Egomotion Estimation with Large Field-of-View Vision

John Jin Keat Lim

1 September 2010

A thesis submitted for the degree of Doctor of Philosophy
of the Australian National University

Declaration

The work in this thesis is my own except where otherwise stated.

J. Lim

Acknowledgements

I would like to thank Nick Barnes for his continuous supervision and mentorship throughout the course of my PhD research. This work would not have been possible without his guidance and encouragement.

I am also grateful to Hongdong Li and Richard Hartley, whose advice and support have benefited me in many ways throughout this time. I also thank the other academics and staff at the College of Engineering and Computer Science and at NICTA, who provided a supportive and friendly research environment.

My appreciation also goes to the PhD students in the lab, in particular Chris Mccarthy, Luping Zhou, Peter Carr, Gary Overett, Tamir Yedidya, Pengdong Xiao, David Shaw and many others, who were a pleasure to work with. I particularly thank Luke Cole for his invaluable help with getting the robot up and running.

I would also like to thank Novi Quadrianto, Debdeep Banerjee, Cong Phuoc Huynh and Akshay Astana for their pleasant company and many stimulating discussions over dinner and other meals. My thanks also to Wang Wei, Carolyn and David for their friendship and warm hospitality, and to Jolyn for listening.

I am as always, deeply grateful for my parents and family who have been loving and supportive to me over the years. Finally, I thank God, who is a revealer of mysteries and an ever present help in times of trouble.

Abstract

This thesis investigates the problem of egomotion estimation in a monocular, large Field-of-View (FOV) camera from two views of the scene. Our focus is on developing new constraints and algorithms that exploit the larger information content of wide FOV images and the geometry of image spheres in order to aid and simplify the egomotion recovery task. We will consider both the scenario of small or differential camera motions, as well as the more general case of discrete camera motions.

Beginning with the equations relating differential camera egomotion and optical flow, we show that the directions of flow measured at antipodal points on the image sphere will constrain the directions of egomotion to some subset region on the sphere. By considering the flow at many such antipodal point pairs, it is shown that the intersection of all subset regions arising from each pair yields an estimate of the directions of motion. These constraints were used in an algorithm that performs Hough-reminiscent voting in 2-dimensions to robustly recover motion.

Furthermore, we showed that by summing the optical flow vectors at antipodal points, the camera translation may be constrained to lie on a plane. Two or more pairs of antipodal points will then give multiple such planes, and their intersection gives some estimate of the translation direction (rotation may be recovered via a second step). We demonstrate the use of our constraints with two robust and practical algorithms, one based on the RANSAC sampling strategy, and one based on Hough-like voting.

The main drawback of the previous two approaches was that they were limited to scenarios where camera motions were small. For estimating larger, discrete camera motions, a different formulation of the problem is required. To this end, we introduce the antipodal-epipolar constraints on relative camera motion. By using antipodal points, the translational and rotational motions of a camera are geometrically decoupled, allowing them to be separately estimated as two prob-

lems in smaller dimensions. Two robust algorithms, based on RANSAC and Hough voting, are proposed to demonstrate these constraints.

Experiments demonstrated that our constraints and algorithms work competitively with the state-of-the-art in noisy simulations and on real image sequences, with the advantage of improved robustness to outlier noise in the data. Furthermore, by breaking up the problem and solving them separately, more efficient algorithms were possible, leading to reduced sampling time for the RANSAC based schemes, and the development of efficient Hough voting algorithms which perform in constant time under increasing outlier probabilities.

In addition to these contributions, we also investigated the problem of 'relaxed egomotion', where the accuracy of estimates is traded off for speed and less demanding computational requirements. We show that estimates which are inaccurate but still robust to outliers are of practical use as long as measurable bounds on the maximum error are maintained. In the context of the visual homing problem, we demonstrate algorithms that give coarse estimates of translation, but which still result in provably successful homing. Experiments involving simulations and homing in real robots demonstrated the robust performance of these methods in noisy, outlier-prone conditions.

List of Publications

Publications by the Candidate Relevant to the Thesis

All publications are available online, and may be obtained from the author's website http://users.rsise.anu.edu.au/∼johnlim/ or from the CD included with this thesis.

1. John Lim and Nick Barnes. Directions of Egomotion from Antipodal Points, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, Anchorage, 2008.

2. John Lim, Nick Barnes and Hongdong Li. Estimating Relative Camera Motion from the Antipodal-Epipolar Constraint, *IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI)*, October 2010.

3. John Lim and Nick Barnes. Estimation of the Epipole using Optical Flow at Antipodal Points, *Computer Vision and Image Understanding (CVIU)*, 2009.

4. John Lim and Nick Barnes. Robust Visual Homing with Landmark Angles, in *Proceedings of 2009 Robotics: Science and Systems Conference (RSS '09)*, Seattle, 2009.

5. John Lim and Nick Barnes. Estimation of the Epipole using Optical Flow at Antipodal Points, *7th Workshop on Omnidirectional Vision (OMNIVIS '07), in conjunction with ICCV '07*, Brazil, 2007.

Other Publications Relevant to but not Forming Part of the Thesis

1. John Lim, Chris McCarthy, David Shaw, Nick Barnes and Luke Cole. Insect Inspired Robotics, in *Proceedings of The Australasian Conference on Robotics and Automation (ACRA '06)*, Auckland, 2006.

2. Rebecca Dengate, Nick Barnes, John Lim, Chi Luu and Robyn Guymer. Real Time Motion Recovery using a Hemispherical Sensor, in *Proceedings of Australasian Conference on Robotics and Automation, (ACRA '08)*, Canberra, 2008.

3. Novi Quadrianto, Tiberio S. Caetano, John Lim, Dale Schuurmans. Convex Relaxation of Mixture Regression with Efficient Algorithms, in *Advances in Neural Information Processing Systems (NIPS '09)*, 2009.

Contents

List of Figures

Part I

Background

Outline of Part I

In this part, we present a general overview of various concepts necessary for understanding the following chapters of the thesis.

Chapter 1: We introduce the problem of camera egomotion estimation, state the motivations for finding egomotion, and give an overview of our approach to the problem in this thesis.

Chapter 2: We review the geometry of image formation in cameras, and consider the various existing methods for measuring image motion.

Chapter 3: We next perform a survey of existing research on egomotion estimation and the related area of visual homing.

Chapter 1

Introduction

> In walking along, the objects that are at rest by the wayside stay behind us; that is, they appear to glide past us in our field of view in the opposite direction to that in which we are advancing. More distant objects do the same way, only more slowly, while very remote bodies like the stars maintain their permanent positions in the field of view, provided the direction of the head and body keep their same directions [98].
>
> *Herman von Helmholtz, 1867*

Egomotion or self-motion is the movement of an observer with respect to the surrounding environment. The German philosopher and physicist, Helmholtz, wrote that the effect of his egomotion was the visual observation of the rest of the world appearing to move relative to him. The focus of this thesis will work in the reverse - that is, to infer from those visual observations, the self-motion that gave rise to the apparent visual motion of the world.

Whilst egomotion can also be estimated from non-visual observations such as the measurements of gyroscopes, Inertial Motion Units (IMUs), or Global Positioning System (GPS) devices, estimates from visual inputs remain invaluable for any moving system. In fact, the estimates from visual and non-visual inputs are complementary, and can be fused, like the human visual and vestibular systems, to give an improved overall estimate of self-motion.

The perception of self-motion from vision has been the subject of research since antiquity. The Greek mathematician, Euclid [56] believed that rays of light emanated from the eye and that an object viewed by the eye was then enclosed by a cone of such rays. He observed that the apex angle of such a cone would be

smaller when an observer was far from the object and larger when the observer was nearer. This effectively allows the observer to infer its displacement relative to the object, and hence, its self-motion (in an object centred frame of reference).

This leads us to the observation that egomotion must always be measured *relative* to some frame or point of reference. An observer could be displaced relative to its previous location in some previous time, or it could be displaced relative to one or more objects in its environment. Imagine an insect flying down a cardboard tunnel which is in fact being pulled forward at the same speed as the insect's forward velocity[1]. The insect would observe that the cardboard world surrounding it would not have moved at all, no matter how hard it flapped its wings. Like the Red Queen in Lewis Carroll's *Through the Looking Glass* [34], the insect would be "running (flying) just to stay in the same place" - its resulting egomotion relative to the cardboard tunnel is nil.

What, then, should the point of reference be, in a dynamic environment where various objects move variously, and some do not move at all (and the observer does not know which is which)? In this work, we measure egomotion with respect to the largest set of objects or points which have a single consistent motion. In practice, this often equates to the set of static points in the environment. In general however, this is not always true, and there is no easy answer to the question.

1.1 Why Find Egomotion?

What then, does knowledge of self-motion tell us? Estimating egomotion is not trivial, and good reasons are required to justify our troubles (and also to justify writing or reading this thesis).

Estimates of self-motion are invaluable to any moving observer that wishes to explore or interact with its environment. Many key behaviours that are essential for navigation require some knowledge of egomotion. Below, we examine a number of these to motivate our research into this problem of egomotion recovery.

A mobile system is typically interested in achieving certain motions necessary for performing specific tasks. For example, a human may attempt to drive a car in a straight line. To do this, a driver would estimate the heading direction of the car, correct the steering angle and then continually repeat this to keep the

[1] Biologists investigating the link between optic flow and flight speed in honeybees performed experiments that did exactly this (Barron and Srinivasan [16]).

car on the straight path. Likewise, airborne systems need to detect and minimize unwanted yaw, pitch and roll rotations to ensure stable flight. For such **closed-loop control of self-motion**, an estimate of egomotion is necessary.

Another piece of information that would be invaluable to a mobile observer is the 3D structure of its surroundings. Given two views or images of the world taken from different positions, stereopsis gives the relative distances to points in the imaged scene (first observed by Wheatstone [263]). For a moving monocular observer[2], this requires finding the relative motion between the two positions from which the images were observed. This approach to estimating scene structure is called **structure from motion**. Other depth cues also arise from self-motion, for example, the occlusion and self-occlusion of objects in the scene.

Knowing self-motion also makes it easier to detect **independently moving objects** in a scene, that is, to distinguish dynamic bodies from static ones. For example, to a forward moving observer, the static points of the world would appear to diverge outwards from the Focus of Expansion (see Chapter 2, Figure 2.4). Hence, points which do not behave as predicted may correspond to dynamic objects. Identifying dynamic objects and planning one's motion accordingly in order to avoid or to interact with them is a vital component of many intelligent mobile systems.

Integrating estimates of self-motion over time leads to **visual odometry**, where the distance and direction traveled over some journey is estimated. Bees for example, integrate the image motion perceived during flight to estimate the distance between a food source and their hive - this distance is then communicated to other bees via a pattern of bodily motions known as the 'waggle dance' (Esch et al. [55]).

Related to this notion of odometry is the concept of **visual homing**, where the goal is to return to some previously visited location, such as a nest, hive, docking station or landing strip. We observe that visual homing is in fact a *relaxed* version of the egomotion problem, and we will discuss this useful and interesting special case in greater detail later (Section 1.3.2).

In general, knowledge of self-motion is a critical component in many mobile systems. Research on egomotion estimation may lead not only to advances in robotics and autonomous vehicle navigation, but also to a deeper understanding of the visual biology of humans, animals and insects.

[2]Note that this is distinct from the case of a *binocular* observer which has two eyes or cameras, at a known, fixed distance from each other. Our focus is mainly on a one-eyed observer, which can perform stereopsis only when that eye is displaced.

1.2 What is Observed?

Thus far, we have not precisely stated what the inputs from which we hope to infer egomotion are. Let us return to Helmholtz's observation that the world appears to move past an observer as it moves forward. A ray of light falls on a point on some object (a 3D 'world point' or 'scene point'). The ray is then reflected and enters the eye of the observer, where the projected ray forms an image on the retina (a 2D 'image point').

Alternatively, suppose instead that this 'eye' is a modern camera - the ray then falls on the image plane of the camera, typically an array of light sensitive CCD pixels. As the observer moves, the ray is projected differently and the 3D world point ends up being imaged by some other neighbouring pixel.

In this work, we will refer to this phenomena as **image motion**. It is the apparent motion of points on the image due to camera motion and/or motion of the actual world point. Image motion may be found or estimated using a large variety of methods, which we categorize into two classes - methods finding dense **optical flow**, and algorithms that find sparser **point correspondences**. In the next chapter, we will discuss these further.

1.3 What is Recovered?

We now seek to be more specific about the particular problems that will be investigated. We first discuss the egomotion estimation problem in greater detail. Then, we look at a *relaxed* version of the same problem with an application to visual homing.

1.3.1 Egomotion

We consider a monocular observer, that is, an observer or camera that has a single view of the scene at any one time. We assume that this camera and all the world points in the scene undergo purely rigid-body motion, the kinematics of which, may then be characterized purely by the translations and rotations of the camera and the world points. Some of the world points may, in fact, be moving independently of the camera (that is, the points lie on some moving or non-static object). In general, the distances from the world points to the camera are unknown.

The camera captures the rays of light reflected from each world point and

records the images of those points. After the camera moves, a second image is captured and the different positions of the imaged points in the first and second images constitute our measurements of image motion. As the camera keeps moving, more images may be captured and used to estimate egomotion, but in this work, we focus on the minimum case of estimation from only *two images* of the scene.

Whilst vision is a rich source of information, it is also an inherently noisy one. The extracted information, in our case image motion data, is liable to be polluted by noise and outliers arising from pixel noise, illumination changes, contrast saturation, violations of the rigid-body motion assumption, and myriad other sources of error, some of which may be removed, but most of which result in inherent ambiguities within the observed data.

Our goal then, is to *estimate the translation and rotation* of the camera in a manner that is *robust* to the noise and outliers present in the image motion data. To this end, this thesis proposes new constraints on egomotion and develops practical algorithms that are based on these constraints. In general, algorithms for egomotion recovery may be categorized into methods that assume the camera undergoes a small or *differential* motion between frames, and methods that handle larger, *discrete* camera motions. In this work, we will introduce new algorithms in both categories.

1.3.2 Egomotion Relaxed - Visual Homing

Whilst accurate estimates of egomotion are useful, obtaining them is, in general, non-trivial and computationally expensive. This is particularly problematic for applications such as navigation on small robots that typically have limited computational resources. Fortunately, we find that many tasks can in fact be performed with relaxed estimates of egomotion, which are computationally much cheaper to find. Here, we apply this idea to the particular problem of visual homing.

Consider a mobile system that wishes to return to some previously visited location, which we call the goal or home position. The system observes a view of the world from its current position and compares that with a stored image of the world as viewed from the home position. If we estimate the translation vector between the current and goal positions, and move in the direction of that vector, the mobile agent will arrive at the home position successfully, and *in the most direct path* (the straight line).

However, visual homing is still a success if the path taken is less than straight. What matters is that the agent arrives home. This can be viewed as a 'relaxed' version of the egomotion estimation problem. Firstly, the relative rotation between the current and goal views need not be estimated. Furthermore, the translation estimate can be a rough, approximate measure, with accuracy being of secondary importance. As we will see in Chapter 8, as long as certain bounds on the error of the estimate are met, homing will be provably successful.

As a result, homing algorithms can be designed to be computationally much cheaper compared to algorithms estimating full egomotion. This trade-off between computational expense and the number of motion parameters that are to be found as well as lower demands on their accuracy, leads to fast algorithms that can run in real-time on mobile systems without imposing heavy burdens on the computing resources of the system.

1.4 Characteristics of Our Approach

Although egomotion has been extensively researched for decades (or even millennia, if we include the work of Euclid and other ancients), obtaining estimates both robustly and quickly still remains a challenge[3]. Here, we attempt to approach the problem from a perspective different from that of most traditional egomotion recovery methods.

Our inspiration comes from the observation that biological systems, particularly insects, must solve many of the problems associated with motion and scene understanding, in order to successfully navigate their environments. However, in spite of their relatively simple brains, insects appear to somehow solve these difficult problems robustly and at great speeds. The vision researcher may be tempted to conjecture that somewhere in the visual-neural system of the insect, some neurons may be computing robust egomotion estimates in real-time. The question then, is how a relatively simple organism like an insect, might perform this difficult task (if indeed, it was doing so).

We observe that a biological solution would characteristically involve simple, basic operations, but these would occur within a massively parallel processing architecture. Parallel computing is one of the new frontiers of processor design but

[3]Only recently, has real-time egomotion recovery been achieved (e.g. Nistér [180] performs robust estimation of relative camera motion from two views using the 5-point algorithm [181] within a preemptive-RANSAC framework), but extending this to work reliably in complex, real environments outside the laboratory is non-trivial. Refer Chapter 3 for a review.

the algorithms that researchers produce are still, more often than not, sequential in nature. We would like to focus on algorithms that are **fundamentally parallel**. Methods which are fundamentally sequential by nature can be adapted for more efficient parallel implementations, however, the speedups experienced would be less significant compared to fundamentally parallel algorithms running on a parallel processing architecture.

Our work also investigates the advantages of using a **large camera field-of-view (FOV)**. Humans have a larger FOV than the standard camera, whilst many birds and insects possess compound eyes and panoramic vision with a much greater FOV than human eyes. Although, research has shown that existing egomotion recovery methods work more accurately and robustly on large FOV cameras [48, 28], few algorithms explicitly exploit this large FOV property to simplify the estimation process.

An image captured by a large FOV camera may be represented naturally with an **image sphere**. The projection of light rays onto some surface forms an image, and many surfaces are possible, the most common one being a plane. However, a perspective projection onto an image plane suffers from distortions - for example, a circle projected onto the plane is an ellipse. The differences between planar and spherical projections were observed even by Leonardo da Vinci, who noted that classical perspective projections gave a different image from that observed by the spherical human eye [46]. Modern studies [64] have indicated that a spherical projection may be more optimal for egomotion recovery compared to a planar one.

Our work will explore the geometry of large FOV, spherical images and exploit their properties in order to develop new algorithms for egomotion estimation, with a focus on robust and potentially real-time methods. In recent years, catadioptric devices, fish-eye lenses and other large FOV vision devices have emerged as common tools in vision and robotics research laboratories. As such, the results of our research will not only be of theoretical interest, but also of practical value to the community.

1.5 Contributions

In the following, we list the main contributions of this thesis:

- A novel geometrical constraint on camera egomotion (translation and rotation) is developed, which exploits the geometry of a spherical image and

the larger information content inherently present in a large FOV image.

- We propose another new constraint that is closely related to the first one, but which results in stronger constraints on motion. However, this method estimates only camera translation, and rotation needs to be found via a second step. Both the first and second constraints work under differential (small) camera motions.

- Using these constraints, algorithms are developed for robust egomotion estimation under the differential motion assumption. Experiments found the constraints to perform more robustly compared to the state-of-the-art under very noisy conditions.

- We introduce novel geometrical constraints for egomotion estimation under discrete (large) camera motion. Once again, we exploit the geometry of large FOV vision to simplify the problem. Our constraints geometrically decouple the translational and rotational components of motion, which, to our knowledge has not been achieved by prior discrete motion methods.

- Novel algorithms based on the discrete motion constraints were developed. These take advantage of the decoupling of motions to increase the efficiency and efficacy of robust estimation algorithms.

- A novel visual homing algorithm that converges provably was introduced. This was based on constraints developed for 'relaxed egomotion'. The algorithm homed robustly in experiments including robot trials in various environments.

1.6 Layout of Chapters

This work is organized into three parts (see the chapter summary below). In the first part of our work, the general ideas and approaches used are discussed and a review of various key concepts required for understanding the following chapters is given.

The second and third parts of the thesis divide our work into methods that estimate egomotion under the differential camera motion assumption and approaches that consider the more general scenario of larger, discrete camera motions.

Part I - Background

Chapter 1: We introduce the problem considered, state the motivations for solving it and describe the characteristics of our approach.
Chapter 2: Next, we review the geometry of image formation in cameras, and consider the various existing methods for measuring image motion.
Chapter 3: We then perform a survey of existing research on egomotion estimation and the related area of visual homing.

Part II - Differential Camera Motion

Chapter 4: In this chapter, we introduce the notion of antipodal points, and present constraints on translation direction and the rotation axis from the directions of optical flow at antipodal points.
Chapter 5: Next, a different but closely related constraint on translation is derived from antipodal flow vectors.
Chapter 6: Here, the algorithms based on the constraints of the last two chapters are demonstrated to work robustly and accurately via simulations and real image sequences. We compare the performance of these methods against each other, and against the state-of-the-art.

Part III - Discrete Camera Motion

Chapter 7: We introduce the antipodal-epipolar constraints for the recovery of discrete camera motion and demonstrate their use with robust algorithms that were tested in simulations and real images.
Chapter 8: Next, we look at the idea of relaxed egomotion and the visual homing problem. We introduce weak constraints on camera translation which lead to convergent homing algorithms that were tested in simulations, with real images, and on a mobile robot.

Chapter 9: We conclude this work with a summary of our findings and possible future work.

Chapter 2

Cameras, Images and Image Motion

Cameras are devices that map points in 3D (the real world) onto points in 2D (the image). We will begin this thesis by considering the geometry of how this image formation process occurs. We consider some mathematical models of cameras and the optical projection process which gives rise to an image within a camera. Also, we will examine how large field-of-view (FOV) cameras are able to capture omnidirectional and panoramic images of the scene.

Next, we look more closely at the problem of measuring image motion, first introduced in the last chapter. We will take a tour of the myriad methods by which image motion may be measured. These include methods that find point correspondences, as well as ones which recover optical flow. Their differences and relative advantages and drawbacks are discussed.

2.1 Cameras and Calibration

2.1.1 Pinhole Camera Model

Medieval scientists built the *camera obscura*, a dark chamber with a small hole in the wall and an image screen opposite it, to investigate the properties of light and optics [257]. The pinhole camera model works on the same principles as this first, primitive camera.

Let the projection centre or **camera centre**, **C**, coincide with the origin of the world coordinate frame, XYZ, as shown in Figure 2.1. The image plane is at a distance, f, from **C** and the centre of the image coordinate frame, xyz, coincides

with the intersection of the Z-axis and the image plane.

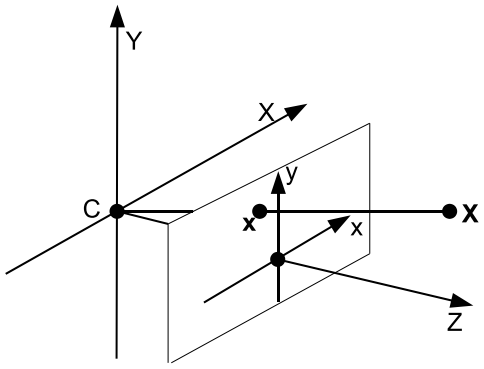


Figure 2.1: The pinhole camera.

X is a point in the 3D world, which we refer to as a **scene point** or a **world point**. A ray of light joining **X** and **C** intersects the image plane at the point where the image of the scene point forms. In the image coordinate frame, the **image point** is **x**. The projection maps a 3D point in Euclidean space $\mathbb{R}^3$ to a 2D image point in $\mathbb{R}^2$. With $\mathbf{X} = [X\ Y\ Z]^T$, the projected image point is given by (where we ignore the last coordinate):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} fX/Z \\ fY/Z \end{bmatrix} \tag{2.1}$$

Alternatively, using a more convenient homogeneous coordinate representation, the projection can then be expressed as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

Following [93], we rewrite this in shorthand as:

$$\mathbf{x} = \mathsf{K}[\,\mathsf{I} \mid \mathbf{0}]\mathbf{X} \tag{2.3}$$

where K is the **calibration matrix** containing the intrinsic parameters, including the focal length, the skew, and the scale factors. More generally, we can write:

$$\mathbf{x} = \mathsf{P}\mathbf{X}, \quad \textit{where } \mathsf{P} = \mathsf{K}\mathsf{R}[\,\mathsf{I} \mid -\tilde{\mathbf{C}}] \tag{2.4}$$

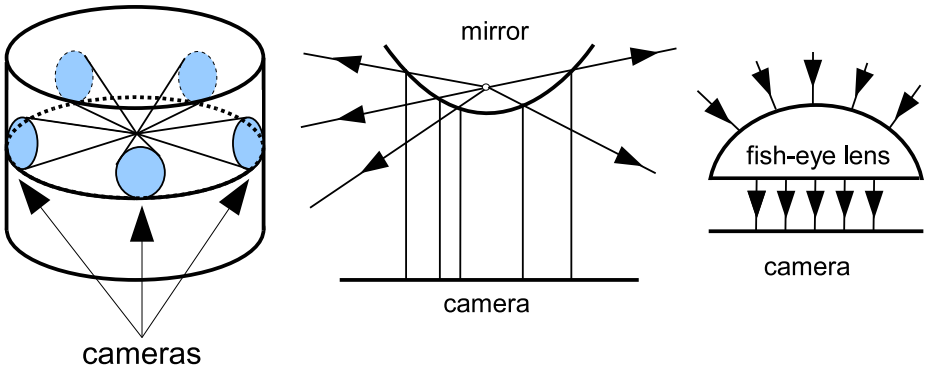


Figure 2.2: A schematic of some large FOV cameras: A rig with multiple cameras pointing in different directions, a catadioptric camera, and a camera with a fish-eye lens.

where P is the **camera matrix** which contains both the intrinsic parameters, K, as well as the extrinsic parameters, $\tilde{\mathbf{C}}$ and R. Whilst we previously assumed the camera and world coordinate frames to be the same, this need not be true in general, as demonstrated by Equation 2.4. Here, $\tilde{\mathbf{C}}$ and R pertain to the translation and the rotation relating the camera coordinate frame to the world coordinate frame (we use the notation $\tilde{\mathbf{C}}$ and $\mathbf{C}$ to differentiate between the camera centres in the two different coordinate frames).

With this, we have laid down the basic geometry of a simple camera model. For a more detailed discussion, we refer the reader to the many excellent books on this subject, such as Hartley and Zisserman [93] or Faugeras et al. [57].

2.1.2 Omnidirectional and Panoramic Cameras

Following the nomenclature of Pajdla et al. [184], a **directed** camera has a FOV which is contained in a hemisphere (this includes most conventional, limited FOV cameras); a **panoramic** camera has a FOV which contains at least one great circle; whilst the FOV of an **omnidirectional** camera will cover the entire **view-sphere**. Few cameras are truly omnidirectional although some panoramic ones do approach omnidirectionality.

Panoramic cameras may further be classified according to their method of construction. Panoramic images may be achieved by mosaicing images taken from rotating or moving cameras (such as [169, 25]), or from multiple cameras oriented in different directions [13]. Large FOV images can also be obtained using wide-angle 'fish-eye' lenses (e.g. [189, 18]) and reflective elements such as one or

more mirrors ([36, 171]). Cameras using a combination of lenses and mirrors are called catadioptric. Figure 2.2 illustrates the setup of some of these cameras.

Cameras may also be categorized as **central** or **non-central**. In central cameras, which are also known as single viewpoint cameras, all rays of light entering the camera will intersect at a single point. This is not the case with non-central cameras, leading to images that appear distorted to humans. Central cameras also lead to simpler analysis, and most computer vision algorithms are designed for use with the perspectively correct images that may be obtained from central cameras.

In practice however, central cameras with very large FOV are more difficult to build compared to non-central ones. Hence some large FOV cameras are only *approximately* central, in that the light rays entering the camera intersect at approximately a single point (see also [52]). For example, to achieve a FOV covering 75% of the view-sphere, the Ladybug multi-camera rig [187] consists of six cameras pointing in different directions, and these are mounted such that the projection centres of each of the six constituent cameras are close to each other, resulting in an approximately central camera.

In this work, *we focus on central cameras* - that is, we assume the camera may be modeled as having a single, finite, constant, camera centre. The pinhole camera discussed previously models central, narrow FOV cameras. However, its equations may also be adapted for use in central, large FOV cameras, where the camera matrix, P, is still a mapping from the direction of a 3D scene point to a 2D image point (albeit a more complicated one). For example, Pajdla et al. [184] derives the camera matrix for central, catadioptric camera systems where hyperbolic and parabolic mirrors are used to reflect light onto a conventional camera.

We refer the reader to books such as [24] for an in depth treatment of the subject.

2.1.3 Camera Calibration

In this work, we assume that cameras have been calibrated - that is, given an image point, some set of calibration parameters will allow us to obtain the direction of a ray joining the camera centre and the scene point.

Calibration recovers the intrinsic parameters (and extrinsic parameters if the camera coordinate frame is not aligned with the world frame). Calibration methods may be loosely classified into two categories. The first involves the use of

a calibration object, which is an object with certain geometrical properties that are known *a priori*, for example, a checkerboard pattern with known dimensions. The second category does not require calibration objects, using instead, the geometric properties inherent in scene features. For example, vanishing points and the absolute conic may be used to recover the intrinsics.

Many calibration techniques are available, such as the DLT method [93, 266], the methods of Tsai [246], of Zhang [278] and many others [88, 121, 97, 160, 242, 188] which can be found in reviews such as [72, 203, 279].

In addition, calibration methods developed specifically for central, large FOV cameras include [273, 275, 274, 161, 269, 75, 73, 122], whilst works such as [162, 79, 222, 186, 198] deal with non-central and more generic camera systems.

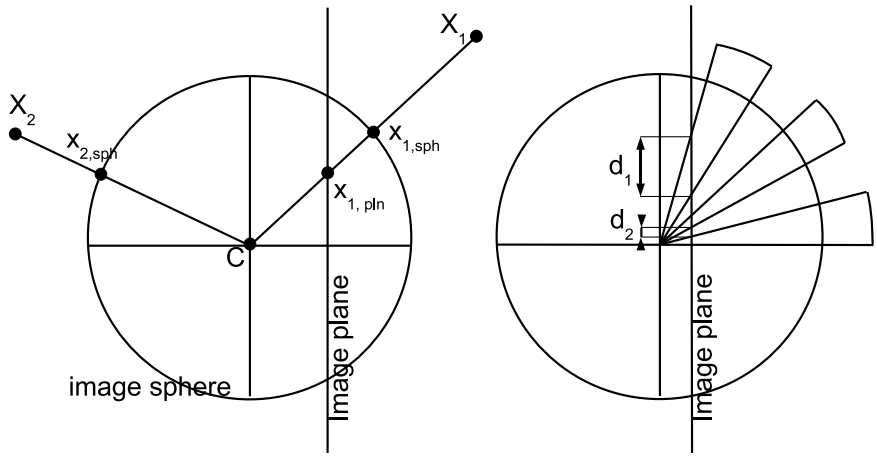
2.1.4 The Image Sphere

The calibration step discussed above aims to undo the physical distortions and warping of light rays entering the camera as a result of refractions, radial distortions, reflections (catadioptric cameras) and such. Once that has been accomplished, given an image point, we can then map it to a ray in space joining the camera centre, $\mathbf{C}$, to some 3D world point, $\mathbf{X}$ - for example, the rays joining points $\mathbf{C}$, $\mathbf{X}_1$ or $\mathbf{C}$, $\mathbf{X}_2$ in Figure 2.3(a).

It is then common, particularly in conventional, narrow FOV cameras, to reproject these rays onto some canonical image plane; for example, in Figure 2.3(a), the ray joining points $\mathbf{C}$, $\mathbf{X}_1$ is reprojected onto an image plane to give the calibrated image point $\mathbf{x}_{1,\mathsf{pln}}$. At first glance, choosing a planar image in our camera model (as we did earlier, in the pinhole camera model of Figure 2.1) seems reasonable, since the *physical* imaging surfaces used in cameras are in fact planar (such as a planar, photo-senstitive CCD array, or a planar sheet of photographic film).

However, reprojecting onto an image plane is problematic for large FOV cameras, as an image plane cannot represent images that cover fields-of-view in excess of 180°. This is clearly seen in Figure 2.3(a), where the point $\mathbf{X}_2$ which is in the opposite hemisphere to $\mathbf{X}_1$ cannot be represented on the image plane shown.

Furthermore, the result of projecting the 3D world onto a planar surface is that the image has to be distorted in order to fit all the rays entering the camera onto a plane. This is illustrated in Figure 2.3(b) where the rays shown intersecting at the camera centre subtend equal angles of 30° each. However, their projections onto the image plane result in images of unequal size (note, for example, d_1 is



(a) (b)

Figure 2.3: Image sphere versus the image plane: (a) $\mathbf{X}_1$ is the world point projecting onto the camera centre $\mathbf{C}$. $\mathbf{x}_{1,\text{pln}}$ is the image of that point on the image plane, whilst $\mathbf{x}_{1,\text{sph}}$ is the image if an image sphere was used. The image plane is unable to represent points in the opposite hemisphere, such as $\mathbf{X}_2$, whereas the image sphere has no problem with this. (b) The perspective distortion on an image plane. The visual angles marked out are the same but their projections onto the image plane are of a different size.

larger than d_2). The effects of this distortion of visual angles are most obvious at the edges of the image and are particularly severe as the camera FOV approaches 180°.

This leads us to the concept of an **image sphere**, where the calibrated rays are reprojected onto some canonical, spherical imaging surface instead of an image plane. For the calibrated ray, Equation 2.4 becomes:

$$\mathbf{x} = \frac{\mathbf{X}}{|\mathbf{X}|} \tag{2.5}$$

where $\mathbf{x}$ is the ray or direction vector joining camera centre and the 3D scene point, $\mathbf{X}$. Another way of thinking about this is to imagine the image point $\mathbf{x}$ as a point lying on the surface of a unit sphere such that $x^2 + y^2 + z^2 = 1$.

Unlike the image plane, the image sphere naturally represents points over the entire viewsphere (e.g. in Figure 2.3(a), both the world points $\mathbf{X}_1$ and $\mathbf{X}_2$ project onto the image sphere to give the image points $\mathbf{x}_{1,\text{sph}}$ and $\mathbf{x}_{2,\text{sph}}$). Furthermore, the problem of distorted visual angles does not occur for such a spherical projection. In general, the image sphere gives a simple framework in which to think about the problem and, as this thesis will show, has certain nice properties which will

aid us in the task of egomotion estimation.

In this work, we focus on calibrated, central cameras, and image points can generally be assumed to satisfy Equation 2.5, unless it is clear from the context that this is not the case (e.g. when dealing with uncalibrated cameras). We also refer to the image point $\mathbf{x}$ and the ray in the direction of $\mathbf{x}$ interchangeably.

2.2 Measures of Image Motion

Taxonomy

Consider two or more images of a scene taken by a moving camera. Given a point in one image, we wish to identify the same point in a different image - that is, the two matching image points should correspond to the same world point. This is known as the **correspondence problem**.

Decades of research have yielded a plethora of algorithms for measuring image motion and for solving the correspondence problem. In our review, we categorize them as **optical flow** algorithms and **point correspondence** methods.

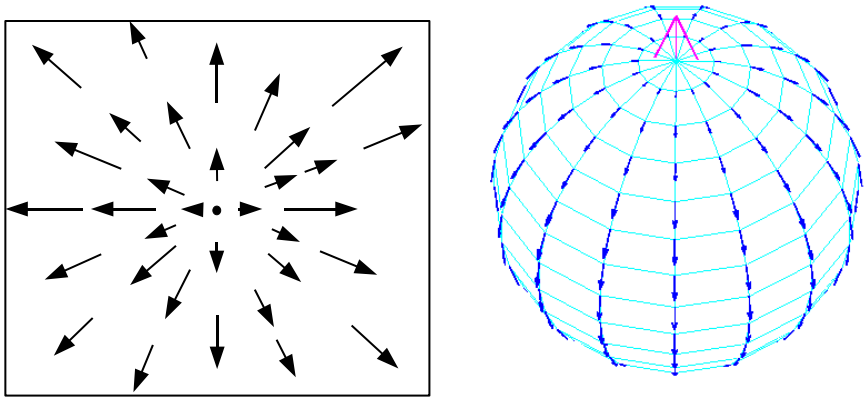
Optical flow methods typically estimate image motion from constraints on the spatiotemporal variations of image intensity. These methods are often used to compute image motion densely over the entire image, often represented as a field of image motion vectors known as the optical flow field (e.g. Figure 2.4).

Meanwhile, point correspondence methods attempt to obtain a robust and discriminative signature from the intensity gradient patterns in the neighbourhood of an image point. This signature is called a *feature descriptor* and should be discriminative enough for that image point to be correctly matched with the corresponding point in a second image.

The key difference between the two classes of methods is that most optical flow methods make approximations (e.g. the brightness constancy assumption, see Section 2.2.1) that limit the range of motion sizes they can handle, whereas point correspondence methods do not do so. This means that in practice, optical flow methods are typically more suited to smaller image motions. In contrast, point correspondence methods work robustly even under large image motions (which may result, for example, from large camera translations and rotations).

Representations of Image Motion

There are two ways by which image motion may be represented:



(a) (b)

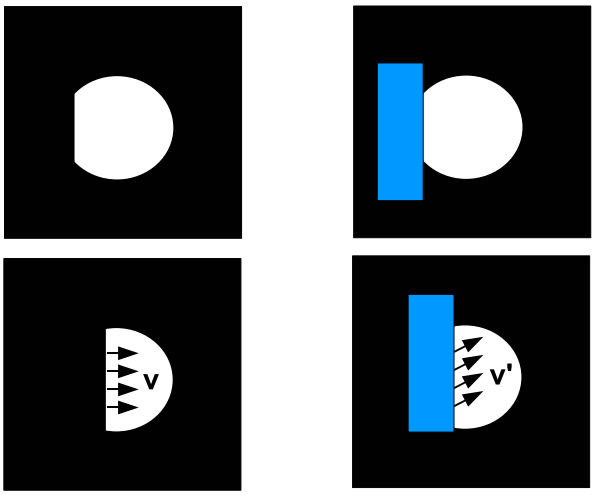
Figure 2.4: (a) A forward moving observer sees a diverging optical flow field. As the observer moves, almost every image point will undergo some motion, leading to an array of optical flow vectors, where each vector starts at the original position of the image point and terminates at its new position. The dot indicates the focus of expansion. (b) Flow on an image sphere. The large arrow indicates the focus of expansion.

- the image coordinates of the point before and after moving, or
- the vector difference between those coordinates

The latter may be plotted on the image to give a vector field called the **optical flow field**. Figure 2.4(a) shows the flow on the image plane whilst Figure 2.4(b) illustrates flow on the image sphere.

In the literature, the term 'optical flow' is sometimes used to refer to the class of dense optic flow *algorithms*; and at other times, used to refer to the vector field *representation* of image motion. The nomenclature used by researchers is somewhat confusing because the vector field representation may actually be used for image motion measured with algorithms from either the dense optic flow or the point correspondence categories.

In practice, if the image motion is small, both dense optic flow algorithms and point correspondence methods will work, and the outputs of both types of methods may be represented with the vector field representation of flow. However, if the image motion is fairly large, only point correspondence methods are used because of their robustness to scale, orientation and other changes in the image. The dense optic flow algorithms do not handle large image motions well, and the vector field representation may break down (for example, if the camera were to



(a) (b)

Figure 2.5: The aperture problem. (a) The motion of points on a moving bar appears to be v when viewed through the circular aperture. (b) However, if the entire bar was visible, its actual motion would be v′.

rotate by some angle larger than 90°, it is not possible to represent the resulting image motion with the vector field representation).

Ambiguities

It is important to note that the *observed* motion of image points does not necessarily equate the *true* motion of the points. This is because image information is often limited, leading to ambiguities in the estimates of the motion of world points.

This may be illustrated by a phenomenon known as the *aperture problem* [250]. Consider Figure 2.5, where a bar appears to move from left to right (direction v) when viewed through an aperture. However, the true motion of the bar, if the entire bar was visible, is in fact upwards and to the right (direction v′). When viewed through the aperture, the observer is only able to measure the component of motion that is in the direction of the intensity gradient. The component of motion that is perpendicular to the intensity gradient is not measurable. In the example, the observed v, is the component of v′ that is perpendicular to the light-to-dark edge. v is called the *normal* component of image motion, since it is normal to the edge.

Furthermore, it is sometimes simply impossible to estimate image motion due

to a lack of features or textures in the environment. For example, consider an attempt to estimate from images, the motion of world points lying on a blank, white wall. This is unlikely to result in anything useful since an image of the blank wall would contain insufficient visual information. Likewise, within environments with repeated structures, points of very similar appearance may easily be confused with each other, leading to erroneous estimates of motion.

Consequently, due to the ambiguities discussed above, the possibility of error always exists in egomotion estimates derived from image motion measurements. In general, however, image motion measurements give a fairly good approximation of the true motion field.

Next, we will discuss methods from both the optical flow and point correspondence classes, and compare their advantages and disadvantages in order to understand how they impact the task of egomotion estimation.

2.2.1 Optical Flow Algorithms

Following the taxonomy of Barron et al. [17], we will discuss three major categories of methods recovering dense optical flow. They are 'differential methods', 'correlation methods' and 'frequency methods'.

Methods in the optical flow category typically assume that the image intensity $I(\mathbf{x}, t)$ at an image point $\mathbf{x}$ and at the time instant t, is approximately constant over a short duration and within a small neighbourhood of $\mathbf{x}$ (that is, the brightness constancy assumption):

$$I(\mathbf{x}, t) \approx I(\mathbf{x} + \delta\mathbf{x}, t + \delta t) \tag{2.6}$$

Taking a first order approximation gives the *optical flow constraint equation*:

$$\nabla I \cdot \mathbf{v} + I_t = 0 \tag{2.7}$$

where ∇I gives the spatial partial derivatives of intensity, I_t its temporal derivative, and $\mathbf{v}$ is the optical flow vector. This is a constraint on image motion from the local spatial intensity gradient about the point $\mathbf{x}$. This equation is ill-posed [17], since only the component of $\mathbf{v}$ that is in the direction of the spatial gradient can be found - which is the aperture problem discussed previously. To resolve the ambiguity inherent in the equation, various additional constraints are used, and we discuss them in the following.

Differential Methods: *Local methods* only consider the intensity gradient over a local window. For example, the method of Lucas-Kanade [146] solves Equation

2.7 for $\mathbf{v}$ via weighted least squares. Chu and Delp [40] and Weber et al. [260] estimated the flow using total least squares, whilst Bab-Hadiashar and Suter [12] introduced a more robust solution using what they term weighted-total least squares.

A more general formulation of the Lucas and Kanade method [14] considers the problem as that of minimizing:

$$\sum_{\mathbf{x}} [\ I(W(\mathbf{x}, \mathbf{p})) - T(\mathbf{x})\ ]^2 \tag{2.8}$$

where a patch of the image I, in the neighbourhood of point $\mathbf{x}$, is warped under the function W (with warping parameters $\mathbf{p}$), such that it is aligned with a template image patch, $T(\mathbf{x})$. This warping can be an affine or similarity transformation, in which case, the problem is non-linear and may be solved by Gauss-Newton gradient descent methods.

