\Delta \rho_x\), \(\Delta \rho_y\), and \(\Delta \rho_z\), while satisfying as closely as possible the constraints of Equation 5.7.

Note that because the features are part of a single rigid body, the translational offset is common. Also note that, based on Equation 5.4, \(X_o^C\) is a function of the orientation estimate at the latest time-step \(\bar{\theta}_t\) (as predicted by Equation 4.14), and an initial guess at target position \(\bar{\rho}_t\).
Rewriting Equation 5.7 as a linear equation in the translational offset vector:

\[
\begin{bmatrix}
  f_c x_{1o}^C - u_1 z_{1o}^C \\
  f_c x_{1o}^C - v_1 z_{1o}^C \\
  \vdots \\
  f_c x_{jo}^C - u_j z_{jo}^C \\
  f_c x_{jo}^C - v_j z_{jo}^C \\
  \vdots \\
  f_c x_{N_o}^C - u_N z_{N_o}^C \\
  f_c x_{N_o}^C - v_N z_{N_o}^C 
\end{bmatrix}
= \begin{bmatrix}
  -f_c & 0 & u_1 \\
  0 & -f_c & v_1 \\
  \vdots & \vdots & \vdots \\
  -f_c & 0 & u_j \\
  0 & -f_c & v_j \\
  \vdots & \vdots & \vdots \\
  -f_c & 0 & u_N \\
  0 & -f_c & v_N 
\end{bmatrix}
\begin{bmatrix}
  \Delta \rho_x \\
  \Delta \rho_y \\
  \Delta \rho_z
\end{bmatrix}^C
\]

(5.8)

\[\bar{b} = A \Delta \bar{\rho}^C\]

Translational offset \(\Delta \bar{\rho}^C\) is solved for via Least Squares:

\[\Delta \bar{\rho}_{LS}^C = (A^T A)^{-1} A^T \bar{b}\]  

(5.9)

In practice, a better way to calculate this translational offset is with weighted Least Squares, where the weights reflect the relative confidence of the estimated feature locations.

\[\Delta \bar{\rho}_{WLS}^C = (A^T (\bar{w}^T I \bar{w}) A)^{-1} A^T (\bar{w}^T I \bar{w}) \bar{b}\]  

(5.10)

\[\bar{w} = [w_1 \ w_2 \ \cdots \ w_j \ \cdots \ w_N]^T\]

Good choices for the weights \(w_j\) are either the determinant of each feature’s position covariance, or alternatively the number of time steps that each feature has been observed (this is what is done for the experiments in Chapter 6).

The 3D position shift to the rigid body in the camera frame has now been calculated:

\[\bar{p}_t = \bar{p}_{t_o} + \Delta \bar{\rho}_{WLS}^C\]  

(5.11)
CHAPTER 5. HYBRID POSE ESTIMATION

Because this is a linear optimization problem, the initial guess at the vehicle position this time-step ($\hat{\rho}_t$) is unimportant; the optimization process will yield a value for $\Delta \hat{\rho}_C$ which always adjusts to the same optimal vehicle position $\hat{\rho}_t$.

To summarize, the position estimate of the rigid body reference point $\hat{\rho}_t$ is based on a best fit alignment of the feature positions’ image plane projections to the latest measurements. This inversion of the camera model is represented in functional form by:

$$\hat{\rho}_t = h_{\hat{\rho}}(Z_t, X, \hat{\theta}_t) = g^{-1}_{\mid(X, \hat{\theta}_t)}(Z_t)$$ (5.12)

Note the similarity to Equation 3.6, but with the relative orientation at the current time-step now also assumed to be given.

Translation Estimation as an Optimization Problem

The process for estimating target translation as given in Equations 5.8, 5.9/5.10, and 5.11, represents the minimization of a cost function:

$$\tilde{u}(\hat{\rho}) = \frac{1}{N} \sum_{j=1}^{N} \xi_j^2 \left( \bar{z}_{j,t} - g(\hat{\rho}, \hat{\theta}_t, \bar{x}_j) \right)^T \left( \bar{z}_{j,t} - g(\hat{\rho}, \hat{\theta}_t, \bar{x}_j) \right)$$ (5.13)

$$\hat{\rho}_t = \arg\min_{\hat{\rho}} \tilde{u}(\hat{\rho})$$ (5.14)

Comparing $\tilde{u}(\hat{\rho})$ (Equation 5.13) to the traditional cost function that drives measurement-only optimization $u(\hat{s})$ (Equation 3.25), the former is only a function in 3 degrees of freedom (just position) as opposed to all 6 degrees of freedom (orientation and position) of pose.

A more important difference between $u(\hat{s})$ and $\tilde{u}(\hat{\rho})$ is in the underlying error
measures, respectively, \(e(\bar{s})\) and \(\tilde{e}(\bar{\rho})\).

\[
u(\bar{s}) = \frac{1}{N} \sum_{j=1}^{N} e(\bar{s})^T e(\bar{s})
\]
\[
e(\bar{s}) = \bar{z}_{j,t} - g(\bar{s}, \bar{x}_j) = \begin{bmatrix} u_{j,t} \\ v_{j,t} \end{bmatrix} - \frac{f_c}{z_j^C} \begin{bmatrix} x_j^C \\ y_j^C \end{bmatrix}
\] (5.15)

\[
\tilde{u}(\bar{\rho}) = \frac{1}{N} \sum_{j=1}^{N} \tilde{e}(\bar{\rho})^T \tilde{e}(\bar{\rho})
\]
\[
\tilde{e}(\bar{\rho}) = z_j^C \left( \bar{z}_{j,t} - g(\bar{\rho}, \hat{\theta}_t, \bar{x}_j) \right) = \begin{bmatrix} u_{j,t} z_j^C \\ v_{j,t} z_j^C \end{bmatrix} - \begin{bmatrix} f_c x_j^C \\ f_c y_j^C \end{bmatrix}
\] (5.16)

Note that the error measure has been modified; traditionally it is the 2D pixel error between the predicted and actual measurement of a feature (Equation 5.15), while for the process described in this chapter, it is the product of this 2D projection error with the feature’s depth from the camera \(z_j^C\) (Equation 5.16), i.e., the \(z\)-component of the position vector of feature \(j\) in the camera’s frame.