On the other hand, approaches such as that of Gupta and Kanal [83] side-step the computation of intensity gradients by using the Gauss divergence theorem to convert the problem into one of local integration over surfaces and volumes of intensity.

Global methods generally enforce some notion of global smoothness in addition to Equation 2.7. Well-known approaches include that of Horn and Schunck [102], which minimizes a global energy function which is solved from the Euler-Lagrange equations for the function via the iterative Gauss-Seidel algorithm. Some methods also relax the brightness constancy assumption to take into account more general transformations of the flow field [207, 174, 192]. The work of Brox et al. [30] is a recently proposed coarse-to-fine warping algorithm giving highly accurate estimates; it is representative of variational methods that combine various constraints such as brightness constancy, gradient constancy and various smoothness terms to estimate flow.

Various other methods also exist, including approaches that obtain flow from contours and surface models [99, 32, 265], as well as approaches using linearly independent filters [214, 215].

Frequency Methods: These methods rely on spatiotemporal filters that work in the Fourier domain. Under certain conditions, these methods can be mathematically equivalent to computing flow from differential methods [213]. The optic flow for certain types of stimuli are more easily estimated using these methods compared to other flow approaches [1]; for example, differential and correlation

methods would have difficulties with sparse patterns of moving dots. These frequency methods are often used as models of motion sensing in biology.

The methods of Heeger [94] and of Fleet and Jepson [67], estimate flow by using banks of spatiotemporal Gabor filters (the product of spatiotemporal Gaussian functions with trigonometric functions) tuned to different spatiotemporal frequencies to find the strongest velocity orientation about an image point. Related research includes phase-based methods [66, 41, 113], and methods inspired by the behaviour of motion sensitive cells in the visual cortex [258, 82].

Correlation Methods: Correlation based methods attempt to find matching image patches by maximizing some similarity measure between them under the assumption that the image patch has not been overly distorted over a local region. Such methods may work in cases of high noise and low temporal support where numerical differentiation methods are not as practical. These methods are typically used for finding stereo matches for the task of recovering depth from stereopsis.

Representative work includes that of Sutton et al. [225], which allows a linear deformation of the matched neighbourhood, and the work of Anandan [6], which computes the correlation between two patches via Sum-of-Squared-Differences (SSD) within a coarse-to-fine scheme. Variations on this idea include the work of [33, 80, 182]. Recent work on correlation-based matching use graph cuts to enforce spatial consistency whilst taking into account occlusions [26].

Correlation methods are perhaps the predecessors of the point correspondence class of methods that we will discuss next. Whilst the former typically use SSD, pixel-differencing and warping functions to establish the correlation between image patches, point correspondence methods extract more robust and discriminative descriptors from the image patches, leading to much better matching under large image and camera motions.

2.2.2 Point Correspondence Algorithms

Point correspondence methods seek to recover feature descriptors that summarize the image information in the neighbourhood surrounding an image point. These descriptors need to be sufficiently *discriminative* such that the correct correspondence can be established between the same world point observed in different images. The feature must discriminate between the correct match and other potential (but wrong) matches.

Since the other image is typically captured after the camera has undergone some motion, the appearance of the image patch around the point of interest will have changed due to differences in scale, rotation and illumination. Hence, the descriptor also needs to be *robust* to these changes as well as to other sources of noise.

One may infer from the discussion up till now, that not every part of an image is conducive for the recovery of features that can be robustly matched or tracked. Edges are easy to detect but estimates of edge motion may be ambiguous due to the aperture problem. Nagel [170] showed that image points with high Gaussian curvature - often corresponding to corner or junction points - allow the recovery of full image motion (not just the normal component).

The Moravec interest point detector [166] and the Harris corner detector [85] were some of the earliest investigations into image points that could be recovered repeatably. Zhang et al. [280] placed an additional correlation window around Harris corners and showed that such points could be tracked over fairly large camera motions. This was perhaps the precursor to the idea of feature descriptors. Schmid and Mohr [206] used a rotationally invariant window instead of the regular correlation window for matching Harris points.

This was followed by the introduction of the Scale-Invariant Feature Transform (SIFT) descriptor [144, 145] that was invariant to rotation, scale and illumination changes. Features that were invariant to affine deformations soon followed [19, 164, 248], leading to successful matching even for image patches suffering fairly large perspective foreshortening effects. Maximally-Stable Extremal Regions (MSER) [157] are another example of features that are robust and stable over large camera motions.

Key issues for these point correspondence algorithms are the computational overhead as well as the sparsity of the computed interest points. Speeded-Up Robust Features (SURF) [21] used integral images to reduce the computational time needed for performing multiple Gaussian convolutions on the input intensity image. More recently, the DAISY descriptor [236] represents a step towards fast, densely computed features. However, although DAISY was designed for efficient dense computation, the problem of quickly matching the large numbers of features found remains a challenge.

In the following, we discuss the various components often occurring in point correspondence algorithms:

Feature Descriptors: Point correspondence algorithms typically compute descriptors from local image structures such as the intensity gradients in the

vicinity of a feature point. The SIFT algorithm for example, parcels out the local neighbourhood of the point into several square sub-windows, within which the orientations of intensity gradient edges are computed. These observations contribute to a series of histograms which encode the local intensity gradient structure of the point in a robust manner. The resulting histograms are concatenated as a feature vector, which form a discriminative signature of that point. To match the point with potential correspondences, some similarity measure (such as the Euclidean distance) is found between the feature vectors, and the most similar one will be the correct match.

Rotation Invariance: In a different view of the scene, certain parts of the image may have rotated relative to the reference image. The square windows used in the SIFT or SURF algorithms would then straddle slightly different image regions and the descriptors computed may differ quite significantly from the descriptors found in the reference image.

To compensate for rotation, point correspondence methods usually recover a dominant orientation for each feature point in the reference image, based on some robust average of the orientations of the local intensity gradients. In the new image, a rotation of some image object would result in a rotation of the local dominant orientation as well, and the feature descriptor can then be computed with respect to this orientation. This leads to the rotational invariance of descriptors and results in better matching.

Scale Invariance: In order to match regions of interest, the *scale* of the region needs to be considered as well. When an object is viewed up close, it occupies a larger area on the image, and the object structure is seen in greater detail. However, as the camera backs away, the image area occupied by the object shrinks and the details on the object are blurred. Scale invariance is the ability to match an image patch with a corresponding patch which is at a different scale.

In order to achieve scale invariance, one approach is to search for points that are robust to scale changes in image space and in scale space. Scale space [264] consists of the collection of images at varying scales (like a set of images taken under various levels of 'zoom' or magnification). To obtain this, an image is convolved with Gaussian kernels of progressively increasing standard deviation. The blurring introduced by the Gaussian convolution simulates the destruction of detail arising from a decrease in scale [124, 139].

The SIFT algorithm locates local maxima and minima points in scale-space by searching a pyramid of Difference-of-Gaussian (DoG) images (that is, a stack of images blurred under Gaussian kernels of increasing standard deviation, which is

then subtracted with an image of a neighbouring scale). This is an approximation of the scale-normalized Laplacian-of-Gaussian which is necessary for true scale invariance [139]. These local scale-space optima points tend to make very stable image features [164].

The DAISY feature [236] takes an alternative approach by considering a *multi-scale* feature, rather than a scale-invariant one. There, the feature descriptor is found at a range of scales, leading to a higher dimension feature compared to that of the SIFT descriptor. This approach is also reported to successfully match feature points that have undergone changes in scale.

2.2.3 Which Method to Use?

The main advantage of optical flow methods is that they can be computed densely over the entire image quite quickly compared to point correspondence methods. However, the accuracy of their estimates are typically worse than those of point correspondence methods. Furthermore, many optical flow methods rely on assumptions such as brightness constancy (Equation 2.7) which only hold under certain conditions (and the error is worse when these conditions are not met). Therefore, a practical system must consider various issues (such as the trade-off between processing time and accuracy) when choosing which class of methods to employ.

Density of estimates. Egomotion can be recovered from a small set of accurately recovered correspondences or optical flow. For example, a minimum of five points is sufficient for recovering the three parameters of camera rotation and two parameters of translation (see Chapter 3). However, in many applications, recovering egomotion is not the end of the story: after egomotion is found, it may be necessary to then compute a depth map, which would require a dense computation of image motion.

Accuracy. Whilst optical flow methods are typically less accurate, many applications do not necessarily require highly accurate optic flow. Robot navigation is possible without the sub-pixel accuracy of point correspondence motion estimates and navigation from qualitative (rather than quantitative) measures of flow is even possible. On the other hand, the reconstruction of scenes for movie post-production purposes or 3D modeling, requires highly accurate image motion estimates, and point correspondence methods which are both accurate and densely computed (such as DAISY) may be more suitable. Moreover, recent advances in dense, optical flow methods (for example, Brox et al. [30]) have

accuracies approaching those of point correspondence approaches.

Speed. Whilst the DAISY feature is designed for fast and dense computation, *matching* these features remains a slow process. The discriminative power of features exacts a price in the form of the high dimension of the feature vectors. To perform matching, some distance or similarity measure has to be calculated between many high-dimensional feature vectors. As a result, it is not possible at present, to obtain dense *and* fast point correspondence estimates of image motion. Sparse point correspondence methods, however, may be computed in real-time but this requires special hardware. For example, SIFT features may be found at high speeds using Graphical Processing Units (GPUs), which are highly parallelized processors adapted for graphics applications [268].

Motion Size. Perhaps the most marked difference between the point correspondence and optical flow methods is the size of the image motion considered. Optical flow methods work for small motions and hierarchical or multi-scale extensions of these methods can work for somewhat larger image motions. However, when the motions are large enough that the local image regions undergo significant affine and perspective distortions or changes in scale, these methods generally tend to break down. Point correspondence methods, however, work robustly for small image motions as well as large ones (up to a point, of course), due to the reasons discussed earlier.

Conditions Apply... Those optical flow methods that solve for flow from Equation 2.7 rely on the assumption of brightness constancy. This assumption holds under: (1) (locally) uniform scene illumination, (2) Lambertian surface reflectance, (3) orthographic projection and (4) pure translation parallel to the image plane (see [17]). However, some of the optical flow methods discussed previously are exempt, for example Lucas-Kanade under an affine warping model [14], as well as the methods of [207, 174, 192]. On the other hand, point correspondence methods are more robust in general, and tend to perform well in general conditions without any of these restrictions.

In the experiments conducted as part of this work, the size of the image motion was the main consideration for our choice of the image motion measurement algorithms used. For small image motions (corresponding to experiments involving small camera motions in Part II) we found the Lucas-Kanade optical flow method to be very useful. Meanwhile, experiments involving mainly large image motions (Part III) generally used SIFT features. The other methods discussed in this chapter are equally valid choices and may be substituted, subject to the

various system criteria (speed, accuracy etc) mentioned above.

2.3 Summary

Beginning with the pinhole camera model, we described the process by which world points project to image points within a camera. These image points may be represented as lying on 2D surfaces such as planes or spheres - with the latter representation having various advantages, such as being more suitable for large FOV images.

We then considered the phenomenon of image motion which arises when the camera and the world points move relative to each other. We also surveyed a number of algorithms for measuring image motion, which were categorized into optical flow methods and point correspondence approaches.

Chapter 3

Review of Egomotion Recovery

3.1 Egomotion Estimation

In the last chapter, we considered the geometry of cameras and image formation, and various methods by which the motion of image points may be measured. Now, we will look at one of the major mechanisms causing image motion - that is, the motion of the camera itself. We consider the equations governing the geometrical effects of camera motion upon image motion, and look at various approaches to estimating camera egomotion.

Egomotion recovery is a classical problem and the last few decades have seen intensive research in this area. The general problem of self-motion estimation may be solved using visual and non-visual means (such as measurements made by Inertial Motion Units (IMUs), Global Positioning System devices (GPS), sonar, radar, laser range-finders and such). Even with vision, there are multiple approaches ranging from systems using a single camera (monocular), a pair of stereo cameras (binocular), a network of cameras (n-ocular), cameras attached with time-of-flight sensors (which estimate scene depth), or even cameras operating in the non-visible spectrum.

In this thesis, we focus on monocular, visual images in the visible spectrum. We summarize the characteristics and assumptions of our approach below:

- a moving monocular camera
- estimate from two camera views[1]
- a single camera centre

[1]This also implies that no knowledge of the past state of the camera is assumed.

- calibrated camera
- using image motion measurements[2]
- unknown scene depth
- rigid body motion (characterized fully by a translation and rotation)

3.1.1 On the Size of Camera Motions

Existing egomotion recovery research may be categorized into two classes - those assuming *differential* camera motion and those assuming *discrete* camera motions. Differential motion methods assume that the camera undergoes an *instantaneous* translational velocity and an *instantaneous* angular velocity (i.e. rotation). The discrete motion methods, on the other hand, assume that the translation and rotation between successive views is large.

If the motion is large, the instantaneous methods become less accurate and eventually break down. It was previously accepted that if the motion was too small, then the discrete methods may also become less accurate. However, some recent research seems to suggest that certain discrete motion methods may work well even for small motions [138, 152, 119, 243, 20]. For example, Lin et al. [138] showed theoretically (under certain conditions and assuming a proportional noise model) that the discrete motion algorithm they considered performed well for both discrete and differential motions.

In general, the choice of using discrete or differential algorithms is often decided by the target application considered. Certain applications consider views taken from cameras that are far apart, and the discrete motion methods are more suitable. On the other hand, many applications involving cameras that are moving (e.g. mounted on a car, robot, or person) would be well-suited to the use of differential motion methods.

However, in the latter case of moving cameras, it must be noted that since all cameras have a finite frame-rate (rate at which images are captured), the motion undergone between two successive frames would be small, but never truly differential (as the camera would have moved a finite distance in the time between the capture of one image and that of the next image). Therefore, many applications may in fact find themselves operating in conditions that are somewhere

[2]We assume that the image motion observations are a good approximation of the true motion field.

in between strictly discrete or strictly differential motions. In practice, there is a range of motion sizes (not too large and not too small) for which both methods work comparably. Comparisons between the operating range of some discrete and differential motion methods can be seen in work such as [138, 221, 255, 148].

Since these discrete and differential camera egomotion estimation algorithms take image motion measurements as their inputs, the size of the camera motions also impacts which image motion method to use. Large camera translations and rotations typically result in large image motions which need to be measured with point correspondence methods. Conversely, small camera motions tend to give rise to small image motions, which may be measured with either optical flow or point correspondence methods (and the factors affecting one's choice are discussed in Section 2.2.3, page 29).

In the early days of computer vision, only small image motions could be automatically measured, using the optical flow techniques that were available at the time. This perhaps explains the focus then on differential motion methods, which resulted in papers such as [31, 281, 120, 191, 114, 199, 100, 190, 237, 173, 103, 63, 212, 220, 4, 112, 272, 130, 228, 2, 159]. However, the introduction of point correspondence measures of image motion enabled image points to be matched even under large camera translations and rotations. This suddenly made discrete motion methods practical (previously, researchers had to virtually hand mark point matches when the images were too different for optical flow methods to work) and the last decade has seen much interest and research in discrete motion methods (such as [141, 247, 262, 87, 181, 136, 93, 277, 147, 86, 59, 181, 136, 179, 89, 90, 91, 117]).

In this chapter, we will discuss the various algorithms and methods that have been devised for egomotion recovery in the last three or four decades.

3.1.2 On Differential Motion Algorithms

Without loss of generality, let the camera centre be the origin of both the world and camera coordinate frames, and assume unit camera focal length. Then, the equation relating optical flow and egomotion under perspective projection onto a planar image by a pinhole camera model, was derived by Longuet-Higgins and Prazdny [142] to be:

$$\dot{\mathbf{x}}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -x_2 \end{bmatrix} \left(\frac{\mathbf{t}}{|\mathbf{X}|} + \mathbf{w} \times \mathbf{x} \right) \tag{3.1}$$

where $\dot{\mathbf{x}}(\mathbf{x})$ is the optical flow at an image point, $\mathbf{x} = [x_1\ x_2\ 1]^T$, and the depth of the corresponding scene point is $|\mathbf{X}|$. $\mathbf{t}$ is the camera translation, $\mathbf{w}$ is the camera rotation[3], and $\times$ denotes the vector cross product.

Under the spherical image representation, with the spherical image point, $\mathbf{r} = [r_x\ r_y\ r_z]^T$, such that $r_x^2 + r_y^2 + r_z^2 = 1$, we have [190]:

$$\dot{\mathbf{r}}(\mathbf{r}) = \frac{1}{|\ \mathbf{R}\ |}((\mathbf{t} \cdot \mathbf{r})\mathbf{r} - \mathbf{t}) - \mathbf{w} \times \mathbf{r} \tag{3.2}$$

where we use $\mathbf{r}$ ($r$adius) instead of $\mathbf{x}$ to emphasize the difference between Equations 3.2 and 3.1, that is, the image point in 3.2 resides on the image sphere. (To keep the notation self-consistent, we also denote the scene point as $\mathbf{R}$ instead of $\mathbf{X}$.)

As before, the point $\mathbf{r}$ on the unit image sphere has the same direction as a ray joining the camera centre and the scene point (which is at a distance of $|\mathbf{R}|$ units away). Hence, we will use the spherical image point $\mathbf{r}$ and the ray in the direction $\mathbf{r}$ interchangeably, in our discussions.

These equations (3.1 and 3.2) are essentially a first-order Taylor approximation of the geometrical effects of egomotion on the perceived image motion [221]. This linearization is possible because the nonlinearities of these effects are insignificant when the size of camera translation and rotation is small. However, if camera motion increases, the approximation becomes less accurate and eventually breaks down. These methods are therefore limited to scenarios where camera motions are small, and under larger camera motions, the discrete motion techniques described in Section 3.1.3 should be used instead.

Scale Ambiguity

The are a total of 7 unknowns in Equation 3.2 (and also 3.1). The depth, $|\mathbf{R}|$, is 1 unknown. Rotation, $\mathbf{w}$, has 3 unknowns. The translation, $\mathbf{t}$, also has 3 unknowns. However, with only image motion information, we can only recover 2 of the 3 parameters of translation. This is due to an ambiguity in the scale of the translation.

Consider the first term in Equation 3.2. We can factor out the magnitude of the translation vector to get $\left|\frac{\mathbf{t}}{\mathbf{R}}\right|((\hat{\mathbf{t}}\cdot\mathbf{r})\mathbf{r}-\hat{\mathbf{t}})$. The $\left|\frac{\mathbf{t}}{\mathbf{R}}\right|$ factor shows that the scale of translation is ambiguous: a larger translation and correspondingly smaller depth

[3]Using the Euler axis-angle representation, where $\mathbf{w}$ is the rotation axis and $|\mathbf{w}|$ is the rotation angle.

can give rise to the same image motion observation as a smaller translation and correspondingly larger depth.

As a result, only 6 out of the 7 unknown parameters can be recovered - 5 are parameters of motion (3 for rotation, 2 for translation) and 1 is the unknown scene depth. Egomotion recovery involves finding the 5 motion parameters. Once that has been found, depth can be estimated - up to a scale as well - which is variously termed as relative scene depth, scaled depth or sometimes time-to-contact (since the inverse of $\left|\frac{\mathbf{t}}{\mathbf{R}}\right|$ has units in time).

Differential: Decoupling of Motion Components

As discussed above, under the assumption of differential camera motion, the first-order approximation of the relationship between egomotion and image motion is described in Equations 3.1 and 3.2. From inspection, we see that under this approximation, the two equations are linear in translation and in rotation. This means that it is not difficult to *decouple* the estimation of translation from the recovery of rotation with some algebraic manipulation. Many of the algorithms which we will discuss next take advantage of this to split up the problem so that translation and rotation may be separately estimated.

Algorithms

A straightforward least squares solution to Equation 3.1 or 3.2 is impossible since scene depth, $\mathbf{R}$, is a function of $\mathbf{r}$. Therefore, every additional flow observation introduces an extra constraint, but also an extra unknown in depth, so that the system of equations is under-constrained.

The approaches of Bruss and Horn [31] and of Maclean et al. [150] remove the dependence on depth algebraically. For example, taking a dot product with $(\mathbf{t} \times \mathbf{r})$ on both sides of the egomotion equation gives us an equation bilinear in $\mathbf{t}$ and $\mathbf{w}$:

$$\begin{aligned} (\mathbf{t} \times \mathbf{r}) \cdot \dot{\mathbf{r}} &= \frac{1}{|\mathbf{R}|}(\mathbf{t} \times \mathbf{r}) \cdot ((\mathbf{t} \cdot \mathbf{r})\mathbf{r} - \mathbf{t}) - (\mathbf{t} \times \mathbf{r}) \cdot (\mathbf{w} \times \mathbf{r}) \\ \Rightarrow (\mathbf{t} \times \mathbf{r}) \cdot \dot{\mathbf{r}} &= -(\mathbf{t} \times \mathbf{r}) \cdot (\mathbf{w} \times \mathbf{r}) \end{aligned} \tag{3.3}$$

A least squares estimate of rotation can then be found as a function of translation. Back-substitution into the equation gives a nonlinear equation for translation which can be solved by nonlinear numerical optimization.

Another nonlinear approach is that of Prazdny [190], which obtains a set of 3^{rd} order polynomials by algebraically removing translation from Equation 3.2.

It is found that rotation can be constrained from a triplet of points via:

$$\mathbf{n_3} \cdot (\mathbf{n_1} \times \mathbf{n_2}) = 0, \quad where \;\; \mathbf{n_i} = (\mathbf{w} \times \mathbf{r_i} + \dot{\mathbf{r}}_\mathbf{i}) \times \mathbf{r_i} \tag{3.4}$$

Tomasi and Shi [237] proposed a different constraint on motion which uses the angle between any two rays corresponding to two image points $\mathbf{r_i}$ and $\mathbf{r_j}$. The measured angle between two rays changes as the camera moves and the image is said to deform. This measure of image motion is rotationally invariant as a purely rotational movement will preserve the angle between two rays. With this approach, equations that are bilinear in translation and scene depth at the two points are obtained. Once again the resulting equations are solved by nonlinear methods.

Another constraint comes from the observation that image points which are close to each other will have nearly similar rotational components, but their translational components can be quite different if the depths at the two points are different (this is easy to see from Equation 3.2) [190]. This corresponds to neighbouring points that straddle a depth discontinuity. Subtracting the flow at two such points eliminates the rotational component and the remaining terms depend purely on translation. This same observation spawned a number of approaches, both classic [199, 100] and recent [105].

However, Prazdny [191] also observed that in general, adding or subtracting the flow at two or more points in some special way (not necessarily at a depth discontinuity), can result in the rotational components canceling out and leaving a constraint on translation. Linear subspace methods achieve this by a linear combination of flow vectors with appropriately chosen weights. How these weights are computed is detailed in the works of Jepson and Heeger [114, 115].

The differential epipolar constraint is the differential motion analogue to the epipolar constraint used in discrete motions (which we discuss in the next section). Kanatani et al. [118, 120] and Zhuang et al. [281] used the differential epipolar constraint to linearly solve for the *flow fundamental matrix*, which encapsulates the translation and rotation parameters.

More recently, Stewenius et al. [221] presented a nonlinear minimal solver[4] for the problem, known as the infinitesimal (or differential) five-point algorithm. The idea is that the direction of translation, a point and the de-rotated optical flow vector at that point all lie on the same plane. Rearranging Equation 3.2, we

[4]i.e. requiring the minimal number of points (5 points) to solve for the 5 unknown motion parameters (translation found up to a scale) in a calibrated camera.

have the de-rotated flow:

$$\mathbf{m} = \frac{1}{\mid \mathbf{R} \mid}((\mathbf{t} \cdot \mathbf{r})\mathbf{r} - \mathbf{t}) = \dot{\mathbf{r}} + \mathbf{w} \times \mathbf{r} \tag{3.5}$$

Due to coplanarity, we have:

$$\begin{aligned} \mathbf{t} \cdot (\mathbf{r} \times \mathbf{m}) &= 0 \\ \Rightarrow \mathbf{t}^T [\mathbf{r}]_\times \begin{bmatrix} [\mathbf{r}]_\times & \dot{\mathbf{r}} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix} &= 0 \\ \Rightarrow \mathbf{a}_i \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix} &= 0 \end{aligned} \tag{3.6}$$

where $\mathbf{a}_i$ is a 1×4 row vector. Five observations ($i = 1 \cdots 5$) will give five equations which may be stacked into a matrix $\mathsf{A} = [\mathbf{a}_1 \cdots \mathbf{a}_5]^T$. From rank-nullity constraints, the five sub-determinants of A must vanish. A tenth degree polynomial is obtained and solved, using either Gröbner basis or Sturm sequence based root solvers.

Recognizing the ambiguity in optic flow measurements due to the aperture problem, so-called 'direct' estimation methods use the 'normal flow' (that is, the component of flow in the direction of the intensity gradient; see Section 2.2, page 23) instead of full optical flow. The normal flow may be observed directly from the spatiotemporal gradients of brightness and is less challenging to find. These direct methods include the work of [172, 173, 103, 63, 60, 62, 211, 212, 220]. Among these, [63, 60, 62, 211, 212] solve the problem by imposing constraints from normal flow that reduce the motion solutions to some restricted subspace.

Koenderink and van Doorn [125, 126] investigated the use of divergence, curl and shear deformations of the optic flow field, in order to extract invariant properties in the camera motion and scene surface shape. Using these results, first and second order image motion flow fields were used in methods proposed by [190, 259] to solve for motion in scenes involving rigidly planar surfaces.

Sequential algorithms suggest using multiple images filtered temporally to estimate motion. Unlike the other methods mentioned in this thesis, this approach generally requires more than two views of the scene. Extended Kalman filtering methods were proposed by [29, 58, 5, 158, 95, 96]. An initial guess of the parameters is updated in these schemes until it converges. Particle filtering and sequential monte-carlo approaches include the work of [219, 193, 194] and a review may be found at [3].

There are also methods which take a *qualitative* approach rather than a quantitative one [63, 252]. For example, the Focus of Expansion of the flow field can

be detected qualitatively. It is not unreasonable then, for translation and rotation to be found from qualitative measurements also.

Other methods of note include those using Hough-like voting [15], those relying on the depth-positivity constraint [28], those using fixation to simplify motion estimation [61, 229, 47], as well as the work of [4, 112, 272, 130, 228, 2, 159] and a great many more, which can be found in reviews such as [235].

3.1.3 On Discrete Motion Algorithms

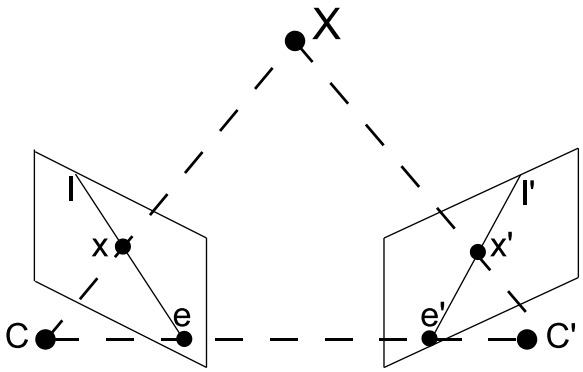


Figure 3.1: The epipolar plane.

Early work on discrete camera motion is represented by works such as Ullman [250] or Roach and Aggarwal [200]. However, it was the classic paper of Longuet-Higgins [141] that introduced the *epipolar constraint* on which most discrete motion algorithms are based (such as [247, 262, 141, 87, 181, 136]).

The geometry of the epipolar constraint may be understood as follows. Given two views of the scene which are related by some rigid motion, the two camera centres $\mathbf{C}$ and $\mathbf{C}'$ and any scene point seen by both cameras will lie on the same plane, termed the *epipolar plane* (given by the dotted lines in Figure 3.1). The observations are correspondences between image points $\mathbf{x}$ and $\mathbf{x}'$. The translation vector $\mathbf{t} = \overrightarrow{\mathbf{CC}'}$ intersects the image plane at the *epipoles* $\mathbf{e}$ and $\mathbf{e}'$.

From the coplanarity condition, the epipolar constraint equation can be derived to be [93]:

$$\mathbf{x}'^{T}\mathsf{F}\mathbf{x} = 0, \quad where \quad \mathsf{F} = \mathsf{K}'^{-T}\mathsf{E}\mathsf{K}^{-1} \tag{3.7}$$

F is the *fundamental matrix* which encapsulates calibration and motion parameters; K and K' are the calibration matrices of the two cameras; and E is the *essential matrix*. The essential matrix describes only the motion - translation $\mathbf{t}$

and rotation R - of the camera:

$$\mathsf{E} = [\mathbf{t}]_\times \mathsf{R} \tag{3.8}$$

(R is the matrix representation of rotation whereas $\mathbf{w}$, which previously appeared, uses the axis-angle representation.) Recovering F or E becomes a linear problem using the epipolar constraint.

For uncalibrated cameras, the fundamental matrix may be recovered by the eight-point algorithm of Longuet-Higgins [141], Hartley's normalized eight-point algorithm [87] or other similar variants [168, 39]. Each point correspondence gives a linear constraint, and eight or more points gives a unique solution for F which may be found by linear least-squares via methods such as Singular Value Decomposition (SVD). Alternatively, if an additional singularity constraint is enforced, a minimum of seven points may give a solution via the nonlinear seven-point algorithm (which yields 3 solutions) [93]. Other variations on these exist (see reviews [277, 147]), but all are based on the same principle of the epipolar constraint.

In the calibrated case (which is our focus), the essential matrix may be recovered from a minimum of five point correspondences [86, 59] using the 5-point algorithms of Nistér [181, 179], Li and Hartley [136] or Kukelova et al. [129]. These five-point algorithms involve a nonlinear solution where the roots of a tenth order polynomial are found using methods based on Guass-Jordan elimination, Gröbner basis techniques and such.

The discrete motion methods mentioned so far minimize the *algebraic distance*, that is, they minimize the L2 norm of $||\mathsf{A}\mathbf{h}||$ (Equation 3.7 can be expressed in the form $\mathsf{A}\mathbf{h}$ where A contains the entries of F, and $\mathbf{h}$ is a vector containing cross-terms of the xyz-components of vectors $\mathbf{x}$ and $\mathbf{x}'$). The advantage of using the algebraic distance is that the resulting cost function admits a linear solution (if 8 or more points are used). However, its disadvantage is that minimizing the algebraic distance is not geometrically meaningful [93].

As a result, for greater accuracy and optimality, the estimate from one of the above methods is often used to initialize a second, refinement stage which is instead based on minimizing the *geometric distance*, which in this problem is the reprojection error:

$$\sum_i [d(\mathbf{x_i}, \hat{\mathbf{x}}_\mathbf{i})^2 + d(\mathbf{x}'_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i})^2] \tag{3.9}$$

where $\hat{\mathbf{x}}_\mathbf{i}$ and $\hat{\mathbf{x}}'_\mathbf{i}$ are the 'true' point correspondences that satisfy the epipolar constraint exactly. This is a maximum-likelihood estimate under the Gaussian

noise assumption and can be solved using iterative Levenberg-Marquardt and other bundle adjustment methods. Due to the nonlinear cost function, these methods converge on local minima, hence a good initialization is necessary. We refer the reader to [93, 245] for a detailed treatment.

More recently, research has considered globally optimal methods of minimizing cost functions based on the L_∞ norm rather than the L2 norm. With known rotation, translation can then be found optimally under the L_∞ norm, using Second Order Cone Programming (SOCP) [91, 117]. Hartley and Kahl [89, 90] later showed that it is possible to optimally solve for both translation and rotation by a search strategy based on the branch and bound algorithm.

Note that within the literature, the problem of recovering discrete camera motion is variously called the 'relative pose' or the 'relative orientation' problem. These terms are used in order to include the more general problem of *two* different, stationary cameras observing the scene, instead of the problem of a *single, moving* camera observing the scene at two time instances. When there are multiple cameras, the calibration parameters, and hence the fundamental matrix may vary across cameras. However, for the case of calibrated cameras, this difference is not important and the relative pose and egomotion estimation problems are effectively equivalent.

Discrete: The Coupling of Translation and Rotation

We earlier saw that in the case of differential camera motion, the image motion could be approximated as a linear function of camera translation and rotation. However, in the case of discrete or large motions, Equations 3.1 and 3.2 no longer hold, and we instead use the epipolar equation, Equation 3.7, to constrain camera motion.

Under discrete camera motion, we see that the translational and rotational components of motion are *coupled* together in the essential or fundamental matrices. In particular, the function is *bilinear* in translation and rotation, i.e. $\mathsf{E} = [\mathbf{t}]_\times \mathsf{R}$. As a result, the various 'tricks' used in differential egomotion algorithms for eliminating translation and first solving for rotation, or vice-versa, will not work here. Therefore, attempting to decouple the translation and rotation in order to obtain linear constraints on translation and rotation is non-trivial and

was not possible up till now[5].

In Chapter 7 we will derive linear constraints on translation that are independent of rotation, and separately, linear constraints on rotation independent of translation. We demonstrate that under the special condition of antipodal points, the generally nonlinear/bilinear problem is decoupled to become linear in translation and linear in rotation (and this is not an approximation, as in the case of differential motions). This leads to advantages for the speed and robustness of algorithms.

Note that the term 'decoupled' is sometimes used in papers [22, 8] that first recover rotation (e.g. using vanishing points to rotationally register images) and then recover translation using the rotation-compensated points. Although, the translation and rotation are separately found, the underlying constraint is still coupled and the translation estimate is a function of the rotation estimate. Our use of the term here refers to finding translation and rotation *independently* of each other.

3.1.4 On the Advantages of Large FOV Vision and Image Spheres

Whilst egomotion estimation has been extensively studied in the case of cameras with limited FOV, the introduction of large FOV sensors such as catadioptric devices and fish-eye lenses, necessitates further study into the intrinsic advantages available from the wider FOV. Eye design in biology, is also suggestive of such advantages, where flying creatures such as insects and birds commonly have eyes that give them vision over nearly the entire viewsphere.

Different camera motions may sometimes give rise to near-similar optical flow fields if the camera FOV is small, leading to potentially erroneous egomotion estimates. A larger FOV is known to reduce this ambiguity [76, 175, 64]. For example, in Figure 3.2(a-b) an observer with narrow FOV undergoing a rightward translation, $\mathbf{t}$, may observe a motion field that is very similar (especially in the presence of noise) to the motion field observed when it rotates about the vertical axis, $\mathbf{w}$. This may lead to a confusion between rotational and translational motions when recovering egomotion. Conversely, as Figure 3.2(c-d) illustrates, a large FOV observer would observe significantly different motion fields depending

[5]Of course, if E has already been estimated, extracting $\mathbf{t}$ and R from it is a standard procedure. Here, we are not referring to this. Instead, we refer to obtaining decoupled constraints for $\mathbf{t}$ and R, where E is still unknown.

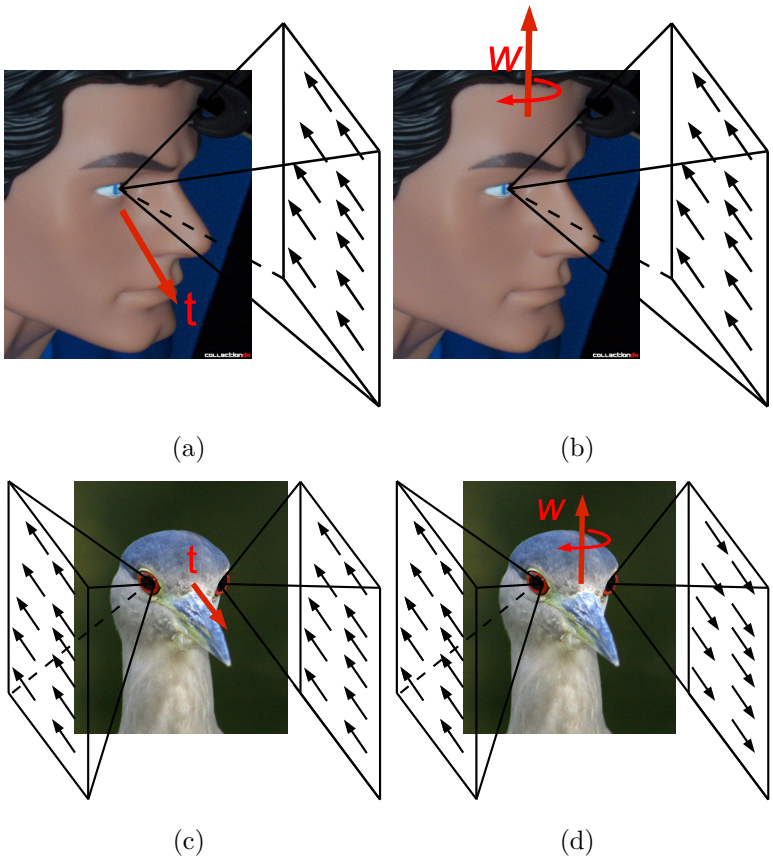


Figure 3.2: Narrow versus Wide FOV: (a-b) A narrow FOV observer (e.g. human eyes) sees a quite similar optical flow field for a rightward translation, and for a rotation about the vertical axis. (c-d) Conversely, a large FOV observer (e.g. bird eyes) sees very different optical flow fields for the translation and rotation shown. (Note how the direction of flow is in opposite directions in the two eyes when the bird rotates.)

on whether it was translating or rotating.

A large FOV is also known to minimize the effects of other ambiguities in motion recovery such as the bas-relief ambiguity and the forward bias problem. The bas-relief ambiguity [138, 38, 49, 106, 270] arises in the form of a local minima in the residual cost function, resulting in potentially erroneous egomotion estimates. This ambiguity is minimized when a large FOV is used. Errors may also arise due to the forward bias problem [256, 138], where translation estimates are biased towards the forward direction of the camera, that is, the direction of the principal axis (or the axis normal to the image plane). This problem is observed to affect methods such as the un-normalized 8-point algorithm [141] or

the method of Jepson and Heeger [114]. However, when an image sphere is used instead of an image plane, this problem does not arise, since the sphere does not have any forward direction (or conversely, every direction on the sphere may be the principal axis).

Fermüller and Aloimonos [64] suggests that a large FOV on a spherical image may be optimal for the recovery of self-motion, leading to higher accuracy and greater stability. The larger information content inherently present in images with large FOV also reduces the ambiguity in motion estimation [177]. Brodsky et al. [28] suggests that two different rigid motions cannot give rise to the same image motion fields when the entire view-sphere is considered whereas such confusions may occur when a smaller FOV is used.

In general, the performance of any of the discrete or differential estimation methods discussed previously would improve when used with a larger FOV camera. However, relatively few papers actually exploit the properties of large FOV cameras to their full advantage. Shakernia et al. [208] and Gluckman and Nayar [76] adapted some differential egomotion methods (including the methods of [114] and [31]) for omnidirectional and large FOV vision. Meanwhile, the work of Svoboda and Pajdla [226], Svoboda et al. [227], and of Micusik and Pajdla [163], extended the discrete epipolar geometry formulation for structure and motion estimation to large FOV catadioptric cameras.

The approaches of Nelson and Aloimonos [175], Fermüller and Aloimonos [63], and Silva and Santos-Victor [211, 212], obtained constraints on motion from qualitative measures of normal optic flow and performed a 2D search to estimate egomotion. Meanwhile, research by Neumann et al. [178, 176] considered how the advantages of large FOV polydioptric cameras may simplify the motion estimation task. Also relevant is the work of Makadia et al. [154, 153] and Geyer and Daniilidis [74], which exploited the properties of image spheres for egomotion estimation.

Following in the footsteps of such previous work, later chapters will explore novel ways of exploiting the geometry of large FOV vision for constraining and recovering egomotion robustly and accurately.

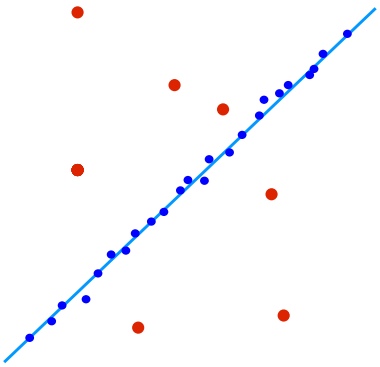


Figure 3.3: The problem of finding a linear regression line from outlier prone data. Outlier data points are the large red points whilst inliers are shown as small, blue points.

3.1.5 On Robust Algorithms and the Need for Outlier Rejection

Sources of Outliers

The methods discussed up to this point are typically able to deal with image motion observations that are affected by small scale noise. In particular, if small Gaussian noise is present, methods that minimize least-squares cost functions will give the optimal estimate.

In practice, however, the data is often polluted by measurements with large error, in the form of *outlier* data points (whilst data points that are not outliers, are termed *inliers*). This is illustrated in Figure 3.3 for a linear regression problem. Dealing with outliers and large noise is a challenging problem for all methods in general. For example, least-squares (L2 norm) minimization, which is used by many of the methods discussed, is well-known to be sensitive to outliers.

Outliers in image motion data typically come from:

1. mismatched points
2. independently moving objects
3. non-rigid motion
4. transparent objects, specularities, reflections, refractions etc

Mismatched points refer to the case where the two image point matches returned by the image motion algorithm do not in fact correspond to the same 3D scene

point. Some of these mismatches may arise due to the ambiguities inherent in image motion measurements, discussed in Section 2.2, page 23. However, even if the image motion algorithm returns the correct correspondence, if the data violates underlying assumptions in the algorithms used, the data are still outliers (the case of reasons 2 to 4).

Points on an independently moving object will move simultaneously to the camera motion. As a result, the effective motion of the camera relative to that point may be different compared to its motion relative to the static world. Applying egomotion algorithms to image motion data coming from a moving object will recover the motion of the camera relative to that object (which is not what we wish to find). Applying egomotion algorithms to image motion data coming from some static points and some non-static points results in some mixture estimate (which is the wrong estimate).

Point correspondences or optical flow from non-rigidly moving objects, transparent objects, reflective or refractive objects and such, violate basic assumptions inherent in the egomotion methods and these are some other causes of error.

Robust statistical frameworks

Since these outlier image motion measurements will inevitably occur in real-world environments, practical applications require egomotion methods that are robust (insensitive or less sensitive) to them. The standard approach is to use the previously discussed egomotion algorithms within a robust statistical framework.

In computer vision, one of the most popular frameworks is **Random Sample And Consensus (RANSAC)** [65, 93]. The RANSAC algorithm depends on Monte Carlo-type random sampling to generate hypothesis solutions. The data is then used to score these solutions and this is iterated until a good solution is found with high probability. This is summarized in Algorithm 3.1.