The benefit of modifying the cost function in this way is that minimization is now solved via linear least squares (that is, the process outlined in Section 5.1.2), with a single calculation, as opposed to via nonlinear least squares, which would require several iterations to converge. The speed up in calculation will be shown to be beneficial in the next section, when a specific implementation of the hybrid estimation scheme using particle filters is described.
5.2 Implementation with Rao-Blackwellized Particle Filters

The previous section described the general approach to hybrid pose tracking. A specific vision-only SLAM/SFM implementation is now given, modified from a Rao-Blackwellized particle filter [53, 68]. Rao-Blackwellized particle filters have advantages to alternative Bayesian forms such as the extended Kalman filter, in that they allow for multimodal probability distributions and can maintain a much larger map of features [49, 16]. They have been previously applied in the vision-only SLAM/SFM problem [3, 19], although with pose tracking done via standard Bayesian estimation. Accurate operation has been achieved at frame-rate (30Hz) with 50 particles [19].

5.2.1 Representation of States

The particle filter consists of many particles (referred to by index $i$), each containing a hypothesis for the kinematic state at the latest time step $t$, $s^{[i]}_t$. As discussed in Chapter 3, Section 3.3.1, within each particle, each feature can be estimated independently of the others. This algorithm duplicates the approach of FastSLAM [49, 50], which has each particle maintain a set of small EKFs, one for each feature. In other words, within each particle $i$, the distribution $p(\bar{x}^M_j | \bar{s}^{[i]}_t)$ is assumed to be Gaussian, $\mathcal{N}(\mu^{[i]}_j, \Sigma^{[i]}_j)$.

To summarize, each particle contains $N$ EKFs; that is, for particle $i$, every feature $j$ has a mean $\bar{\mu}^{[i]}_j$ and covariance $\Sigma^{[i]}_j$ to estimate its true 3D location in the target map frame $(\bar{x}^M_j)$.

5.2.2 Algorithm

This section describes the three steps of hybrid estimation with a Rao-Blackwellized particle filter: ‘Time Update’, ‘Measurement Update’, and ‘Particle Weighting and Resampling’. Figure 5.1 gives the overall flow diagram of the filter, showing how Equations 5.17-5.35 process data to determine an estimate of target pose and shape at each time-step.
Figure 5.1: Hybrid Estimator Flow Diagram
CHAPTER 5. HYBRID POSE ESTIMATION

Time Update

The pose prediction step is run for each particle \(i\) in the filter (containing \(M\) particles in total). Following the hybrid approach of Section 5.1, the pose is predicted using Equations 5.3 and 5.12 as follows:

\[
\hat{s}_i^t = \begin{bmatrix}
  f_\theta(\bar{\theta}_t^{i-1}, \bar{\omega}_t^{i-1}) \\
  f_\omega(\bar{\theta}_t^{i-1}, \bar{\omega}_t^{i-1}) \\
  h_\rho(Z_t, X_t^i, f_\theta(\bar{\theta}_t^{i-1}, \bar{\omega}_t^{i-1}))
\end{bmatrix}
\]  

(5.17)

These predictions are now used for generating a diverse distribution of the likely relative pose. One way to do this would be to perturb each particle’s predicted pose with additive Gaussian process noise. This is the method of FastSLAM 1.0 [49]:

\[
\tilde{s}_i^t = \hat{s}_i^t + \tilde{\nu}
\]  

(5.18)

A better method is to generate a proposal distribution that takes account of the latest measurement \(Z_t\), following the approach of the FastSLAM 2.0 algorithm [50]. FastSLAM 2.0 helps focus the proposal particle cloud towards the highest likelihood regions of the state space. This is the method adopted here, and used for the experiments in Chapter 6.

For each particle, a pose is drawn from a Gaussian proposal distribution:

\[
\tilde{s}_i^t \sim \mathcal{N}(\mu_{prop}^i, \Sigma_{prop}^i)
\]  

(5.19)

The covariance \((\Sigma_{prop}^i)\) and mean \((\mu_{prop}^i)\) of the proposal distribution are:

\[
\Sigma_{prop}^i = \left[ \sum_j G_{s}^i T \left( Q_{j}^i \right)^{-1} G_{s}^i + P^{-1} \right]^{-1}
\]  

(5.20)

\[
\mu_{prop}^i = \Sigma_{prop}^i \sum_j G_{s}^i T \left( Q_{j}^i \right)^{-1} \left( \tilde{z}_j - \hat{z}_j^{[i]} \right) + \hat{s}_i^t
\]  

(5.21)
where $\hat{s}_t[i]$ is determined via Equation 5.17, $P$ is additive noise covariance (empirically chosen), and:

$$\hat{z}_j[i] = g(\hat{s}_t[i], \hat{\mu}_j[i]) \tag{5.22}$$

$$G_{\hat{x}_j} = \left. \frac{\partial g}{\partial \hat{x}_j} \right|_{(\hat{s}_t[i], \hat{\mu}_j[i])} \tag{5.23}$$

$$G_{\hat{s}} = \left. \frac{\partial g}{\partial \hat{s}} \right|_{(\hat{s}_t[i], \hat{\mu}_j[i])} \tag{5.24}$$

$$Q_{j}^{[i]} = R_{prop} + G_{\hat{x}_j} \Sigma_{j}^{[i]} G_{\hat{x}_j}^T \tag{5.25}$$

Once the pose states have been proposed, the filter then advances to the measurement update step.

**Measurement Update**

This process is done for every feature $j$ observed, for each particle $i$ (because each particle maintains its own estimates of the feature locations). To update a given feature’s estimate for a given particle, the measurement model $g(\bar{s}, \bar{x}_j)$ is used to predict the next measurement of that feature, and calculate its Jacobians:

$$\hat{z}_j[i] = g(\hat{s}_t[i], \hat{\mu}_j[i]) \tag{5.26}$$

$$G_{\hat{x}_j} = \left. \frac{\partial g}{\partial \hat{x}_j} \right|_{(\hat{s}_t[i], \hat{\mu}_j[i])} \tag{5.27}$$

$$G_{\hat{s}} = \left. \frac{\partial g}{\partial \hat{s}} \right|_{(\hat{s}_t[i], \hat{\mu}_j[i])} \tag{5.28}$$

This is followed by a standard EKF measurement update, performed on the feature’s mean and covariance (Equations 5.29-5.32). Note that $\sigma_{meas}$ is the noise standard deviation. This parameter is a measure of the importance with which the latest
CHAPTER 5. HYBRID POSE ESTIMATION

measurement is treated.

$$Q = \left( G_s P G_s^T + G_{x_j} \Sigma_{x_j}^{[i]} G_{x_j}^T + \sigma_{\text{meas}}^2 I \right)$$ (5.29)

$$K = \Sigma_{x_j}^{[i]} G_{x_j}^T Q^{-1}$$ (5.30)

$$\bar{\mu}_j^{[i]} = \bar{\mu}_j^{[i]} + K(\bar{x}_j - \hat{\bar{x}}_j^{[i]})$$ (5.31)

$$\Sigma_j^{[i]} = (I - KG_{x_j}) \Sigma_j^{[i]}$$ (5.32)

Particle Weighting and Resampling

At each time step, the particle filter updates an importance weighting for each of the particles. This is a measure of the likelihood of the particle, given the measurements observed up to the given time-step. The importance weighting of particle $i$ is denoted $w_i$. There are many choices available for weighting measures. For example, inlier/outlier counts based on pixel error thresholding have been used in previous SLAM/SFM particle filters[12, 58].

Given the choice made above to model $p(\bar{x}_j|\bar{s}^{[i]})$ as ‘Gaussian-like’ ($\mathcal{N}(\bar{\mu}_j^{[i]}, \Sigma_j^{[i]})$), a natural choice of weighting is the Gaussian likelihood (as is done in the original FastSLAM algorithm[49, 68]). The measurement of feature $j$, $\bar{z}_j$, has likelihood denoted $w_j^{[i]}$, and calculated by:

$$w_j^{[i]} = |2\pi Q|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\bar{z}_j - \hat{\bar{z}}_j^{[i]})^T Q^{-1}(\bar{z}_j - \hat{\bar{z}}_j^{[i]}) \right\}$$ (5.33)

The overall likelihood of a specific particle, $w^{[i]}$, is the cumulative likelihoods of that particle’s measurements:

$$w^{[i]} = \prod_j w_j^{[i]}$$ (5.34)
Chapter 5. Hybrid Pose Estimation

Periodically, the particles are resampled, in order to remove the least likely hypotheses, and focus on the most likely. This algorithm resamples based on a calculation of effective sample size $M_{\text{eff}}$ [18]:

$$M_{\text{eff}} = \frac{1}{\sum_i (w[i])^2}$$  \hspace{1cm} (5.35)

When $M_{\text{eff}}$ is less than half the total number of particles $M$, resampling takes place. In order to resample efficiently, low-variance resampling is used [68]. After resampling, all particles are set to equal weight.

5.2.3 Feature Initialization

Because the measurement is bearings-only, a feature cannot be initialized with just one measurement. Two or more views of a feature are required before its associated EKF can be initialized.

In selecting the number of views to use at initialization, a balance must be struck between competing objectives. From the standpoint of initialization accuracy, it is better to have more views of a feature, since this is equivalent in general to having a wider ‘baseline’. On the other hand, it is desirable to initialize a feature as quickly as possible (i.e., with less views), so that it is mapped and taken into account when weighting the particles. An initialization size of 4 views was empirically chosen for the experiments performed in Chapter 6.

The initialization is performed via least squares minimization. Based on the measurement model, variables are rearranged to obtain a linear equation of the form $\bar{b} = A\bar{x}_j$, where $\bar{x}_j$ is $\bar{x}_{i/\text{map}}^M$, the 3D position of the feature in target body (map) coordinates. Alternatively, a more accurate nonlinear minimization technique could be used (i.e., where the standard cost function based on 2D pixel error is used, as in single-view motion-only Bundle Adjustment [17]). The strength of initialization via least squares is its speed. This trade-off is identical to that made in Section 5.1.2 for calculating target translation via optimization, where the standard pixel error cost function (Equation 5.15) was replaced with a quadratic cost function (Equation 5.16) which can be solved with a linear calculation in one step.
Expanding Equation 2.6, the relationship to transform feature positions into camera coordinates is expressed as:

\[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}_{\text{Cam}}^{C} = 
\begin{bmatrix}
c_{11} & c_{12} & c_{13} \\
c_{21} & c_{22} & c_{23} \\
c_{31} & c_{32} & c_{33}
\end{bmatrix}
\begin{bmatrix}
 r_{M/1} & r_{M/1} & r_{M/1} \\
 r_{M/2} & r_{M/2} & r_{M/2} \\
 r_{M/3} & r_{M/3} & r_{M/3}
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}_{\text{Map}}^{M} +
\begin{bmatrix}
\rho_{x} \\
\rho_{y} \\
\rho_{z}
\end{bmatrix}_{\text{Map/Cam}}^{\text{Cam}}
\]

(5.36)

The above matrix equation can be broken into three scalar equations:

\[
x_{\text{Cam}} = a \cdot x_{\text{Map}} + b \cdot y_{\text{Map}} + c \cdot z_{\text{Map}} + d
\]

(5.37)

\[
y_{\text{Cam}} = e \cdot x_{\text{Map}} + f \cdot y_{\text{Map}} + g \cdot z_{\text{Map}} + h
\]

(5.38)

\[
z_{\text{Cam}} = l \cdot x_{\text{Map}} + m \cdot y_{\text{Map}} + n \cdot z_{\text{Map}} + o
\]

(5.39)
The coefficients in Equations 5.37-5.39 are:

\[ a = r_{11}^{c} r_{11}^{m} + r_{12}^{c} r_{12}^{m} + r_{13}^{c} r_{13}^{m} \]
\[ b = r_{11}^{c} r_{21}^{m} + r_{12}^{c} r_{22}^{m} + r_{13}^{c} r_{23}^{m} \]
\[ c = r_{11}^{c} r_{31}^{m} + r_{12}^{c} r_{32}^{m} + r_{13}^{c} r_{33}^{m} \]
\[ d = r_{11}^{c} \rho_x + r_{12}^{c} \rho_y + r_{13}^{c} \rho_z \]
\[ e = r_{21}^{c} r_{11}^{m} + r_{22}^{c} r_{12}^{m} + r_{23}^{c} r_{13}^{m} \]
\[ f = r_{21}^{c} r_{21}^{m} + r_{22}^{c} r_{22}^{m} + r_{23}^{c} r_{23}^{m} \]
\[ g = r_{21}^{c} r_{31}^{m} + r_{22}^{c} r_{32}^{m} + r_{23}^{c} r_{33}^{m} \]
\[ h = r_{21}^{c} \rho_x + r_{22}^{c} \rho_y + r_{23}^{c} \rho_z \]
\[ l = r_{31}^{c} r_{11}^{m} + r_{32}^{c} r_{12}^{m} + r_{33}^{c} r_{13}^{m} \]
\[ m = r_{31}^{c} r_{21}^{m} + r_{32}^{c} r_{22}^{m} + r_{33}^{c} r_{23}^{m} \]
\[ n = r_{31}^{c} r_{31}^{m} + r_{32}^{c} r_{32}^{m} + r_{33}^{c} r_{33}^{m} \]
\[ o = r_{31}^{c} \rho_x + r_{32}^{c} \rho_y + r_{33}^{c} \rho_z \]