If M_i is large enough ($> M_T$), the algorithm terminates. Otherwise, it is iterated N times. N may be adaptively updated at every iteration according to the formula [93]:

$$N = \frac{log(1-p)}{log(1-(1-\epsilon)^s)} \tag{3.10}$$

Here, ϵ is the probability of picking an outlier from the total set S, therefore $(1-\epsilon)^s$ is the probability that all s sample points picked are inliers. p is the probability that one of the hypotheses generated by the algorithm was generated purely from inlier points (i.e. a good estimate). This method of updating N

Algorithm 3.1 The RANSAC framework

1: **while** $M_i < M_T$ and $i < N$ **do**
2: Select s points randomly from a total of S data points.
3: Generate the ith hypothesis solution from s points.
4: Find all data points fitting the hypothesis to within some threshold distance, t. This is the consensus set M_i of inliers for the ith hypothesis.
5: Update N, increment i.
6: **end while**
7: Refine the solution by fitting to all inliers in the consensus set of the winning hypothesis.

means that the algorithm iterates until the probability that at least one good hypothesis was generated, is higher than p.

p is a set parameter (usually around 0.99 or 0.999) whilst ϵ is updated as the number of inliers in the largest consensus set so far, divided by the total number of points, S.

s is usually the minimum number of points needed - since from Equation 3.10, if s increases, then more iterations (N) are needed. For example, in the egomotion recovery system of Nistér [180], a minimal 5-point solver [181] is used to generate hypothesis estimates of the camera essential matrix within a RANSAC-type framework.

Various variants to the original RANSAC strategy exist. Some aim to speed up the performance of RANSAC or to optimally trade off between processing time and robustness. Matas and Chum's **R-RANSAC** method [155] reduces the hypothesis evaluation time via a statistical test performed on d random points - the hypothesis is evaluated on the remaining $S - d$ points only if the first d points turn out to be inliers. **Optimal R-RANSAC** [156] uses sequential decision-making theory to propose an optimal way for making this trade off, and is reported to run faster than comparable existing schemes.

Preemptive-RANSAC [180] was designed for time critical applications. It considers only a *fixed* number of hypotheses and selects the best out of these. Hypotheses are scored with a fraction of data points and those with the worst scores are rejected. The goal of this scheme is to achieve constant run-time by finding the best result within a fixed number of hypotheses, and within a scheduled time. From Equation 3.10, we know that for the same probability guarantee on performance (p), the number of models considered (N) depends on the outlier probability (ϵ). Therefore, by fixing N, Preemptive-RANSAC

implicitly assumes some lower bound on the outlier probability of the data.

Another robust estimator that can be used is the **Least-Median of Squares (LMedS)** method [84, 202]. Like RANSAC, s points are picked randomly to generate hypothesis solutions. The hypothesis is then scored against the data, and the median of the squared residuals is kept as the score (rather than the number of inliers, as in RANSAC). The rest of the scheme, is quite similar to RANSAC.

Robust M-estimators work by replacing the squared residuals of a standard least-squares cost function with a more robust loss function. That is, instead of $\min \sum_i r_i^2$, we use:

$$\min \sum_i \rho(r_i) \tag{3.11}$$

where ρ is a symmetric, positive-definite function growing sub-quadratically and having unique minimum at zero. This may be solved as a re-weighted least squares problem (see, for example, [107]). The M-estimator SAmple Consenseus (MSAC) [240] is one algorithm using this approach. Also, in the **Maximum-likelihood SAmple Consensus (MLESAC)** method of Torr and Zisserman [241], the log-likelihood of a hypothesis is used to score it. Other related work on RANSAC-type algorithms may be found in reviews such as [197].

Voting is a very robust method which has wide applications in many areas of vision [104, 54, 110]. Ballard and Kimball [15] used the generalized Hough-transform to estimate egomotion, where votes were cast in a five-dimensional (parameters of translation and rotation) space and the peak in the vote space was taken as the solution. Interestingly, the presence of independently moving objects manifested as additional peaks in the vote space, leading to multi-motion recovery in addition to a robust estimate of self-motion.

More recently, den Hollander and Hanjalic [51] used a combined RANSAC-Hough approach for egomotion recovery. The method randomly samples less than the full number of data points needed to obtain a unique solution. The result is a subspace of solutions, and votes are cast for this space of solutions; repeating many times yields a peak at the correct estimate.

Makadia et al. [154] used the Radon transform (which is closely related to the Hough transform) for performing self-motion estimation using *soft* correspondences. As opposed to hard correspondences, here, point matches are given weights indicating the probability of a correct correspondence. Voting was performed on the five-dimensional coefficient space after taking Spherical Fourier Transforms of an epipolar constraint-based cost function. The method was re-

ported to perform accurately and, in addition, it was also able to detect and estimate the motions of independently moving objects.

The drawback with most of these voting approaches is the sparsity problem caused by the high dimensional space in which voting takes place (e.g. a minimum of 5 dimensions for essential matrix recovery). The number of voting bins increases rapidly as the dimension of the solution space increases, and to fill these bins, a large number of data points is needed. If only a small amount of data is available, it is difficult to obtain good estimates with this method since the voting bins will only be filled sparsely.

A key advantage of Hough-like voting however, is that the algorithm is well-suited for implementation in parallel processing architectures. There exists much research on the efficient implementation of various Hough-like voting algorithms on parallel processing devices such as Graphical Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs). Some example publications include [133, 204, 185, 137, 123, 230]. Whilst many algorithms that are fundamentally sequential in nature may be faster when implemented on parallel hardware, naturally parallel algorithms such as Hough voting will show more significant improvements in speed.

3.1.6 What is the State-of-the-Art?

Having been presented with the bewildering array of alternative methods in the preceding sections, two questions may come to mind:

1. What is the State-of-the-Art? and

2. Do we really need yet *another* egomotion recovery method?

What appear to be the standard methods for solving the problem are as follows:

In the case of uncalibrated cameras, either the eight-point algorithm or the seven-point algorithm is used; and in the calibrated case, the five-point algorithm is used. These are typically employed within a RANSAC-type statistical framework. If accuracy is critical, the result is then used to initialize a nonlinear bundle adjustment step in order to further refine the solution (the n-point algorithm and RANSAC framework is expected to give a sufficiently good initialization such that bundle adjustment converges on the global optimum). Furthermore, these methods generally perform more accurately and stably under a larger camera

FOV.

In general, the computer vision community has focused more on discrete motion methods in the last decade. These have become stable and accurate, and are used in many practical applications involving vast numbers of images, such as movie post-production or the large-scale structure-from-motion reconstructions of buildings, monuments and entire cities.

Today, differential motion methods appear to be less popular in computer vision as they are more sensitive to noise (for the same level of noise, the signal-to-noise ratio is clearly better for a large motion than for a small one). These methods are perhaps less suitable for applications such as reconstruction and model building where accuracy is of prime concern and the processing time taken is of secondary importance. However, in many applications where the camera is undergoing differential motion, differential methods seem a more natural framework to work in. Hence they remain widely used in areas such as robotics and autonomous vehicle navigation (for example, some recent work includes [10, 50, 77, 224]).

Challenges

Whilst highly accurate and optimal self-motion estimation is possible, and there has been some progress in real-time estimation, we believe that **fast egomotion recovery in the most general situations is still not well solved**.

A multiple-frame tracking approach based on the extended-Kalman filter (EKF) gave real-time performance in the work of Jin et al. [116]. Thanh et al. [232] obtains real-time performance on panoramic cameras by attempting to identify distant features from which rotation is estimated; translation is found in a second step using nearer features. Other methods using temporal filtering include Simultaneous Localization and Mapping (SLAM) techniques such as that of Davison [50], or the approach of Mouragnon et al. [167] which uses generic camera models. However, robust real-time estimation without assumptions on the past state of the camera and without estimates or assumptions on scene depth, remains a challenge even after many decades of research.

A robust, real-time solution from two-views was presented by Nistér [180]. This approach used the five-point algorithm [181] within a preemptive-RANSAC framework. However, as discussed previously, the real-time performance of this method depends on an assumed lower-bound on outlier probability. As a result, the method may not work well for many real, complex scenes where the proportion

of outliers in the data is unknown, time-varying, and may be higher than the proportion implicitly assumed by the fixed number of hypotheses used. (The experiments in the paper involved static scenes with quite sparse SIFT features, some small proportion of which were mismatched.)

In real scenes, large proportions of outliers will arise due to the presence of moving shadows, changing illumination and multiple dynamic objects. Visual egomotion for applications such as navigating a robot through a crowd of moving pedestrians, or autonomously driving a car in a dynamic, high-speed traffic scene is extremely challenging. In order for visual egomotion recovery to work in these practical, real-time applications, robustness to large outlier and noise proportions needs to be achieved in constant time.

Our focus is to investigate this problem under the very general case of a monocular, calibrated observer moving in a general, dynamic scene of unknown depth. From two-views of the scene, we extract noisy, outlier-prone measurements of image motion and estimate the camera translation and rotation from such inputs. We will consider the problem for both differential and discrete camera motions, leading to theoretical and algorithmic contributions under both scenarios.

3.2 Relaxed Egomotion: Visual Homing

In addition to our contributions on accurate and precise egomotion estimation (which form the bulk of this thesis), we also explore the estimation and applications of approximate measures of egomotion, which we term the 'relaxed' egomotion approach.

We observe that robust egomotion estimates are useful for many applications *even when not found with great accuracy.* In human navigation, for example, a human is able to perform many tasks without the precise egomotion estimates that state-of-the-art computer vision algorithms attempt to achieve. Using only approximate estimates within a closed-loop servo control framework, humans are able to perform highly accurate actions.

One behaviour that is useful for any autonomously navigating agent is **visual homing**. Consider an agent that observes a view of the world from a particular position, G. Suppose, it is then displaced from this position (for example, to perform some task or to explore the environment). The agent now finds itself at a new position, C, where it has a different view of the world.

Comparing the current view and the stored view recorded at the home position

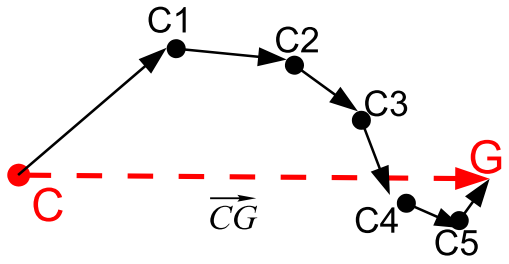


Figure 3.4: Visual homing: whilst $\overrightarrow{CG}$ gives the most direct path home, continual updates of relaxed egomotion estimates can still get a robot home, albeit with a less direct path. This reduces the computational burden of having to obtain highly accurate egomotion estimates.

G, visual homing aims to find the direction of motion that will take the agent home to G. Recovering the translation direction $\overrightarrow{CG}$, leads to successful homing and the egomotion estimation algorithms discussed previously can be applied here.

However, because fast and robust egomotion is challenging and computationally expensive, an alternative is to use a relaxed approach which still enables successful and robust homing whilst working in real-time and imposing only a small computational burden on the hardware of the agent (see Figure 3.4).

In Chapter 8, we show that **successful homing is guaranteed even with highly inaccurate estimates of translation** as long as the error is within some bound. Basically, the intuition is that as long as we can guarantee the estimate is not too wrong, moving the agent in that estimated direction will bring it closer to home instead of further away. If, at each step along the way, we can guarantee the same condition is met, then the distance between the agent and its home will monotonically decrease, and at the limit, the agent is guaranteed to reach home.

The condition necessary for this is easy to satisfy. It is also observable by the system and steps can be taken to improve the estimate if the condition is not met. We develop *robust* algorithms that home successfully within a closed-loop control framework in real, dynamic, outlier-prone environments.

In the following section, we discuss existing approaches whilst our method will be discussed in Chapter 8.

3.2.1 Visual homing algorithms

The homing direction can be taken as the estimated camera translation found using any of the egomotion methods discussed. If the distance between the current and goal positions of the agent, $|\overrightarrow{CG}|$ is large, any of the discrete motion algorithms will work but if it is small, then the differential motion algorithms are viable. Furthermore, visual servoing [108] and other approaches [132, 101] are also possible.

The general homing problem may also be solved using some metric map of the environment, where the agent can plan its return path using this map. The SLAM framework [234, 50] offers a principled way for robots to explore unknown environments and fuse the acquired estimates of scene depth into a single map encoding the 3D structure of that environment. In the robotics literature, scene depth is typically found using laser, sonar, radar or time-of-flight methods. A visual approach may use stereo cameras or multi-view reconstruction. However, whilst 3D maps are obviously invaluable for the planning and execution of many complex, higher-order tasks, they are arguably, not essential to the homing task.

Here, we focus on the class of algorithms termed as **local visual homing** methods. These do *not* rely on metric, reconstructions of the 3D world. Local visual homing algorithms are typically used within graph-like **topological maps** [78, 71, 45, 7, 69, 68, 128] where connected nodes represent nearby locations in the environment. A large FOV or panoramic image of the scene is captured at each position (which corresponds to a node) and stored. A short-range local homing algorithm is then used to drive the agent from one node to the next. By iteratively moving from node to node, long-range global homing may be accomplished by traversing the topological map.

Various local homing methods exist, including [23, 131, 140, 261, 81, 70, 35, 276] and many others, which can be found in reviews such as [251]. The elegance and simplicity of these methods make them attractive not only for robotics, but also for modeling homing behavior in insects.

Indeed, many of these algorithms were directly inspired by or aimed at explaining insect homing behaviour in ants, bees and other relatively simple biological systems. Many of these creatures do not possess stereo vision or other means of recovering 3D structure directly. Solving structure from two views of the scene is a theoretical possibility but the computational complexity of the task seems to suggest it may be beyond the neural capabilities of such systems. Likewise, these computationally cheap local homing algorithms would also be well suited

for small robots with very limited payloads and therefore a limited suite of sensors and scarce computing resources.

In particular we are interested in methods that use only the bearing angles of feature points (which in the homing literature, are called **landmarks**). Landmark points are typically image features obtained, matched and tracked with methods such as SIFT matching [145], Harris corners [85], KLT [209] and others. We focus on such 'naturally' occurring landmark points, instead of objects placed in the scene for the sole purpose of being an easily tracked and unmoving landmark.

Local visual homing approaches include the Snapshot Model of Cartwright and Collett [35], which matches sectors of the image snapshot taken at the goal position with the sectors in the current image to obtain 'tangential' and 'radial' vectors used to perform homing. Another method, the Average Landmark Vector method of Lambrinos et al. [131] computes the average of several unit **landmark vectors** (a vector pointing from the agent to some landmark point) and obtains a homing vector by subtracting the average landmark vectors at the current and goal positions. The Average Displacement Model of Franz et al. [70] and the work of Möller [165] are examples of other relevant methods.

All of the above rely on *known rotation* between the goal and current images in order to perform a correct comparison of landmark vectors observed at the current and goal positions. This rotation can be recovered from images, as in the work of Zeil et al. [276], or via some globally observable compass direction.

More recently, Bekris et al. [23] and Argyros et al. [9] proposed a *rotation independent* framework which works for any combination of start, goal and landmark positions on the entire plane. This approach uses a 'basic' and a 'complementary' control law that performs homing with three or more landmarks. Furthermore, Argyros et al. [9] demonstrated successful homing through extensive simulations (whereas the previously discussed local homing methods had neither theoretical nor empirical guarantees of successful convergence onto the home or goal location).

Loizou and Kumar [140] provided the first planar homing algorithm with a proof of convergence. One drawback is that this method requires known rotation between the goal and current robot positions. Furthermore, the landmarks are assumed to lie on the same plane as the robot (this rather unrealistic assumption also occurs in virtually all the other local homing papers cited).

It is important to note that all of the existing local homing methods assume that a correct correspondence has been found between landmarks seen in the goal and current positions. However, if some of these landmarks have moved, are

moving, were occluded, or were erroneously matched, there is often no provision to ensure successful homing in the presence of these *outlier landmarks*.

We will later see that our proposed method (Chapter 8) demonstrates provable convergence, is robust to outliers and does not require known rotation. We also work with more general scenarios where landmarks are not assumed to lie in the same plane as the robot, and where the robot may either move on the ground plane, or in 3D.

3.3 Conclusion

This chapter reviewed existing methods of egomotion estimation. We saw that they can be categorized into methods that work under the differential camera motion assumption, and methods that work in the more general, discrete motion scenario. We also reviewed homing as a relaxed version of the egomotion recovery problem.

In the following chapters, we investigate constraints and algorithms for the differential motion case in Part II and for the discrete motion case in Part III.

Part II

Differential Camera Motion

Outline of Part II

In Part II, we introduce the idea of *antipodal points* and present two different but related constraints on differential camera egomotion based on this idea. We also give complete algorithms using these constraints and demonstrate their use in real and simulated experiments.

1. **Chapter 4:** Firstly, we show that by using only *directions* of antipodal optical flow vectors, we can constrain and recover the *directions* of motion, that is the direction of translation and the axis of rotation. Later, these can be re-substituted into the equations to find the *magnitude* of the rotation angle.

2. **Chapter 5:** Secondly, we show that by *summing* the optical flow at antipodal points, the component of flow due to rotational motion will cancel, leading to a constraint on translation. After translation direction has been recovered, rotation may be found by re-substituting translation into the equations.

3. **Chapter 6:** Finally, real and simulated experiments are conducted to investigate the performance of algorithms based on these two different approaches. We compare the performance of the two with each other, and with the state-of-the-art.

Chapter 4

Egomotion from Antipodal Flow Directions

We first consider the case of *differential* camera motion, that is, the translation and rotation between two views of the camera, is assumed to be small. (Discrete camera motions are dealt with in Part III.)

Throughout our discussion of differential camera motion, we will use the term 'optical flow' to refer to the vector field representation of image motion, with the understanding that it may be measured using either dense optic flow methods or point correspondence methods (refer 2.2 page 21).

Here we explore the geometry of large field-of-view (FOV) vision and of image-spheres in order to exploit their properties for egomotion estimation. Starting from the differential motion equations relating camera motion and image motion, we will derive new constraints on camera motion based on the idea of **antipodal points**, which are found only in large FOV cameras.

In this chapter, we will consider pairs of optical flow vectors measured at antipodal points on the image sphere. Using only the *directions* of the flow vectors (and not their magnitudes), we present novel constraints for recovering the *directions* of motion - that is, the direction of translation and the axis of rotation. Note that whilst the *magnitude* of the rotation angle is not directly recovered, it may be found in a second step by re-substituting the translation direction and rotation axis into the motion equations. Using these constraints, we also develop an efficient and practical algorithm that is robust to noise and outliers.

Later, in Chapter 5, we will detail a different but closely related constraint on egomotion, whilst Chapter 6 will focus on experiments that evaluate the perfor-

mance of the algorithms introduced in Chapters 4 and 5.

4.1 What are Antipodal Points?

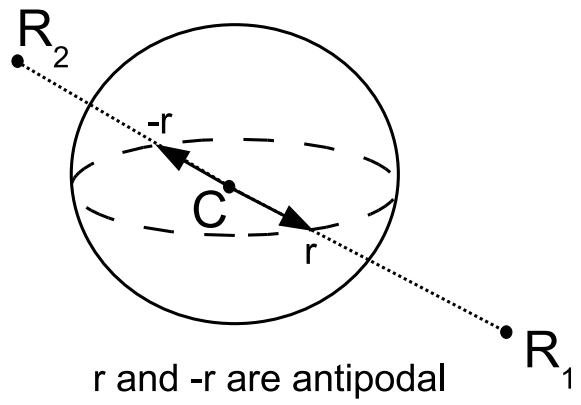


Figure 4.1: An illustration showing antipodal points. The world points $\mathbf{R}_1$ and $\mathbf{R}_2$ are imaged by the camera centered on $\mathbf{C}$. The ray $-\mathbf{r}$ is antipodal to $\mathbf{r}$. Also shown is the image-sphere centered on $\mathbf{C}$.

Figure 4.1 shows the geometry of antipodal points. With the origin of the camera coordinate frame being the camera centre, $\mathbf{C}$, we say that two rays are antipodal if they are collinear and point in opposite directions, i.e. rays $\mathbf{r}$ and $-\mathbf{r}$ are antipodal. These antipodal rays intersect the image sphere at points which are diametrically opposite each other on the sphere - i.e. at antipodal points. (The North and South poles of the earth are an example of antipodal points.)

If a pair of world points ($\mathbf{R}_1$ and $\mathbf{R}_2$) happen to lie on these antipodal rays as shown in the diagram, we can write down constraints on the camera motion based on the image motion observed at these antipodal points.

In practice, antipodal points are present in any central camera (i.e. single centre of projection, see Section 2.1.2, page 17) with a FOV larger than 180°. Figure 4.2 shows some examples. Depicted, is the schematic of a camera rig consisting of multiple cameras pointing in different directions but having approximately a single center of projection. Also shown is a catadioptric camera where a curved mirror enhances the FOV. In the illustration, the rays $\mathbf{r}$ and $-\mathbf{r}$ show an example of points that are antipodal with respect to the projection center.

In practice, points which are sufficiently close to antipodal (within some threshold) may be approximated as antipodal without introducing too much er-

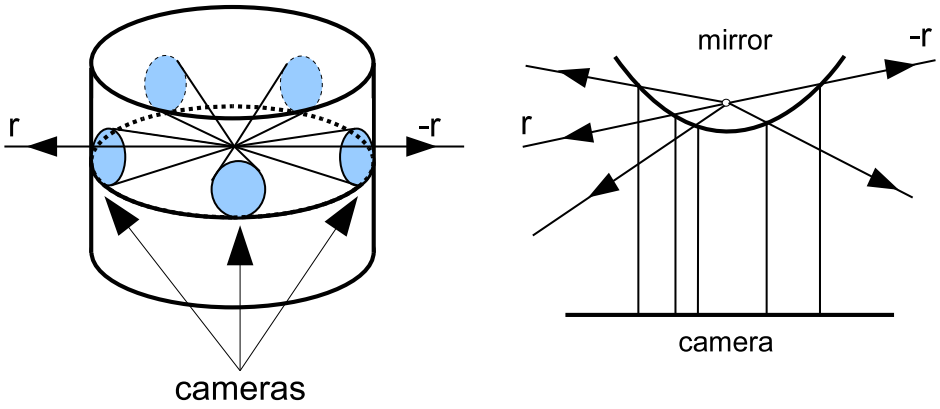


Figure 4.2: Antipodal points in some large Field-of-View cameras, including a rig with multiple cameras pointing in different directions and a catadioptric camera. $\mathbf{r}$ and $-\mathbf{r}$ are an example of antipodal rays occurring in a camera.

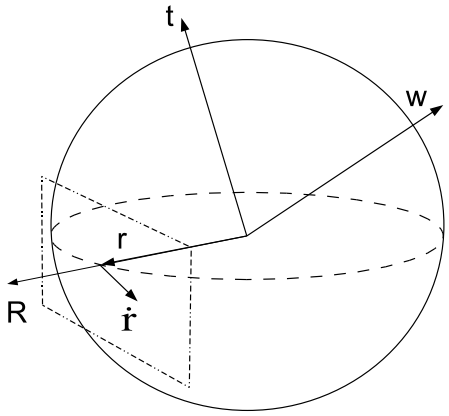


Figure 4.3: Optical flow on the image sphere.

ror. Experiments will later show that antipodal points occur in real images in quantities much larger than the minimum number required.

4.2 Background

In Chapter 3, we first introduced the optical flow equation for a spherical imaging surface (reproduced below for the reader's convenience):

$$\begin{aligned} \dot{\mathbf{r}} &= f_{tr}(\mathbf{r}) + f_{rot}(\mathbf{r}) \\ &= \frac{1}{|\,\mathbf{R}\,|}((\mathbf{t}\cdot\mathbf{r})\mathbf{r} - \mathbf{t}) - \mathbf{w}\times\mathbf{r} \end{aligned} \tag{4.1}$$

The flow at an image point $\mathbf{r}$ corresponding to scene depth $|\ \mathbf{R}\ |$ is given by $\dot{\mathbf{r}}$ when the camera translates in direction $\mathbf{t}$ whilst simultaneously rotating with rotation $\mathbf{w}$. This equation does not model noise. Figure 4.3 shows an illustration of the set up.

Note that Equation 4.1 consists of two terms - the term due to translational motion, $f_{tr}(\mathbf{r})$, and the term due to rotational motion, $f_{rot}(\mathbf{r})$. This means that the equation is linear in translation and rotation, a consequence of the first order Taylor approximation under the differential motion assumption (refer Section 3.1.2, page 35).

Recall that the scale of translation is not recoverable [93], hence there will be five unknown motion parameters in total to estimate - two unknowns for translation direction and three unknowns for the rotation. Also, the relative depth to every point in the scene is unknown - that is, scene depth, $\mathbf{R}$, is a function of $\mathbf{r}$. Consequently, a straightforward least squares solution is impossible since the system of equations is under-constrained. A standard method for overcoming this is to introduce additional constraints, for example, the epipolar constraint.

Here, we suggest an alternative approach using constraints on the directions of motion arising from the spherical geometry of antipodal points. We show in Section 4.3, that from the flow at a single pair of antipodes, one can obtain geometric constraint regions on the translation direction and on the rotation axis. Next, Section 4.3.2 shows that many pairs of antipodal points will give many such constraint regions and the translation direction or rotation axis will lie in the intersection of all these constraints. Then, we develop a practical and robust algorithm based on Hough-reminiscent voting in Section 4.4.

4.3 Constraints from Antipodal Flow Directions

For the following discussion, refer to Figure 4.4. Let us consider two antipodal points $\mathbf{r}_1$ and $\mathbf{r}_2$ on the image sphere. There exist infinitely many great circles that will pass through both points (just as an infinite number of longitudinal circles pass through the North and South poles of the earth), and we consider one of them, the circle C.

The flow vector at a point $\mathbf{r}$ inhabits the tangent plane to the image sphere at that point. Let us define a unit vector $\mathbf{c}$, at a point $\mathbf{r}$ that is tangent to the image sphere (therefore, $\mathbf{c}$ and the flow vector $\dot{\mathbf{r}}$ both lie on the same tangent plane as depicted in Figure 4.4). We further require vector $\mathbf{c}$ to lie in the plane of the

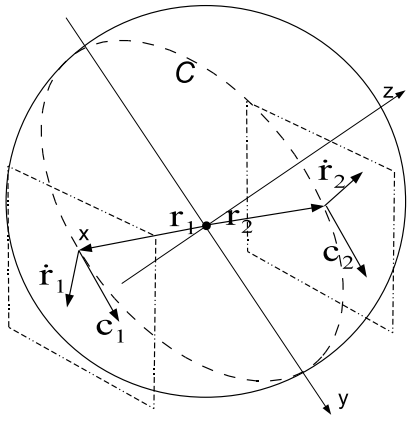


Figure 4.4: $\dot{\mathbf{r}_1}$, $\mathbf{c}_1$ and $\dot{\mathbf{r}_2}$, $\mathbf{c}_2$ lie on two parallel tangent planes if $\mathbf{r}_2 = -\mathbf{r}_1$ (i.e. they are antipodal).

great circle C. Therefore, $\mathbf{c}$ is both tangent to great circle C and also tangent to the image sphere.

Let the projection of the flow vector $\dot{\mathbf{r}}$ onto the vector $\mathbf{c}$ be termed as *projected flow* and denoted as $proj(\dot{\mathbf{r}})$ where $proj(\dot{\mathbf{r}}) = \dot{\mathbf{r}} \cdot \mathbf{c}$. Then, we may show the following lemma:

Lemma 4.1. If the projected flow vectors, $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$, at two antipodal points $\mathbf{r_1}$ and $\mathbf{r_2}$ are in the same direction (same sign), there must exist a component of translation, t_c, that is parallel and in the opposite direction to $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$ (Figure 4.5a). Likewise, if $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$ are in opposite directions (opposite signs), a component of rotation, w_c, that is normal to the plane of the great circle C must exist (Figure 4.5b).

Proof. Without loss of generality, let us choose a Cartesian coordinate frame such that our antipodal points $\mathbf{r}_1$ and $\mathbf{r}_2$ are at $[1\ 0\ 0]^T$ and $[-1\ 0\ 0]^T$ and such that the great circle C lies on the x-y plane. From Equation 4.1, we substitute for $\mathbf{r}_1$ and $\mathbf{r}_2$, and expand the translation vector $\mathbf{t}$ as its magnitude multiplied by the unit vector in the direction of translation, that is, $\mathbf{t} = |\mathbf{t}|\ [t_x\ t_y\ t_z]^T$. Also, the instantaneous rotation vector is given by $\mathbf{w} = [w_x\ w_y\ w_z]^T$. Furthermore, we project the flow vectors $\dot{\mathbf{r}_1}$ and $\dot{\mathbf{r}_2}$ onto $\mathbf{c}_1$ and $\mathbf{c}_2$ which in this case are both simply $[0\ 1\ 0]^T$ to obtain:

$$\begin{aligned} proj(\dot{\mathbf{r}_1}) &= -\ |\ \frac{\mathbf{t}}{\mathbf{R}_1}\ |\ t_y - w_z \\ proj(\dot{\mathbf{r}_2}) &= -\ |\ \frac{\mathbf{t}}{\mathbf{R}_2}\ |\ t_y + w_z \end{aligned} \tag{4.2}$$

where $\mid \mathbf{R}_1 \mid$ and $\mid \mathbf{R}_2 \mid$ are the depths of the scene at $\mathbf{r}_1$ and $\mathbf{r}_2$. Note that the term $\mid \mathbf{t} \mid t_y$ comes from expressing the translation vector as its magnitude multiplied by its unit direction vector (so Equation 4.2 is actually linear in translation, as Equation 4.1 would imply). The unknowns are reduced to t_y, and w_z, which are mutually orthogonal ($\mid \frac{\mathbf{t}}{\mathbf{R}_i} \mid$ for $i = 1, 2$, is also unknown and cannot be recovered).

Here, we are interested in the directions (i.e. the sign) of $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$. Equation 4.2 shows that the projected flow consists of two components, the translational component, $\mid \frac{\mathbf{t}}{\mathbf{R}_i} \mid t_y$ (for $i = 1, 2$), and the rotational component, w_z. We find that the translational components of both $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$ act in the same direction because they have the same sign whilst the rotational components have opposite signs and therefore, act in opposing directions.

Logically, we can conclude that if we observed $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$ to have the same sign (same direction), then some translational component of projected flow, $\mid \frac{\mathbf{t}}{\mathbf{R}_1} \mid t_y$, must exist. This is because the rotational component of projected flow, w_z, acts in opposite directions at $\mathbf{r_1}$ and $\mathbf{r_2}$ and, on its own, could never give rise to what we observe.

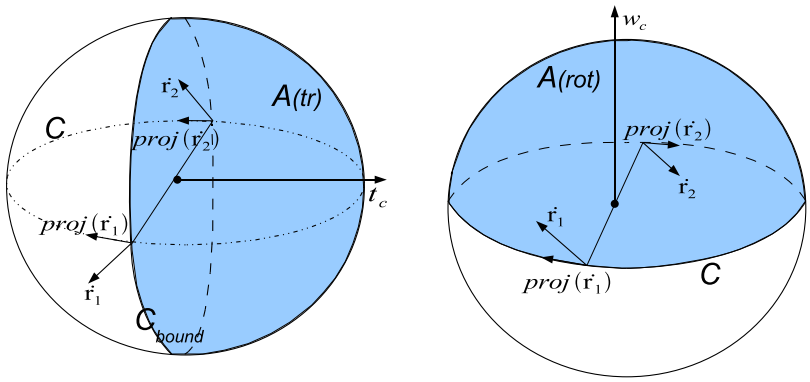
Similarly, if we observed $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$ to be in opposite directions (opposite signs), then there must exist some rotational component of flow that is causing it as the translational component could not, on its own, give rise to the observed signs. These conclusions hold true for any general pair of antipodal points since the coordinate frame can always be chosen such that it is aligned conveniently in the above manner, leading us to the statements of Lemma 4.1. □

This result trivially leads to constraints on egomotion as per Theorem 4.1 (illustrated by Figure 4.5):

Theorem 4.1. If the projected flow vectors, $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$, at two antipodal points $\mathbf{r_1}$ and $\mathbf{r_2}$ are in the same direction (same sign), then the translation is constrained to lie in a hemisphere bounded by a great circle C_{bound} that passes through $\mathbf{r_1}$ and $\mathbf{r_2}$ and is perpendicular to the great circle C. The hemisphere is such that $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$ point away from it.

Likewise, if $proj(\dot{\mathbf{r}_1})$ and $proj(\dot{\mathbf{r}_2})$ are in opposite directions (opposite signs), the direction of the rotation axis is constrained to lie in a hemisphere bounded by C. Which side of C this hemisphere lies on is determined by the cross product in the rotational component of flow, $-\mathbf{w} \times \mathbf{r}$.

Proof. From Lemma 4.1, t_c is parallel to, and lies in the opposite direction to $proj(\dot{r_1})$ and $proj(\dot{r_2})$. The translation, $\mathbf{t}$, is then constrained to lie in a hemispherical region, $A(tr)$, shown in Figure 4.5. If $\mathbf{t}$ was outside $A(tr)$, then t_c



(a) (b)

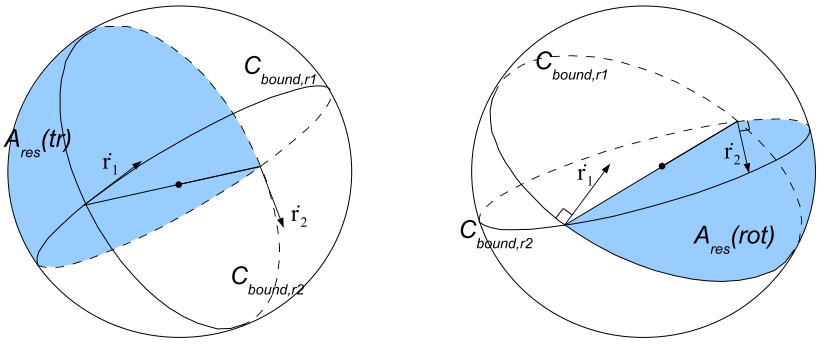
Figure 4.5: Hemispheres constraining directions of motion. (a) If t_c exists, the translation direction is constrained to lie in the shaded hemisphere $A(tr)$. (b) The rotation axis is constrained to lie in the hemisphere $A(rot)$. Since $proj(\dot{\mathbf{r}}_\mathbf{1})$ and $proj(\dot{\mathbf{r}}_\mathbf{2})$ are in a clockwise configuration, the direction of w_c is upward due to the cross product, $-\mathbf{w} \times \mathbf{r}$ (the left hand grip rule).

would be parallel to, but in the *same* direction as the projected flows $proj(\dot{r}_1)$ and $proj(\dot{r}_2)$; thus contradicting Lemma 1. This hemisphere, $A(tr)$ is bounded by the great circle, C_{bound}, obtained by the intersection of a plane with normal vector t_c with the image sphere. Similarly, for the rotation axis to have a component in the direction w_c, it must lie in a similar hemispherical region $A(rot)$. □

4.3.1 The Constraints from All Great Circles Passing Through a Pair of Antipodal Points

In Theorem 4.1 we saw that a great circle passing through a pair of antipodal points will yield a hemispherical region, $A(tr)$, that constrains the translation and another region, $A(rot)$, that constrains the rotation axis. Since there exist infinitely many great circles that will pass through these two antipodal points, an infinite set of hemispherical constraint regions, $A_1, A_2, \cdots, A_i, \cdots$ will result. As the motion must simultaneously satisfy all these constraints, the direction of motion must lie within the intersection of all these hemispherical constraint regions, $A_{res} = A_1 \cap A_2 \cap \cdots \cap A_i \cdots$.

Let C_i be the i^{th} great circle in the infinite set of great circles passing through $\mathbf{r}_1$ and $\mathbf{r}_2$. Associated with every C_i, is the vector $\mathbf{c}_i$ that is tangent to the sphere



(a) (b)

Figure 4.6: Lunes constraining directions of motion. (a) Shaded lune $A_{res}(tr)$ is the resultant constraint region on translation (b) $A_{res}(rot)$ is the resultant constraint region on rotation.

and lying in the plane of the circle C_i as defined earlier. The intersection of constraints can then be described in Theorem 4.2 and illustrated by Figure 4.6:

Theorem 4.2. The optical flow $\dot{\mathbf{r}}_\mathbf{1}$ and $\dot{\mathbf{r}}_\mathbf{2}$ observed at any two antipodal points $\mathbf{r}_1$ and $\mathbf{r}_2$ on the image sphere will give rise to a constraint on the translation direction and a constraint on the rotation axis.

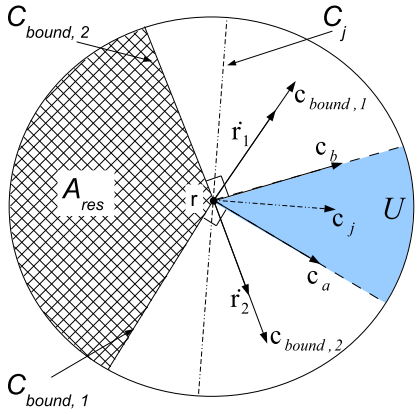
The translation direction lies in a region on the sphere bounded by great circles C_{bound,r_1} and C_{bound,r_2} that pass through $\mathbf{r}_1$ and $\mathbf{r}_2$ and are such that the associated $\mathbf{c}_i$ vectors are $\mathbf{c}_{bound,r1} = \dot{\mathbf{r}}_1$ and $\mathbf{c}_{bound,r2} = \dot{\mathbf{r}}_2$.

The direction of the rotation axis lies in a region on the sphere bounded by great circles C_{bound,r_1} and C_{bound,r_2} such that $\mathbf{c}_{bound,r1}$ is perpendicular to $\dot{\mathbf{r}}_1$ and $\mathbf{c}_{bound,r2}$ is perpendicular to $\dot{\mathbf{r}}_2$.

Theorem 4.2 can be shown using a geometrical argument. For simplicity, we consider only the vectors inhabiting the two tangent planes to the sphere at $\mathbf{r}_1$ and $\mathbf{r}_2$. They are the flow vectors and the $\mathbf{c}_i$ vectors that are associated with the great circles. As the two tangent planes are parallel in $\mathbb{R}^3$, we can represent everything in just one plane, as is done in Figure 4.7. Every great circle passing through a point $\mathbf{r}$ is characterized completely by $\mathbf{r}$ and $\mathbf{c}_i$, so in Figure 4.7, great circles C_i and vectors $\mathbf{c}_i$ are synonymous.

Recall that $proj_i(\dot{\mathbf{r}}) = \mathbf{c}_i \cdot \dot{\mathbf{r}}$. So, for the case of translation, only the great circles with $\mathbf{c}_i$ satisfying the following condition will give constraints on translation:

$$sgn\,(proj(\dot{\mathbf{r}}_\mathbf{1})) > 0 \quad and \quad sgn\,(proj(\dot{\mathbf{r}}_\mathbf{2})) > 0 \tag{4.3}$$



(a)

Figure 4.7: The two tangent planes containing $\dot{\mathbf{r}}_\mathbf{1}$, $\dot{\mathbf{r}}_\mathbf{2}$, $\mathbf{c}_1$ and $\mathbf{c}_2$ are parallel to the plane of the page. Intersection of all hemispherical constraints on translation is the lune A_{res}.

That is to say, the angles between $\dot{\mathbf{r}}_\mathbf{1}$ and $\mathbf{c}_i$ and between $\dot{\mathbf{r}}_\mathbf{2}$ and $\mathbf{c}_i$ must both be less than $\pi/2$ (i.e. acute angles). Such $\mathbf{c}_i$ vectors lie in the shaded region U in Figure 4.7. Consider the two great circles C_a and C_b with $\mathbf{c}_i$'s exactly perpendicular to $\dot{\mathbf{r}}_\mathbf{1}$ and $\dot{\mathbf{r}}_\mathbf{2}$. Let their $\mathbf{c}_i$ vectors be $\mathbf{c}_a$ and $\mathbf{c}_b$. From Theorem 4.1, we obtain two hemispherical regions constraining translation that are bounded by C_{bound1} and C_{bound2} which are again perpendicular to C_a and C_b. This means that the $\mathbf{c}_i$ vectors for C_{bound1} and C_{bound2} are actually $\mathbf{c}_{bound,1} = \dot{\mathbf{r}}_1$ and $\mathbf{c}_{bound,2} = \dot{\mathbf{r}}_2$. The translation must lie within the intersection of these two hemispherical regions - the region A_{res} of Figure 4.7.

In fact, C_{bound1} and C_{bound2} are the boundary cases. All other great circles that give valid constraints on translation - those with $\mathbf{c}_i$'s lying between $\mathbf{c}_a$ and $\mathbf{c}_b$ - will give rise to hemispherical regions that contain A_{res}. For example, see the great circle represented by the vector $\mathbf{c}_j$ in Figure 4.7. It gives rise to the hemispherical region bounded by the great circle C_j and that hemisphere contains A_{res}. So the A_{res} bounded by C_{bound1} and C_{bound2} is the smallest such region that satisfies all the constraints. In geometry, these wedge-shaped constraint regions are termed *lunes*.

To show the case of constraining rotation, simply reiterate the previous argument, but now let $\mathbf{c}_{bound,1}$ be perpendicular to $\dot{\mathbf{r}}_1$ and $\mathbf{c}_{bound,2}$ be perpendicular to $\dot{\mathbf{r}}_2$.

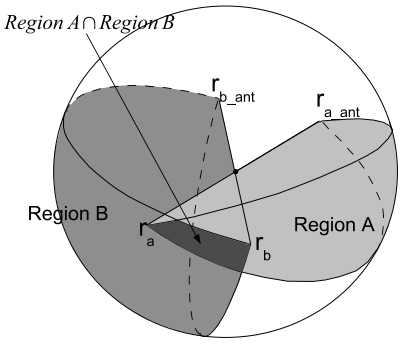


Figure 4.8: Point $\mathbf{r}_a$ and its antipode $\mathbf{r}_{a\_ant}$ gave rise to constraint region A. Then, point $\mathbf{r}_b$ was chosen such that it was inside region A. $\mathbf{r}_b$ and $\mathbf{r}_{b\_ant}$ gave rise to region B. Region A $\cap$ region B is guaranteed to be convex and smaller in size than regions A and B.

4.3.2 Motion Estimation with Constraints from N Antipodal Point Pairs

Thus far, we have shown that once the direction of flow $\dot{\mathbf{r}}_1$ and $\dot{\mathbf{r}}_2$ at antipodal points are computed, finding the boundaries of the lune shaped constraint regions is trivial. N pairs of antipodal points on the image sphere will give rise to N constraint regions for the translation direction and N regions for the rotation axis direction. The intersection of these will give some estimate of the directions of motion. By sampling sufficiently many antipodal points from all over the image sphere, this will give a global estimate of motion.