Equations 5.37, 5.38, and 5.39 are substituted into the measurement model (Equation 2.5), and rearranged to express it as a linear equation on the feature’s body coordinates:

\[
\begin{bmatrix}
  u_j * o - f_c * d \\
  v_j * o - f_c * h
\end{bmatrix}_t = \begin{bmatrix}
  f_c * a - u_j * l & f_c * b - u_j * m & f_c * c - u_j * n \\
  f_c * e - v_j * l & f_c * f - v_j * m & f_c * g - v_j * n
\end{bmatrix}_t \bar{x}_j
\]
\[ \bar{b}_t = A_t \bar{x}_j \]  

(5.41)

In Equation 5.41, the index \( t \) indicates the terms are based on measurement and vehicle data at a specific time. Taking data at multiple time steps, the matrices are
CHAPTER 5. HYBRID POSE ESTIMATION

stacked:

\[
\begin{bmatrix}
\tilde{b}_{t_1} \\
\vdots \\
\tilde{b}_{t_4}
\end{bmatrix} =
\begin{bmatrix}
A_{t_1} \\
\vdots \\
A_{t_4}
\end{bmatrix}
\begin{bmatrix}
\bar{x}_{j}
\end{bmatrix}
\tag{5.42}
\]

Finally, least squares estimation returns the estimate of feature position:

\[
\hat{x}_{j} = (A^T A)^{-1} A^T \tilde{b}
\tag{5.43}
\]

This estimate is used to initialize the mean of the EKF, \( \mu_{j}^{[i]} \).

To initialize the covariance of the feature’s location (\( \Sigma_{j}^{[i]} \)), the Jacobian matrices (\( G_t \)) are determined by evaluating Equation 5.23 at \( \mu_{j}^{[i]} \) for each of the poses \( \bar{s}_{t}^{[i]} \) for which a measurement was recorded. The covariance is then initialized as given in Equation 5.44 below.

\[
\Sigma_{j}^{[i]} = \sigma_{\text{meas}}^2 \left( \begin{bmatrix}
G_{t_1} \\
\vdots \\
G_{t_4}
\end{bmatrix} \right)^{-1}
\tag{5.44}
\]

5.3 Summary

This chapter presented a new, hybrid technique for SLAM/SFM, which can track and reconstruct moving targets accurately and is real-time capable. The broad concept, to estimate rotational pose via Bayesian filtering and translational pose via optimization, was explained. A Rao-Blackwellized particle filter implementation of this hybrid estimator was described in detail. Figure 5.1 shows the computational flow of this estimator, and the partitioning of estimation of orientation (via Bayesian filtering, through the ‘Rotational Motion Model’ block) and position (via optimization of a cost function, through the ‘Translation Optimization’ block).
Chapter 6

Simulations and Experiments

Three sets of simulations and experiments were conceived to test the performance of the new hybrid algorithm for moving target SLAM/SFM: simulations with a synthetic target and camera, laboratory experiments, and field experiments.

The simulations were performed (6.1) to compare the estimated pose and shape against truth. Additionally, simulations were used to verify the decrease in computational cost attained by switching from a standard Bayesian estimation process (as in Chapter 4, Section 4.2) to the hybrid estimation process explained in Chapter 5. The simulation results show that only by switching to a hybrid estimation process can the target be tracked and reconstructed accurately in real-time today. Furthermore, simulation results show that even a standard Bayesian estimator with a tenfold increase in computational capability still does not match the performance of the original hybrid estimator.

Laboratory experiments were performed (6.2) using a camera and target. These expanded on the simulations by introducing the complexities of vision processing (as explained in Chapter 2). The hybrid algorithm’s performance was tested, with feature tracking now dependent on detecting and matching SIFTs successfully with real measurement noise. Target pose estimates were compared against an independent set of measurements obtained via an infrared marker-based system, and the target shape estimate was compared against a wireframe model of the true shape. These tests show that the algorithm can operate successfully with vision processing of actual...
images operating as the front-end to the hybrid estimator.

Finally, field experiments (6.3) were performed in Monterey Bay, California, using a Remotely Operated Vehicle (ROV) and underwater target in conjunction with the Monterey Bay Aquarium Research Institute (MBARI). These tests involved difficult lighting conditions underwater, as well as highly uncertain motion of the target due to ocean currents and tether forces. The target shape estimate was compared against a wireframe model of the true shape. While no truth signal or independent estimate was available for comparison against the hybrid algorithm’s estimate of target pose, the target’s pose trajectory was analyzed qualitatively, based on ROV pilot observations. Results demonstrate the ability of the hybrid algorithm to track a real underwater target undergoing random forces and torques.

### 6.1 Simulations

Two SLAM/SFM algorithms are compared in simulation: a standard Bayes filter like those in Chapter 3, Section 3.3.1 (specifically, a Rao-Blackwellized particle filter[50, 19, 3, 2]), and the hybrid algorithm of Chapter 5, Section 5.2 (a modified Rao-Blackwellized particle filter [4]).

These algorithms differ in how the pose estimation process is performed, but have the same computationally complexity: $O(MN_{\text{obs}})$, where $M$ is the number of particles, and $N_{\text{obs}}$ is the number of features observed at a given time-step. Therefore, the relative computational burden can be compared via the number of particles that each algorithm requires to estimate accurately the pose and shape of the synthetic target.

As mentioned in Chapter 5, Section 5.2, previous researchers[19] have found that Rao-Blackwellized particle filters are capable of running in real-time with 50 particles. Reflecting this finding, both the hybrid filter and the standard Bayes filter were run with 50 particles.

A synthetic target was generated, specifically as a set of 100 reference features arranged to sit on four sides of a cube with side length 2 meters. The synthetic target is shown in Figure 6.1.
Relative to the synthetic camera, the target is placed 3 meters away, along the camera’s boresight. The target exhibits 1DOF rotation, at a rate of \(0.01\pi\) radians per time-step, around a rotation axis perpendicular to the camera’s boresight. This is camera-relative motion in one of the degrees of freedom of \(\bar{s}\) that is difficult to determine via a measurement-only approach, due to low-confidence and potential ambiguity in estimation (see Chapter 4, Section 4.1).

### 6.1.1 Simulation Results

Results for pose tracking are shown in Figures 6.2 and 6.3 for the new hybrid estimator and the standard Bayes filter, respectively. Each pair of plots shows 10 independent runs of the filters (in green), with the true pose in black. Note that the pose in Figures 6.2 and 6.3 is expressed in the camera frame, with the z-axis in the direction of the camera’s boresight.

Examining these plots, it is observed that the hybrid estimator tracks orientation and position successfully, while the standard Bayesian estimator fails at tracking the target.

Results for target reconstruction \(\mathbf{X}\) are compared in Figure 6.4. The plots show the RMS error in the feature locations, for 10 different runs of the hybrid estimator.
CHAPTER 6. SIMULATIONS AND EXPERIMENTS

79

(a) Hybrid Estimator (50 particles)  
(b) Bayes Filter (50 particles)

Figure 6.2: Simulations: Orientation Tracking

(a) Hybrid Estimator (50 particles)  
(b) Bayes Filter (50 particles)

Figure 6.3: Simulations: Position Tracking
CHAPTER 6. SIMULATIONS AND EXPERIMENTS

(a) Hybrid Estimator (50 particles)  
(b) Bayes Filter (50 particles)

Figure 6.4: Simulations : RMS Reconstruction Error

The plots show the results of simulations for different estimators. The hybrid estimator (50 particles) and the Bayes filter (50 particles) are compared. The left plot (hybrid estimator) shows reconstructions converging to an accuracy on the order of 0.05 m RMS error. The right plot (Bayes filter) fails to reconstruct accurately the synthetic target’s shape.

A metric for comparison with later tests is to normalize this RMS error with the average depth of the target from the camera:

$$e_{\text{normalized}} = \frac{e_{\text{RMS}}}{z_{\text{avg}}} \approx \frac{0.05m}{3m} = 0.0167$$  \hspace{1cm} (6.1)

In contrast to the hybrid estimator, the standard Bayes filter fails to reconstruct accurately the synthetic target’s shape.

For a given particle filter, expected accuracy is proportional to the number of particles. The results in Figures 6.2-6.4 are preliminary indications that the standard Bayes filter is incapable of tracking the pose of the target, or reconstructing its shape, without requiring a number of particles that it is too large to run in real-time.

To test this hypothesis, an additional estimator was also run on the synthetic data: the standard Bayesian particle filter running with 500 particles. This factor of ten increase in the number of particles pushes the estimator out of the realm of real-time operation. The accuracy of this Bayes filter versus the same 50-particle hybrid filter was again compared (Figures 6.5, 6.6, and 6.7).

As expected, the Bayes filter with 500 particles tracks and reconstructs the target with greater accuracy than the Bayes filter with 50 particles. However, Figures 6.5-6.7
CHAPTER 6. SIMULATIONS AND EXPERIMENTS

Figure 6.5: Simulations: Orientation Tracking

(a) Hybrid Estimator (50 particles)  (b) Bayes Filter (500 particles)

Figure 6.6: Simulations: Position Tracking

(a) Hybrid Estimator (50 particles)  (b) Bayes Filter (500 particles)
CHAPTER 6. SIMULATIONS AND EXPERIMENTS

Figure 6.7: Simulations: RMS Reconstruction Error

show that the Bayes filter implementation would require still more particles than 500 to estimate the pose as accurately and consistently as the hybrid estimator running with only 50 particles.

From these plots, it can be concluded that hybrid estimation is more accurate (in both pose and shape) and less computationally burdensome (i.e., less particles are needed to operate with accuracy) than standard Bayesian filtering for the tumbling target problem. Furthermore, only the hybrid algorithm is capable of tracking/reconstructing the moving target with a number of particles that allows for real-time operation.

6.2 Laboratory Experiments

To test the efficacy of the hybrid algorithm under actual vision sensing, physical experiments were conducted in the controlled environment of the Aerospace Robotics Lab at Stanford. The previous section demonstrated the theoretical ability of the hybrid algorithm to reconstruct and track a target in real-time, where measurement correspondence was perfect. However it was important to test the abilities of the algorithm given (noisy) measurements $Z_t$ detected from SIFT tracking on a real image stream, as described in Chapter 2.
Camera sensing can involve low illumination, high illumination, shadow, glint, and/or glare. These will cause the image processing front-end to detect/match too few SIFTs or spurious SIFTs (i.e., features which aren’t located on the target body, or are not stationary on the body). Furthermore, the quantization of the camera’s inputs (i.e., the binning of incoming light into discrete pixels or ‘pixelization’) is a source of noise in the measurements $Z_t$. A successful vision-based estimation algorithm must overcome these issues.

These lab experiments were meant to involve similar target motion to the simulations, and verify that the hybrid algorithm’s performance is not seriously degraded when using measurements from a real sensor. The target is located at an average distance of 60 cm from the camera. The target exhibits 3DOF rotation, with the primary rotation being along an axis perpendicular to the camera boresight, at a rate of $0.0083\pi$ radians per time-step ($0.25\pi$ radians per second, with the camera feed running at 30Hz). As with the simulations, this is camera-relative motion in one of the degrees of freedom of $\bar{s}$ that is difficult to determine via a measurement-only approach, due to low-confidence and potential ambiguity in estimation (Chapter 4, Section 4.1).

### 6.2.1 Laboratory Hardware

The Aerospace Robotics Lab maintains a 6DOF robotic gantry for performing robotic experiments. A calibrated[8] Sony XC-999 camera was attached to the end-effector of the gantry. These pieces of hardware, shown in Figure 6.8, represented the exploration robot and its camera sensor. Digital images were acquired by feeding the image stream to a computer equipped with a Matrox Morphis framegrabber.

A cube-shaped cardboard box was used as the target (see Figure 6.9). Such a simple shape made it easier to verify whether a reconstruction is accurate or not, as a 3D plot of the estimated feature locations on the target should resemble a point cloud representation of a cube. The side length of the target is 16.5 cm.

To generate the general 3DOF rotation of this target, a programmable rotational motion demonstrator[61] was used. This piece of hardware (shown in Figure 6.10),
(a) Camera  
(b) 6DOF Robotic Gantry

Figure 6.8: Lab Robotic Hardware

Figure 6.9: Lab Target

Figure 6.10: Rotational Motion Demonstrator
designed and built by fellow lab member Stephen Russell, is capable of generating general rotations in all 3 degrees of freedom. This makes it ideal for simulating tumbling targets. An example of the capability of the demonstrator is shown in Figure 6.11, where a satellite model mounted on the hardware was made to rotate as it would in space (i.e., via torque-free motion).