However, does this method converge to the true directions of motion as N becomes large? We argue that if the next antipodal point pair is always chosen such that one of the points lie inside the current intersection of all previously obtained constraint regions, the intersection of the new constraint with the current intersection region will be a convex region smaller than the current region (Figure 4.8).

Lunes are convex regions. Since the intersections of convex regions are themselves convex, the intersection of the lunes must also be convex. Let the region of intersection from all previous lunes be A and the constraint region from a newly chosen antipodal point pair be B. One of the points in this pair is P, so $P \in B$. Let us pick P subject to two conditions. Firstly, it is inside region A, that is, $P \in A$ and therefore $P \in (B \cap A)$. Secondly, P is not on the boundary of A, that is, there is an open ball, C, around P such that $C \subset A$. Since P, the antipodal

point, lies on the boundary of B there exists some point Q in the neighborhood of P such that $Q \in A$ but $Q \notin B$, so $Q \notin (B \cap A)$. Therefore, $(A \cap B) \subset A$, i.e. the intersection of the original constraint region and the lune is smaller than the original region.

Using such a scheme to choose the antipodal point pairs, as N becomes large, the constraint region will converge to the true motion direction. Thus, for the noise-free case of dense optical flow, we have a convergent algorithm.

In practice, we need to incorporate Hough-like voting, simulated annealing or other such methods to account for noise, outliers and independently moving objects that would give erroneous constraint regions. Furthermore, once the solution has been constrained to some region smaller than a hemisphere, a gnomonic projection can map great circles on the sphere onto straight lines on a plane and intersecting lunes onto polygons. This reduces the convex problem into a linear one for which linear programming methods are a possible solution.

Once the translation and rotation axis have been recovered, estimating the magnitude of rotation is easy and several methods are possible. Here is one suggestion: $\dot{\mathbf{r}} = f_{tr}(\mathbf{r}) + f_{rot}(\mathbf{r})$ (Equation 4.1) is a vector sum triangle. We now know the directions of $f_{tr}(\mathbf{r})$ and $f_{rot}(\mathbf{r})$, that is, we know the angles between the sides of that triangle. Thus, magnitude of $f_{rot}(\mathbf{r})$ can be found by the sine rule of triangles. Repeating for several points on the image, an overconstrained system of linear equations is obtained and a least squares solution to rotation angle may be found.

4.4 Algorithm and Implementation

The quickly converging method earlier described could be used if speed was critical, but here, we use a simple but robust vote-based algorithm summarized in Algorithm 4.1. The steps for estimating translation and for finding the rotation axis are similar.

Our implementation uses coarse-to-fine voting. First, we pick the flow at M_{coarse} pairs of antipodes and obtain M_{coarse} constraints as detailed in Theorem 4.2. Having divided the image sphere into a two-dimensional voting table according to the azimuth-elevation convention, we proceed to vote in each lune or constraint region obtained. This is the coarse voting stage and the area on the sphere represented by a voting bin can be fairly large. The bin with maximum votes is then found, giving us a coarse estimate of either the translation or the

Algorithm 4.1 Estimating the translation (or rotation axis) by voting

1: **for** $i = 1$ to M_{coarse} **do**
2: Select a pair of antipodes $\mathbf{r_1}$, $\mathbf{r_2}$ with optical flow $\dot{\mathbf{r}}_\mathbf{1}$, $\dot{\mathbf{r}}_\mathbf{2}$.
3: Obtain the constraint region on translation (or rotation axis) and the great circles bounding the lune as per Theorem 4.2.
4: The spherical coordinates (θ, ϕ) of points on the lune is calculated and coarse voting is performed.
5: **end for**
6: Find the bin(s) with maximum votes. This gives the coarse estimate.
7: Choose a projection plane at or close to the coarse estimate.
8: **for** $j = 1$ to M_{fine} **do**
9: Repeat steps 2 to 3 to obtain the lune and its bounding great circles.
10: Project the lune onto the projection plane. This maps the bounding great circles onto straight lines.
11: Vote on the projected lune on the plane.
12: **end for**
13: Find the bin(s) with maximum votes. This gives the fine estimate.

rotation axis. If more than one bin has maximum votes, the average direction is used instead. An example of the result of coarse voting on the sphere is shown in Figure 4.9.

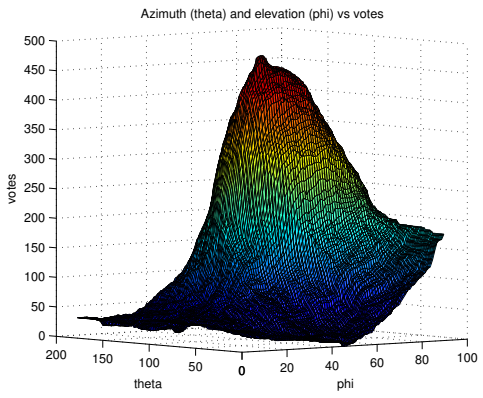


Figure 4.9: Voting on the sphere.

For the fine voting stage, a region of the image sphere in the neighborhood of the coarse estimate is projected onto a plane that is tangent to a point on the sphere (refer Figure 4.10). The point can be at the location of the coarse estimate found earlier or anywhere near it. The center of projection is the sphere center

and the mapping will project points on the surface of the sphere onto the tangent plane. Let the sphere centre be the origin and $\mathbf{r_{sph}}$ be a point that lies on the unit image sphere. If the plane is tangent to the sphere at a point $\mathbf{n}$, then $\mathbf{n}$ is also a unit normal of that plane. In that case, the projection of $\mathbf{r_{sph}}$ onto the plane is a point, $\mathbf{r_{pln}}$, that lies on the tangent plane and is given by $\mathbf{r_{pln}} = \mathbf{r_{sph}}/(\mathbf{r_{sph}} \cdot \mathbf{n})$.

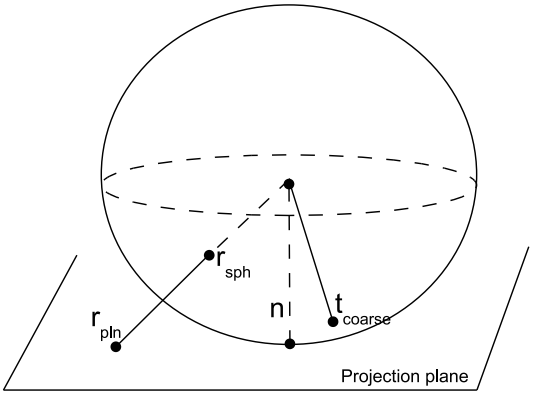


Figure 4.10: Projection of a point $\mathbf{r_{sph}}$ which lies on the image sphere, to a point $\mathbf{r_{pln}}$ which lies on the plane that is tangent to the sphere at point $\mathbf{n}$. $\mathbf{t}_{coarse}$ is the coarse estimate of translation found after the coarse voting stage (the same applies for estimating rotation axis).

This is called a gnomonic projection [44]. This projection will map great circles on the sphere onto straight lines on the plane[1]. The motivation for performing the fine voting stage on a projection plane rather than on the image sphere is the greater efficiency and simplicity in voting on the plane.

After each voting stage, the centre of mass of the bin(s) with the highest votes gives the Maximum Likelihood estimate of motion direction. If necessary, the planar voting stage can be repeated for progressively finer scales such as with the schemes proposed by [11, 109]. We chose a vote-based approach because it is highly robust to noise and outliers in the flow. Its disadvantage is that accuracy is limited by voting bin resolution.

Other variations on voting are also possible. For example, many optical flow calculation methods return a covariance matrix which gives an uncertainty measure of the accuracy of the calculated flow. Hence, lunes resulting from flow with high accuracy can be given higher weights during voting compared to lunes

[1]Note: A stereographic projection is subtly different from the gnomonic projection. The centre of projection for the former may lie anywhere on the surface of the sphere, whilst the projection centre for the latter is the sphere centre.

arising from less accurate flow. In our experiments however, we found that the unmodified version of the algorithm was sufficiently robust and accurate and we did not require any such extra measures.

4.5 Discussion and Summary

In summary, our method directly recovers both the translation and rotation axis using two sets of constraints. These constraints on translation and rotation are geometrically dual to each other. Both the translation and rotation axis can be recovered in parallel since they can be estimated separately from each other. Once these directions of motion are known, the magnitude of rotation may also be recovered.

Features of the voting algorithm: The algorithm we presented is a 'non-numerical' approach - that is, the constraints are obtained without numerically solving equations. In fact, no mathematical operations more complex than that required for voting will be performed. Furthermore, run-time is constant with increasing noise and outliers (see experiments in Chapter 6), and that, together with the naturally parallelizable nature of voting makes it potentially viable for implementation as a real-time egomotion recovery system.

Directions of flow: An interesting concept in this approach was the use of only the directions of flow vectors, and not their magnitudes. This leads to weaker, lune-shaped constraints. However, by using only directions of flow, we are able to extend antipodal constraints to rotation as well (in the next chapter, we derive constraints using both the directions and magnitudes of antipodal flow, but that only constrains translation direction). Other benefits include improved robustness since it is independent of noise in the magnitude of flow vectors - only noise in the direction of flow affects it. It is also theoretically interesting because our constraint regions (the lunes) arise immediately from the flow vectors without further mathematical processing, be it addition, multiplication or more complex operations.

Fail-soft. Insufficient antipodal points does not break the algorithm - the motion only becomes more ambiguous. The vote-table is a probability map, where regions of higher votes indicate higher probability of the motion being in that direction. Ambiguity occurs when constraints fail to further disambiguate motion, and a patch of directions has tied votes. The true solution lies within that patch with maximum probability given the available antipodes. This is

better than many other motion methods that return a nonsensical number when they fail.

Chapter 5

Translation from Flow at Antipodal Points

In the last chapter, we presented constraints on both the translation direction as well as the axis of rotation using purely directions of antipodal flow. In this chapter, we consider a different approach which uses the *magnitude* of the antipodal flow vectors as well. Once again, recall that we are dealing with differential camera egomotion (discrete camera motions are considered in Part III).

In brief, we will use the optical flow at two antipodal points to constrain the translation direction to lie on a great circle (that is, the intersection of a plane through the camera center with the image sphere). The rotational contribution to the measured optical flow is eliminated, leaving an equation dependent only on the translational component of motion. Multiple such constraints are then used to estimate the translation uniquely (up to a scale, of course).

Compared to the approach in the last chapter, we obtain much stronger constraints on the direction of translation (we constrain it to a great circle, instead of a lune). This happens because we use all the information from the antipodal flow - i.e. both the magnitudes as well as the directions. However, the downside is that no constraints on rotation arise directly since we effectively cancel the rotational effects with our method. Even so, we will show that once translation has been found, a least squares estimate of rotation that is *robust* to outliers would not be difficult to recover.

We note that Thomas and Simoncelli [233] observed an antipodal point constraint that is related to the constraint presented in this chapter. However, significant differences in the theoretical derivation and in the resulting constraint exist between the method of [233], and our work (see Section 5.1.1). Our constraint

is somewhat simpler to derive and use compared to [233] and it succeeds in certain scene depth configurations where theirs fails. Both the constraint presented here and that of [233] may be thought of as special cases of the linear subspace methods investigated by Jepson and Heeger [114].

5.1 Removing Rotation by Summing Flow at Antipodal Points

Once again, we consider the optical flow, $\dot{\mathbf{r}}_1$, at some point, $\mathbf{r}_1$, on an image sphere:

$$\dot{\mathbf{r}_1} = \frac{1}{\mid \mathbf{R}(\mathbf{r}_1) \mid}((\mathbf{t} \cdot \mathbf{r}_1)\mathbf{r}_1 - \mathbf{t}) - \mathbf{w} \times \mathbf{r}_1 \tag{5.1}$$

Suppose there is another point, $\mathbf{r}_2$, that is antipodal to $\mathbf{r}_1$, that is, $\mathbf{r}_2 = -\mathbf{r}_1$. Then the optical flow at the second point is:

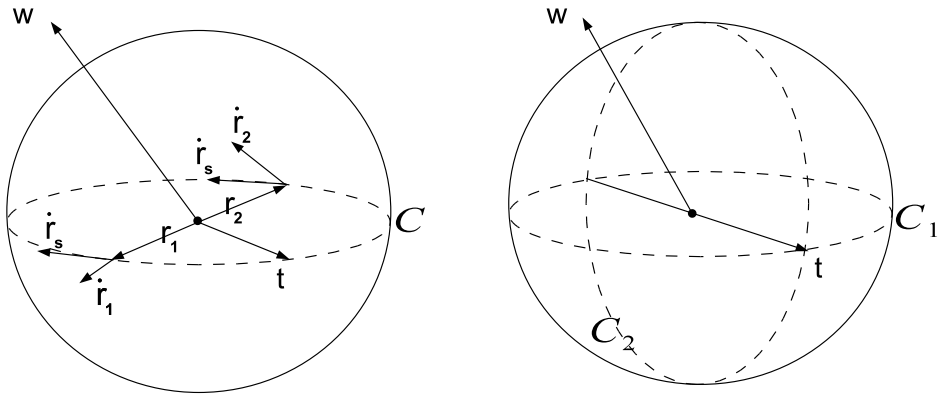
$$\begin{aligned} \dot{\mathbf{r}_2} &= \frac{1}{\mid \mathbf{R}(\mathbf{r}_2) \mid}((\mathbf{t} \cdot \mathbf{r}_2)\mathbf{r}_2 - \mathbf{t}) - \mathbf{w} \times \mathbf{r}_2 \\ &= \frac{1}{\mid \mathbf{R}(-\mathbf{r}_1) \mid}((\mathbf{t} \cdot \mathbf{r}_1)\mathbf{r}_1 - \mathbf{t}) + \mathbf{w} \times \mathbf{r}_1 \end{aligned} \tag{5.2}$$

By summing Equations 5.1 and 5.2, we have an expression that arises purely from the translational component of motion. The rotational components cancel out.

$$\begin{aligned} \dot{\mathbf{r}}_\mathbf{s} &= \dot{\mathbf{r}_1} + \dot{\mathbf{r}_2} \\ &= (\frac{1}{\mid \mathbf{R}(\mathbf{r}_1) \mid} + \frac{1}{\mid \mathbf{R}(-\mathbf{r}_1) \mid})((\mathbf{t} \cdot \mathbf{r}_1)\mathbf{r}_1 - \mathbf{t}) \\ &= A((\mathbf{t} \cdot \mathbf{r}_1)\mathbf{r}_1 - \mathbf{t}) \end{aligned} \tag{5.3}$$

From Equation 5.3, we see that vectors $\mathbf{r}_1$, $\dot{\mathbf{r}}_\mathbf{s}$ and $\mathbf{t}$ are coplanar. The normal of that plane is given by $\mathbf{r}_1 \times \dot{\mathbf{r}}_\mathbf{s}$ where $\times$ denotes the vector cross product. The direction of translation, $\mathbf{t}$, lies on that plane. The intersection of that plane with the image sphere gives a great circle. See Figure 5.1(a) for an illustration.

By picking another pair of antipodal points (that do not lie on the first great circle) and repeating, we obtain a second such plane or great circle. The intersection of the two planes or great circles gives an estimate of translation, $\mathbf{t}$. See Figure 5.1(b) for an illustration.



(a) (b)

Figure 5.1: (a) Summing the flow $\dot{\mathbf{r}}_1$ and $\dot{\mathbf{r}}_2$ yields the vector $\dot{\mathbf{r}}_s$, which, together with $\mathbf{r}_1$ (or $\mathbf{r}_2$), gives rise to a plane on which $\mathbf{t}$ is constrained to lie. The intersection of that plane with the sphere is the great circle C. (b) From two pairs of antipodal points, two great circles C_1 and C_2 are obtained. $\mathbf{t}$ is the intersection of these two circles.

Of course, the intersection of the two great circles actually yields two points corresponding to $\mathbf{t}$ and $-\mathbf{t}$. To disambiguate between the two, one may pick one of the two points and calculate the angle between it and the vector $\dot{\mathbf{r}}_s$. If the angle is larger than $\pi/2$ radians, then that point is $\mathbf{t}$. Otherwise, it is $-\mathbf{t}$.

5.1.1 Comparison with Thomas and Simoncelli

The underlying principle here was first observed in Thomas and Simoncelli [233]. However, major differences differentiate this work from that of [233]. The approach used there defined *angular flow*, which is obtained by taking the cross product of optical flow at a point with the direction of that point. In effect, a dual representation of flow is obtained and [233] showed that if the angular flow at two antipodal points was *subtracted* from each other, the rotational component would vanish.

Both the constraint developed here and the one in [233] probably stem from a similar geometrical property inherent to antipodal points but the methods by which they were derived and the final results are quite different. The constraint in this paper does not require the transformation of optical flow into angular flow as in [233], a step which incurs an additional cross product.

Furthermore, the method of [233] requires a *subtraction* of angular flow at antipodal points and thus, that method fails when the world points projecting onto the antipodal points on the imaging device are at equal distances from the camera centre (subtracting the angular flow in that case yields zero) - a problem observed by the authors of that paper. An example of this would be the distances to the two opposite walls when the camera is exactly in the middle of a corridor (note that this is an important and commonly occurring scenario in many autonomously navigating systems, for example, the corridor centring behaviour of honeybees and robots [217, 216, 218]). The method presented in this paper *sums* the optical flow at antipodal points and therefore, does not encounter such instabilities when antipodal scene points are at equal, or close to being at equal distances.

Finally, because omnidirectional and panoramic sensors were not widely available then, the method of [233] was not fully developed into an algorithm and no implementations or experiments were ever conducted. Here two complete algorithms are presented, tested on simulations under increasing noise and outliers, and demonstrated to work on fairly difficult real image sequences.

Closely related, is the more recent work of Hu and Cheong [106], which develops a constraint similar to that of [233], where the optical flow at antipodal points is subtracted to yield a constraint on translation[1]. As a result, [106] is also degenerate when the world points corresponding to the antipodal rays used are at near-equal distances from the camera centre.

5.2 Obtaining robust estimates of translation

We now have a method for obtaining some estimate of the direction of $\mathbf{t}$ given the flow at any pair of antipodal points. In this section, we outline methods for doing this *robustly*. Two classes of approaches are possible and we present an algorithm representative of each class in Sections 5.2.1 and 5.2.2.

Class 1: Point-Based - Firstly, the flow at two pairs of antipodal points gives a unique (up to a scale) solution for the translation from the intersection of two great circles. Repeatedly picking 2 pairs from a set of N antipodal point pairs

[1]In [106], the authors work with what they term 'matching rays', which come from two laterally mounted cameras pointing in opposite directions such that their FOVs do not overlap (a setup reminiscent of the laterally directed eyes of many birds or fish). However, they then assume that the separation between the two cameras is small, which effectively reduces these 'matching rays' to antipodal points.

can give up to ${}^{N}C_2 = \frac{N!}{2!(N-2)!}$ possible solutions, where each candidate solution is a point on the image sphere. We loosely term the class of methods that estimates the best solution from a set of point solutions, *point-based* algorithms.

Section 5.2.1 presents an algorithm representative of this class that finds the solution point best supported by the optical flow data using the RANSAC framework. Moreover, since the points all lie in a Lie-group (which a sphere is), Tuzel et al. [249] showed that the mean shift algorithm [43] may also be used for finding the mode of the points. K-means is another possibility for clustering points on a sphere [238]. Other algorithms in this class include robust estimators such as [37], and variants of RANSAC such as [241, 155, 156, 180].

Class 2: Line/Curve-Based - Alternatively, since the flow at every pair of antipodes yields a great circle which must intersect $\mathbf{t}$, the translation could be estimated by simultaneously finding the best intersection of many of these great circles. We will later show this can be reduced to the problem of intersecting straight lines. Once again, many algorithms exist, including linear and convex programming. We observe that the problem is quite similar to that of intersecting many vanishing lines to find the vanishing point [196, 151, 210], and inspired by a popular solution in this area, we use a Hough-reminiscent voting approach. There are of course myriad variations to Hough voting (such as [104, 54, 110, 11, 271, 109]) and we present an algorithm representative of the class which is not necessarily the best solution, but which nevertheless works very well in the experiments.

Algorithms that are a combination of the two classes are also possible, such as finding the best supported solution with RANSAC, and then doing linear programming using the inlier flow measurements that have been found.

5.2.1 RANSAC

Algorithm 5.1 summarizes our use of the constraint within the RANSAC sampling algorithm. M is the number of iterations of the algorithm and is calculated adaptively as:

$$M = \frac{log(1-p)}{log(1-w^s)} \tag{5.4}$$

where p is the probability of choosing at least one sample of data points free of outliers, w is the inlier probability, and s is the number of data points required to obtain a candidate solution.

Algorithm 5.1 Estimating translation direction - RANSAC algorithm

1: Set $M = \infty$, $count = 0$
2: **while** $M > count$ AND $count < MAX$ **do**
3: Select a pair of antipodes $\mathbf{r_1}$, $\mathbf{r_2}$ with optical flow $\dot{\mathbf{r}}_\mathbf{1}$, $\dot{\mathbf{r}}_\mathbf{2}$.
4: $\dot{\mathbf{r}}_\mathbf{s} = \dot{\mathbf{r}}_\mathbf{1} + \dot{\mathbf{r}}_\mathbf{2}$ and $\mathbf{n_1} = \dot{\mathbf{r}}_\mathbf{s} \times \mathbf{r_1}$
5: Repeat for a different pair of antipodes to obtain $\mathbf{n_2}$ such that $\mathbf{n_1} \neq \mathbf{n_2}$
6: Take cross product $\hat{\mathbf{t}} = \mathbf{n_1} \times \mathbf{n_2}$
7: **for** $i = 1$ to N **do**
8: For the ith antipodal pair, find the plane normal $\mathbf{n_i}$
9: If $\frac{\pi}{2} - cos^{-1}(\mathbf{n_i} \cdot \hat{\mathbf{t}}) < \theta_{thres}$, increment support for $\hat{\mathbf{t}}$
10: **end for**
11: Update M, increment $count$
12: **end while**
13: The best supported $\hat{\mathbf{t}}$ is the solution.

N is the number of antipodal flow measurements available. MAX is the upper limit on number of iterations to guarantee termination of the algorithm. If several solutions have the same maximum support, then some mean of their directions can be used instead. Details of the RANSAC framework may be found in Section 3.1.5, page 47.

In this algorithm, lines 3 to 6 randomly pick antipodal points and generate hypothesis solutions from the intersection of two planes as discussed earlier. In lines 8 and 9, the candidate solution $\hat{\mathbf{t}}$ is then scored according to the number of antipodal flow data vectors that support it. An antipodal flow pair supports the candidate translation if the angle between the translation vector and the constraint plane generated by that flow pair is less than some threshold angle, θ_{thres}. The candidate with the most support from the flow data is the estimated translation.

The method is quite simple but very robust, as the results in Chapter 6 will show. However, some tuning of the parameters is required for best results. Furthermore, as with all RANSAC-type algorithms, the processing time increases with the probability of outliers and noise in the data. If speed is critical, then a more efficient method may be necessary, which leads us to the voting approach in the next section.

Algorithm 5.2 Estimating translation direction - voting algorithm

1: **for** $i = 1$ to M_{coarse} **do**
2: Select a pair of antipodes $\mathbf{r_1}$, $\mathbf{r_2}$ with optical flow $\dot{\mathbf{r}}_\mathbf{1}$, $\dot{\mathbf{r}}_\mathbf{2}$.
3: $\dot{\mathbf{r}}_\mathbf{s} = \dot{\mathbf{r}}_\mathbf{1} + \dot{\mathbf{r}}_\mathbf{2}$
4: The sphere center, vectors $\dot{\mathbf{r}}_\mathbf{s}$ and $\mathbf{r_1}$ (or its antipode, $\mathbf{r_2}$) define a great circle, C_i on the image sphere.
5: The spherical coordinates (θ, ϕ) of points on great circle C_i are calculated and coarse voting is performed.
6: **end for**
7: Find the bin with maximum votes. This is the coarse estimate.
8: Choose a projection plane at or close to coarse estimate.
9: **for** $j = 1$ to M_{fine} **do**
10: Repeat steps 2 to 4 to obtain a great circle C_j.
11: Project C_j onto the projection plane to obtain a straight line L_j.
12: Perform fine voting by voting along the straight line L_j.
13: **end for**
14: Find the bin with maximum votes. This is the fine estimate.

5.2.2 Hough-reminiscent Voting

Another approach that we identified for robustly estimating translation direction, $\mathbf{t}$, was to simultaneously find the best intersection of all the great circles arising from our method of summing flow at antipodal points. An efficient way of doing this is via a Hough-reminiscent voting method as summarized in Algorithm 5.2. The procedure follows closely the algorithm described in Chapter 4, however, here we vote on great circle constraints, instead of the lune constraint regions of the last chapter.

In the coarse voting stage, we pick the flow at M_{coarse} pairs of antipodes and obtain M_{coarse} great circles with the steps detailed in Section 5.1. We vote on these great circles and find the peak in the vote table to obtain the coarse estimate of translation, $\mathbf{t_{coarse}}$.

Once again, we project a region on the image sphere in the neighborhood of the coarse estimate onto a plane via the gnomonic projection and perform our voting on the projections of great circles - i.e. on straight lines on that plane. This is an advantageous mapping since very efficient algorithms for voting along straight lines exist, such as the Bresenham line algorithm [27]. Once again, voting may be repeated for finer scales if necessary. An example of the result of fine voting

is shown in Figure 5.2, where the peak in the vote is clear even when there are up to 60% outliers in the data.

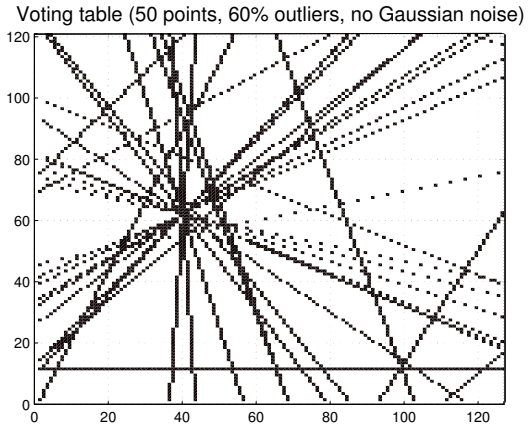


Figure 5.2: The result of fine voting under 60% outliers in the data (This is a window into the vote space and many more outliers fall outside the window).

One possible variation (which we did not find necessary in our experiments) to this algorithm involves 'blurring' out the votes for the straight lines in a Gaussian fashion. This means the highest votes are cast along the center of the line, with the votes falling off in a Gaussian manner in the direction normal to the line. Also, straight lines resulting from flow with high accuracy can be given higher weights during voting compared to lines arising from flow of uncertain accuracy.

5.3 Discussion and Summary

Improving the Accuracy of the Voting Method: A consequence of the voting step was the segmentation of antipodal flow data into inliers and outliers. Inliers are the antipodal flows that gave rise to great circles or straight lines (after projection) that intersected at the estimated translation direction.

If a more accurate translation estimate is desired (one not limited by voting bin resolution), a maximum likelihood estimate (assuming Gaussian noise) can be found by finding the intersection of the inlier straight lines by linear least squares. This additional step incurs little extra effort and we note that it is standard practice in motion estimation algorithms to add a linear or non-linear (e.g. bundle adjustment) optimization step at the end.

Finding Rotation Robustly: We remind the reader that at this point, we have only recovered the direction of translation. The rotation was not estimated directly from our method. However, knowing translation, it becomes much easier to then estimate rotation. For instance, taking dot products on both sides of Equation 5.1 with $\mathbf{t} \times \mathbf{r}$ eliminates dependence on depth:

$$\begin{aligned}(\mathbf{t} \times \mathbf{r}) \cdot \dot{\mathbf{r}} &= (\mathbf{t} \times \mathbf{r}) \cdot (\frac{1}{\mid \mathbf{R}(\mathbf{r}) \mid}((\mathbf{t} \cdot \mathbf{r})\mathbf{r} - \mathbf{t})) - (\mathbf{t} \times \mathbf{r}) \cdot (\mathbf{w} \times \mathbf{r}) \\ (\mathbf{t} \times \mathbf{r}) \cdot \dot{\mathbf{r}} &= (\mathbf{t} \times \mathbf{r}) \cdot (\mathbf{w} \times \mathbf{r})\end{aligned} \tag{5.5}$$

The estimated translation direction is substituted into Equation 5.5, giving us an equation linear in rotation, $\mathbf{w}$. Taking flow at several points, we can build a system of over-constrained linear equations which is solvable by the method of linear least squares. This would be done using only the inlier flow vectors obtained from the translation estimation stage, so the rotation estimate would also be independent of outliers. However, whilst almost all outlier flow vectors were discarded, it is still possible for a few outliers to slip in (called 'leverage points' in statistics). This happens if, by chance, the noise in two outlier antipodal flow vectors cancel when added, giving a great circle that passes through the translation direction. Therefore, another layer of outlier rejection may be needed in practice. Fortunately, since previous steps would have reduced the number of unknowns and weeded out almost all outliers, this should converge very quickly.

In summary, we have presented a simple but effective constraint on the translation direction from antipodal points. The constraint restricts the translation to lie on a plane (or great circle on the image sphere) and is a stronger constraint compared to the one introduced in Chapter 4. However, rotation then needs to be found via a second step, by substituting the translation back into the motion equations. Also, errors in the translation estimate may propagate forward to the rotation estimate. Hence, one solution may be to use the stronger great circle constraints to estimate translation, and to use the lune constraints of Chapter 4 to recover rotation.

Chapter 6

Experiments and Comparison of Results

In the last two chapters, we presented two different antipodal point-based constraints for egomotion recovery. The constraint of Chapter 4 used the directions of antipodal flow vectors to constrain the directions of egomotion. Meanwhile, in Chapter 5, we summed the flow at antipodal points to obtain constraints on translation. We also introduced a robust Hough-like voting algorithm that utilized the first constraint, whilst another two algorithms (based on Hough-voting and on RANSAC) utilized the second constraint.

In this chapter, we demonstrate that the constraints presented in Chapters 4 and 5 work, and that the usage of antipodal points is practical for egomotion recovery. The algorithms based on our two constraints are tested in experiments using real images, as well as in simulations under noise and outliers. The results in the following sections will show that algorithms based on our antipodal point constraints performed competitively with the state-of-the-art whilst demonstrating advantages in their robustness to large outlier proportions and in the processing time required.

Comparison between our two approaches: The Hough-like voting algorithm of Chapter 4 is referred to as the "Lune+voting" method since it votes on lune-shaped constraint regions. Meanwhile, the RANSAC-based and voting-based algorithms of Chapter 5 are referred to as "GC+RANSAC" and "GC+voting" methods respectively (the "GC-" prefix refers to the great circle regions arising from the intersection of the planar constraints with the image sphere).

Lune+voting is used to recover the camera translation direction and rotation axis. Meanwhile, GC+RANSAC and GC+voting only directly recover transla-

tion[1]. The angle between the estimated motion direction and the true motion (known in simulations, and measured in real experiments) then gives the error in the estimates.

Comparison against the state-of-the-art: Since we are working with calibrated cameras, we compare the performance of our methods against the 5-point algorithm of Li and Hartley [136] within a robustifying RANSAC framework[2]. We will refer to this as the "5-point+RANSAC" method.

We chose 5-point+RANSAC as a comparison since it is well-known, well-studied, widely used and code is widely available. Code for the five-point algorithm was made available by the authors of [136]. This five-point minimal solver worked within a RANSAC framework using code from [127]. Sampling was adaptive and the probability with which a good solution was to be found was set to $p = 0.99$. As discussed in page 47 (refer also [93]), the method attempts to estimate the proportion of outliers in the data and adaptively adjusts the number of samples used. The quality of hypothesis solutions was measured as the Sampson distance approximation of reprojection error, which was set at a threshold of 0.01.

6.1 Simulations

6.1.1 Experimental Method

We simulated the camera to simultaneously rotate and translate such that an optical flow field was generated. Since the imaging surface is a sphere and flow is available in every direction, we can fix the direction of translation without any loss of generality as the result would be the same in any direction. The axis of rotation however, was varied randomly from trial to trial since the angle made between the direction of translation and axis of rotation does in fact influence the outcome. Baseline was 2 units and the rotation angle 0.2 radians. 500 pairs of

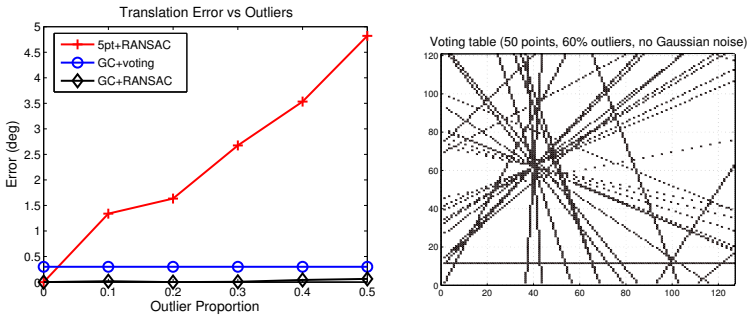
[1] As discussed before, with known translation, the rotation can be easily recovered in a second step. However, we will not focus on that here as our emphasis is on testing and validating the antipodal constraints.

[2] Although the 5-point algorithm is considered to be a discrete-motion method, it works quite well for a large range of motion sizes, including fairly small motions. Various other discrete-motion methods have also been found to work well for small motions (refer Section 3.1.1, page 34), and the 5-point algorithm has previously been used as a benchmark in papers on differential motion methods (e.g. Stewenius et al. [221]). For the motion sizes used in our experiments, we found 5-point to work reasonably well with sub-pixel accuracy for zero noise and $< 1^\circ$ error for small noise (Figures 6.1, 6.2 and 6.3).

random antipodal scene points were uniformly distributed in all directions with depth ranging from 10 to 15 units. Results were averaged over 100 trials.

In the simulations, 5-point used the point matches as input whilst our method took the flow vectors as input. Experiments were conducted for increasing probability of outliers (with no Gaussian noise) and also for an increasing level of Gaussian noise (with no added outliers). Outliers were simulated by randomly replacing matches or flow with errors. To simulate Gaussian noise, the position of a matched point was perturbed from its true value by a noise vector modeled as a 2D Gaussian distribution with standard deviation σ and zero mean.

6.1.2 Robustness to Outliers



(a) (b)

Figure 6.1: (a) Error in estimated translation as proportion of outliers in data increases. The benchmark method - 5-point+RANSAC - is also shown. Comparison between the two great-circle (GC) constraint based algorithms. (b) The result of the fine voting stage of GC+voting.

GC+RANSAC versus GC+voting: The errors for the two algorithms that summed antipodal flow to obtain great-circle constraints on translation are shown in Figure 6.1(a). As the figure demonstrates, all methods performed robustly under increasing outlier proportions. GC+RANSAC and GC+voting showed no appreciable increase in error even for outlier proportions of as high as 50%. In contrast, errors for 5-point+RANSAC increased visibly by up to 5° as the proportion of outliers grew. Note that the errors for GC+voting were slightly higher than GC+RANSAC due to quantization error during voting (which may be further mitigated by using finer scales, and performing linear or nonlinear refinement on these results).

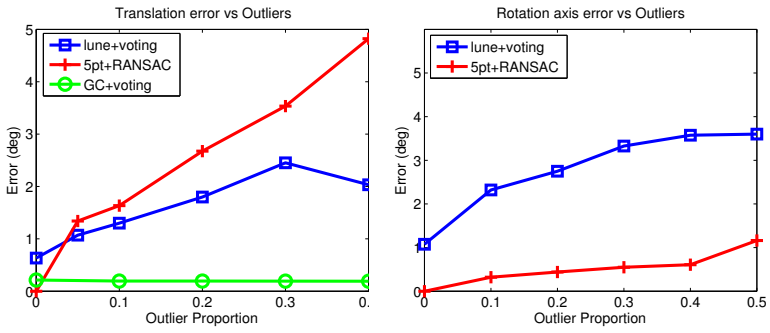
Here, errors in 5-point+RANSAC arise because some outliers fall within the

model distance threshold used (0.01). Reducing the RANSAC distance threshold improves performance; but in general, both our antipodal methods tended to outperform 5-point+RANSAC for very large outlier proportions. This is an interesting result and there are several reasons for it, as discussed in the following.

The Hough-like voting algorithm is extremely resistant to outliers. Figure 6.1(b) shows the votes cast in a voting table for 60% random outliers in the data. The figure shows the straight lines obtained from projecting great circles onto a plane during the fine-voting stage of the GC+voting algorithm. Inlier straight lines intersect at a common point whilst the outlier straight lines do not. Most of them fall outside the region of the plane shown in the figure but quite a few outliers can still be seen. From the figure, it is clear that the intersection can easily be found even for this high proportion of outliers.

The GC+RANSAC framework worked better because it needed only 2 points for a constraint as opposed to 5 points for the 5-point+RANSAC algorithm. If q is the probability of an inlier, then the probability of obtaining a good constraint is q^2 for the antipodal method and q^5 for 5-point. Therefore, both antipodal methods are always more likely to obtain more good constraints when flow at antipodal points is available.

Performance of Lune+voting: Figure 6.2(a) compares the results of lune+voting with that of 5-point+RANSAC and of GC+voting (GC+RANSAC gives very similar results, which we do not show). Unsurprisingly, 5-point+RANSAC estimates translation most accurately for low noise since voting bin resolution limits our methods in this implementation. Interestingly, both lune+voting and GC+voting outperformed 5-point+RANSAC as outliers grew. Compared to GC+voting, the lune+voting method has higher error, since it uses a weaker constraint.



(a) (b)

Figure 6.2: Error in estimated motion as proportion of outliers in data increases. (a) Performance of lune+voting method for translation and (b) for rotation.

However, 5-point with RANSAC outperformed lune+voting in finding rotation (Figure 6.2(b)). Compared to estimating translation, our method performed the same but 5-point was much more accurate in estimating rotation. This may be because 5-point estimates rotation more easily than translation, as observed by Nistér [181]. The important point is that our method degrades at a *comparable rate* to 5-point for rotation errors, implying similar robustness to increasing outliers in the two methods.

6.1.3 Under Gaussian Noise

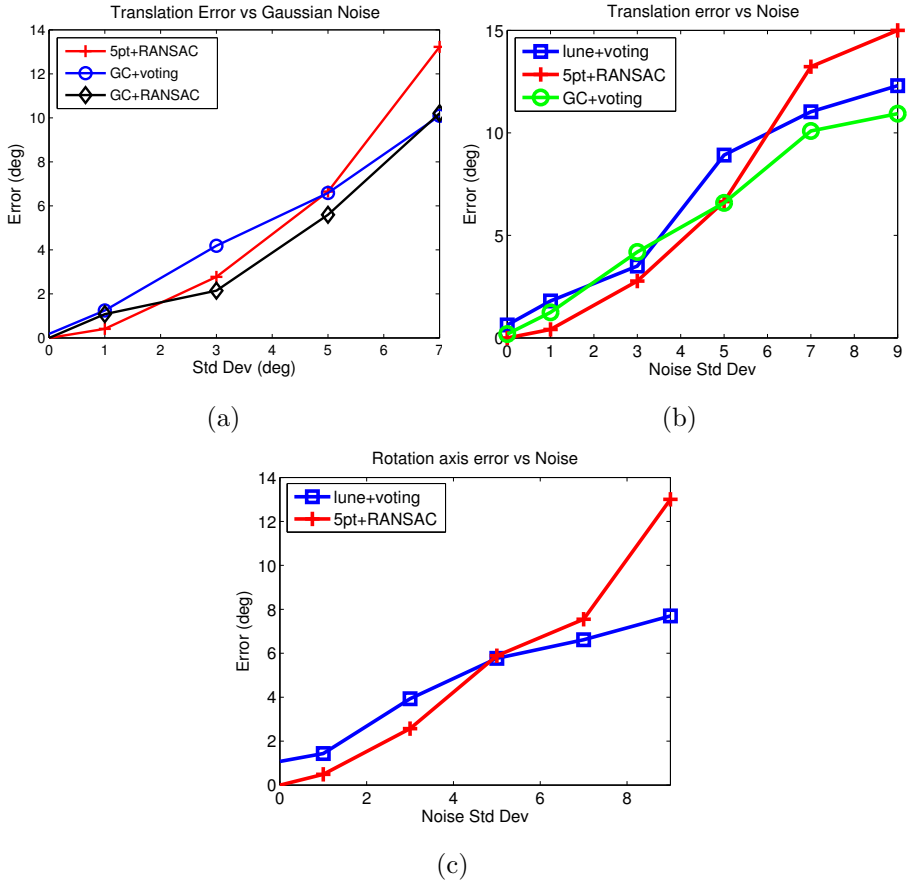


Figure 6.3: Error in estimated motion as the Gaussian noise level increases. (a) Comparison of translation errors between the two great-circle (GC) constraint based algorithms and 5point+RANSAC. (b) Performance of the lune+voting method for translation and (c) for rotation.

Figure 6.3(a-c) shows the translational and rotational errors under increasing Gaussian noise. The trend is that all three methods degrade gracefully with

increasing noise. However, for large Gaussian noise, our antipodal point methods tended to perform better than 5-point+RANSAC since some of the noisy data points were falling within the set threshold (for distinguishing inliers from outliers) of the RANSAC framework.

6.1.4 Processing time

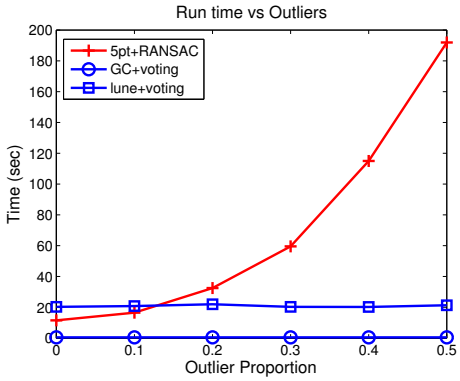


Figure 6.4: Run time versus increasing outlier proportions (Matlab implementations running on an Intel Pentium Dual Core CPU 3GHz, 1GB RAM).

We have seen that our vote-based antipodal algorithms (GC+voting and lune+voting) show comparable robustness with 5-point and RANSAC. Figure 6.4 further demonstrates that our algorithms achieve this with *constant* processing time under increasing proportions of outliers. Conversely, run-time for RANSAC increases quickly as the outlier proportion increases (in the experiment of Figure 6.4, the increase is approximately cubic). Our algorithm is constant time because voting is efficient and fast enough to use all available flow, whereas 5-point+RANSAC samples points adaptively according to $N = log(1-p)/log(1-(1-\epsilon)^s)$ (see page 47). The numbers in Figure 6.4 are obviously implementation dependent but the *trend* will remain the same.