Unlike the simulations, no truth was available for judging the hybrid algorithm’s performance. However, for comparison purposes, an independent pose estimation system was utilized. This consists of infrared light reflectors (Figure 6.12a) affixed to the periphery of the rotating base and a suite of 6 infrared cameras (Figure 6.12b) arrayed around the target, all from the commercial OptiTrack system made by NaturalPoint. Once calibrated the Optitrack system provides an independent estimate of the 3DOF orientation of the tumbling target (see Figure 6.13).
6.2.2 Laboratory Results

A time lapse of the hybrid estimation process in action (running with 100 particles) is shown in Figure 6.14. The images from the video stream are shown at the right of each frame, with the SIFTs detected and matched overlaid as red dots. The motion of the target is visible clearly over this set of images. The left frames show the 3D configuration of the camera and the positions of the image features relative to the camera. Red dots are positions of features actively being observed by the camera, and blue dots are positions of features which are no longer being observed by the camera. The black wireframe cube is the shape of the target cube shown in Figure 6.9, projected at the pose estimated by the Optitrack system shown in Figure 6.13.

Pose

The ability of the hybrid estimator (running with 100 particles) to track target pose is shown in Figures 6.15 and 6.16. The pose estimate based on visual tracking (i.e., the hybrid algorithm described in this thesis) is shown in green, and the estimate from the Optitrack infrared system is shown in black for comparison (spikes in the black line are outages in the estimate). Note that the pose in Figures 6.15 and 6.16
Figure 6.14: Lab Estimation Sequence
Figure 6.15: Lab Experiment: Orientation Tracking ($\bar{\theta}_t$)

Figure 6.16: Lab Experiment: Position Tracking ($\bar{\rho}_t$)
is expressed in an inertial frame, with the z-axis perpendicular to the lab floor. This is to better show the dominant axis of rotation of the target.

Qualitatively, the plots show good agreement between the visual estimate of target pose and the infrared camera and marker system for estimating pose. In particular, tracking is quite precise in the degree of freedom that has the largest change, namely, the Euler angle $\psi$. That is, the estimates agree on the dominant motion of the target. As this represents ‘out-of-plane’ rotational motion that can be the hardest to track with monocular sensing, this agreement is an important demonstration of the hybrid algorithm’s accuracy.

Figure 6.16 gives the estimates of the inertial position of the center of the lab target versus time. The visual tracking algorithm and the infrared system are similar qualitatively in their estimates, in that neither drifts and both display the periodic nature of the motion of the target’s center. However, there is a bit more quantitative disagreement between the visual tracking estimate and the infrared comparison estimate.

There are two potential causes for this disagreement. The first is error in the visual tracking estimate. Recall from Section 4.1 that a measurement model-only approach is susceptible to low confidence estimates in two of the rotational degrees of freedom (corresponding to ‘out-of-plane’ rotation, Figures 4.6 and 4.7) and one of the translational degrees of freedom (corresponding to translation along the boresight, Figure 4.5). The hybrid approach partially addresses this by using a rotational motion model to increase confidence in the rotational degrees of freedom. However, the hybrid approach estimates translation via optimization without a motion model, so translation along the boresight (i.e., ‘out-of-plane’) can still be subject to larger uncertainty.

The other potential cause of disagreement is error in the infrared camera system’s position estimate. The system developed in lab only explicitly estimates orientation, while position is a derived estimate using imprecise knowledge of the offset between the lab target’s reference center and the true rotation center of the rotational motion demonstrator (Figure 6.10). It is interesting to note that this issue stems from the
same underlying problem that plagued the SLAM/SFM Bayes filter’s ability to estimate pose (Chapter 4, Section 4.2): unknown offset $\bar{r}$ between the reference center and rotation center of the rigid body target.

**Shape**

Figure 6.17 compares the reconstructed map with the true shape of the target. For the alignment shown, the RMS error is 0.51 cm. This value is determined by using the distance between each feature and the closest planar face on the wireframe truth model as a measure of individual feature error. Note that the alignment between map and truth in Figure 6.17 was determined by hand, so the true RMS error of reconstruction is less than 0.51 cm. Normalizing the RMS error of feature reconstruction by the approximate distance to target along the boresight:

$$e_{\text{normalized}} = \frac{e_{\text{RMS}}}{z_{\text{avg}}} = \frac{0.51\text{cm}}{60\text{cm}} = 0.0085$$

(a) Top-Down View  
(b) Perspective View

*Figure 6.17: Lab Target Reconstruction*
6.3 Field Experiments

The hybrid algorithm was tested in the field, with real vision and complex lighting conditions, to demonstrate its real-time capability for tracking and reconstructing a moving target with \textit{a priori} unknown shape and motion properties. Tests were conducted with a remote submersible observing an underwater target mounted on a tether, in Monterey Bay off the coast of California.

Building on the results in the previous sections, the results in this section show the successful performance of the hybrid algorithm, with a small number of particles, in more difficult lighting conditions than the lab, while tracking an object undergoing realistic random rotational and translational motion.

6.3.1 Field Hardware

Testing was conducted in conjunction with the Monterey Bay Aquarium Research Institute (MBARI), using the ROV \textit{Ventana} as the observation vehicle (shown in Figure 6.18). The ROV is deployed and operated from a surface vessel (described further below), with a long tether connecting the ROV back to the surface vessel.

![ROV Ventana](image)

(a) Deployment  
(b) Front View

Figure 6.18: ROV \textit{Ventana}
The imaging feed used for estimation is provided by the main science camera, visible in the view of the front side of *Ventana* shown in Figure 6.18. The feed from this camera is fed back through the tether to a computer on the surface vessel, where the simultaneous target pose and shape estimation process takes place.

The ROV is deployed from the MBARI research vessel *Point Lobos* (shown in Figure 6.19). Aboard the *Point Lobos*, ROV operations are run from the control room by a dedicated team of pilots.

The underwater float target is shown in Figure 6.20. It is a syntactic foam block used for underwater moorings, connected to a tether with an eye-bolt. The long axis of the float is 46 cm in length. It is representative of similar underwater science instruments (such as in Figure 1.3) for which autonomous rendezvous and docking are desirable.

The target underwent unknown, unpredictable rotation and translation, due to forcing and torqueing by ocean currents, tether forces, and buoyancy effects. A comparison signal for the target’s pose does not exist (the underwater float was not instrumented), so judging the veracity of the pose estimate generated by the hybrid algorithm can only be done qualitatively.

During the field experiment, the ROV and the underwater target were roughly 2 meters apart.
6.3.2 Field Results

A time lapse of the estimation process deployed in the field, running with 100 particles, is shown in Figure 6.21. The images from the video stream are shown at the right of each frame, from which SIFTs detected. The left frames show the 3D configuration of the camera and the positions of the image features relative to the camera.