Note also that the processing time of RANSAC is dependent on the distance threshold used for distinguishing inliers from outliers. The Sampson distance threshold used was 0.01; if a stricter threshold was used, 5-point+RANSAC would perform better in the outlier and Gaussian noise experiments, however, its processing time performance would then worsen correspondingly. Our methods do not trade-off accuracy for run-time.

Compared to GC+voting, the lune+voting algorithm obviously takes more time, since the constraint regions it needs to vote over are larger (lunes versus

great-circles).

Overall, the processing times for our voting algorithms are quite fast and always constant, regardless of the probability of outliers or amount of noise in the data. They involve quite simple mathematical operations - vote casting and some dot products (in contrast to the 5-point algorithm which requires solving a degree ten polynomial). Furthermore, voting is by its very nature well suited for a parallel implementation, which could lead to further time savings.

6.2 Real Image Sequences

6.2.1 Experimental Method

(a) (b) (c) (d) (e)

Figure 6.5: Real image experiments. (a-e) shows 5 images captured by the Ladybug camera in an experiment. Going from left to right, we have the views from cam0, cam1, cam2, cam3 and cam4, which then wraps back round to the cam0 view.

Real sequences were captured with a Ladybug camera [187] which returns 5 images positioned in a ring, as shown in Figure 6.5. These are mapped onto an image sphere according to a calibration method supplied by Point-Grey Research, the makers of the camera system. Between each frame in the sequence, the camera translated along the ground for some fixed distance in the direction of the x-axis, which is parallel to the ground plane. It simultaneously rotated by some rotation angle about the z-axis, which is perpendicular to the ground plane. Our algorithm ran on consecutive frames whilst for 5-point, we skipped a frame for every estimate, to afford it larger baselines and rotation. The estimates were compared with measured ground truth.

For our algorithms, image motion was measured using either SIFT matching

[145], or the multi-scale Lucas-Kanade method [146][3]. Meanwhile, the 5-point with RANSAC method used only SIFT to obtain point correspondences.

SIFT descriptors were computed using code from [143] and matched to give point correspondences. The vector difference between these point matches then gave the sparse optic flow inputs used by our algorithms. Our algorithms were also tested with dense optic flow obtained by the Lucas-Kanade method within an iterative multi-scale refinement process, using code from OpenCV [111]. An example of the typically noisy Lucas-Kanade flow found is shown in Figure 6.6.

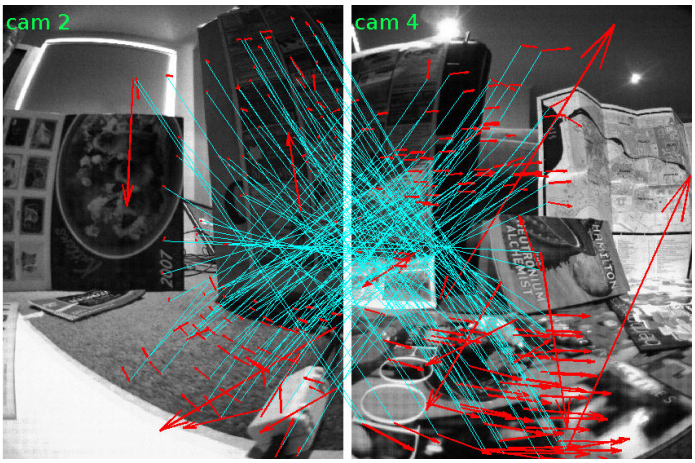
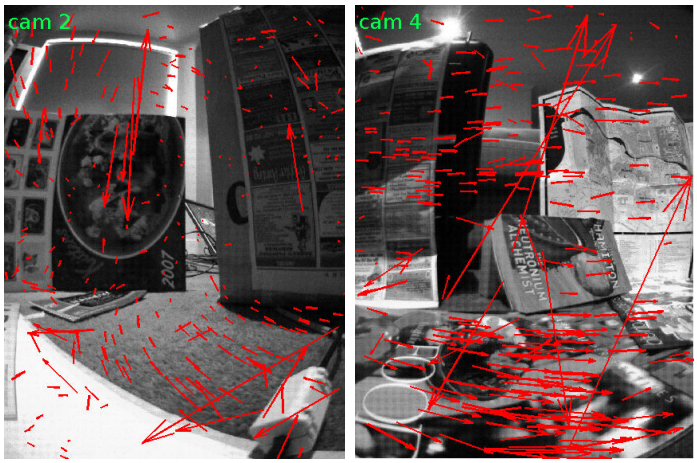


Figure 6.6: Blue lines indicate corresponding antipodes in two cameras (cam2 and cam4) facing different directions. Flow vectors found by the Lucas-Kanade method are shown in red.

Let us label the ring of 5 cameras on the Ladybug camera rig as cam0, cam1, ... cam4, thus completing the ring. Figure 6.6 shows two images taken from cameras on the Ladybug multi-camera rig that were facing different directions. Blue lines pair up antipodes where optic flow (red vectors) was found stably at both points using the Lucas-Kanade method. The blue lines show the pairing of points in cam2 with their antipodes in cam4.

Note that Figure 6.6 only shows the antipodal pairs for cam2 and cam4. More antipodes for the cam2 image exist in cam0, and more antipodes for cam4 exist in cam1. Figure 6.7(a) shows the flow at points in cam2 that had antipodes in cam0 and cam4. Figure 6.7(b) shows the flow at points in cam4 that had antipodes in cam1 and cam2.

[3]Recall from Section 2.2 page 21, that for small image motions, both methods from the dense optical flow category (e.g. methods of Lucas-Kanade, Horn-Schunck etc) and from the point correspondence class (such as SIFT, SURF etc) may be used.



(a) (b)

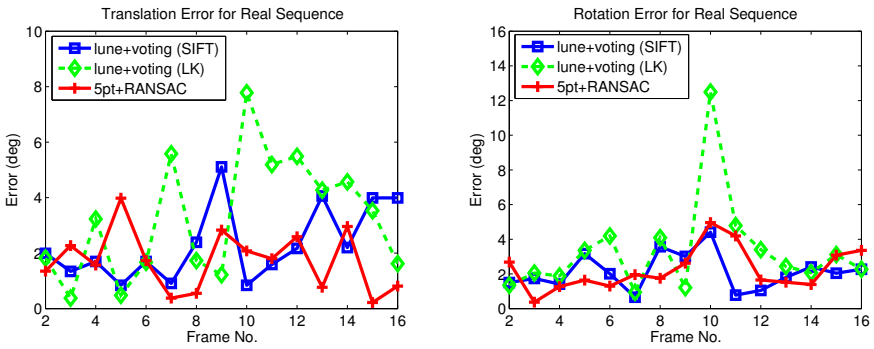
Figure 6.7: (a) All the antipodal flow that was found for the cam2 image using views from cam0 and cam4. (b) All antipodal flow found for the cam4 image using views from cam1 and cam2.

The Lucas-Kanade method computed flow at fixed positions in a grid, so that the antipodal points were already user defined. Flow vectors with large error (the flow algorithm returns an uncertainty measure) were thrown away, but a few obvious mismatches remain. However, most of the flow appears reasonably stable. Over the entire view-sphere, for these experiments, flow was found for about 500 - 700 pairs of antipodes, which is generally more than our algorithms need for a good estimate.

The SIFT algorithm computes SIFT points at particular positions in the image that the algorithm deems likely to be robust to changes in scale [145]. Therefore, to find antipodal points, one has to perform an exhaustive search over all SIFT matches returned by the algorithm (this is a fairly slow operation). We consider two points which are within $\sim 0.5^\circ$ of being antipodal, to be antipodal. In the real images of cluttered indoor scenes, experiments using SIFT matches obtained flow at typically 300 to 800 of these near antipodal points, which is also more than sufficient for our purposes. Increasing this margin finds more antipodes, but at the expense of accuracy.

6.2.2 Experiments with Lune+voting

Figure 6.8 shows the results of translation and rotation axis estimation for the lune+voting algorithm on a real image sequence. In this experiment, the camera



(a) (b)

Figure 6.8: Performance of the lune+voting algorithm for real image experiments.

was moved as described previously, and the translation baseline used was 2-units per frame and the rotation angle was 5° per frame.

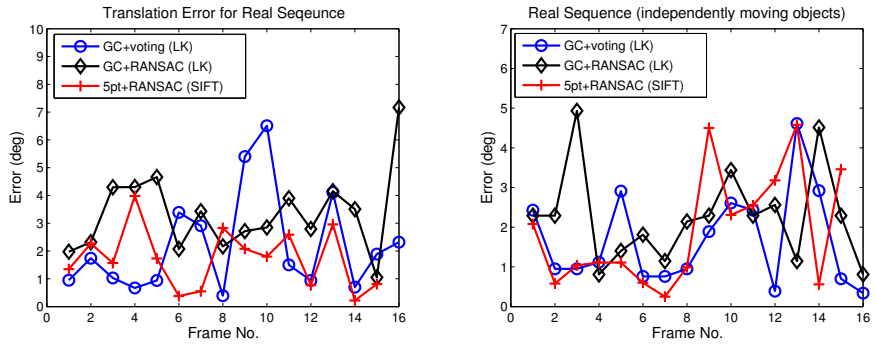
Experiments using SIFT inputs for our method (marked 'lune+voting(SIFT)') showed little difference in performance when compared with 5-point+RANSAC. Average errors over the whole sequence range from 2° to 3° for both methods, with the average slightly lower for 5-point (1° lower for translation and 0.5° lower for rotation axis). Experiments with pyramidal Lucas-Kanade flow inputs (marked 'lune+voting(LK)') for our algorithm showed results that were, on average, about 1° worse than those obtained using matched SIFT points as inputs. These errors were within the bounds on the accuracy of ground truth measurements.

In general, SIFT can be slower than Lucas-Kanade flow and is less suitable for real-time, parallel implementations. On the other hand, SIFT tends to be more accurate compared to the noisier, pyramidal Lucas-Kanade. However, the experiments showed that using Lucas-Kanade flow resulted in only a little degradation in performance, which reinforces the point about the robustness of our methods.

6.2.3 Experiments with GC+voting and GC+RANSAC

The GC+voting and GC+RANSAC methods which find translation were tested with real image sequences and the results shown in Figure 6.9.

Figure 6.9(a) is the result for a sequence with translation baselines of 2cm and rotation angles of 5° between frames (same sequence as before). Also, the scene was static. Meanwhile, Figure 6.9(b) shows a dynamic scene containing moving objects, with translation baselines of 2cm and rotation angles of 2° between frames. The moving objects make the sequence more challenging by adding to



(a) (b)

Figure 6.9: Performance of GC+voting and GC+RANSAC for real image sequences in a (a) static scene and (b) dynamic scene.

the outliers in the scene.

Since the previous results showed that dense Lucas-Kanade gave only slightly less accurate results, whilst being much faster, we will use Lucas-Kanade flow for this set of experiments. In terms of a comparison on the accuracy of 5-point+RANSAC (using SIFT) versus the antipodal point methods (using Lucas-Kanade), this gives a slight disadvantage to the antipodal methods since their inputs are noisier. However, as the results show, the performance of both approaches on real images is roughly equivalent.

The differences in the average errors of different methods were quite small - around 1° to 2° - which is within the bounds on the accuracy of the ground truth measurements. For supplementary videos, refer Appendix A.

6.3 Discussion and Summary

Are antipodal points practical? The experiments demonstrate that antipodal points are plentiful in real scenes and may be used to accurately constrain and estimate camera egomotion. Also, it should be noted that the Ladybug multi-camera system used in real image experiments is not exactly single-viewpoint, leading to points being less than ideally antipodal. Nevertheless, the results demonstrate that with those real images, reliable image motion measurements can be found at sufficient numbers of antipodes, with sufficiently strong constraints on motion, such that accuracies comparable to 5-point are achievable.

Complex scenes. The voting algorithms' resistance to outliers (Figures 6.1

and 6.2) and their constant processing time with respect to increasing outliers (Figure 6.4) would be well suited for certain applications and scene environments - for instance, a moving camera in a busy urban scene with many pedestrians and moving vehicles. In such unstructured and dynamic environments, the simulations suggest that this algorithm would be at least as accurate as 5-point with RANSAC, while running much faster (in fact, in constant time). The results also suggest combining the various antipodal point algorithms. Such a hybrid can find translation with GC+voting since it gives stronger constraints on translation, whilst rotation is recovered with lune+voting.

Speed: The voting approach could potentially be implemented to recover egomotion at high speeds. The flow calculation is typically performed for only a few hundred antipodal points ($\sim$200 to 500) and the mathematical operations used in the voting algorithm are simple - a few dot products, addition and the casting of votes along circles (the coarse-voting stage) or lines (fine-voting). Furthermore, since both the pyramidal Lucas-Kanade flow calculation and the voting step are naturally parallelizable, it is possible to obtain fast implementations using parallel computing architectures such as FPGAs and GPUs.

Doesn't voting resolution limit accuracy? Voting is an approach that works well here and in other motion estimation methods (eg. [154, 51]). However, alternative implementations are possible, such as linear programming, as suggested in the earlier chapters. Nevertheless, the real image experiments demonstrate that under practical levels of noise and calibration errors, voting is comparable to exact methods like 5-point-RANSAC. To further improve accuracy, the estimates obtained by methods like 5-point or antipodal point can be refined by nonlinear optimization methods such as bundle adjustment [245]. Also, voting allows variable resolutions, which is suited to the multi-scale nature of real world tasks like robot navigation.

Field of View: As Equation 4.1 shows, flow is an entanglement of translational and rotational components. The flow in certain parts of the viewsphere are strongly influenced by translation or rotation. For example, if the camera rotates about the z-axis, flow near the equator (x-y plane) gives strong constraints on rotation. Also, if either translation or rotation is large relative to the other, it becomes harder to estimate the weaker motion and easier to find the dominant one. Figure 6.10 shows that for baseline of 2 units, as rotation angle varies from 3° to 18°, rotation becomes easier to estimate and translation harder to estimate.

It is important to note that Figure 6.10 does *not* imply that rotation can be estimated with arbitrary accuracy as the rotation angle increases indefinitely.

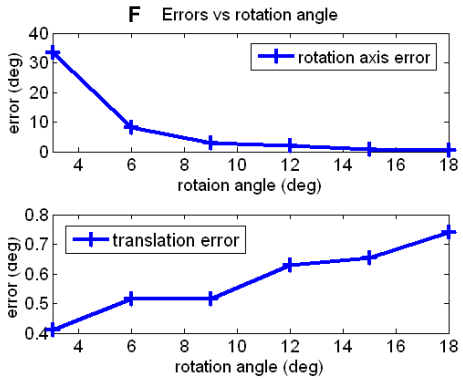


Figure 6.10: Errors as rotation angle increases (for lune+voting algorithm).

For large rotation angles, the errors would actually be larger than the simulation results shown in the figure, since Equation 4.1 is no longer a good model of the effects of egomotion on image motion.

Indeed, the methods presented up till now really work best when the camera is undergoing differential or instantaneous motion, but break down for large camera translations and rotations. This leads us to the next part of this work, where we consider methods that work for large or discrete motions, and are not limited by the assumption of small motions.

Part III

Discrete Camera Motion

Outline of Part III

In Part III, we consider egomotion recovery for cameras undergoing discrete motions.

1. **Chapter 7:** Here, we present the antipodal-epipolar constraints on relative camera motion. These make use of antipodal points to obtain separate linear constraints for translation and rotation. Robust algorithms are introduced, and their performance demonstrated with experiments.

2. **Chapter 8:** Next, we consider the idea of relaxed egomotion recovery, where we relax the requirement for highly accurate estimates whilst still remaining robust to outliers. Compared to full egomotion recovery, this is much faster to compute, and such estimates are still useful for many applications. We apply this idea to the problem of visual homing, where a relaxed estimate of translation is used to perform robust robot homing in real environments.

Chapter 7

The Antipodal-Epipolar Constraint

Previously, in Part II, we considered egomotion recovery under the assumption that the camera was undergoing differential motion. The main drawback with those approaches was that their usefulness was limited to scenarios where camera motions were small. Here in Part III, we drop this assumption and consider *discrete* camera motions instead. This leads to a new formulation of the problem, where we develop useful constraints and algorithms that work over a wider range of camera motion sizes.

In this chapter we will introduce novel **antipodal-epipolar constraints** on relative camera motion. We show that the idea of antipodal points is useful not only under the assumption of differential motions, but also in the case of discrete camera motions. We demonstrate the use of our constraints with two robust and practical algorithms, one based on the RANSAC sampling strategy, and one based on Hough-like voting. Lastly, we show that the proposed constraints work and that the algorithms using them perform accurately and robustly in noisy simulations and in real image sequences.

We previously saw that translation and rotation were decoupled in the motion equations under the differential motion assumption, whereas, for the more general discrete motion case, they were nonlinearly coupled together in the essential or fundamental matrices (pages 37 and 42). Here, we show that even for the discrete case, translation and rotation may be geometrically decoupled using the properties of antipodal points, allowing them to be estimated separately, independent of each other.

In particular, through the antipodal-epipolar formulation, we can express con-

straints that are *linear in translation*, and independently, constraints that are *linear in rotation*. In contrast, the conventional epipolar constraint is *bilinear* in translation and rotation. This is theoretically significant, as it means that under the special condition of antipodal points, a problem that is nonlinear in general becomes linear. This has not been achieved by any other method that we know of.

As a result of the decoupling of motion components, we find ourselves faced with the task of solving *two smaller dimension problems* instead of a single higher dimensional one. This simplifies the task, leading to important consequences for designing algorithms that are robust to outliers and noise. Basically, it is easier to separate the inlier data from the outliers in a lower dimensional space than in a higher dimensional one. This naturally results in more efficient and robust algorithms, including ones which may be implemented to run in *constant-time* regardless of outlier proportions in the data.

Furthermore, since translation can be estimated independently of rotation, we propose a method by which one may reconstruct scene structure purely from translation estimates, and *without* recovering relative camera rotation explicitly (the rotation is implicitly constrained). This translation-only structure-from-motion (SfM) algorithm was previously not a sensible option since past methods recovered rotation and translation simultaneously (as the essential or fundamental matrices). The new SfM algorithm is not only of theoretical interest but also has practical advantages in terms of computation speed, compared to standard SfM techniques.

In Section 7.1, we will investigate the geometry of the problem and derive our constraints on camera motion. We present robust algorithms utilizing these constraints in Section 7.2 and discuss our reconstruction method in Section 7.3. Section 7.4 demonstrates the performance of our methods with some experiments and we end with some discussion in Section 7.5.

7.1 Theory

In Figure 7.1(a), a camera centred on the point $\mathbf{C}$ undergoes some rigid body motion - a translation $\mathbf{t}$ and rotation R - that takes it to a new position $\mathbf{C}'$ and some new orientation. The image-spheres centered on cameras $\mathbf{C}$ and $\mathbf{C}'$ are also shown. The world point $\mathbf{P}$ is imaged by the two cameras giving the point correspondence $\mathbf{p}$ and $\mathbf{p}'$. The ray $-\mathbf{p}$ is antipodal to $\mathbf{p}$. If there happens to be

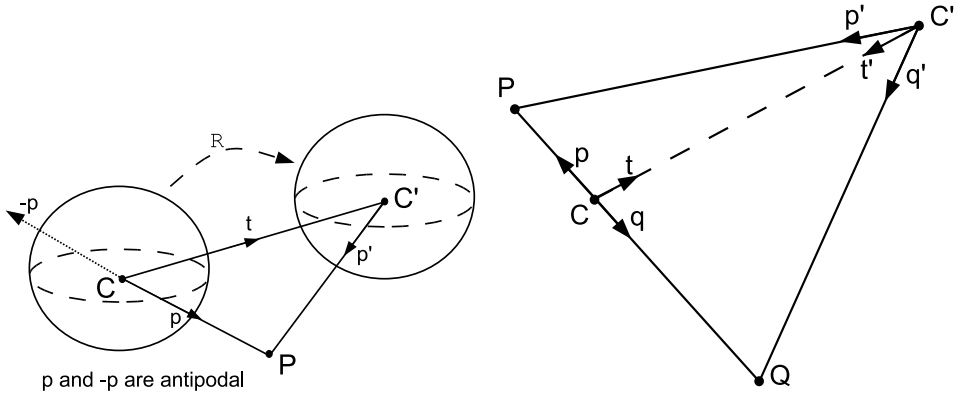


Figure 7.1: (a) Some translation $\mathbf{t}$ and rotation matrix R relates the two cameras at $\mathbf{C}$ and $\mathbf{C}'$. (b) Cameras $\mathbf{C}$ and $\mathbf{C}'$ and world points $\mathbf{P}$ and $\mathbf{Q}$. Rays $\mathbf{p}$ and $\mathbf{q}$ (i.e. $\mathbf{q} = -\mathbf{p}$) are antipodal in the camera $\mathbf{C}$ view. $\mathbf{p}$ and $\mathbf{p}'$ are one pair of point correspondences and $\mathbf{q}$ and $\mathbf{q}'$ are another. $\mathbf{t}$ is the translation directed from $\mathbf{C}$ to $\mathbf{C}'$ in the coordinate frame of $\mathbf{C}$, and $\mathbf{t}'$ is directed from $\mathbf{C}'$ to $\mathbf{C}$ in the frame of $\mathbf{C}'$. All points and rays shown lie on the same epipolar plane.

another world point lying on the ray $-\mathbf{p}$, and that point is also visible to camera $\mathbf{C}'$, then we can obtain antipodal-epipolar constraints on $\mathbf{t}$, and on R.

Such a situation is shown in Figure 7.1(b) (essentially the same setup as (a) but without the spheres) where a pair of world points, $\mathbf{P}$ and $\mathbf{Q}$, project onto $\mathbf{C}$ to give the rays $\mathbf{p}$ and $\mathbf{q}$. Suppose these world points, and hence the rays, are antipodal relative to $\mathbf{C}$, that is, $\mathbf{q} = -\mathbf{p}$ (where the rays are expressed in the image coordinate frame of $\mathbf{C}$).

Meanwhile, the projection of the world points onto the second camera, $\mathbf{C}'$ gives the rays $\mathbf{p}'$ and $\mathbf{q}'$ (i.e. the rays are expressed in the coordinate frame of $\mathbf{C}'$). Therefore, $\mathbf{p}$ and $\mathbf{p}'$ are a pair of point correspondences and $\mathbf{q}$ and $\mathbf{q}'$ are another pair.

$\mathbf{t}$ is the translation directed from $\mathbf{C}$ to $\mathbf{C}'$, expressed in the coordinate frame of $\mathbf{C}$. Conversely, $\mathbf{t}'$ is the translation directed from $\mathbf{C}'$ to $\mathbf{C}$, expressed in the coordinate frame of $\mathbf{C}'$.

As before, we can only recover $\mathbf{t}$ (or $\mathbf{t}'$) up to a scale [93], so that gives 2 unknowns to be recovered. R contains another 3 unknowns; so altogether there are 5 unknown motion parameters to be solved for. Let all vectors denoted with lower case letters be of unit length.

7.1.1 Constraint on translation

All the points and rays in Figure 7.1(b) lie on a single plane if $\mathbf{p}$ and $\mathbf{q}$ are antipodal. $\mathbf{p}$, $\mathbf{p}'$ and $\mathbf{t}$ lie on a plane - the epipolar plane. Likewise, $\mathbf{q}$, $\mathbf{q}'$ and $\mathbf{t}$ also lie on a plane, and if $\mathbf{p}$ and $\mathbf{q}$ are antipodal, then the two planes are one and the same. At first glance, it seems as though the extra pair of correspondences $\mathbf{q}$ and $\mathbf{q}'$ may be giving redundant information since the epipolar equation arising from it describes the same plane as the one arising from the pair $\mathbf{p}$ and $\mathbf{p}'$.

However, notice that we can express the equation of the epipolar plane purely using the rays $\mathbf{p}'$ and $\mathbf{q}'$. The normal vector to the plane is simply given by $\mathbf{p}' \times \mathbf{q}'$ where $\times$ denotes the cross product. Then, we have:

$$\mathbf{t}'^{T}(\mathbf{p}' \times \mathbf{q}') = 0 \tag{7.1}$$

This is a linear constraint on the translation, $\mathbf{t}'$, independent of rotation, R. Geometrically, Equation 7.1 simply states that $\mathbf{t}'$ lies on the epipolar plane. Let us call this the *antipodal-epipolar constraint on translation*. (We use the 'antipodal-' qualifier to differentiate this from the usual epipolar constraint, $\mathbf{p}'^{T}\mathsf{E}\mathbf{p} = 0$, where E is the essential matrix.) Note that if the points are antipodal in camera $\mathbf{C}$, then the constraint is on $\mathbf{t}'$; conversely, if the points are antipodal in camera $\mathbf{C}'$, then the constraint is on $\mathbf{t}$.

It is also important to note that the sign of translation is *unambiguously* recoverable (whereas in normal essential matrix estimation, it is ambiguous whether the camera translation is $\mathbf{t}$ or $-\mathbf{t}$ [93]). From Figure 7.1(b), it is clear that $\mathbf{t}'$ must lie between $\mathbf{p}'$ and $\mathbf{q}'$, that is, $(\mathbf{t}')^{T}\mathbf{p}' > 0$ and $(\mathbf{t}')^{T}\mathbf{q}' > 0$. Hence it cannot lie in the opposite direction.

Ambiguity arises only when both world points $\mathbf{P}$ and $\mathbf{Q}$ are infinitely far away; then $\mathbf{p}'$ and $\mathbf{q}'$ will also be antipodal and it is not possible to determine the sign of $\mathbf{t}'$. Obviously, infinitely distant points can tell us nothing about translation, and this degenerate case is easily detectable as infinitely distant $\mathbf{P}$ and $\mathbf{Q}$ means $\mathbf{p}$, $\mathbf{q}$ will be antipodal, and $\mathbf{p}'$, $\mathbf{q}'$ will also be antipodal.

A minimum of two pairs of antipodal points (which is four points), will recover translation up to a positive scale (since the sign of $\mathbf{t}$ is known). One pair of antipodal points constrains translation to lie on a plane. Another pair of antipodal points (that do not lie on the first plane), gives another such plane, and the two distinct planes intersect to give the direction of translation (Figure 7.2(a)).

7.1.2 Constraint on rotation

The relative rotation between the two camera coordinate frames may similarly be constrained by the epipolar planes. The ray $\mathbf{p}$ must lie on the plane given by $\mathbf{p}' \times \mathbf{q}'$, so we have:

$$(\mathsf{R}\mathbf{p})^T(\mathbf{p}' \times \mathbf{q}') = 0 \tag{7.2}$$

This gives a linear constraint on rotation, R, independent of the translation. Let us call this the *antipodal-epipolar constraint on rotation*. (Note that $(\mathsf{R}\mathbf{q})^T(\mathbf{p}' \times \mathbf{q}') = 0$ gives no extra information as it is linearly dependent to Equation 7.2, since $\mathbf{p} = -\mathbf{q}$).

The 3×3 rotation matrix contains effectively 3 unknowns, since $\mathsf{R}^T\mathsf{R} = \mathsf{I}$ gives 6 quadratic constraints on the 9 entries of the matrix. Therefore, a minimum of 3 antipodal pairs is sufficient to recover rotation. Geometrically, this corresponds to the situation depicted in Figure 7.2(b). However, this is a nonlinear solution. To obtain a linear solution, one requires a minimum of 9 points to constrain the 9 entries in R.

Just as with the conventional epipolar constraint, the case of purely rotational motion (no translation) is degenerate. This event may be easily detected as all the antipodal points in the first camera will still be antipodal in the second camera. This is effectively similar to the case where all world points are infinitely distant.

7.1.3 Linear and Decoupled Constraints

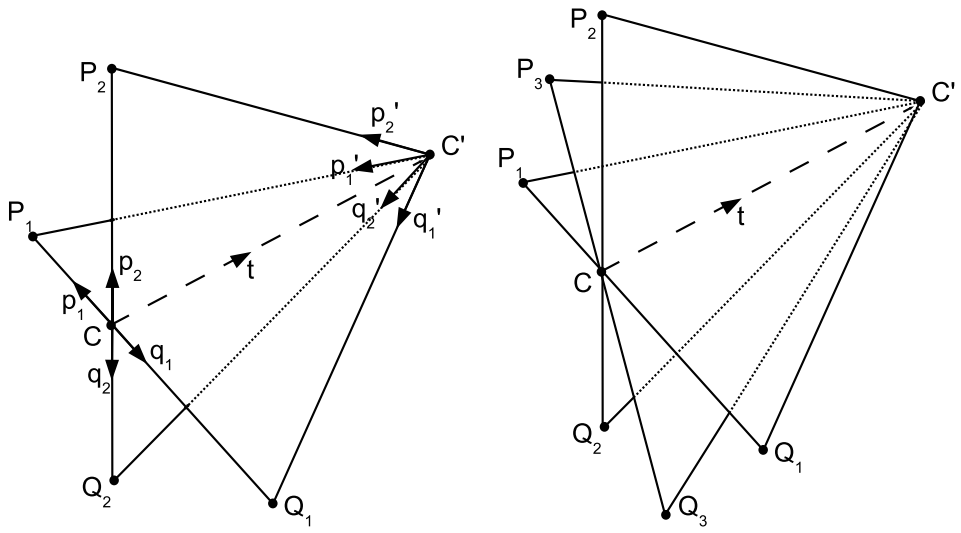
The conventional epipolar equation is given by:

$$\mathbf{p}'^T([\mathbf{t}]_\times\mathsf{R})\mathbf{p} = 0 \tag{7.3}$$

where the essential matrix, E, is the cross product of $\mathbf{t}$ and R. The equation is *bilinear* in both $\mathbf{t}$ and R.

Contrast that with Equation 7.1, which is linear in $\mathbf{t}$ (and independent of R), and with Equation 7.2 which is linear in R (and independent of $\mathbf{t}$). This demonstrates that under the special condition of antipodal points, the constraints on motion are linear, even though the motion equations are nonlinear (or bilinear) in general.

The decoupling of translation and rotation into separate constraint equations leads to several advantages. For example, it means that in theory, errors in the translation estimate will not propagate and contribute to the errors in rotation,



(a) (b)

Figure 7.2: Multiple antipodal pairs give rise to multiple antipodal-epipolar planes. $\mathbf{P}_i$ and $\mathbf{Q}_i$ are pairs of antipodal world points, for $i = 1, 2, 3$. (a) Two antipodal pairs give two planes that intersect to give $\mathbf{t}$. (b) Three pairs give three planes constraining the three unknowns of R.

and vice-versa. Other advantages include breaking up the problem into two lower dimensional ones which are easier to solve. In the next section, we will see how this naturally leads to better algorithms.

By simplifying the problem, the antipodal-epipolar constraints enable us to explore new, robust estimation frameworks - including ones which were not attractive solutions before this. The algorithms proposed will attempt to address some of the issues or drawbacks present in existing approaches.

7.2 Robust Algorithms from Antipodal Constraints

In this section, we present algorithms designed to be robust to outliers in the point correspondence inputs. Like the 5-point or 8-point approaches, if greater accuracy is desired, the output of our algorithms can be used to initialize local, nonlinear refinement methods such as bundle adjustment [245].

Recall that in Chapter 5 we presented RANSAC and Hough-voting algorithms (i.e. GC+RANSAC and GC+voting) that found translation from the intersection

of two or more planes. Meanwhile, in the last section, we saw that for the discrete motion case, the recovery of translation from antipodal-epipolar constraints also requires finding the intersection of planes. Therefore, even though the differential and discrete cases are distinct problems, and different sets of constraints were derived in each case, the same type of algorithms may be applied in both cases to robustly solve for translation.

In the following, we will briefly show how the antipodal-epipolar constraint on translation may be used within a RANSAC-type algorithm and within a Hough-like voting algorithm. For further implementation details of each method we refer the reader to Sections 5.2.1 and 5.2.2 of Chapter 5.

The robust recovery of translation with either RANSAC or voting also results in the segmentation of the data into inliers and outliers. These inliers may then be used in the next stage to recover rotation using the antipodal-epipolar constraint on rotation[1].

The algorithms presented here aim to demonstrate the viability of the novel geometrical constraints introduced, and as such, play an illustrative role rather than a definitive one. Many other algorithms utilizing the antipodal constraints are possible, and the reader may replace them with their favorite robust estimation method.

7.2.1 Robustly Finding Translation

RANSAC

Recall that minimal solvers are crucial for hypothesize-and-test methods such as RANSAC and its relatives. This is because a reduction in the number of points needed to generate a hypothesis greatly increases the probability of picking an outlier-free set of points. This then speeds up the process significantly as the number of hypotheses that need to be tested before a good one is found with high probability, is reduced.

We can obtain a minimal solver requiring *2 antipodal pairs* (which is 4 points). As discussed above, the minimum number of points required to recover translation is 2 antipodal pairs (4 points) and the minimum required to recover rotation is 3

[1]This actually introduces some coupling between the translation and rotation estimates. However, the effect is not too significant and we made this design choice for efficiency purposes. In general, algorithms can be devised such that the two estimates are independent, since the constraints themselves are decoupled. For example, rotation can be found by performing RANSAC over all points, rather than just the inliers found by the translation stage.

pairs (6 points). At first glance, it therefore seems that we need a minimum of 3 pairs. However, because we can estimate translation and rotation separately, we can actually get away with a minimum of 2 pairs instead of 3.

First, translation is found via RANSAC and a 4-point (2 pair) minimal solver. The inlier-outlier segmentation of data points occurs at this stage - any pair that satisfies Equation 7.1 to within some threshold is an inlier. The recovered inliers then give an over-constrained system of homogeneous, linear equations (via Equation 7.2) from which a least squares solution to rotation can be found.

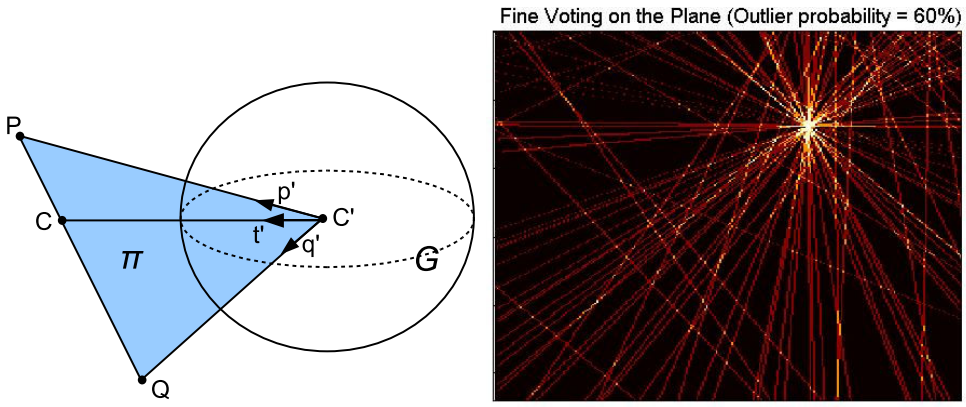
Our antipodal point and RANSAC algorithm performs faster than the usual 5-point and RANSAC algorithm, since a smaller number of points are used for hypothesis generation (4 versus 5 points). Further time savings are incurred due to the fact that the solver solves a simple, linear problem of intersecting two planes (contrast that with the tenth order polynomial that the nonlinear 5-point algorithm needs to solve). Implementation of the required solver is trivial.

However, like all RANSAC-based methods, run-time will still increase with increasing outlier proportions. This leads us to the next algorithm, which does not face this problem, since it avoids random sampling altogether.

Hough-like Voting

Since translation estimation is a fairly low dimensional problem (2D), we suggest that here, Hough-like voting may be an efficient algorithm for robust estimation. Voting has previously been used for estimation of discrete camera motion [15, 154, 51]. However, these generally vote in some higher dimensional space (e.g. at least 5D for estimating essential matrices). Hence, the large number of bins needed to discretize the solution space made previous methods susceptible to the problem of sparsity of points. Here, the decoupling effect of antipodal points enables us to work in a 2D solution space, for which voting is both effective and efficient.

This algorithm represents a break from previous approaches for estimating discrete camera motion, which were largely based on the conventional epipolar constraint and relied on robust frameworks using RANSAC-type, Monte-Carlo hypothesis generation and testing. Voting, in contrast, performs in constant time under increasing outlier proportions, and does so without sacrificing robustness or accuracy (recall that to achieve speed-ups under increasing outliers, methods such as Preemptive-RANSAC, R-RANSAC and Optimal R-RANSAC trade-off on robustness and accuracy for run-time). Also, voting does not require the use



(a) (b)

Figure 7.3: (a) The epipolar plane Π intersects the image-sphere to give the great circle G. (b) Result of the fine voting stage. The peak is obvious even with 60% outliers.

of minimal point solvers.

Each antipodal pair constrains $\mathbf{t}$ to lie on some plane and the intersection of two or more planes will recover $\mathbf{t}$. By voting to each plane, we can robustly find the best ray of intersection from all the planes. Instead of random sampling, we will use the constraints arising from every antipodal pair available. This is because voting is computationally very cheap, hence there is no reason not to use all the available constraints.

To perform voting, we need a representation of the space in which the solutions lie. Borrowing from our previous work in Chapters 5 and 6, we first represent the space of possible directions of translation as the surface of a unit image-sphere centered on the camera center. The intersection of a plane passing through the sphere center with the surface of the sphere is a great circle. The problem of finding the intersection of planes then becomes one of finding the intersection of great circles on the sphere.

This is illustrated in Figure 7.3(a), where the epipolar plane Π with normal vector $\mathbf{p}' \times \mathbf{q}'$ intersects with the surface of the unit sphere at the great circle G. After voting along all the great circles arising from all the antipodal pairs, the peak in the vote table gives a robust estimate of the translation.

For efficiency purposes, we perform multistage coarse-to-fine voting, first on the sphere, and later on a projection plane. For the implementation details of coarse voting on great circles and fine voting on straight lines, we refer the reader to the similar voting framework used for differential camera motion estimation in

Chapter 5. Figure 7.3(b) shows an example of the fine voting stage. The peak is obvious even with 60% outliers in the data (this is a window of the voting space and many more outliers lie outside the window).

Once again, a natural consequence of translation estimation is the segmentation of inliers from outliers in the data. Points that resulted in straight lines (in the fine voting stage) that pass through the translation estimate are inliers (i.e. equivalent to satisfying Equation 7.1). A last linear refinement step is performed using these inliers in order to obtain a least squares estimate of translation. This last step mitigates the effects of voting bin quantization on accuracy.

7.2.2 Finding Rotation from the Linear Constraints

Using only the inliers found when estimating translation with either the RANSAC-based or voting-based methods above, we now go on to perform rotation estimation with Equation 7.2, which is linear in rotation, R. Each pair of inlier points would give one linear constraint on rotation. M inliers will give M linear constraints, giving a system of over-constrained equations which may be written in the form $\mathsf{A}[\mathsf{R}_{vec}] = 0$, where $[\mathsf{R}_{vec}]$ is the vectorized form of the rotation matrix, R. A is a $M \times 9$ matrix, whilst $[\mathsf{R}_{vec}]$ is a 9×1 vector.

To be able to solve for R linearly, we need $M \geq 9$. $\mathsf{A}[\mathsf{R}_{vec}] = 0$ can then be solved by the DLT algorithm [93], which involves performing a SVD, $\mathsf{A} = \mathsf{UDV}^T$. The least-squares solution is given by the column of V corresponding to the smallest singular value in D. A final nonlinear correction step ensures R is a 'proper' rotation matrix by enforcing $\mathsf{RR}^T = I$ and $det(R) = 1$ (so the complete method for rotation recovery is linear except for this final correction step).

In practice, there is a small, but finite probability of outliers slipping past the outlier rejection stage during translation estimation. This happens when the outlier is a 'leverage point' that satisfies the translation constraint of Equation 7.1 but does not satisfy the constraints of full camera motion. Therefore, a practical system may need another RANSAC stage (or other robust algorithm) here. Since the vast majority of outliers (usually all) have been removed, this step generally happens quickly, i.e. terminates in one iteration[2]. In the event of one or two leverage points slipping in, the RANSAC stage will then detect and deal with them.

[2]Since we know the outlier probability is very low, we did not use a nonlinear minimal-point rotation solver since RANSAC will undergo few iterations. Such a minimal-point rotation solver is a possible alternative but it is harder to implement compared to the method used here.

Another factor to take into practical consideration is the size of singular values found in the SVD. Certain singular values in the diagonal matrix, D, can sometimes be very small, and under noise, it can be hard to distinguish the smallest singular value. To remedy this, we test all solutions corresponding to very small singular values by reconstructing several points, and picking the solution where all reconstructed points are in front of both cameras (similar to the test applied to disambiguate twisted-pairs in essential matrix motion recovery [93]).

7.3 Structure from Partial-Motion - Not Finding Full Rotation

One observes that relative scene structure can actually be found *without* recovering the parameters of rotation. Previously, this fact was not particularly useful, since conventional epipolar geometry methods will recover translation and rotation *simultaneously*, that is, they are entangled together in the essential or fundamental matrices which are to be estimated.

Here, however, the two are found *separately*. Therefore, if the goal of some system was structure recovery, then there is no necessity to compute any extra parameters which are not needed for the task. In particular, we will show that recovering translation twice, that is, finding $\mathbf{t}$ and $\mathbf{t}'$, is sufficient for structure computation. Rotation is not explicitly estimated at all.

Computing $\mathbf{t}$ and $\mathbf{t}'$ amounts to recovering rotation up to 1 degree of freedom. $\mathbf{t}$ and $\mathbf{t}'$ both lie on the straight line, $\mathbf{CC}'$, joining the camera centers. If $\mathbf{t}$ and $\mathbf{t}'$ have been found, the two cameras are still free to rotate relative to each other about the axis $\mathbf{CC}'$. However, given a point correspondence, we can assume that the rays $\mathbf{p}$ and $\mathbf{p}'$ intersect at the world point $\mathbf{P}$. This implicitly constrains the last remaining degree of freedom of rotation (without having to explicitly estimate it).