Pose

From observation of the video feed for the test in Figure 6.21, and confirmation by the ROV pilots, the net relative rotation of the target is known to be counterclockwise (looking down on the target). Figure 6.22 shows a plot of target shape and relative target pose expressed in the target’s body frame; i.e., in this frame, the ROV appears to fly around the stationary target. This plot confirms qualitatively the pose estimate by showing that the ROV appears to ‘fly’ around the target in a clockwise sense.

Shape

Figure 6.17 compares the reconstructed feature map with the true shape of the underwater target. The reconstructed shape of the underwater float (green dots) and the true shape of the float’s upper hull (orange wireframe) are shown from a perspective view in Figure 6.24. Similarly, Figure 6.23 displays the 3-axis view of the
Figure 6.21: Underwater Estimation Sequence
Figure 6.22: Relative ROV Trajectory and Target Reconstruction, in Target Frame

Figure 6.23: 3-Axis View of Target and Reconstruction
Figure 6.24: Perspective View of Target and Reconstruction
reconstruction and true shape.

The features in the reconstruction that are clustered below the target correspond to SIFTs that were observed on the tether that connects the target to the seafloor.

Judging qualitatively from these comparisons, the hybrid algorithm does an effective job of reconstructing the partial shape of the target, for the faces observed by the camera during the sequence of Figure 6.21. As can be seen in Figure 6.23, almost all the features fall on the orange hull of the target.

Quantitatively, the RMS error is 1.74 cm (excluding the features detected on the tether). This value is determined by using the distance between each feature and the closest planar face on the wireframe truth model as a measure of individual feature error. Note that the alignment between map and truth in Figure 6.17 was determined by hand, so the true RMS error of reconstruction is less than 1.74 cm.

Normalizing the RMS error of feature reconstruction by the approximate distance to target along the boresight:

\[
e_{\text{normalized}} = \frac{e_{\text{RMS}}}{z_{\text{avg}}} \approx \frac{1.74\text{cm}}{200\text{cm}} = 0.0087
\]  

\[ (6.3) \]

6.4 Summary

The three sets of tests demonstrate the performance of the hybrid algorithm. Section 6.1 showed that the hybrid approach converges and can run with a small enough set of particles to operate in real-time, while the Bayes filter requires too many particles to operate in real-time. Section 6.2 showed that the algorithm works with real vision processing included in the loop, for similar target motion as in the synthetic target simulations. Finally, Section 6.3 showed the successful tracking and reconstruction of a moving target under realistic conditions underwater (one of the environments where the algorithm is expected to be deployed).

Comparing the performance across the three different experiments, it was observed empirically that in simulation the hybrid estimator ran consistently with 50 particles, while the tests with real vision required double the particles (100) to perform consistently. This number is still in the real-time capable range, as opposed to the well
over 500 particles required by the standard Bayes filter even under perfect, synthetic conditions (recall from Section 6.1 that the computation complexity is linear in the number of particles \( M \)). However, it does reflect some of the increased uncertainty that arises when tracking image features corrupted by real measurement noise.

Comparing the RMS reconstruction error of the underwater target with the lab target presented above, the RMS error is nominally about 3 times higher. This was expected, as the test complexity increased correspondingly (due to the more difficult lighting and greater variations in target motion experienced underwater).

However, it is important to note that when normalized by the average distance of the features from the camera’s imaging plane, the error (given in Table 6.1) for both experiments is about equivalent.

<table>
<thead>
<tr>
<th>Test: ( e_{\text{normalized}} )</th>
<th>Synthetic</th>
<th>Lab</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0167</td>
<td>0.0085</td>
<td>0.0087</td>
</tr>
</tbody>
</table>

Table 6.1: Depth Normalized Reconstruction Error

It is hard to compare the reconstruction error between the synthetic simulations and the tests with real vision, because the error metric is different: in the synthetic simulations, RMS error is measured point feature to (truth) point feature, while in the real vision tests, RMS error is measured point feature to (truth) plane. So while the synthetic error value in Table 6.1 is twice the error value for the real vision tests, it should be noted that the latter error metric does not reflect potential reconstruction error that is parallel to the surface plane.
Chapter 7

Conclusion

This thesis has described and demonstrated a novel hybrid SLAM/SFM algorithm, for tracking pose and reconstructing shape of moving rigid bodies. This enables visual relative navigation of robots with respect to tumbling targets for autonomous inspection, servicing, and docking purposes. This chapter reviews the contributions made in this thesis and describes avenues for future research.

7.1 Review of Contributions

The major contribution of this thesis has been to synthesize an estimator capable of 6DOF tracking and 3D reconstruction of an unknown moving target in real-time based solely on monocular camera observations. This estimation algorithm is a novel hybridization of recursive Bayesian filtering and nonlinear optimization, the two dominant approaches to online SLAM/SFM to date.

In the context of SLAM/SFM, the research in this thesis can be considered an extension of the online SLAM/SFM field to problems where the target (as opposed to the camera) is the object whose pose is unknown.

Additional contributions were:

- An improved algorithm for matching the SIFTs to establish correspondence.
  The classic SIFT match algorithm [46] compares descriptors between images
using a relative test: which descriptor is the best match is judged relative to other potential descriptor matches. The improved SIFT match algorithm uses in addition an absolute test: whether the dot product between the descriptors of the potential match exceeds a threshold. This was found to improve the feature matching performance by reducing the mismatches.

- A method for fast optimization-based determination of target position. The hybrid pose estimation scheme estimates rotation with a Bayes filter and translation with an optimization approach. For the latter, the standard approach is to minimize the pixel projection error, performed via nonlinear (i.e., iterative) optimization as the error measure $e(\hat{s})$ is nonlinear. An approach that is computationally faster is to change the error measure to be linear, $\tilde{e}(\hat{s})$, so that the calculation of optimal position is a linear optimization. In this case, the optimum is calculated in one step, as opposed to via iteration. This speed-up is important when used in a particle filter, where the optimization must be performed for each particle at every time-step.

### 7.2 Future Research

There are several avenues for further research that build on the work presented in this thesis. Some broad goals of additional research could be:

- improving the accuracy and performance of moving target tracking and reconstruction
- studying effective methods of closed-loop pose control, given the hybrid algorithm’s estimates

Three examples of the former would be implementing the hybrid estimator with guaranteed real-time performance, via software and/or hardware (Section 7.2.1), examining the proper use of Bundle Adjustment to reduce error in estimation (Section 7.2.2), and fusing an additional range-based sensor into the hybrid estimator (Section 7.2.3). An example of the latter is the application of dual control (Section 7.2.4) to
fuse estimation goals (such as complete and accurate reconstruction of the target) into the closed-loop control algorithm.