Figure 7.4 shows the geometry of the situation. $\mathbf{t}$ is estimated from points which are antipodal in camera $\mathbf{C}'$, whilst $\mathbf{t}'$ is found from points which are antipodal in camera $\mathbf{C}$. Then, from basic trigonometry, we have:

$$tan(\alpha) = \frac{Z_l}{X_l}, \quad tan(\beta) = \frac{Z_r}{X_r}, \quad X_r = |\mathbf{T}| - X_l \tag{7.4}$$

where $|\mathbf{T}|$ is the baseline, the magnitude of which, we set to unit. $\alpha = acos(\mathbf{t}^T\mathbf{p})$ and $\beta = acos(\mathbf{t}'^T\mathbf{p}')$ where all ray vectors are unit length. With $|Z_r| = |Z_l| = Z$,

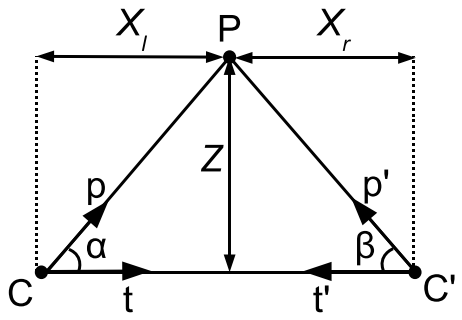


Figure 7.4: Structure from $\mathbf{t}$ and $\mathbf{t}'$ (partial-motion).

we have:

$$Z = \left| \frac{tan(\alpha)tan(\beta)}{tan(\beta) + tan(\alpha)} \right| \tag{7.5}$$

Then the reconstructed world point, $\mathbf{P}$, is $\frac{Z}{sin(\alpha)}$ units away from the camera $\mathbf{C}$ in the direction $\hat{\mathbf{p}}$.

Apart from theoretical interest, this result is also of practical use since $\mathbf{t}$ and $\mathbf{t}'$ can be found simultaneously using parallel processors, whereas in our implementation, the rotation step happens after the translation stage has found the inliers. Furthermore, translation recovery by the Hough-voting approach can be sped up significantly with parallel processing hardware such as GPUs (Graphical Processing Units) or FPGAs (Field-programmable Gate Arrays). The rotation step, which is not as well-suited for parallel implementation compared to voting, may be skipped altogether.

Using $\mathbf{t}$ and $\mathbf{t}'$, reconstruction can be done using the above procedure over all point matches. Again, this may also be computed in parallel. The drawback of the approach is that in the presence of noise, the rays $\mathbf{p}$ and $\mathbf{p}'$ may not intersect exactly. The advantage of the method lies in its speed.

This method is reminiscent of (but different from) plane+parallax methods which use known epipoles and known planes (either real planes or the virtual, infinite plane) to perform reconstruction [244, 201]. Rotation and calibration parameters need not be known as the plane+parallax constraints cancel them. In our case, the camera is calibrated, and we use the angle between the rays and the epipoles for triangulation. The angle between any two rays is invariant to rotations, hence we are able to escape explicitly calculating rotation.

7.4 Experimental Results

We now demonstrate that the antipodal-epipolar constraints work, and that the algorithms based on them work robustly and accurately in practice. We run tests on the translation estimation algorithms (which, for brevity, we term as the 'antipodal+RANSAC' and 'antipodal+voting' methods), as well as on our algorithm for recovering rotation (referred to as 'antipodal' in the legends of the graphs for errors in rotation). The results for simulations under outliers and noise are shown in Figures 7.5 and 7.7; whilst results for real image experiments are in Figure 7.8.

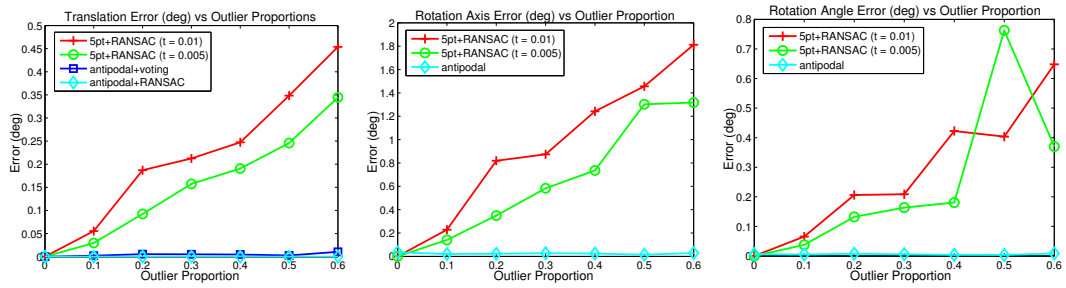
Results were benchmarked against the performance of 5point+RANSAC, which consists of the 5-point algorithm of Li and Hartley [136] within the RANSAC code of [127] (see Section 3.1.5, for details of RANSAC). As before, the algorithm will estimate the outlier probability from the sampled data and use that to adaptively determine the amount of samples required such that the statistical probability of sampling at least one outlier free measurement is $p = 0.99$. We tested using Sampson distance thresholds of 0.01 and 0.005 (the experiments will demonstrate the effects of lowering the threshold from 0.01 to 0.005 on RANSAC's robustness to outliers and on its run-time).

The antipodal+RANSAC method also worked with the RANSAC code of [127]. Adaptive sampling probability was $p = 0.99$ and the threshold used was 0.5° (angle between translation and the antipodal-epipolar plane). The antipodal+voting method performed coarse-to-fine voting in two stages, with the estimate taken as the centre of mass of voting bin(s) with the highest votes.

We measure the accuracy of the estimates on translation direction, rotation axis and rotation angle. All errors are measured in units of degrees (e.g. the angle between the true and estimated translation directions). All algorithms used were Matlab implementations.

7.4.1 Simulations

Method: A series of simulated experiments were performed in Matlab to test performance under increasing outlier proportions and under increasing Gaussian noise. Two views of a 3D scene were generated with random translations and rotations between cameras. The magnitude of the translation varied randomly between 5 and 10 units; the rotation angle varied randomly between 10° and 50°; whilst the distances to the world points varied between 5 and 10 units. Results



(a) (b) (c)

Figure 7.5: Errors under increasing outlier proportion. All methods performed robustly, giving small errors for outlier proportions of up to 60%. However, our methods gave much smaller increases in error compared to the benchmark approach. Here, t is the Sampson distance threshold used for 5-point+RANSAC.

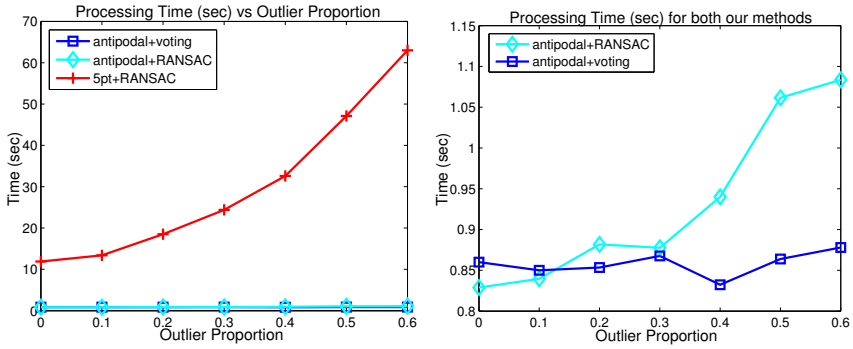
were averaged over 100 trials.

We tested on data contaminated with outliers in proportions ranging from 10% to 60%; and on data under Gaussian noise with zero mean and standard deviations of up to 0.3° (In a camera with viewing angle 60° and a 640x480 pixel image, 0.3° corresponds to around 2 to 3 pixels of noise). This corresponds to realistic noise levels in feature matching methods (e.g. SIFT), which are sub-pixel accurate and typically exhibit errors of at most a couple of pixels. Outliers consisted of randomly mismatched rays, whilst Gaussian noise was simulated by perturbing the directions of image rays according to a Gaussian distribution.

Robustness to Outliers: The antipodal point algorithms proved to be remarkably resistant to increasing outlier proportions. Figures 7.5(a-c) demonstrate that increasing the outlier proportions up to 60%, caused little appreciable increase in the error of motion estimates from the antipodal+voting and antipodal+RANSAC methods. Over the same increase in outliers, the errors for the 5-point+RANSAC method were still small, but increased more quickly compared to our methods.

The improved outlier robustness of our methods compared to 5-point+RANSAC is not an unusual result since our methods perform robust estimation in a 2-dimensional space instead of a higher, 5-dimensional one. It is simply more efficient and effective to perform the task of clustering inliers from outliers in a lower dimensional space. Also, since each constraint used less points compared to the 5-point algorithm, the probability of obtaining a good constraint is significantly increased, which is important for the sampling done in the antipodal+RANSAC framework. Similar improvements in robustness were also observed in Part II for

our algorithms that estimated differential camera motion.



(a) (b)

Figure 7.6: (a) Run-time versus increasing outlier proportion. (b) Using a smaller scale for the y-axis (zooming in) to compare the performance of the two antipodal methods.

Run-time: It is important to keep in mind that the above results for the antipodal+voting method happened in constant processing time regardless of the outlier proportion in the data[3]. In contrast, Figure 7.6(a) shows that the time taken for 5-point+RANSAC to arrive at a good solution increased quickly with increasing outlier proportions. Meanwhile, Figure 7.6(b) compares the timing for the antipodal+RANSAC and antipodal+voting methods. We see that run-time for antipodal+RANSAC also increased, but at a slower rate compared to 5-point+RANSAC, since it uses less points and solves a much simpler, intersection of two planes problem. The numbers in the graphs are obviously implementation dependent, but the *trend* will remain the same.

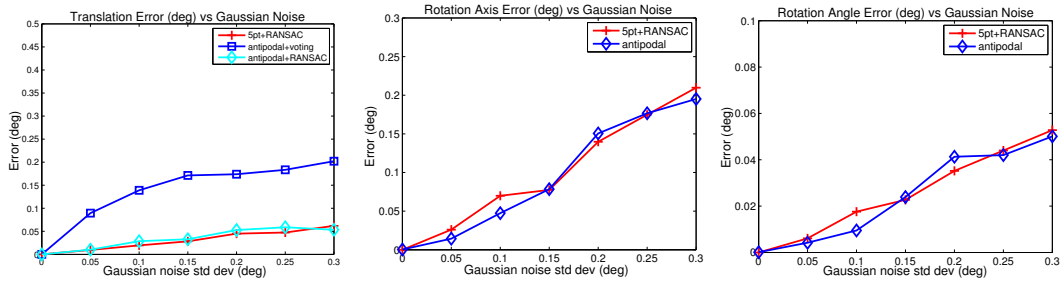
As outlier proportion increases, RANSAC increases the number of solutions, N, that are evaluated and tested, in order to maintain a high chance of hitting on a good one. N is adaptively increased according to $N = log(1-p)/log(1-q^s)$, where the number of points needed for generating a solution is $s = 5$ for the 5-point algorithm. q is the inlier probability, and p is the probability of there being at least one sample of outlier-free points in the N samples chosen. The antipodal+RANSAC algorithm also faces the same problem of increasing run-time but the increase is less rapid.

Another parameter influencing RANSAC processing time is the distance threshold used for distinguishing outliers from inliers. If the threshold is made stricter,

[3]The run-time behaviour of antipodal+voting is similar to that observed in the tests on voting algorithms for finding differential egomotion in Chapter 6.

the errors in the estimate may be further lowered. For example, in Figure 7.5, by reducing the distance threshold of 5point+RANSAC from 0.01 to 0.005, the error is improved in general. However, RANSAC will then have to iterate many more times and take even longer to obtain the solution.

Our implementation of the antipodal+voting method is constant-time because the approach is sufficiently fast such that using all the constraints arising from every antipodal pair is viable. Also, if M antipodal pairs are available, there are only M antipodal-epipolar constraints to consider. Hence the run-time is linear in the number of antipodal points used for estimation.



(a) (b) (c)

Figure 7.7: Performance degrades gracefully under increasing Gaussian noise.

Gaussian Noise: Figures 7.7(a-c) show that all methods tested gave small errors, with performance degrading gracefully under increasing Gaussian noise. Figure 7.7(a) shows that errors for the translation estimates of antipodal+RANSAC were about similar to that of 5-point+RANSAC. Translation errors from antipodal+voting were slightly higher (less than 0.2° difference) due to quantization error from dividing the solution space into a finite set of voting bins; its performance may be improved by voting with finer resolutions (we used only 2 coarse-to-fine stages). The errors for rotation axis and angle in 7.7(b-c) were also comparable to 5-point+RANSAC.

7.4.2 Real Image Sequences

Method: Experiments were conducted with real image sequences obtained from a Ladybug omnidirectional camera [187]. In Image Sequence A, the camera was placed on the ground, where it translated by about 4cm and rotated by 10° between frames. The scene was static. The motion was estimated and compared with measured ground truth values. Figures 7.9(a-e) show the images captured by the Ladybug omnidirectional camera rig at one time instance in sequence A.

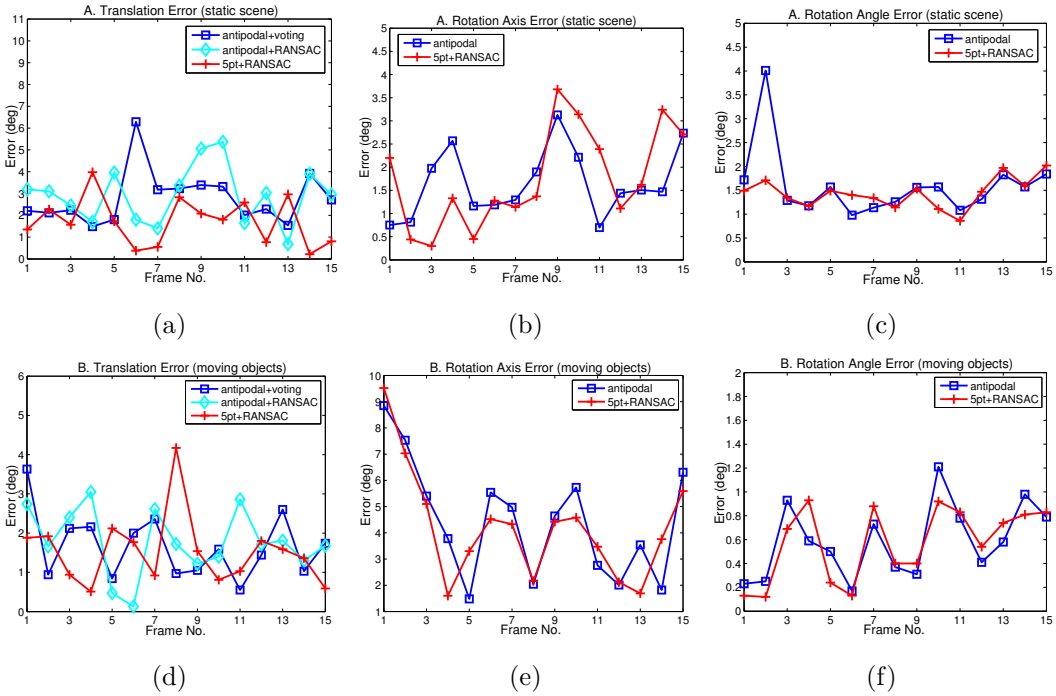


Figure 7.8: Experiments on real images. (a-c) Image Sequence A - static scene, translation 4cm, rotation 10° per frame. (d-f) Image Sequence B - dynamic scene containing moving objects, translation 4cm, rotation 4° per frame.

In another experiment, Image Sequence B, the camera translated by 4cm and rotated by 4° between frames; furthermore, the scene contained independently moving objects, leading to greater outlier probabilities. Also, the fact that motion was planar, was *not* used by any of the algorithms to simplify estimation.

Note that these are the same sequences tested on the GC+RANSAC and GC+voting algorithms which also find intersections of planes (Section 6.2.3). The difference is that here, we skipped every alternate frame in order to obtain larger camera motions between frames.

SIFT features (code from [143]) were used to find point correspondences. In the real images, several hundred antipodal points can typically be found, which are many more than the minimum needed. Points that are within 0.5° of being antipodal are approximated as antipodal.

Motion Estimates: Figures 7.8 (a-c) show errors for image sequence A (static scene) whilst Figures 7.8 (d-f) show errors for sequence B (contains moving objects). In these experiments, the three methods performed comparably. The differences in average errors of all three methods were small - less than 1° - which is within the bounds of the accuracy of our ground truth measurements.

The accuracy of all methods were affected by practical issues such as the fact that the Ladybug camera rig is only approximately central - the centres of its constituent cameras dont exactly coincide. Also, our methods approximated nearly antipodal points to be antipodal. In spite of this additional source of error, our methods performed comparably to 5-point.

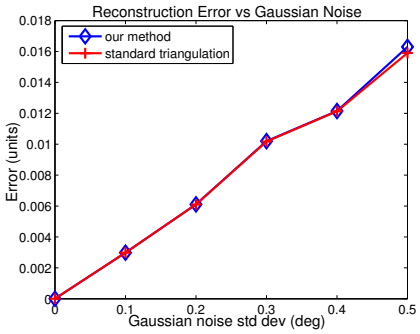
The rotation angle in sequence A (10°) was larger than in sequence B (4°), resulting in better rotation estimates and worse translation estimates for A (and the reverse for results of B). Videos showing the experiments and recovered estimates are included as supplementary material (see Appendix A).

7.4.3 Structure Estimates

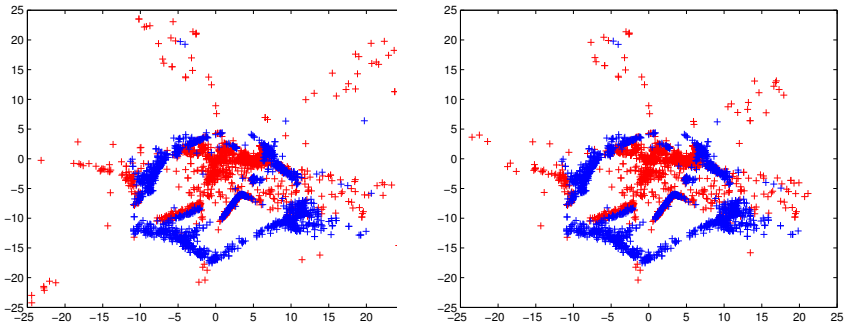
Our structure from partial-motion method is compared with a standard linear triangulation method (see [92, 93] for details) in simulations under noise and in reconstructions of real scenes.

Under increasing Gaussian noise, the simulation results of Figure 7.9(f) show that our partial-motion method and linear triangulation performed comparably, with both giving small average reconstruction errors. Both methods used true camera motions and noisy correspondence data as inputs. Errors were measured as the Euclidean distance between a reconstructed point and the (scaled) true

(a) (b) (c) (d) (e)



(f)



(g) (h)

Figure 7.9: (a-e) Example images from Image Sequence A taken by the 5 cameras on the Ladybug omnidirectional camera rig. (f) Gaussian noise simulation comparing our partial-motion structure method with standard linear triangulation. (g-h) Top-down view of structure estimates for Image Sequence A using our method and using linear triangulation respectively, with ground truth motion as the inputs. Points on the ground are marked red and points corresponding to objects above the ground are marked blue.

world point.

We also performed reconstruction on Image Sequence A, and Figure 7.9(g-h) shows a bird's eye view of the reconstructed environment. Using measured ground truth motion as inputs, both methods gave very similar reconstructions. Blue denotes points belonging to objects above the ground whilst points on the ground are marked red.

All plots show results *before* performing any nonlinear refinement, which can be applied here to improve accuracy. The scale of the plotted structure is arbitrary since it is not recoverable.

7.5 Discussion

Real-time implementations: Like our previous voting implementations, the antipodal+voting algorithm is also fundamentally parallel, and implementation on a GPU or FPGA should further speed it up as the hardware would be able to take advantage of this. Furthermore, since it works in constant-time under increasing outliers, it can potentially give robust, fast performance even in complex scenes with large, variable outlier proportions.

Decoupling leads to improved algorithms: Motion decoupling leads to novel algorithms that were previously not efficient or not possible for estimating the discrete camera motions. For example, bringing the problem down to lower dimensions enabled us to efficiently perform Hough-voting. It also led to the structure from partial-motion algorithm, which saves time by not having to explicitly find rotation.

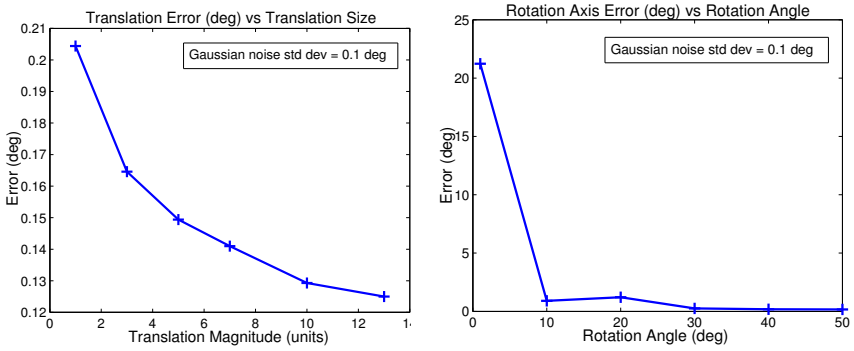
Further improvements on accuracy: Whilst the results show that the methods discussed give accurate motion estimates, further linear and nonlinear refinement may be performed to obtain even more accurate results. Once translation and rotation have been estimated, *all* inlier points may be used to linearly refine the estimated motions (whereas in the experiments above, linear refinement was performed using only the inlier antipodal points and points without antipodes were not used). Further refinement using nonlinear techniques such as bundle adjustment are also standard.

Effect of motion size: Figures 7.10(a-b) show the effects of different magnitudes of camera motion. In (a), as the magnitude of translation increases, the translation is estimated with increasing accuracy. Conversely, in (b), as the rotation angle becomes small, rotation is increasingly difficult to recover. These

effects are in accordance with the behavior of virtually all other motion estimation methods.

Finding antipodal points efficiently: In our experiments, we searched through all available correspondences for antipodes, which is not very efficient. (Recall that in Section 6.2.1, for differential motions, we were able to efficiently obtain antipodal flow via the Lucas-Kanade method. However, for large discrete motions, this method is not suitable.) For a faster method, we recommend avoiding this search by using DAISY features [236]. Whilst SIFT features are meant to be computed only at certain positions in the image (at the local extrema in a stack of Difference-of-Gaussian images), DAISY features are designed to be computed anywhere in the image. The antipodal points can then be chosen in advance and the DAISY descriptor is computed at those points.

Deterministic solutions: It is interesting to note that the solution given by RANSAC is random - i.e. 100 different runs can give 100 different solutions scattered around the 'true' solution. This happens since sampling is random, so slightly different sets of inliers will be found each time [135]. On the other hand, the voting approach suggested here gives deterministic solutions since random sampling is not used.



(a) (b)

Figure 7.10: Performance for different motion sizes.

7.6 Discrete versus Differential

There has been some debate over the range of translation magnitudes or baselines, for which discrete egomotion methods work. It was initially believed that discrete egomotion methods may become less stable under very small (i.e. near differential) motions. Recently however, some researchers have suggested that, under small

camera motions, discrete motion methods work well and may even outperform differential methods [119, 138, 243, 152, 20][4].

For example, Lin et al. [138] showed that under certain conditions and the assumption of proportional noise, the discrete 8-point algorithm may perform similarly for both small and large motions. Also, in simulations under differential camera motion, they found that three variants of the discrete 8-point algorithm [87, 168, 141] performed better than the differential methods of Ma et al. [149], and of Jepson and Heeger [114]. The main reasons cited by the authors for the poor performance of those differential methods were the bas-relief ambiguity and the forward bias.

In view of these findings, this section seeks to compare the performance of our discrete and differential antipodal constraints via simulations with different motion sizes. In particular, we consider the differential antipodal constraint obtained by summing antipodal flow (Chapter 5) and the discrete antipodal-epipolar constraint discussed earlier in this chapter, since the two methods are geometrically dual to each other - i.e. they both involve finding translation from the intersection of multiple planes. In the following, we will refer to these as the 'sum-antipodal' and 'antipodal-epipolar' constraints respectively.

In this set of simulations, the level of Gaussian noise used was fairly small and there were no outliers present. Hence, outlier rejection schemes like RANSAC and voting were not used. From the image motion observed at 100 antipodal points, the differential sum-antipodal and discrete antipodal-epipolar constraints each gave a set of equations for which least squares solutions to translation were found.

We also compared performance with a discrete method for linearly recovering the essential matrix, termed 'discrete essential matrix'. With the same set of point correspondences (a total of 200 points, since there were 100 antipodal pairs), the conventional epipolar constraint gave a system of linear equations from which a least squares solution to the essential matrix may be found by the DLT algorithm. The translation was then extracted from the matrix.

Figures 7.11(a-c) show the effects of decreasing translation baseline by several orders of magnitude, on the performance of the three approaches. The distance

[4]Such research lends credence to our use of the discrete 5-point algorithm as a benchmark against which we compared both our differential and discrete antipodal constraints in earlier sections. Note that, to be on the safe side, we did ensure that all experiments involving 5-point used motion sizes that were not overly small. The resulting performance for 5-point was quite good in all the experiments.

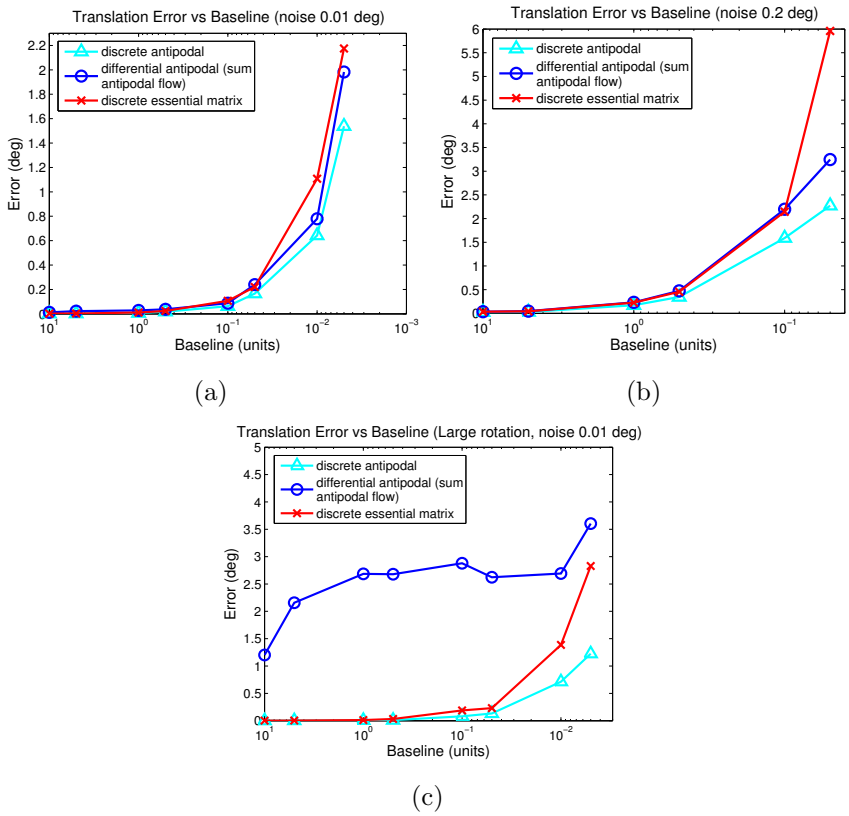
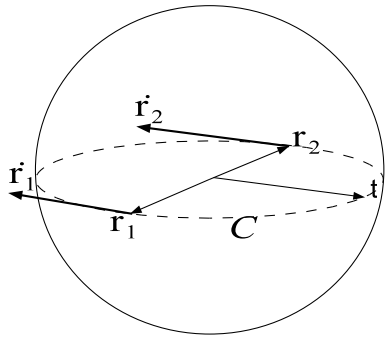


Figure 7.11: Error in translation estimate for decreasing translation baselines. The experiments were performed with the following Gaussian noise standard deviations (s.d.) and rotation angles: (a) s.d. $= 0.01°$, angle $= 0.1°$. (b) s.d. $= 0.2°$, angle $= 0.1°$. (c) s.d. $= 0.01°$, angle $= 10°$.

to world points varied randomly between 10 and 15 units. In Figures 7.11(a) and (c), the Gaussian noise used had a standard deviation of $0.01°$. Meanwhile, Figure 7.11(b) shows the same experiment but with a larger Gaussian noise level of $0.2°$ standard deviation.

In general, as translation magnitude decreased, all three methods had increasing difficulty recovering translation direction. As expected, the larger Gaussian noise in Figure 7.11(b) means that compared to (a), the performance begins to degrade much sooner and the rate of degradation is also increased. This is because decreasing motion size leads to a corresponding decrease in the image motion observed, hence the same level of noise will degrade the observations more severely.

Interestingly, from Figures 7.11(a-b), we see that the differential 'sum-antipodal' method performed comparably with the discrete methods for *large* translation



(a)

Figure 7.12: The sum-antipodal constraint under large translations. With no or small rotation, the plane (or great circle C) constraining translation can still be found accurately for large baselines.

baselines. However, note that the rotation used in (a) and (b) is fairly small - the rotation angle is 0.1°. In Figure 7.11(c), a much larger rotation angle of 10° is used and this results in a significant decrease in the accuracy of the differential method, whilst the discrete methods were hardly affected by the larger rotation.

This means the differential sum-antipodal constraint is valid over a large range of translations, but works best with small or differential rotations. Recall that the sum-antipodal method cancels the effects of rotation by summing antipodal flow vectors in order to obtain a plane constraining translation. If the rotation is large, summing antipodal flow becomes less effective at canceling the rotation. However, the estimation of the constraint plane is not affected by the magnitude of translation. Consider for example, the case of pure translation (no rotation) shown in Figure 7.12. As the magnitude of translation increases, the antipodal flow vectors observed merely grow longer, but they will still lie in the plane of the great circle C, which can be recovered by a cross product of the antipodal flow with any one of the antipodal points.

Figures 7.11(a-b) also demonstrate that for small baselines and small rotation, our differential sum-antipodal method outperformed the discrete method of finding essential matrix. We attribute this to the advantages of working in a lower dimensional space compared to estimating the solution in higher dimensions. Furthermore, our discrete antipodal-epipolar method outperformed the differential sum-antipodal method for small baselines in all experiments.

In short, our findings were largely consistent with previous research into the differences in performance between discrete and differential egomotion approaches. Like Lin et al. [138] and Kanatani [119], we found that our dis-

crete antipodal-epipolar method performed as well or better than the differential method for small motions. However, unlike Lin et al. [138], where the differential methods tested performed very poorly due to large errors caused by the bas-relief ambiguity and the forward bias problem, our differential sum-antipodal method worked quite well for small motions, surpassing the discrete essential matrix approach. This is because the large FOV used by antipodal point methods means that the bas-relief problem was virtually non-existent. Also, due to the image sphere formulation, our method is unaffected by forward bias (refer Section 3.1.4, page 43).

7.7 Summary

The main contribution of this chapter was the antipodal-epipolar constraint on camera motion. We demonstrated that the geometry of antipodal points resulted in linear, decoupled constraints on the translational and rotational components of motion. We also showed that the antipodal-epipolar constraint works quite well for both large and small motions.

We demonstrated the use of our constraints within the antipodal+RANSAC algorithm which is trivial to implement, and shows improvements in robustness to outliers and in performance time compared to the state-of-the-art. The second, Hough-voting algorithm showed similar improvements, with the added advantage of being well suited for implementation on a parallel machine.

We believe the decoupling of motion components simplifies the problem, and the advantages of this were demonstrated in the novel algorithms proposed for finding motion and estimating structure. This feature may be further exploited in other applications as well - for example, certain camera odometry applications do not strictly require rotation recovery, and a similar algorithm that skips finding rotation may be possible, leading to speedier estimation. Conversely, if an application needs to recover rotation (e.g. for attitude stabilization of a flying vehicle) then translation estimation may be skipped and only rotation is found.

Chapter 8

Relaxed Egomotion and Visual Homing

8.1 Introduction

In this chapter, we look at the problem of estimating observer egomotion from a slightly different perspective. Previously, we have been preoccupied with obtaining estimates of egomotion as accurately as possible. Here, however, we look at what we term **relaxed egomotion**, where we relax requirements for high accuracy in the egomotion estimates whilst still maintaining stable and robust performance under outliers. This generally leads to a simpler problem which may be easier and faster to solve. We will study the usefulness of these considerably less accurate estimates in the context of visual navigation.

In fact, we have already encountered this idea of relaxed egomotion in the previous chapters of this work. Earlier, we saw how voting along constraint regions (lunes, great circles and straight lines) led to a *probability map* of the solution space, where more likely solutions had higher votes and the best solution had the most votes. We mentioned that if a single, sharp peak in the voting distribution could not be obtained, then the result obtained was less accurate and precise, but still correct in the sense that the true solution lay within the set of bins with maximum votes (we called this the 'fail-soft' property of the voting approach, page 74). See Figure 8.1.

This is a relaxed egomotion estimate - a robust but less accurate answer with known uncertainty is obtained. The uncertainty is bounded by how many bins of maximum vote there are and how widely spread out they are over the solution space. As long as certain bounds on uncertainty are met - i.e. it is not too large

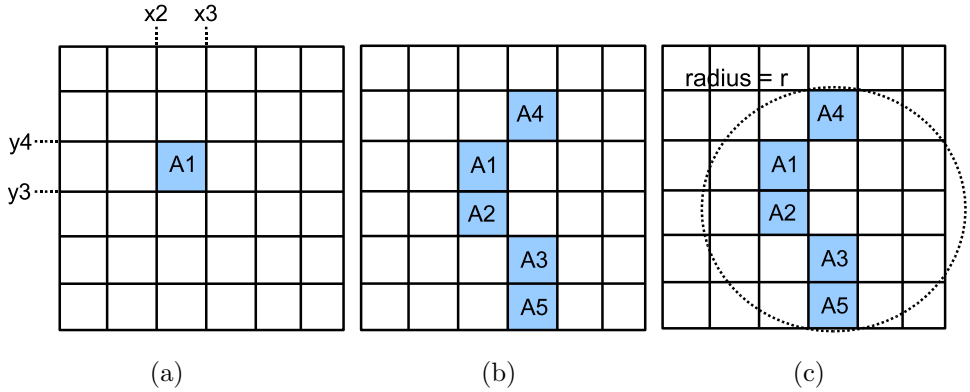
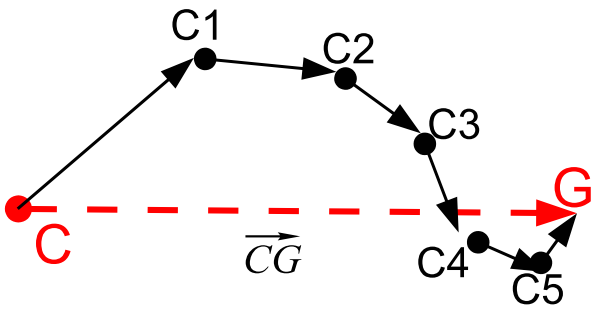


Figure 8.1: Shaded bins indicate voting bins with maximum vote. (a) Voting bins are a discretized representation of the solution space. If bin A1 had the most votes, the solution may lie in the shaded region $x2 < x < x3$ and $y3 < y < y4$. (b) However, if more than one bin has the highest vote, then the set of possible solutions is the union of all such bins (not necessarily a connected set). The solution then lies with maximum probability in the region A1 $\cup$ A2 $\cup$ A3 $\cup$ A4 $\cup$ A5. (c) The uncertainty in the estimate is bounded by the smallest circle encompassing all bins with highest votes, that is, the diameter of that circle (= 2r) is the largest error possible (for example, when a point on one of the upper vertices of bin A4 is picked as the estimate but the true solution is a point on one of the lower vertices of A5).

- this relaxed estimate is useful for many applications and navigational tasks. This approach has the advantage of faster estimation and lower computational demands, which may be useful for small robots with limited payloads. For example, small quadrotor aircraft with 250g-300g payloads [53], or microflyers that weigh only 10g-30g in total [282, 267], would carry limited computing power on board and may benefit from simple and cheap homing algorithms.

Alternatively, a system could give a fast and rough answer first, and later return with a more accurate answer after further processing. This would be a coarse-to-fine approach, with the relaxed egomotion method giving a coarse estimate first; and one of the conventional egomotion estimation algorithms giving an accurate and precise answer later.

Here, we will consider relaxed egomotion in the context of a **visual homing** application. Given two views of the world - the home view and the current view - a visual homing algorithm attempts to infer directions to move in, that would take the agent from the current location, C, to its desired home location, G



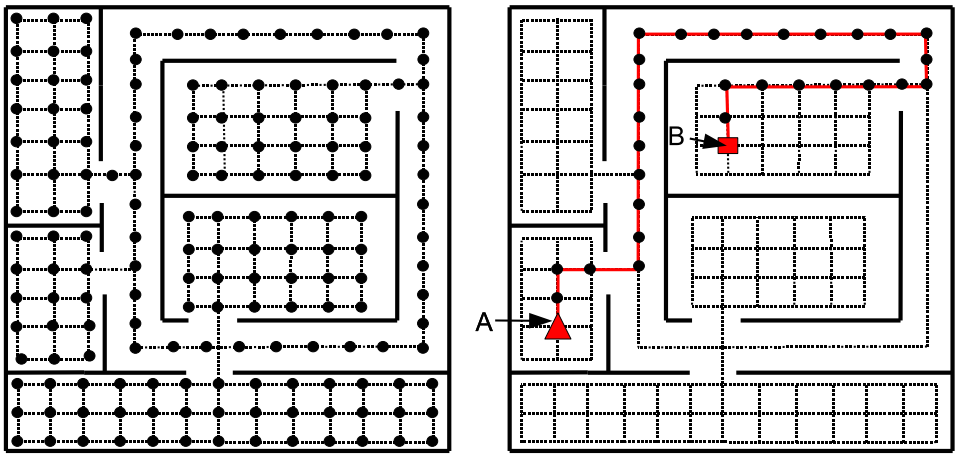
(a)

Figure 8.2: Visual homing: whilst $\overrightarrow{CG}$ gives the most direct path home, continual updates of relaxed egomotion estimates can still get a robot home, albeit with a less direct path. This reduces the computational burden of having to obtain highly accurate egomotion estimates.

(refer Figure 8.2). We show that any two image points (which do not have to be antipodal) give weak constraints on the direction of translation, $\overrightarrow{CG}$, under certain conditions. Information from several such constraints may be pooled together via a Hough-like voting framework to give a relaxed estimate of translation. Rather than a single homing direction, a set of directions (corresponding to all the bins with maximum vote) is returned.

We show that if the uncertainty meets certain easily satisfiable conditions, homing directions may be picked such that the agent is guaranteed to reach the home location successfully. In fact, we will argue that these conditions for successful homing are satisfied even with very 'rough' estimates of the possible homing direction, which allows for quite coarsely sampled voting tables (e.g. the 360° of directions in which a robot moving on the ground plane may move in can be represented with 360 bins, or even 36 bins), which makes the method computationally cheap.

The inputs to the homing algorithm are point correspondences between two views of the scene, found using methods such as SIFT, SURF etc. In this chapter, following the terminology of homing literature, we will call these point matches **landmarks**. These are 'naturally' occurring landmark points, rather than objects placed in the scene for the sole purpose of being an easily tracked and unmoving landmark. This strategy deals with *local, short-range* homing where the current and goal positions are not too far apart, such that there are sufficient point correspondences between the two views of the world (i.e. over the same distances



(a) (b)

Figure 8.3: (a) The topological map of an indoor environment. The round dots mark nodes in the map. To get from a node to any neighbour connected to it by a dotted line, a local homing algorithm is used. (b) To navigate over a larger distance, for example to get from node A to node B, a path is found and all the intermediate nodes in that path are noted. Local homing is then used to get from one intermediate node to the next, until the robot arrives at B.

that one would use discrete camera egomotion methods).

Such a homing strategy may be extended to perform localization and long-range navigation within the framework of a **topological map** [78, 71, 45, 7, 69, 68, 128]. A topological map is a graph-like structure of connected nodes representing locations in the environment (see Figure 8.3). Any two connected nodes are locations which are sufficiently close together such that a local homing algorithm may be used to move the agent from one node to the other. Each node is represented by either an image taken by the agent at the corresponding location, or a list of landmark points extracted from that image. Long-range homing, localization and other such behaviours may be realized in such a framework using the map and a local homing algorithm.

We will focus on a robust approach, leading to successful homing even in real, dynamic environments where significant numbers of landmark points are wrong or missing (i.e. outliers). Section 8.2 introduces three algorithms. Firstly, a homing algorithm with both the robot and landmark points lying on the same plane is presented and shown to converge theoretically. Next, we investigate the case of planar homing with general, 3D landmark points and present two algorithms - the *conservative* method, which has theoretically guaranteed convergence and a

non-conservative method which is demonstrated to converge empirically. We also show that it is possible to extend this to homing for robots or agents moving in 3D. Implementation details are in Section 8.3 and experimental results are in Section 8.4.

8.2 Theory

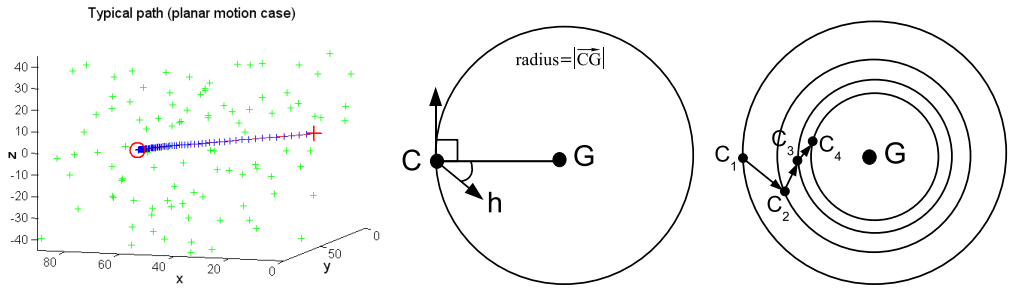


Figure 8.4: (a) An experiment for homing amidst a cloud of landmark points (green crosses). The starting position is marked with a red cross and the robot moves from one position to the next (blue crosses) until it converges on the goal position (red circle). (b) For a small motion, the robot moves closer to G (into the circle) as long as the angle between $\overrightarrow{CG}$ and $\mathbf{h}$ is less than 90°. (c) If this is repeated, the robot gets closer and closer to G.

Let the goal or home be a point G and the current location of the robot be the point C. Images of the environment are taken at both points. The robot uses this image pair to compute some homing direction, $\mathbf{h}$. It then moves in that direction for a certain distance, which we call the step size of the robot. Consequently, the current location of the robot changes and a new current view of the world is observed. This is repeated, with the current position of the robot, C, changing with each iteration, and homing is successful when C converges on G (see Figure 8.2 and Figure 8.4(a)).

Convergence of the robot onto G may be defined as moving the robot such that, at the limit, the distance between the robot and the goal, $|\overrightarrow{CG}| \rightarrow 0$. This will be the definition that we refer to in the theoretical discussions that follow in the rest of Section 8.2.