### 7.2.1 Real-time Implementation

The simulations and experiments in Chapter 6 were processed via non-real-time implementations of the real-time capable hybrid algorithm (due to software and operating system limitations). The product of this thesis was an algorithm which reduces the number of particles required for accurate performance to a manageable (i.e., real-time capable) amount. Studying effective methods for actually implementing this hybrid algorithm in a real-time manner is an interesting avenue for future research.

One topic of study is algorithmic, software simplifications, such as ‘locking’ the position (i.e., discontinuing the measurement updates of Equations 5.29-5.32) for certain map features once the magnitude of their position covariances $\Sigma_{ij}$ fall below a threshold. Such simplifications are worth examining, to quantify the trade-off between computational speed-up vs. decreases in accuracy.

Another topic is the application of parallel processing hardware, such as GPUs, which have begun to be harnessed by the online SLAM/SFM research community[14]. As discussed in Section 2.1.3 in the context of image feature detection, GPUs are ideal for computational tasks which require the same set of instructions to be performed many time over. The particle filter is such a task, as the same computational pipeline (shown in Figure 5.1, described by Equations 5.17-5.35) is repeated for all $M$ particles at every time-step. A GPU would allow for all particles to be processed concurrently.

### 7.2.2 The Role of Bundle Adjustment

Some existing online SLAM/SFM algorithms (Chapter 3), with nonlinear optimization-based pose estimation (Section 3.3.2), additionally make use of multi-view Bundle Adjustment for periodic correction of the less accurate, real-time generated estimates.

These periodic corrections take place by running Bundle Adjustment using the current frame and some number of the immediately preceding frames [20, 67, 51] (i.e., temporal locality), or by running Bundle Adjustment over the current frame
and some number of the nearby, previously stored frames [38] (i.e., spatial locality). As Bundle Adjustment involves a non-convex cost function, it needs a good prior at initialization, to ensure convergence within the correct local minimum. This initialization is provided by the best estimate from the real-time, recursive pose algorithm (i.e., the single-view motion-only Bundle Adjustment).

In a sense, these approaches perform a combination of Bundle Adjustments, ‘single-view motion-only’ at every time-step (to estimate pose in real-time) and ‘multi-view’ at periodic intervals \textit{a posteriori} (to improve the quality of the estimates).

The addition of periodic multi-view Bundle Adjustment to the hybrid estimation approach described in this thesis could potentially improve the accuracy of the reconstruction and improve pose tracking. As the target is rotating and faces of it are viewed repeatedly, batch corrections should theoretically improve the hybrid algorithm’s recursive estimate of target map $X$ and latest pose $\bar{s}_t$.

An interesting aspect of this research topic is that multi-view Bundle Adjustment has not been used previously for improving Bayesian filtering-based online SLAM/SFM algorithms (such as those described in Chapter 3, Section 3.3.1). In fact, contrasts are usually drawn between ‘Bayesian filtering only’ approaches and ‘nonlinear optimization + multi-view Bundle Adjustment’ approaches[66]. As the hybrid algorithm is partially based on Bayesian filtering, the inclusion of periodic multi-view Bundle Adjustment (i.e., a ‘Bayesian filtering + multi-view Bundle Adjustment’ approach) would represent a novel direction for the online SLAM/SFM field.

### 7.2.3 Fusion with Range-based Sensors

As discussed in the Introduction, the goal of this thesis was an algorithm that only used a monocular camera, due to the availability and ease of integration of this type of sensor. The hybrid algorithm represents an improvement over other monocular vision-only SLAM/SFM estimation approaches. It would be worthwhile to investigate the augmentation of the hybrid vision algorithm with other types of ‘simple’ sensors, and whether such augmentation yields additional improvements that offset
the cost/complexity of adding a second sensor.

LIDAR or sonar is an example of an augmenting sensor that, if used smartly in conjunction with the hybrid vision algorithm, could yield additional improvements in accuracy and robustness. Range-based sensors like LIDAR and feature-based sensors like vision can be viewed as complementary. Vision sensing has advantages over range sensing due to the computational efficiency of tracking sparse features as opposed to dense range returns. Range sensing has advantages over vision sensing due to the robustness of being an active sensor: it is unaffected by illumination changes, glint, shadow, etc.

Recent research in the Aerospace Robotics Lab has focused on tracking the pose of a satellite using only a planar scanning LIDAR[60] (Figure 7.1). A good place to begin sensor fusion is by investigating the relative strengths and weaknesses of the hybrid vision algorithm and this range-based approach.

Figure 7.1: Simulated Range Scans of Satellite (Credit: Stephen Russell)
7.2.4 Dual Control

With the development of the hybrid algorithm developed in this thesis, the real-time estimates of pose and structure of the target are now available, for use as signals in a feedback control scheme. However, there are other high-level goals besides the immediate commanding of the robot to desired 6DOF pose offsets with respect to the target. For example, generating a complete and accurate shape estimate of the target, without unobserved faces, might be the task that is desired in a spacecraft damage inspection scenario.

Novel methods exist for fusing the estimation goals and closed-loop control processes of a robot. In particular, the subject of ‘dual control’ has been studied [37, 36, 32], where the high level goals of estimation (to reduce down the estimation uncertainty) are coupled into the closed-loop pose control algorithm, as additional states in the controller.

For example, the desired states to be controlled (e.g., 6DOF robot pose $\bar{s}_{\text{robot}}$) can be augmented with states which are measures of the covariances of the feature position estimates $\Sigma_{\bar{x}_j}$. The desired values for these states are set to zero, and a controller is synthesized to both achieve the original goals of closed-loop control (to command 6DOF pose of the robot) and the achieve the estimation-related goals (to decrease the uncertainty in the target’s reconstruction).

7.3 Conclusion

Navigational autonomy for robots in remote environments increases the suite of tasks which can be performed. An example is the robotic capability to perform autonomous proximity and docking operations about moving targets. This robotic capability reduces mission costs, by replacing humans in situations where human-based intervention is possible but expensive (e.g., repairing the Hubble Space Telescope, servicing underwater science instruments). In addition, it enables new missions in situations where humans cannot currently reach and rigid body-relative navigation is required (e.g., repairing satellites in Geosynchronous Earth Orbit, autonomous landing on
comets or asteroids). The product of this research has been a real-time capable estimation algorithm, to aid in making moving target-relative autonomous navigation a reality.
List of References