However, this idea is not meaningful in practical systems as random noise in the observer and controller means that there will always be some error such that $|\overrightarrow{CG}|$ is never quite zero (e.g. this may manifest as oscillatory behavior about

G as the robot repeatedly tries and fails to arrive exactly at G). Consequently, a more practical definition of convergence is that of moving the robot to within some small neighbourhood of G and having it remain inside this neighbourhood. In other words, at the limit, we desire $|\overrightarrow{CG}| < \epsilon$, where ϵ is generally small. We will refer to this **practical convergence** when we discuss actual implementations and experimental results in Sections 8.3 and 8.4.

We observe that:

Observation 8.1. The robot converges, if at each iteration it takes a small step in the computed direction $\mathbf{h}$, such that the angle between $\overrightarrow{CG}$ and $\mathbf{h}$ is less than 90°.

C lies on a circle centered on G with radius $|\overrightarrow{CG}|$ (see Figure 8.4(b)). If the movement of the robot is small, it will always move *into* the circle if homing direction $\mathbf{h}$ deviates from $\overrightarrow{CG}$ by less than 90°. (At exactly 90° from $\overrightarrow{CG}$, the robot moves off the circle in a tangent direction.) This ensures that each step the robot makes will take it a little closer to the goal (since it is moving into the circle), as shown in Figure 8.4(c). Hence the robot-goal distance decreases monotonically, and at the limit, it converges on G.

The above is conditioned on the robot step size being small compared to $|\overrightarrow{CG}|$. This is necessary in order for the assertion that the robot always moves into the circle if the angle between $\mathbf{h}$ and $\overrightarrow{CG}$ is less than 90°, to be true. In practice, robot motion is not infinitesimally small, but as long as the angle between $\mathbf{h}$ and $\overrightarrow{CG}$ is not too close to 90°, the above still holds.

The condition of a small step size relative to $|\overrightarrow{CG}|$ also implies that the step size decreases as the robot approaches G (i.e. as $|\overrightarrow{CG}|$ decreases). This ensures that the robot does not repeatedly overshoot the goal and oscillate about the point G.

8.2.1 The Case of the Planar World

Consider the case in which both robot and landmark points lie on a plane. Given 2 landmarks L_1 and L_2 (Figure 8.5), and the current position at point C, the *inter-landmark angle* is the angle between the two rays $\overrightarrow{CL_1}$ and $\overrightarrow{CL_2}$. All points on circular arc L_1CL_2 observe the same angle $\angle L_1CL_2$. Following Argyros et al. [9], we define a convention where angle $\angle L_1CL_2$ is consistently measured, going from $\overrightarrow{CL_1}$ to $\overrightarrow{CL_2}$, in a anticlockwise (or clockwise) direction. Then any point

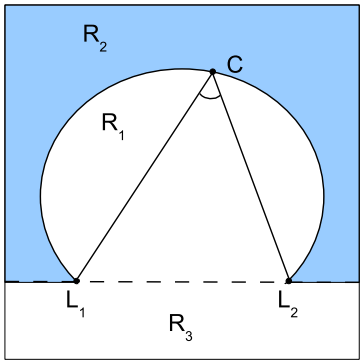


Figure 8.5: Horopter L_1CL_2 and line L_1L_2 splits the plane into 3 regions.

on arc L_1CL_2 will have angle $\angle L_1CL_2 < 180°$ (or $> 180°$, if measuring in the clockwise direction). The inter-landmark angle observed at the current position, $\angle L_1CL_2$, is the current angle, whilst that observed at the goal position, $\angle L_1GL_2$, is the goal angle.

The circular arc L_1CL_2 is termed a *horopter*. The dashed landmark line L_1L_2 splits the plane into upper and lower half-planes. Let the set of points on the horopter be R_0; let the region in the upper half-plane and within the horopter be R_1; the region outside the horopter (shaded region in Figure 8.5) be R_2; and the region in the lower half-plane be R_3.

In the configuration of Figure 8.5 and using the anticlockwise angle measurement convention, if C lies on the horopter, and G lies within R_1, that is $G \in R_1$, then $\angle L_1GL_2 > \angle L_1CL_2$. However, if $G \in R_2$, then $\angle L_1GL_2 < \angle L_1CL_2$. Furthermore, the anticlockwise convention for measuring angles implies that if point $P \in R_1 \cup R_2$, then $\angle L_1PL_2 < 180°$, but if $P \in R_3$, then $\angle L_1PL_2 > 180°$. Therefore, $C \in R_0 \cup R_1 \cup R_2$ and $G \in R_3$, implies $\angle L_1CL_2 < \angle L_1GL_2$.

The angle size relationships are reversed if C lies on the other side of the line L_1L_2 (the lower half-plane). Angle $\angle L_1CL_2$ is now less than 180° when measured in the clockwise direction and more than 180° if measured in the anticlockwise. An analysis will yield relationships symmetrical to the above.

With this, given knowledge of whether the current or the goal angle is larger, one can constrain the location of G to one of regions R_1, R_2, R_3. However, since distance from the current point to the landmarks is unknown, we know neither the structure of the horopter nor that of the line L_1L_2. Even so, one can still obtain valid but weaker constraints on the location of G purely from the directions of the landmark rays.

These constraints on G come in two types. Type 1: Define a region R_{A1}

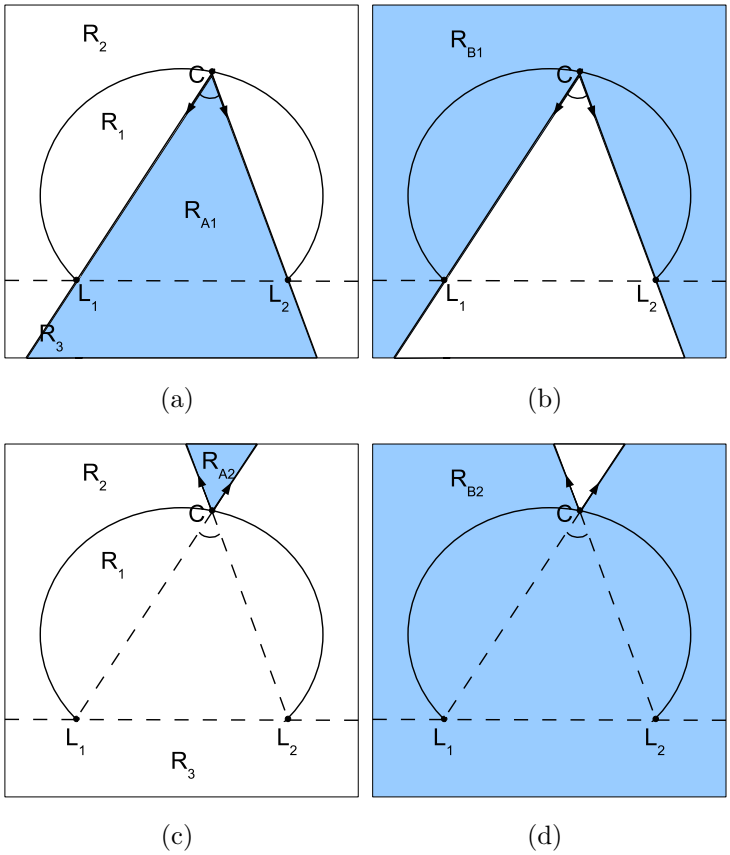


Figure 8.6: Regions R_{A1} and R_{A2} - shaded regions in (a) and (c). Type 1 and Type 2 constraint regions are given by the complement of R_{A1} and R_{A2}, that is, the regions R_{B1} and R_{B2} which are the shaded areas in (b) and (d).

that lies between vectors $\overrightarrow{CL_1}$ and $\overrightarrow{CL_2}$ (shaded region in Figure 8.6(a)). Let the complement of R_{A1} be $R_{B1} = R_{A1}^c = \Pi \setminus R_{A1}$, where Π is the entire plane and $\setminus$ denotes set difference. R_{B1} is the Type 1 constraint region (the entire shaded region in Figure 8.6(b)) and G must lie within R_{B1}.

Type 2: Let R_{A2} lie between the vectors $-\overrightarrow{CL_1}$ and $-\overrightarrow{CL_2}$ (shaded region in Figure 8.6(c)). The Type 2 constraint region is the complement set, $R_{B2} = R_{A2}^c$ (the shaded region in Figure 8.6(d)).

Lemma 8.1. In the case of a current angle which is less than 180°, if it is greater than the goal angle, then $G \in R_{B1}$, but if it is less than the goal angle, then $G \in R_{B2}$. In the case of a current angle larger than 180°, if it is greater than the goal angle, then $G \in R_{B2}$ but if it is less, then $G \in R_{B1}$.

Proof. If $\angle L_1CL_2 < 180°$ and $\angle L_1CL_2 > \angle L_1GL_2$, the goal, G, must lie in R_2. R_{A1} is the shaded area in Figure 8.6(a), which does not intersect R_2. One can

see that for any C lying on the horopter, R_{A1} never intersects R_2. R_2 is a subset of R_{B1}. Hence, $G \in R_2 \Rightarrow G \in R_{B1}$.

Conversely, if $\angle L_1CL_2 < 180°$ and $\angle L_1CL_2 < \angle L_1GL_2$, then $G \in R_1 \cup R_3$ and $G \notin R_2$. R_{A2} is the shaded area in Figure 8.6(c) and $R_{A2} \cap (R_1 \cup R_3) = \emptyset$ for any point C on the horopter. So, $G \in (R_1 \cup R_3) \subset R_{B2}$.

For $\angle L_1CL_2 > 180°$ (which occurs if C is above line L_1L_2 in Figure 8.5 and the clockwise convention is used; or if C is below L_1L_2 and the anticlockwise convention is used), the proof is symmetrical to the above (that is, if $\angle L_1CL_2 > \angle L_1GL_2$, then $G \in R_1 \cup R_3$ and $(R_1 \cup R_3) \subset R_{B2}$ so $G \in R_{B2}$. Also, if $\angle L_1CL_2 < \angle L_1GL_2$, then $G \in R_2$ and $R_2 \subset R_{B1}$ so $G \in R_{B1}$). □

From the constraints on G, we can now define constraints on $\overrightarrow{CG}$. Some region, R, consists of a set of points, $\{P_1, P_2 \cdots P_i, \cdots\}$. We define a set of direction vectors:

$$D = \{\mathbf{d_j} \mid \mathbf{d_j} = \frac{\overrightarrow{CP_i}}{|\overrightarrow{CP_i}|},\ \forall\ i > 0, j \leqslant i\} \tag{8.1}$$

such that for every point in set R there exists some $k > 0$ and some $\mathbf{d_j} \in D$ such that $P_i = C + k\mathbf{d_j}$. (A robot starting at C and moving k units in direction $\mathbf{d_j}$ will arrive at P_i). As an example, consider the case in Figure 8.6(a). Let $\overrightarrow{CL_1}$ correspond to the polar coordinate angle, θ, of $\theta_{\overrightarrow{CL_1}} = 0$, and let the direction of θ vary in the anticlockwise direction. Then, region R_{A1} maps to the fan of vectors with polar angle $0 < \theta < \theta_{\overrightarrow{CL_2}}$ whilst region R_{B1} maps to the vectors with polar angle $\theta_{\overrightarrow{CL_2}} < \theta < 360°$.

$\overrightarrow{CG}$ is the vector pointing from current to goal location. Recovering it guarantees convergence (moving in the direction $\overrightarrow{CG}$ repeatedly will, inevitably, bring the robot to G). It also gives the most efficient path (the straight line) to the goal. In the following, we are interested only in the direction of $\overrightarrow{CG}$ and we will use $\overrightarrow{CG}$, and the unit vector in the direction $\overrightarrow{CG}$, interchangeably. Then:

Theorem 8.1. From one landmark pair, Lemma 8.1 constrains G within regions R_{B1} or R_{B2}. Equation 8.1 maps this to a set of directions, D, where $\overrightarrow{CG} \in D$. For N landmarks, we have up to ${}^NC_2 = \frac{N!}{(N-2)!2!}$ vector sets $\{D_1, D_2, \cdots D_{{}^NC_2}\}$. $\overrightarrow{CG}$ lies in their intersection, $D_{res} = D_1 \cap D_2 \cap \cdots \cap D_{{}^NC_2}$.

In general, assuming isotropically distributed landmarks, the larger the number of landmarks (N) used, the more constraints there are on $\overrightarrow{CG}$, so the region D_{res} will be smaller. In practice, successful homing only requires N to be large

enough such that a homing direction can be chosen, that is less than 90° from *every* vector in D_{res}. It will then be less than 90° from $\overrightarrow{CG}$ and from Observation 8.1, the robot will converge. A stricter condition is to have the maximum angle between any two vectors in D_{res} less than 90°. Then all vectors in D_{res} will be less than 90° from $\overrightarrow{CG}$, that is, choosing any vector in D_{res} as the homing direction will move the robot closer to G.

8.2.2 Robot on the Plane and Landmarks in 3D

Whilst the robot can move on some plane with normal vector, $\mathbf{n}$, the observed landmark points will, in general, not lie on that plane. Here, we extend the previous results to this more general situation. It is not unreasonable to assume that the robot knows which direction is 'up', that is, the normal to the ground plane[1]. Then, we can measure angles according to the same clockwise or anticlockwise conventions as before.

Let C, G lie on the x-y plane (so normal vector $\mathbf{n}$ is the z-axis) and suppose C lies in the negative-x region. There always exists some plane passing through two 3D points such that this plane is orthogonal to the x-y plane. So, without loss of generality, we can let the landmark pair L_1, L_2 lie on the y-z plane. There exists a circular arc passing through L_1, C and L_2. Revolving the arc about line L_1L_2 sweeps out a surface of revolution. The portion of the surface that lies in the negative-x region of $\mathbb{R}^3$ is the horopter surface. All points on the horopter have inter-landmark angles equal to $\angle L_1CL_2$.

The intersection of this 3D horopter surface with the x-y plane containing C and G is a 2D horopter curve (some examples shown in Figure 8.7 (a-f)). Rather than a circular arc, the curve is elliptical. Let region R_1 be the set of all points inside the horopter curve and in the negative-x region; let R_2 be the set of points in the negative-x region and outside the horopter; let R_3 be the set of points in the positive-x region. Then, the inequality relationships between $\angle L_1CL_2$ and $\angle L_1GL_2$ if C lay on the horopter and G in one of regions R_1, R_2 or R_3, are similar to those discussed in Section 8.2.1.

Since landmark distances are unknown, once again, we attempt to constrain G using only landmark directions. The difference is that we now use the projected landmark vectors $\overrightarrow{C\tilde{L}_1}$ and $\overrightarrow{C\tilde{L}_2}$ instead of the landmark vectors as was previously the case. The projection of a landmark ray vector, $\overrightarrow{CL_1}$, onto the plane is $\overrightarrow{C\tilde{L}_1} =$

[1]Note that this is *not* the same as finding the ground plane which involves segmenting pixels into ground and non-ground categories, and is in the most general case, a non-trivial task.

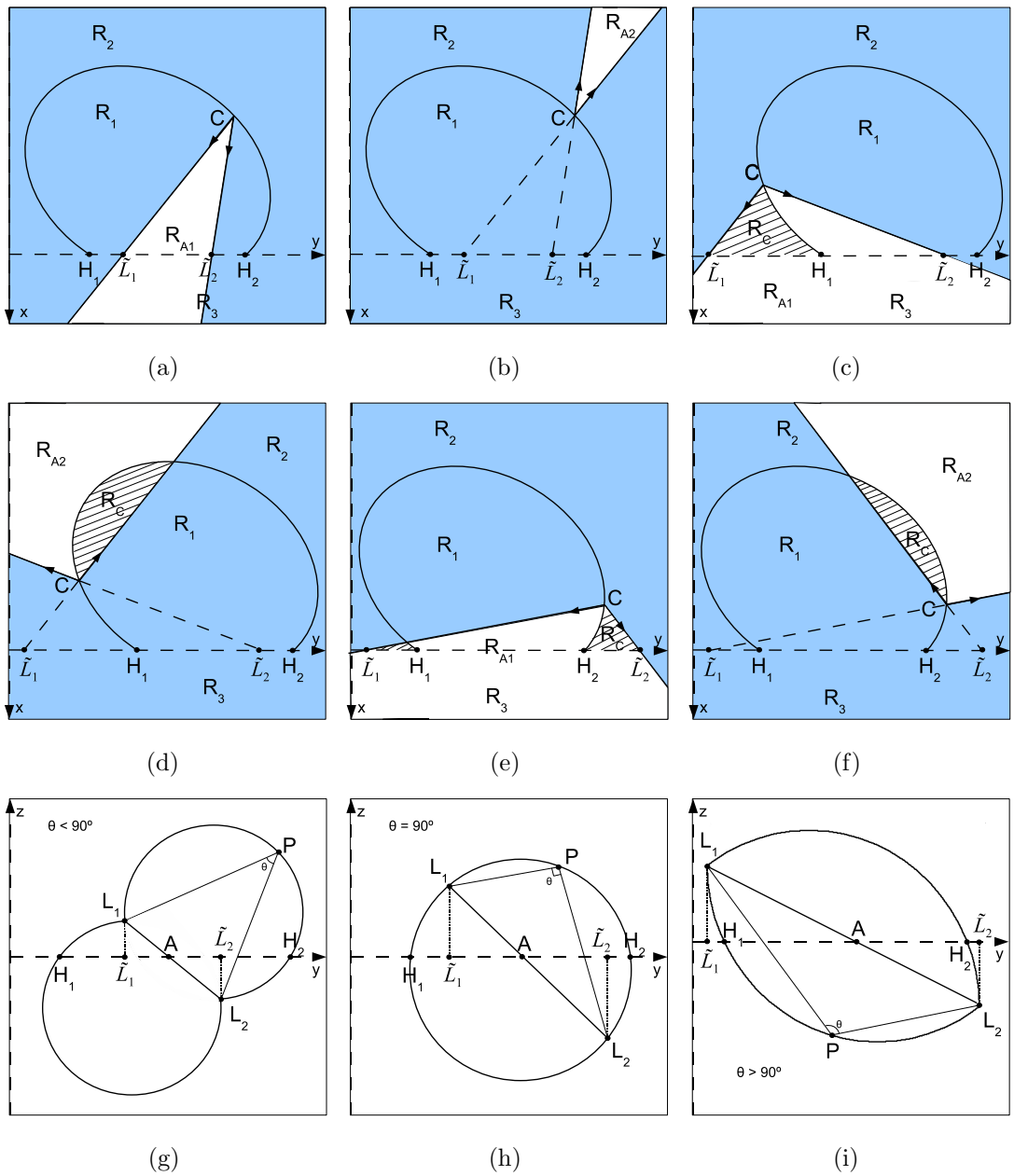


Figure 8.7: Intersection of horopter with x-y plane. Regions R_{B1}, R_{B2} constraining location of G are shaded. The complement regions, R_{A1}, R_{A2} are unshaded. (a-b) Case1: both $\tilde{L_{1y}}, \tilde{L_{2y}}$ in $[H_{1y}, H_{2y}]$. (c-d) Case 2: one of $\tilde{L_{1y}}$ or $\tilde{L_{2y}}$ outside $[H_{1y}, H_{2y}]$. (e-f) Case 3: both $\tilde{L_{1y}}, \tilde{L_{2y}}$ outside $[H_{1y}, H_{2y}]$. (g-i) Intersection of horopter surface with y-z plane. 3 cases arising from $\theta < 90°$, $\theta = 90°$ and $\theta > 90°$. P is any point that lies on the horopter.

$\mathbf{n} \times (\overrightarrow{C\tilde{L}_1} \times \mathbf{n})$ where $\tilde{L}_1$ is the projection of L_1 onto the plane and $\times$ is the cross product.

The horopter curve intersects the y-axis at H_1, H_2 and the projected landmark rays, $\overrightarrow{C\tilde{L}_1}$ and $\overrightarrow{C\tilde{L}_2}$ intersect the y-axis at $\tilde{L}_1, \tilde{L}_2$. However, whilst L_1, L_2 lie on the horopter, the projected landmark rays are such that $\tilde{L}_1, \tilde{L}_2$ do not lie on the horopter in general. (If they did, then the problem is reduced to that of Section 8.2.1.) Let P_y be the y-coordinate of point P. Three cases arise: (Case 1) both $\tilde{L_{1y}}, \tilde{L_{2y}}$ lie in the interval $[H_{1y}, H_{2y}]$; (Case 2) one of $\tilde{L_{1y}}$ or $\tilde{L_{2y}}$ lies outside that interval; (Case 3) both $\tilde{L_{1y}}, \tilde{L_{2y}}$ lie outside $[H_{1y}, H_{2y}]$.

For Case (1), the results of Lemma 8.1 hold except that the Type 1 and 2 constraint regions are now bounded by projected landmark rays. Figure 8.7(a) illustrates the case of $G \in R_2$. The shaded region is $R_{B1} = R_{A1}^c$, which is a Type 1 constraint bounded by $\overrightarrow{C\tilde{L}_1}, \overrightarrow{C\tilde{L}_2}$ such that $G \in R_{B1}$ (since $R_2 \subset R_{B1}$) for any C on the horopter curve. Meanwhile, Figure 8.7(b) illustrates the case of $G \in R_1 \cup R_3$ and the shaded region is the Type 2 constraint region, $R_{B2} = R_{A2}^c$ bounded by $-\overrightarrow{C\tilde{L}_1}, -\overrightarrow{C\tilde{L}_2}$ such that $G \in (R_1 \cup R_3) \subset R_{B2}$. Using Equation 8.1, these regions can be mapped to sets of possible homing directions.

Unfortunately, Lemma 8.1 does not always hold for Cases (2) and (3) such as in the example configurations in Figures 8.7(c, d) for Case (2) and (e, f) for Case (3). Regions R_C are marked with diagonal hatching. In Figure 8.7(c), the Type 1 constraint, R_{B1} should contain R_2. However, $R_C \subset R_2$ but $R_C \not\subset R_{B1}$. Likewise, in Figure 8.7(d), the Type 2 constraint, R_{B2} should contain $R_1 \cup R_3$ but it misses out on $R_C \subset R_1$. This means that if G lay in R_C, there would exist configurations of L_1, L_2, C where G would not lie in the constraint region.

Two approaches are possible - the first is a non-conservative approach that uses all constraints arising from all landmark pairs. This method will very probably converge and in all (thousands of) experiments (Section 8.4), was indeed observed to converge. A second approach uses only a subset of all the constraints and its convergence is theoretically guaranteed.

1. Non-conservative Method - Likely Convergence

As a result of the above problem, Theorem 8.1 will no longer hold. D_{res} was previously defined as the intersection of *all* sets of possible homing directions arising from all the landmark pairs. This may not exist, so we will instead define $\bar{D}_{res}$, which is the intersection of the *largest number* of such sets.

One observes that the size of R_C tends to be small relative to the total region in which G can lie. For example, in the case of $G \in R_2$ and R_{B1} should contain

R_2 but misses out the R_C regions (Figures 8.7 (c, e)), R_C is largest when C is at the highest point on the horopter and $\tilde{L_{1y}}, \tilde{L_{2y}}$ approach $\pm\infty$. Even then, R_C is small compared to all of region R_2.

Therefore, the probability of $G \in R_C$ is actually rather small. Assuming N randomly scattered landmarks, out of ${}^NC_2$ constraints, some proportion of constraints would give rise to these 'missed' regions, R_C, where there is potential for error. Of these, an even smaller number will actually be erroneous constraints, i.e. $G \in R_C$. These erroneous constraints map to a set of direction vectors that is 'wrong' in the sense that $\overrightarrow{CG}$ is not in this set.

The direction $\overrightarrow{CG}$ might not lie in $\bar{D}_{res}$. However, there is a good chance that $\overrightarrow{CG}$ lies close to it, and in fact, there is a very high probability that $\overrightarrow{CG}$ is less than 90° from a homing direction chosen from the set $\bar{D}_{res}$ (e.g. the average of all directions in $\bar{D}_{res}$). In the highly unlikely event of a wrong estimate, that is, $\overrightarrow{CG}$ being greater than 90° from the homing direction, the robot will make a step that takes it further away from the goal than it was at its last position.

However, this alone is insufficient to cause non-convergence. The robot will move to a new position, leading to a change in the directions of the landmark rays, and a new estimate of the new homing direction is found. In order for the robot to *not* converge to G, it would have to obtain so many wrong estimates that it was going in the wrong direction most of the time[2]. This requires a stacking of many improbable odds and indeed, in no experiment was non-convergence ever observed. Even so, there is a remote but finite chance of failure, which leads us to the next method.

2. Conservative Method - Guaranteed Convergence Assume the camera is mounted some distance above the ground and the x-y plane (we are working in the camera coordinate frame) is parallel to the ground. Homing motion is then restricted to this x-y plane. As before, landmark pair L_1, L_2 lies on the y-z plane and the horopter surface is the surface swept out by revolving the arc L_1CL_2 about the line L_1L_2, restricted to the half-space that has x-coordinates with the same sign as the sign of C_x.

There is a strategy for picking landmark pairs so that $\tilde{L_{1y}}, \tilde{L_{2y}}$ lie in the interval $[H_{1y}, H_{2y}]$. This is the Case 1 configuration which is free of erroneous constraints:

Lemma 8.2. If a landmark pair, L_1, L_2 is chosen such that one lies above the

[2]Here, the discussion centres on estimates of homing direction. Bear in mind that convergence is subject also to the robot step size, as discussed on page 136 and elsewhere in the chapter.

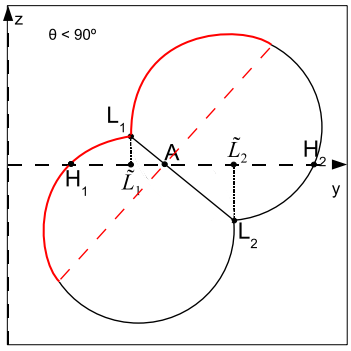


Figure 8.8: L_1 is closer to A than any other point on the red section of the horopter. L_2 is closer to A than any other point on the black section of the horopter.

x-y plane and one lies below it, and such that $\theta = cos^{-1}(\overrightarrow{CL_1} \cdot \overrightarrow{CL_2}) \leqslant 90°$, then $\tilde{L_{1y}}, \tilde{L_{2y}} \in [H_{1y}, H_{2y}]$.

Proof. The angle θ is always less than 180° (since it is measured by a dot product); it differs from the inter-landmark angle which can be greater or less than 180°. The intersection of the horopter surface with the y-z plane is as shown in Figures 8.7(g-i) which depict the three cases arising when the angle, θ, between two landmark rays as observed at any point C on the horopter, is greater than, equal to or less than 90°. With one landmark above and one below the x-y plane, the line segment lying between L_1 and L_2 intersects the y-axis at the point A. $\tilde{L_1}, \tilde{L_2}$ are the projections of L_1 and L_2 onto the x-y plane, whilst H_1 and H_2 are the intersections of the horopter with the plane.

Let P denote any point on the horopter and the Euclidean distance between A and P is $d(A, P)$. The intersection of the horopter surface with the y-z plane is shown in Figure 8.8 for the case of $\theta < 90°$. The dotted red line is a straight line constructed such that it goes through A and is perpendicular to line segment L_1L_2. It effectively separates the horopter curve shown into two parts (the red curve and the black curve).

It is easy to see from the diagram that $d(A, L_1) \leq d(A, P)$, for any point P on the red section of the horopter, including H_1. Likewise, $d(A, L_2) \leq d(A, P)$ for any point P on the black section of the horopter, including H_2. Therefore, the perpendicular projections of L_1 or L_2 onto the y-axis, that is, $\tilde{L_1}$ or $\tilde{L_2}$ will also be closer to A than H_1 or H_2. Hence, $\tilde{L_1}$ and $\tilde{L_2}$ will lie between H_1 and H_2 as stated in the lemma. The same can be shown for the case of $\theta = 90°$ whilst the counterexample of Figure 8.7(i) demonstrates that this argument does not hold

for $\theta > 90°$. □

Using only landmark pairs that have $\theta \leqslant 90°$ is a conservative method that ensures only correct constraints are used, in the noise-free case. However, most of the discarded constraints will in fact be correct, according to the earlier argument that the probability that $G \in R_C$ is low. (Note that if insufficient pairs of landmarks meet the conditions of Lemma 8.2, one can still use the earlier, non-conservative method to home.)

The conservative method ensures Theorem 8.1 holds and $\overrightarrow{CG} \in D_{res}$. Hence, if there are sufficient landmarks such that a vector $\mathbf{h}$ which is within 90° of all direction vectors in D_{res} can be found, then moving in the direction of $\mathbf{h}$ will bring the robot closer to the goal. If this condition is met at each step of the robot's estimate and move cycle, convergence on G is guaranteed as per Observation 8.1, subject to conditions on step size as discussed previously.

8.2.3 Robot and Landmarks in 3D

Now, C and G no longer lie on the x-y plane but we can still let L_1 and L_2 lie on the y-z plane without loss of generality. Also, let C lie in the region with negative x-coordinates. The intersection of the horopter surface with the y-z plane gives 2D horopter curves as shown in Figure 8.7(g-i). For the case of $\theta < \pi/2$ (Figure 8.7(g)), notice that a circle passing through L_1 and L_2 with the segment L_1L_2 as its diameter, is such that it always lies within the horopter curve. For $\theta = \pi/2$ (Figure 8.7(h)) this circle is coincident with the horopter curve.

The current position, C, is a point on the horopter surface. As illustrated in Figure 8.9, C and the circle described above define a cone which must be contained by the horopter in the region of negative x-coordinates, and this cone also extends into the region of positive x-coordinates. This cone, denoted as K, can be used to describe constraints on the location of G that are very similar to those earlier derived for the planar case. However, for $\theta > \pi/2$, the cone will not be contained by the horopter in the region of negative x-coordinates and we will discard constraints where $\theta > \pi/2$.

In the 3D case, let us denote the cone bounded by $\overrightarrow{CL_1}, \overrightarrow{CL_2}$ as K_1. The complement of K_1 is the region $J_1 = \mathbb{R}^3 \setminus K_1$ and corresponds to the Type 1 constraints, R_{B1}, in the planar case. Likewise, the cone bounded by $-\overrightarrow{CL_1}, -\overrightarrow{CL_2}$ is K_2, which has a complement $J_2 = \mathbb{R}^3 \setminus K_2$, corresponding to the Type 2 constraints, R_{B2}, of the planar case. By examining the size of the current angle, $\angle L_1CL_2$, relative to the goal angle, $\angle L_1GL_2$, one applies Lemma 8.1 except that

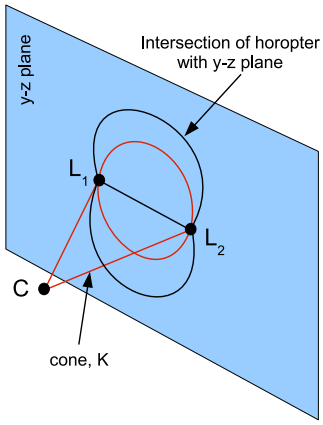


Figure 8.9: Constraining homing direction in 3D. For the case of $\theta < \pi/2$, the circle (in red) with L_1 and L_2 as points on its diameter is bounded by the horopter curve (in black). The circle and point C that lies on the 3D horopter surface defines an oblique circular cone. The cone is contained by the horopter surface (not shown).

now, regions J_1, J_2 are used instead of regions R_{B1}, R_{B2}. These regions are then mapped to a set of possible homing directions, D, according to Equation 8.1. This is repeated for all combinations of landmarks where $\theta \leqslant \pi/2$ to obtain sets of possible homing directions and the intersection of all these sets is D_{res}.

Unfortunately, because distances to the landmarks are not known, the cones K_1, K_2 *cannot* be found purely from the directions of the landmark vectors. However, one can approximate these cone regions as follows. Given the origin O and two *unit* landmark direction vectors $\overrightarrow{CL_1}, \overrightarrow{CL_2}$, let the points $\mathbf{a} = \overrightarrow{OC} + \overrightarrow{CL_1}$ and $\mathbf{b} = \overrightarrow{OC} + \overrightarrow{CL_2}$ define a circle which passes through points $\mathbf{a}$ and $\mathbf{b}$ and has diameter $|\mathbf{a} - \mathbf{b}|$. C together with this circle defines a cone, $\bar{K}_1$ that is an approximate guess of K_1. $\bar{K}_2$ is likewise defined. (If we knew the actual lengths of $\overrightarrow{CL_1}, \overrightarrow{CL_2}$, that is the distances to the landmarks, then we can find the cones K_1, K_2 exactly.)

Since $\bar{K}_1, \bar{K}_2$ are approximations, some of the regions where G may possibly lie in can be 'missed' by the corresponding approximate constraint regions, $\bar{J}_1, \bar{J}_2$. This is analogous to the 'missed' R_C regions in the earlier planar case. From $\bar{J}_1, \bar{J}_2$, we obtain sets of possible homing directions $\{D_1, D_2, \cdots, D_i, \cdots\}$, some of which are wrong in the sense that the $\overrightarrow{CG} \notin D_i$. As in the earlier non-conservative method for planar robots and 3D landmarks, we can find the intersection of the largest number of sets, $\bar{D}_{res}$. As before, it is very probable that $\overrightarrow{CG}$ lies within or close to the set of directions in $\bar{D}_{res}$ and that homing is successful. As the

experiments will later show, we do in fact observe successful convergence in every one of the experiments, including cases when landmark measurements contain large amounts of noise and outliers.

8.3 Algorithms and Implementation

8.3.1 Homing Direction

Algorithm 8.1 Planar Robot, 3D Landmarks

while $StopFlag == 0$ **do**
  for $j = 1$ to K **do**
    Select a pair L_1^j, L_2^j of landmarks with known correspondence.
    if (Conservative) and ((L_1^j, L_2^j are both above the plane or both below) or ($cos^{-1}(\overrightarrow{CL_1^j} \cdot \overrightarrow{CL_2^j}) > 90°$)) **then**
      continue to next j.
    end if
    $\alpha_j = |\angle L_1^j C L_2^j - \angle L_1^j G L_2^j|$.
    Find regions R_{B1}, R_{B2} as per Lemma 8.1 using projected landmark rays, $\overrightarrow{C\tilde{L}_1^j}, \overrightarrow{C\tilde{L}_2^j}$ and $\overrightarrow{G\tilde{L}_1^j}, \overrightarrow{G\tilde{L}_2^j}$
    Find the set of possible homing directions D_j and cast votes.
  end for
  Find bin(s) with highest votes (or with votes $> vote_{thres}$). Mean direction is homing vector, $\mathbf{h}$.
  Run Algorithm 8.3 and move robot by $StepSz * \mathbf{h}$.
end while

The homing approach proposed in Section 8.2 involves finding sets of possible homing directions and obtaining the intersection of all or of the largest number of these sets. A voting framework accomplishes this quickly and robustly. The table of votes divides the space of possible directions of movement into voting bins. For planar motion, voting is done in the range $\theta = [0, 360°)$. For 3D robot motion, we can divide the space using the azimuth-elevation convention (latitude-longitude) with azimuth angle, $\theta = [0, 360°)$ and elevation angle, $\phi = [-90°, 90°]$.

From the sets of possible homing directions $\{D_1, D_2, \cdots, D_i, \cdots\}$, votes for bins corresponding to the directions in each D_i are incremented, and the bin(s) with maximum vote (or with votes exceeding a threshold, $vote_{thres}$) gives D_{res} or

$\bar{D}_{res}$. When more than one bin has maximum vote, we take the average direction to be the homing vector, **h**.

The non-conservative and conservative approaches for homing with planar robots and 3D landmarks are summarized in Algorithm 8.1, whilst Algorithm 8.2 describes homing for 3D robots with 3D landmarks. In the algorithms, if $K = {}^{N}C_2$, then the algorithms will use all possible pairs arising from N landmarks, but if $K < {}^{N}C_2$, then landmark pairs are randomly sampled.

Note that Algorithms 8.1 and 8.2 are more efficiently implemented by using the directions mapped from regions $R_{A1}, R_{A2}, \bar{K}_1, \bar{K}_2$ instead of $R_{B1}, R_{B2}, \bar{J}_1, \bar{J}_2$, since the former regions are typically smaller than the latter. This way, less computations (incrementing of votes) occur. The homing vector, **h**, is then found from the *minimum* vote instead of the maximum.

In all three methods, we have assumed that there is some globally consistent way of measuring the angles according to a clockwise or anticlockwise convention. With planar robots, it is not unreasonable to assume the robot knows which way is 'up', that is, the normal to the ground plane. Clockwise and anticlockwise directions can then be defined according to this normal vector. In the case of robots in 3D however, the 'up' direction may be available from on board accelerometers. Alternatives include the solar compass, the normal to the horizon circle and such.

Algorithm 8.2 3D Robot, 3D Landmarks

while $StopFlag == 0$ **do**
  for $j = 1$ to K **do**
    Select a pair of landmarks L_1^j, L_2^j with known correspondence
    if $(cos^{-1}(\overrightarrow{CL_1^j} \cdot \overrightarrow{CL_2^j}) > \pi/2)$ **then**
      continue to next j.
    end if
    Angular error $\alpha_j = |\angle L_1^j C L_2^j - \angle L_1^j G L_2^j|$
    Find regions $\bar{J}_1, \bar{J}_2$ as per Lemma 8.1
    Find the set of possible homing directions D_j and cast votes
  end for
  Find bin(s) with highest votes (or votes $> vote_{thres}$).
  Take the unit mean direction as homing vector, **h**.
  Run Algorithm 8.3
  Move robot by $StepSz * \mathbf{h}$
end while

Algorithm 8.3 Calculate Stopping Criterion and Robot Step Size

Initialize $count_{in}, count_{out}, ave_{in}, ave_{out}, StopFlag = 0$
for $j = 1$ to K **do**
  if $\alpha_j < \alpha_{thres}$ **then**
    $ave_{in} = ave_{in} + \alpha_j$, increment $count_{in}$
  else
    $ave_{out} = ave_{out} + \alpha_j$, increment $count_{out}$
  end if
end for
$ave_{in} = ave_{in}/count_{in},\ ave_{out} = ave_{out}/count_{out}$
if $count_{in} > count_{out}$ **then**
  if $ave_{in} < ave_{thres}$ **then**
    $StopFlag = 1$
  else
    $StepSz = G_1 * ave_{in}$
  end if
else
  $StepSz = G_2 * ave_{out}$
end if
Return $StopFlag$ and $StepSz$

8.3.2 Homing Step Size

Whilst we have thus far focused on the homing *direction*, here we provide a simple method of controlling the size of the step that the robot should take in the homing direction. The homing step size should be smaller than $|\overrightarrow{CG}|$ as discussed in Section 8.2. However, $|\overrightarrow{CG}|$ is not known, and indeed, from purely 2D images, we cannot recover the *scale* of the 3D structure of the scene, therefore, we have no hope of estimating the scale of the step size. Here, we tune step size by hand such that its magnitude is appropriate for the scale of the environment in which the robot operates. Alternatively, prior knowledge of the size of certain objects viewed by the robot may be used to automatically determine a good step size to use.

The simple proportional control law demonstrated by Lines 14 and 17 in Algorithm 8.3 works well, but this can be replaced with more advanced control laws if necessary. In general, when one is far from the goal, large step sizes to rapidly close the robot-goal distance are desirable, and when one is close to the goal, smaller steps are better. To sense whether the robot is far from or close to the goal, we use the inter-landmark angular error, $\alpha_j = |\angle L_1^j C L_2^j - \angle L_1^j G L_2^j|$, as a measure of how similar the current and goal images are. Ideally, as the robot approaches the goal position, $\alpha_j \rightarrow 0, \forall j$. However, in the presence of noise and outlier data, this will not be so and a robust algorithm for dealing with this is discussed next.

Algorithm 8.3 partitions the angular error measurements into 'small' and 'large' sets where 'small' errors are less than some α_{thres} (a typical value of, say, 5°). The number of measurements in the large and small sets are $count_{out}$ and $count_{in}$ respectively, and the average angles of these sets are ave_{out} and ave_{in}. When the majority of angular error measurements are large ($count_{out} > count_{in}$) the robot is likely to be far from the goal, and conversely, if the majority are small ($count_{in} > count_{out}$), the robot is likely to be close to the goal.

If it is far from the goal, $count_{out} > count_{in}$, we use gain G_2 multiplied by the average angular error ave_{out} to regulate step size. Since ave_{out} is the average of the 'large' error set, it is typically in the order of a few radians and the resulting step size is correspondingly, fairly large. ave_{out} tends to drop as the robot closes in on the goal, leading to decreasing step sizes, until the robot is close enough so that $count_{in} > count_{out}$. We then switch to using ave_{in} multiplied by gain G_1 to regulate step size. Since ave_{in} is the average of the 'small' error set, it will be a small value (less than α_{thres} and typically a tenth or a hundredth of a radian)

leading to much finer steps.

Likewise, the criterion for stopping is met only when the majority of angular error measurements are small ($count_{in} > count_{out}$) and the average of these small errors are below some threshold, $ave_{in} < ave_{thres}$. Both step size and the stopping criterion can be filtered in time for greater stability.

8.4 Experiments and Results

Both simulations and real experiments were conducted to verify convergence and to investigate performance under noisy, outlier prone conditions (note that here, we are referring to the more practical measure of convergence discussed on page 136). The experimental method and results are as follows.

8.4.1 Simulations

Method: Matlab simulations investigated performance under Gaussian noise and outliers in the data measurements. In the simulations, landmarks were randomly scattered within a cube of $90 \times 90 \times 90$ units[3]. The robot moves between randomly generated start and goal positions by iterating the homing behaviour. The algorithms return a set of possible homing directions corresponding to all bins with maximum votes, and we calculated $\mathbf{h}$ as the average of this set.

Firstly, for the noise-free case, 1000 trials were conducted for each of the planar conservative, planar non-conservative and 3D homing cases. Secondly, a series of experiments involving 100 trials each, investigated homing under outliers and Gaussian noise. Outliers were simulated by randomly replacing landmark rays with random vectors. Gaussian noise was simulated by perturbing landmark rays in the current view with a noise vector such that the resulting ray has a mean centred on the original ray and the angle between original and perturbed ray falls off as a Gaussian distribution with standard deviation σ. We tested for up to 40% outliers and for Gaussian noise with standard deviation up to 9°.

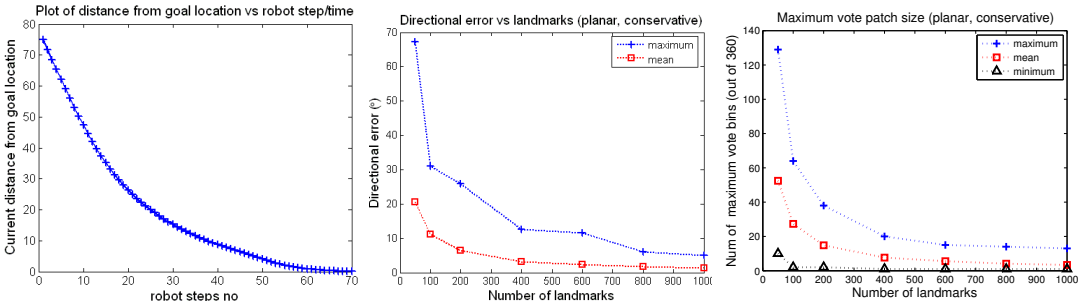
The effects of increasing the numbers of landmarks used, increasing outliers and increasing Gaussian noise were investigated in a series of experiments involving 100 trials each. For the planar robot experiments, the circle of possible

[3]The starting and goal positions were randomly generated and lay inside a cube of 70 units centred within the workspace. This was to ensure landmarks were scattered in all directions around the robot, which is a basic assumption that generally holds in real environments. What happens when this is not true and what to do in this case is further discussed in Section 8.5.

directions of movement was divided into 360 voting bins (1° per bin). For 3D robot motion, where directions can have azimuth and elevation angles ranging from $\theta = [0, 360°)$ and $\phi = [-90°, 90°]$, we divided θ into 180 bins and ϕ into 90 bins. As will be discussed in Section 8.5, convergence is still good with much coarser resolutions but the paths will not be as straight and direct.

In order to examine the quality of homing, we compute the *directional error* (angle between $\overrightarrow{CG}$ and $\mathbf{h}$), which measures how much the homing direction deviates from the straight line home. In Figures 8.10(b) and 8.12, directional error is averaged over 100 trials. For every trial, the robot takes on average about a hundred steps, where a homing estimate is made for each step. Therefore the number of estimates over which directional error is averaged, is in the order of 10^4.

Convergence demonstrated: With 1000 trials each, the three methods - planar conservative, planar non-conservative and 3D homing - *all* converged successfully. These noise-free trials brought the robot, on average, to within 0.17 units of the goal. This confirms the theoretical convergence proof for the planar conservative case. Furthermore, it gives empirical evidence supporting the statistical argument for likely convergence in the planar non-conservative and 3D cases. In all the other trials which follow, including those simulating outliers and noise, convergence was also observed for all tests.



(a) (b) (c)

Figure 8.10: (a) Monotonically decreasing distance from goal. (b) Directional error versus number of landmarks. (c) Size of maximum vote patch versus number of landmarks.

Efficient paths: A typical homing path is shown in Figure 8.4, where a planar robot makes its way from start (red plus) to goal (red circle) amid 100 landmarks (green pluses). Figure 8.10(a) plots the distance between the current position and the goal for each step the robot takes and this results in

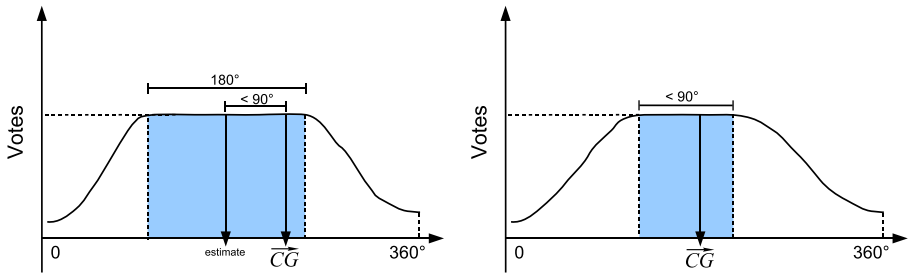
a monotonically decreasing graph that drops to zero. The distance decreases exponentially due to the non-uniform step size resulting from the proportional control law used to vary step size.

Number of Landmarks: Figure 8.10(b) shows the effects of increasing the number of landmarks on the directional error for the case of conservative, planar homing. The smaller the directional error is, the closer the robot's path is to the straight line path to the goal. The mean error drops to less than 5° with more than 300 points. Also shown is the maximum error, which is the maximum over all steps for every one of the 100 trials. Even with as few as 50 points, the homing vector, $\mathbf{h}$, is always less than 68° from $\overrightarrow{CG}$. Therefore, the robot always heads in a direction that brings it closer to the goal. Since this is so, convergence on the goal is certain.

Guaranteed convergence (planar, conservative): Figure 8.10(c) shows, for the same experiment, the size of the patch (connected set) of bins with maximum votes. In the case of 50 landmark points, these bins form a patch that is, at its largest, about 130° wide and $\overrightarrow{CG}$ must lie within this patch. The algorithm takes $\mathbf{h}$ to be the average of all directions with the maximum vote, which is roughly in the middle of this patch. Therefore, $\overrightarrow{CG}$ is less than 90° from $\mathbf{h}$. (In fact the same will hold for any patch that is less than 180° wide, refer Figure 8.11(a).) This is in accordance with the corresponding largest directional error observed for 50 points, which is about 65° (Figure 8.10(b)). This occurs when $\overrightarrow{CG}$ is just at the border of the 130° wide patch.

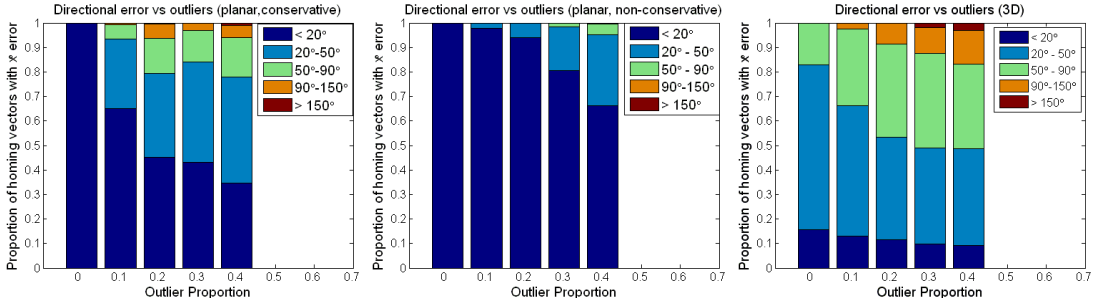
For the case of a 100 landmarks or more, an even stronger condition for convergence is met. As discussed earlier, as long as the maximum angle between two directions in D_{res} is less than 90°, *every* direction in D_{res} will be less than 90° from $\overrightarrow{CG}$ (illustrated in Figure 8.11(b)). From Figure 8.10(c), for 100 points or more, the size of the maximum vote patch (corresponding to the size of D_{res}) is always less than 90°, and the above statement is verified by observing that the maximum directional error (Figure 8.10(a)) is also always less than 90°. This implies both decreasing distance to the goal and successful convergence. The stricter condition is useful in practice - for example, if the selected $\mathbf{h}$ is blocked by an obstacle, every other direction in set D_{res} is a possible alternative that still guarantees taking the robot closer to home.

Outliers: Under increasing proportions of outliers, successful convergence was still observed for every trial. Figures 8.12(a-c) show performance under increasing probability of random outliers in the data. The histograms show the directional error from $\sim 10^4$ homing estimates from 100 trials. 100 landmarks



(a) (b)

Figure 8.11: Bounding the error or deviation from $\overrightarrow{CG}$. (a) If the interval of maximum votes is less than 180° wide, picking the middle of the interval results in an estimate that is always less than 90° from $\overrightarrow{CG}$, since $\overrightarrow{CG}$ must lie in that interval. (b) A stricter condition where the interval of maximum votes is less than 90° wide. Then, any direction picked from this interval gives an estimate that is less than 90° from $\overrightarrow{CG}$.



(a) (b) (c)

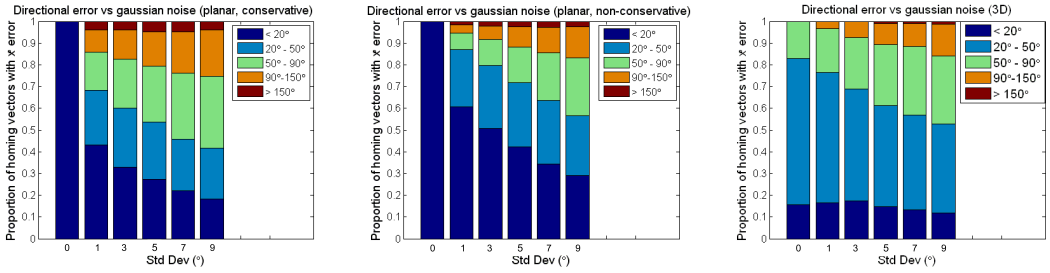
Figure 8.12: (a-c) Homing direction under outliers.

were used. For the case of no outliers, directional error was, as expected, always within 90° of $\overrightarrow{CG}$ for planar conservative and non-conservative homing as well as for 3D homing. Even with up to 40% of the landmark rays being complete nonsense, the number of homing estimates that were more than 90° from $\overrightarrow{CG}$ was insignificant for the planar non-conservative case (Figure 8.12(b)), less than 8% for the planar conservative case (Figure 8.12(a)), and less than 20% for the 3D case (Figure 8.12(c)).

Consequently, even under large outlier proportions, homing was still successful because the robot was heading in the correct direction the vast majority of the time. A plot of distance to the goal at each step however, will reveal that the graph is no longer strictly decreasing. Small bumps in the graph occur where the robot occasionally moved further from the goal than it was at the last position.

Note that we used 100 landmarks for the 3D case as well even though the space of possible homing directions is greatly increased. Using more landmarks would further improve the performance of the 3D case.

Gaussian noise: Successful convergence was observed in all trials involving increasing Gaussian noise. Figures 8.13(a-c) illustrate the performance as the Gaussian noise standard deviation is varied from 0° to 9°. Performance degrades gracefully with noise but the proportion of homing vectors with directional error greater than 90° is once again insufficient to prevent convergence. The non-conservative planar method once again performs better than the conservative planar method. Large Gaussian noise similarly causes the robot to oscillate about the final goal.



(a) (b) (c)

Figure 8.13: (a-c) Homing direction under Gaussian noise.

Planar conservative vs. non-conservative: In experiments under outliers and Gaussian noise, the planar non-conservative case outperformed the planar conservative case in the sense that the homing direction deviated less from $\overrightarrow{CG}$ (convergence was still observed in all the experiments conducted, but the paths are less direct if homing direction deviates more from $\overrightarrow{CG}$).

Although the conservative case is theoretically guaranteed to converge in the noise free case, that guarantee comes at the cost of throwing away all constraints that are not certain to be correct. In the process, many constraints which are perfectly fine are thrown away as well.

Conversely, voting was robust to the incorrect constraints arising in the non-conservative case. Since correct constraints dominate in number, these form a robust peak in the vote space which requires large numbers of incorrect constraints voting *in consistency* with each other, to perturb from its place. However, the incorrect constraints are quite random and do not generally vote to a consistent peak at all; and the 'missed' R_C regions also tend to be small, so the total effect on the performance of the non-conservative case is quite minimal.

8.4.2 Real experiments

Grid trials

A camera captured omnidirectional images of some environment at every 10 cm within a 1 m x 1 m grid. With some point on the grid as the home position, the homing direction was calculated at every other point on the grid. The result is summarized in a vector-field representation where each vector is the direction a robot would move in if it was at that point in the field. By taking each point on the grid in turn to be the home position, we can examine how the algorithm performs in that environment for all combinations of start and goal positions (marked with a '+') within the grid.

All experiments on the planar conservative and planar non-conservative algorithms were successful[4]. Figure 8.14 shows sample results from the three different environments. Landmarks were SIFT features [145] matched between current and goal images with code from [143].

The high-resolution (3840 x 1024) Ladybug camera [187] captured images of a room (Figure 8.14(a)) and an office cubicle (b). Figures 8.14(c-d) are results for the room using conservative and non-conservative algorithms respectively. Room images were taken with the camera 0.85 m above the ground. This is necessary for the conservative method, which requires landmarks both above and below the image horizon. (In contrast, cubicle images were taken with the camera lying on the floor and were unsuitable for the conservative algorithm since everything below the horizon is featureless ground.)

Both conservative and non-conservative methods homed successfully in Figures 8.14(c-d). Since the conservative method discards many constraints, the homing direction deviated further from $\overrightarrow{CG}$, compared to the non-conservative method which gave more direct homing paths. Nevertheless, it is clear that a robot placed anywhere within the grid will follow the vectors and eventually reach the goal position.

Figure 8.14(e) demonstrates successful homing in the cubicle environment using the non-conservative method. We also tested the algorithm on a low-resolution (360 x 143) outdoor image set supplied by authors of [276, 223]. Figure 8.14(f) is an example image from the set and 8.14(g) is a sample result. Interestingly, the low-resolution outdoor images gave smoother vector fields than the high-resolution, indoor ones. We believe this is due to a more even distribution of

[4]We leave real image experiments on 3D homing for future work.

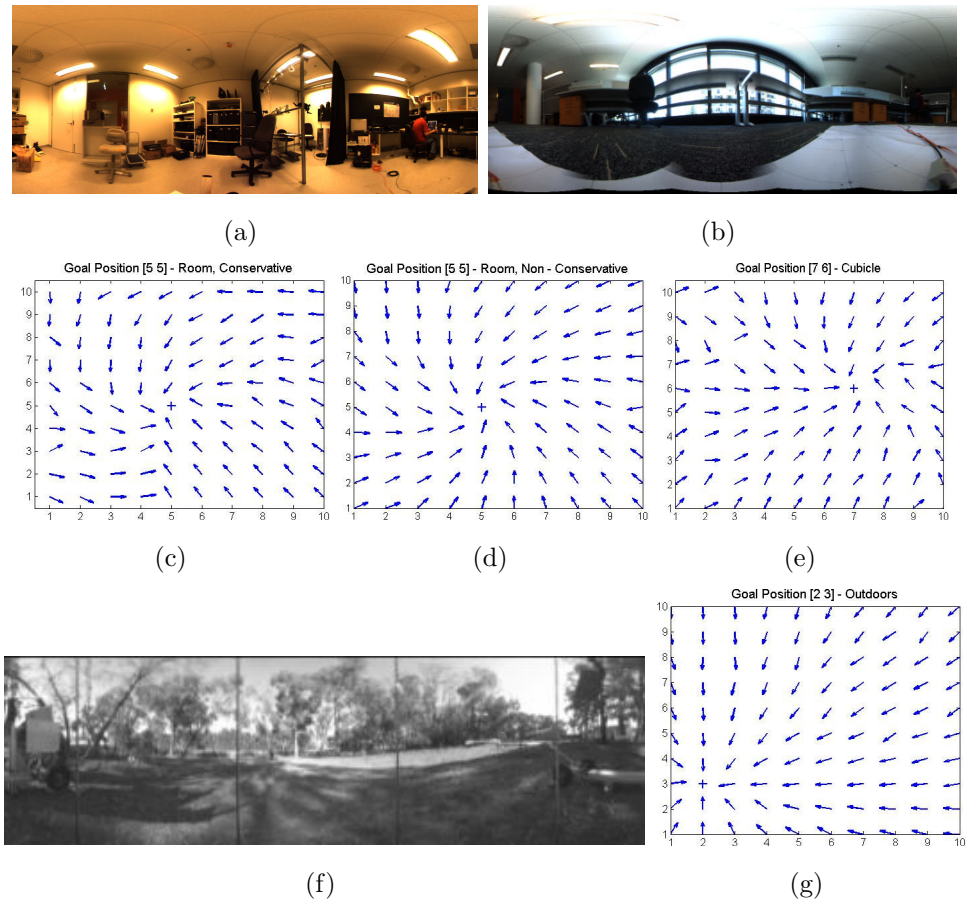
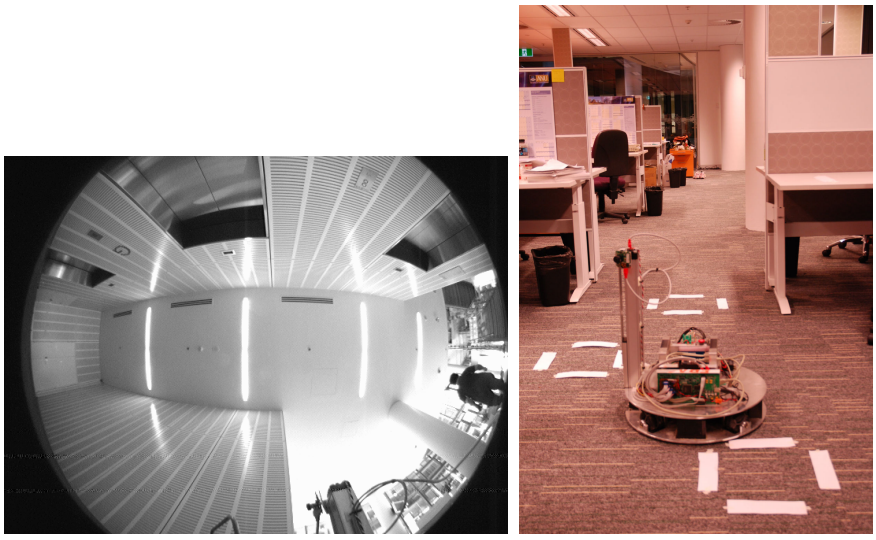


Figure 8.14: Grid trial experiments. (a) High-resolution panorama taken in a room with an elevated camera and (b) taken in a cubicle with camera on the floor. (c-d) Grid trials for room, comparing conservative and non-conservative methods. (e) Grid trial result for the cubicle. (f) Low-resolution panorama taken outdoors. (g) Grid trial result outdoors.

landmarks outdoors, whereas indoors, features tend to be denser in some areas of the image but are very sparse in other areas, leading to a bias in some directions. However, convergence is unaffected and homing paths remain fairly straight.

Robot trials



(a) (b)

Figure 8.15: Robot trial experiments. (a) Robot's view in the atrium experiment from the upward-pointing fish-eye camera. The rim of the image corresponds to the horizon of the world. (b) Robot in the office experiment.

A holonomic wheeled robot [42] homed in various environments including an office and a building atrium. The robot performed local homing multiple times to move along a chain of goal positions (simplest instance of the topological maps mentioned in Section 8.1) in order to get from one part of the environment to another. Images of the world were captured at each intermediate goal position and stored in the map. This was done using an Omnitech Robotics fish-eye lens camera [183], which was mounted pointing upwards, giving views such as Figure 8.15(a), where the image rim corresponds to the horizon.

The robot retrieves from memory the image corresponding to the next intermediate goal position that it needs to get to, compares that with its current view of the world, and uses the homing algorithms to drive there. After arriving, the next image is then retrieved from the map and the procedure is repeated until the final destination is reached.

This is illustrated in Figure 8.15(b), which shows an experiment where the intermediate goal positions are marked with white squares[5]. The robot performs long range homing by visiting these intermediate goal positions in turn until it arrives at its final destination.

In these experiments, homing was successful to within the order of a few centimeters, in the presence of mismatched landmarks, moving objects (leading to bad landmark points) and illumination changes. For supplementary videos, refer Appendix A.

8.5 Discussion

Robustness to outliers: Existing methods (see review in Section 3.2, page 52) often assume that a correct correspondence has been found between landmarks seen in the goal and current positions. However, if some of these landmarks have moved, are moving, were occluded, or were erroneously matched - i.e. outliers are present in the data - there is often no provision to ensure successful homing. In methods such as [131, 140, 23, 9], a series of vectors, $\{M_1, M_2 \cdots M_N\}$, are typically derived from landmark bearings by some rule, and the average direction is taken as the homing vector, $M_{res} = M_1 + M_2 + \cdots + M_N$. Unfortunately, this average is sensitive to outliers which can potentially cause the homing vector to point in any random direction. Conversely, the robust voting framework used in our methods were demonstrated in experiments to be very resistant to significant outlier and noise contamination.

Provable convergence. Homing is not provably successful in most of the cited local homing methods. The work of Argyros et al. [9] showed successful homing through extensive simulations. Loizou and Kumar [140] provided the first planar homing algorithm with a proof of convergence. Our work differentiates itself from Loizou and Kumar by proposing provably convergent algorithms for the more general case where the relative rotation between current and goal positions need not be known (compass-free), and where the landmarks need not lie on the same plane as the robot.

Are there sufficient landmarks? More landmarks lead to more direct homing paths. However, Observation 8.1 suggests that homing is successful as long as homing directions are consistently less than 90° from $\overrightarrow{CG}$. Computer

[5]The white squares were placed there so that the reader has an idea of where the intermediate goal positions are. They were not used for homing by the robot and indeed, the robot does not even see them since they are outside the FOV of the upward-pointing camera.

vision methods such as SIFT matching [145] can typically find and match thousands of feature points for some reasonable distance between the two views. A small subset (a few hundred) of these should be more than enough since we earlier saw that homing can work with even as few as 50 landmarks (Figure 8.10(b)).

Note that our method should work in the subcase of a robot being very close to a blank wall where approximately half the space contains no or little landmarks. This is because the goal position will lie in the half of the space that does contain landmarks (it could not lie inside the blank wall). The homing direction is sufficiently constrained to drive the robot in the general direction of the goal and hence away from the wall, and subsequently the situation improves as the robot moves further from the blank wall. In some circumstances however, there may be the degenerate case of very few landmarks such that motion is not adequately constrained. In such a case, we suggest falling back on methods such as Argyros et al. [9], which work with a minimum of 3 landmarks.

Speed: A C++ implementation of our algorithm runs in 35 msec (28 fps) on a Linux OS, Pentium4, 3GHz, 512Mb RAM machine ($\sim$200 landmarks). The bottleneck in the current system comes from the SIFT matching module which takes several seconds to compute. However, real-time SIFT implementations (for example, on a GPU [268]) do exist. Alternatively, faster feature tracking methods such as KLT are also possible.

Poor Image Resolutions: The trials with low-resolution (360 x 143 pixels) outdoor images demonstrate that the algorithm works well even with poor image resolutions. This is in agreement with the results of the simulations under increasing Gaussian noise. The image noise in most cameras is generally in the order of a couple of pixels (typically less than a degree), which is far less than the amounts of Gaussian noise used in the simulations (up to 9°). Therefore, what these large noise trials do in fact simulate is the degradation of accuracy caused by using very coarse image resolutions causing uncertainty in the landmark ray.

Voting Resolution: A large range of voting resolutions (not to be confused with image resolution) can be used. Lower voting resolutions could achieve even greater speeds without affecting convergence (it will, however, lead to a less direct path home). In general, reducing voting resolution by a small amount leads to little degradation in performance but an appreciable increase in processing speed, especially for the case of 3D homing (order N^2 increase in speed).

In fact, Observation 8.1 suggests that very coarse voting resolutions can be used without affecting convergence. As an extreme example, for homing on the plane, the conditions of Observation 8.1 are satisfied even with as few as 4 voting

bins (90° per bin), if **h** is the center of the bin and conservative voting was used. The homing vector is then at most 45° from $\overrightarrow{CG}$. Such a minimalist version of the method is well-suited for small, resource limited robots.

Although convergence is unaffected, overly coarse resolutions will lead to the robot taking a less than direct path, although that path would still be one where the distance between the goal and the current robot position is a decreasing function with each move that the robot makes.

Flexible Probability Maps: Whilst our experiments (and most existing methods) obtained a single homing vector, the voting table is in fact a weighted map of likely directions for homing. This leads to a flexible and natural framework for incorporating additional navigational constraints such as obstacle avoidance, kinematic constraints (in non-holonomic robots), centering behaviour and path optimality planning. For example, directions blocked by obstacles can be augmented with negative votes so that the robot is discouraged from moving in that direction.

8.6 Conclusion

In summary, we have investigated the usefulness of relaxed egomotion estimates. In particular, we considered relaxed translation estimates, which can be applied to the problem of visual homing. We introduced weak constraints on camera translation and proposed novel methods for robust homing that work in both planar and 3D robots. Theoretical convergence guarantees can be shown for the planar case (conservative method), whilst the planar non-conservative and 3D methods were shown by extensive simulation to converge empirically. The method is fast, flexible and performed well in real and simulated experiments.

Chapter 9

Conclusion

9.1 Summary of Findings

One of the central ideas investigated in this work was how large FOV vision may aid the task of egomotion estimation. To answer this question, we sought to investigate those geometrical features of large FOV cameras which are *not* found in small FOV devices. This, we hoped, would enable us to determine explicitly what may be achieved with large FOV vision that cannot be achieved (or is harder to achieve) with small FOV cameras.

We focused our attention on one such geometrical feature, that is, the antipodal point. We showed that the geometry of antipodal points may be exploited to decouple translation and rotation, leading to novel constraints on egomotion under both differential and discrete camera motions. For the case of differential motion, antipodal point constraints represent a new way of decoupling the camera translation and rotation. Furthermore, under the more general scenario of discrete camera motion, antipodal points lead to the only known method for decoupling motion and rendering a generally nonlinear or bilinear problem, linear.

Motion decoupling reduces the dimensionality of the problem, as it means that translation and rotation may be found independently of each other i.e. the 5-dimensional problem is broken up into two smaller-dimensional ones. This is important for the design of robust egomotion estimation methods, as it simplifies the problem and leads to improvements in accuracy and efficiency. Standard robust algorithms (e.g. RANSAC) based on the decoupled motion constraints showed improved performance, whilst methods that were not as attractive for solving higher dimensional problems (e.g. Hough-like voting) could now be used.

Experiments involving simulations and real images showed that the robust

algorithms based on both our differential and discrete antipodal point constraints performed competitively with the state-of-the-art. Overall, our algorithms showed better robustness to large outlier proportions. Furthermore, they demonstrated reduced run-time under increasing outlier proportions and those algorithms based on Hough-voting in particular, performed in constant-time.

These results are important because they demonstrate a lesser known advantage of large FOV vision. Whilst it was previously well-known that a large FOV improves the accuracy and stability of egomotion recovery methods (Section 3.1.4, page 43), here, we also demonstrated how large FOV vision may *simplify* the task through motion decoupling, resulting in more *efficient* algorithms for fast and robust egomotion recovery[1].

Secondary to our main contributions on egomotion estimation was our work on some applications that rely on knowledge of egomotion, such as the novel structure-from-motion algorithm discussed in Chapter 7. We also proposed novel algorithms for visual homing, and demonstrated that approximate or 'relaxed' egomotion measures can also be useful for certain applications, whilst being much easier and faster to compute.

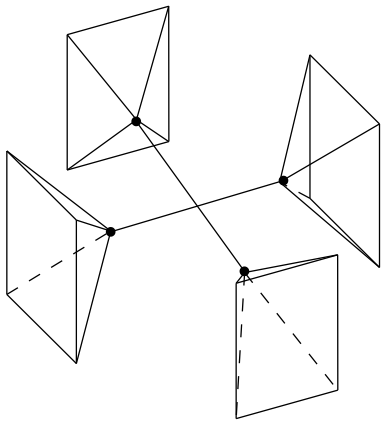
9.2 Future Work

9.2.1 Non-Central Cameras

The use of antipodal points requires a central or single viewpoint, large FOV camera. If the camera is not single viewpoint, then the results of this thesis are no longer applicable. An example of such a camera is shown in Figure 9.1, which depicts the schematic of a multi-camera rig which has four individual camera centres. This then, is motivation for extending the ideas of this thesis beyond single viewpoint cameras to the more general case of non-central camera systems.

A first step in this direction may perhaps be found in the work of Hu and Cheong [106], which formulates, under the differential motion assumption, equations for so-called 'matching rays' in a dual-camera rig (the construction of which, is essentially similar to Figure 9.1, except that it consists of two cameras only). However, in order to simplify the derived equations, Hu and Cheong then assumed that the separation between the two cameras in the rig was negligible.

[1] Another example of a larger FOV resulting in faster motion estimation may be found in the work of Hartley and Kahl [90], where the speed with which their branch-and-bound strategy arrived at the optimal solution is significantly influenced by the size of the camera FOV.



(a)

Figure 9.1: A non-central camera rig consisting of four small FOV cameras. Each pyramid represents the cone of rays projecting onto a separate camera centre (dot at the apex of the pyramid). The four camera centres are distinct, resulting in a non-single viewpoint camera system.

Their assumption effectively reduces the 'matching rays' of the rig into the antipodal points of a central camera. Hence, the problem remains unsolved for the non-central case.

With the equations of Hu and Cheong before the simplifying assumption was made, as a starting point, future work can perhaps explore the theoretical possibility of extending the ideas described in this thesis to non-central, large FOV camera systems. Also, further analysis of the error introduced by the assumption of a central camera when the camera is in fact only approximately central should be undertaken in order to better understand when such an approximation is valid and when it is not.

9.2.2 Multiple Moving Objects

In this work, we have focused on extracting the dominant motion in the scene, that is, the egomotion of the observer. Independently moving objects in the scene gave rise to image motion measurements that were discarded as outliers. However, in many instances, the motion of other moving objects in the scene are also of interest to the observer, which leads to the problem of estimating Multibody Structure and Motion (MSaM).

Some approaches to MSaM first recover the dominant self-motion (and our

algorithms may be used for this first stage) before recovering the motions of other objects in the scene [239, 205]. Other methods however, attempt to simultaneously segment and estimate all motion groups in a scene [254, 253, 134, 231, 195].

It would be interesting to investigate the possibility of extending the work of this thesis to the problem of estimating multiple motions. Whilst we know that a large FOV benefits self-motion recovery, we leave the question of whether large FOV vision can also benefit the estimation of multiple motions, to future work.

9.2.3 Implementation of a Real-time Egomotion Recovery System

This thesis focused on developing the theoretical basis for a number of novel egomotion estimation algorithms. Although implementations of these algorithms have been validated experimentally, going further to develop a practical, real-time egomotion recovery system is non-trivial.

The current form of the algorithms are by no means the most efficient or the fastest way of solving the task. Migrating from the predominantly Matlab implementations to programs written in C is one obvious way of speeding up the algorithms. We also advocate using parallel processing hardware (see the next section) to realize further speedups.

9.2.4 Parallel Algorithms

A recurring theme in this thesis was the use of 2D Hough-reminiscent voting in some of the algorithms presented. As noted previously, Hough-voting is highly suitable for implementation on parallel processors. In comparison, the standard solutions to the problem (e.g. 5-point and RANSAC) are fundamentally sequential in nature, and attempting to implement these on parallel processors would see only limited increases in speed.

As the computer industry undergoes a paradigm shift from increasing the number of transistors in a single processor chip to increasing the number of processors in a system, we believe that further work on parallel algorithms for egomotion recovery should be done in order to take advantage of the massively multi-core processors that are likely to become standard in the near future.

Parallel implementations of some of the algorithms described in this work, as well as of the optical flow or point correspondence stage, could potentially form the basis of a practical, real-time egomotion recovery system.

9.3 Conclusion

This thesis is a step toward faster and more robust egomotion recovery in complex, noisy and outlier-prone, real-world scenes - scenes which humans and animals appear to navigate and operate in, so effortlessly. In light of this, we were inspired by the characteristics of eye and brain design in biology, such as the large FOV vision possessed by many creatures, the spherical retina common to nearly all sighted animals, and the fundamentally parallel architecture of neural systems.

Thus inspired, we investigated the geometry of large FOV cameras and of image spheres, leading to the introduction of a number of constraints on egomotion. Algorithms that utilized these constraints were proposed, including several parallel, voting-based algorithms. The performance and advantages of these were then demonstrated through both real and simulated experiments.

We hope that the ideas proposed in this work will stimulate further research and progress in egomotion estimation, eventuating, some day, in stable and robust technologies that find ubiquitous applications in our cars, homes and workplaces.

Appendix A

Supplementary Videos

Supplementary videos are available from the CD included with this thesis. Alternatively, they are downloadable from the author's website as a single file:
http://users.rsise.anu.edu.au/∼johnlim/

The folder "videos for chapter 6 experiments" show the experiments on real image sequences for the lune+voting and GC+voting differential egomotion algorithms. The estimate errors in Figures 6.8 and 6.9 of Chapter 6 are for the experiments shown in these videos.

Meanwhile, the folder "videos for chapter 7 experiments" demonstrates the results for the real image experiments conducted on the discrete motion algorithms proposed in Chapter 7 (cross reference results of Figure 7.8).

Videos showing robot homing in an office environment and in a larger atrium environment using the algorithms proposed in Chapter 8 are found in the folder "videos for chapter 8 experiments".

Publications relevant to this thesis are also included for the reader's convenience.

Bibliography

[1] E. Adelson and J. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2:284–299, 1985.

[2] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:384–401, 1985.

[3] J. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images: A review. *Proceedings of the IEEE*, 76(8):917–935, August 1988.

[4] A. Agrawal and R. Chellappa. Ego-motion estimation and 3D model refinement in scenes with varying illumination. In *IEEE Workshop on Motion and Video Computing,MOTIONS '05*, volume 2, pages 140–146, 2005.

[5] J. Aisbett. An iterated estimation of the motion parameters of a rigid body from noisy displacement vectors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(11):1092–1098, 1990.

[6] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[7] A. Angeli, D. Filliat, S. Doncieux, and J. Meyer. A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, 2008.

[8] M. E. Antone and S. Teller. Automatic recovery of relative camera rotations for urban scenes. In *Conference on Computer Vision and Pattern Recognition*, 2000.

[9] A. Argyros, C. Bekris, S. Orphanoudakis, and L. Kavraki. Robot homing by exploiting panoramic vision. *Journal of Autonomous Robots*, 19(1):7–25, 2005.

[10] X. Armangue, H. Araujo, and J. Salvi. A review on egomotion by means of differential epipolar geometry applied to the movement of a mobile robot. *Pattern Recognition*, 36(12):2927–2944, December 2003.

[11] M. Atiquzzaman. Multiresolution Hough transform - an efficient method of detecting patterns in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1090–1095, 1992.

[12] A. Bab-Hadiashar and D. Suter. Robust optic flow computation. *International Journal of Computer Vision*, 29(1):59–77, 1998.

[13] P. Baker, A. S. Ogale, and C. Fermüller. The Argus eye: A new tool for robotics. *IEEE Robotics and Automation Magazine*, 11(4), 2004.

[14] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[15] D. Ballard and O. Kimball. Rigid body motion from depth and optical flow. *Computer Vision, Graphics and Image Processing*, 22:95–115, 1984.

[16] A. Barron and M. V. Srinivasan. Visual regulation of ground speed and headwind compensation in freely flying honey bees (Apis mellifera L.). *Journal of Experimental Biology*, 209:978–984, 2006.

[17] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[18] A. Basu and S. Licardie. Alternative models for fish-eye lenses. *Pattern Recognition Letters*, 16(4):433–441, 1995.

[19] A. Baumberg. Reliable feature matching across widely separated views. In *Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[20] L. Baumela, L. Agapito, I. Reid, and P. Bustos. Motion estimation using the differential epipolar equation. In *International Conf. on Pattern Recognition*, volume 3, pages 840–843, 2000.

[21] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[22] J. Bazin, C. Demonceaux, P. Vasseur, and I. Kweon. Motion estimation by decoupling rotation and translation in catadioptric vision. *Computer Vision and Image Understanding*, 2009.

[23] K. Bekris, A. Argyros, and L. Kavraki. Angle-based methods for mobile robot navigation: Reaching the entire plane. In *Proc. Int. Conf. on Robotics and Automation*, pages 2373–2378, 2004.

[24] R. Benosman and S. Kang (editors). *Panoramic Vision*. MIT Springer, 2001.

[25] R. Benosman, T. Maniere, and J. Devars. A generic camera model and calibration method for conventional, wide-angle and fish-eye lenses. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 161–165, 1996.

[26] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimisation via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1222–1239, 2001.

[27] J. Bresenham. Algorithm for computer control of a digital plotters. *IBM Systems Journal*, 4(1):25–30, 1965.

[28] T. Brodsky, C. Fermüller, and Y. Aloimonos. Directions of motion fields are hardly ever ambiguous. *International Journal of Computer Vision*, 26:5–24, 1998.

[29] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):90–99, January 1986.

[30] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision*, pages 25–36, 2004.

[31] A. R. Bruss and B. K. Horn. Passive navigation. *Computer Vision, Graphics and Image Processing*, 21:3–20, 1983.

[32] B. Buxton and H. Buxton. Computation of optic flow from the motion of edge features in image sequences. *Image and Vision Computing*, 2(2):59–75, 1984.

[33] T. Camus. Method for real time correlation of stereo images. In *US Patent 6516087*, 2003.

[34] L. Carrol. *Through the Looking-Glass, and What Alice Found There*. Macmillan, 1871.

[35] B. Cartwright and T. Collett. Landmark learning in bees. *Journal of Comparative Physiology A*, 151(4):521–543, 1983.

[36] J. Chahl and M. Srinivasan. Reflective surfaces for panoramic imaging. *Applied Optics*, 36(31):8275–8285, 1997.

[37] H. Chen and P. Meer. Robust regression with projection based M-estimators. In *IEEE Int. Conference on Computer Vision*, pages 878–885, 2003.

[38] A. Chiuso, R. Brockett, and S. Soatto. Optimal structure from motion: local ambiguities and global estimates. *International Journal on Computer Vision*, 39(3):195–228, 2000.

[39] W. Chojnacki, M. J. Brooks, A. van den Hengel, and D. Gawley. Revisiting Hartley's normalised eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(25):1172–1177, 2003.

[40] C. Chu and E. Delp. Estimating displacement vector from an image sequence. *Journal of the Optical Society of America A*, 6:871–878, 1989.

[41] C. Clifford, K. Langley, and D. Fleet. Centre-frequency adaptive IIR temporal filters for phase-based image velocity estimation. *International Conference on Image Processing and Applications*, pages 173–178, 1995.

[42] L. Cole and N. Barnes. Insect inspired three dimensional centering. In *Proc. Australasian Conf. on Robotics and Automation*, 2008.

[43] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[44] H. Coxeter. *Introduction to Geometry*. New York: Wiley, 1969.

[45] M. Cummins and P. Newman. Highly scalable appearance-only SLAM FAB-MAP 2.0. In *Robotics Science and Systems (RSS)*, Seattle, USA, June 2009.

[46] L. da Vinci. *The Notebooks of Leonardo da Vinci*. circa 1500 AD. (edited J.P. Richter, 1880).

[47] K. Daniilidis. Fixation simplifies 3D motion estimation. *Computer Vision and Image Understanding*, 68:158–169, 1997.

[48] K. Daniilidis and H.-H. Nagel. The coupling of rotation and translation in motion estimation of planar surfaces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 188–193, June 1993.

[49] K. Daniilidis and M. Spetsakis. *Visual navigation: from biological systems to unmanned ground vehicles*, chapter : Understanding noise sensitivity in structure from motion, pages 61–88. Hillsdale: Lawrence Erlbaum, 1997. (editor: Y. Aloimonos).

[50] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. International Conference on Computer Vision*, 2003.

[51] R. den Hollander and A. Hanjalic. A combined RANSAC-Hough transform algorithm for fundamental matrix estimation. In *Proc. 18th British Machine Vision Conference*, 2007.

[52] S. Derrien and K. Konolige. Approximating a single viewpoint in panoramic imaging devices. In *Proc. IEEE Workshop Omnidirectional Vision*, pages 85–90, 2000.

[53] Draganfly Innovations Inc. <http://www.draganfly.com>, 2009.

[54] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[55] H. Esch, S. Zhang, M. Srinivasan, and J. Tautz. Honeybee dances communicate distances measured by optic flow. *Nature*, 411:581–583, 2001.

[56] Euclid. *Optics*. circa 300 BC. (The Arabic version of Euclid's Optics, translated and edited by Elaheh Kheirandish, Springer, 1999).

[57] O. Faugeras, Q. Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001.

[58] O. Faugeras, F. Lustman, and G. Toscani. Motion and structure from point and line matches. In *Proc. International Conference on Computer Vision*, June 1987.

[59] O. Faugeras and S. Maybank. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246, 1990.

[60] C. Fermüller. Passive navigation as a pattern recognition problem. *International Journal of Computer Vision*, 14(2):147–158, 1995.

[61] C. Fermüller and Y. Aloimonos. The role of fixation in visual motion anaylsis. *International Journal of Computer Vision*, 11:165–186, 1993.

[62] C. Fermüller and Y. Aloimonos. Direct perception of three-dimensional motion from patterns of visual motion. *Science*, 270:1973–1976, 1995.

[63] C. Fermüller and Y. Aloimonos. Qualitative egomotion. *International Journal of Computer Vision*, 15:7–29, 1995.

[64] C. Fermüller and Y. Aloimonos. Ambiguity in structure from motion: Sphere versus plane. *International Journal of Computer Vision*, 28:137–154, 1998.

[65] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24:381–395, 1981.

[66] D. Fleet. *Measurement of Image Velocity*. Kluwer Academic Press, 1992.

[67] D. Fleet and A. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–104, 1990.

[68] M. Franz and H. Mallot. Biomimetic robot navigation. *Biomimetic Robots, Robotics and Autonomous Systems*, 30(1):133–153, 2000.

[69] M. Franz, B. Schölkopf, H. Mallot, and H. Bülthoff. Learning view graphs for robot navigation. *Autonomous Robots*, 5(1):111–125, 1998.

[70] M. Franz, B. Schölkopf, H. Mallot, and H. Bülthoff. Where did I take that snapshot? Scene-based homing by image matching. *Biological Cybernetics*, 79(3):191–202, 1998.

[71] F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *International Conference on Intelligent Robots and Systems*, 2007.

[72] A. Fusiello. Uncalibrated Euclidean reconstruction: A review. *Image and Vision Computing*, 18:555–563, 2000.

[73] C. Geyer and K. Daniilidis. Catadioptric camera calibration. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 398–404, 1999.

[74] C. Geyer and K. Daniilidis. Structure and motion from uncalibrated catadioptric views. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 279–286, Dec 2001.

[75] C. Geyer and K. Daniilidis. Paracatadioptric camera calibration. *Pattern Analysis and Machine Intelligence*, 24(5):687–695, May 2002.

[76] J. Gluckman and S. Nayar. Ego-motion and omnidirectional cameras. In *Proc. Int'l Conf. Computer Vision*, 1998.

[77] R. Goecke, A. Asthana, N. Pettersson, and L. Petersson. Visual vehicle egomotion estimation using the Fourier-Mellin transform. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 450–455, June 2007.

[78] T. Goedemé, T. Tuytelaars, and L. V. Gool. Visual topological map building in self-similar environments. In *Int. Conf. on Informatics in Control, Automation and Robotics*, 2006.

[79] N. Gonçalves and H. Araújo. Estimating parameters of noncentral catadioptric systems using bundle adjustment. *Computer Vision and Image Understanding*, 113(1):11–28, 2009.

[80] A. Goshtasby, S. Gage, and J. Bartholic. A two-stage cross correlation approach to template matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6(3):374–378, 1984.

[81] S. Gourichon, J. Meyer, and P. Pirim. Using colored snapshots for short range guidance in mobile robots. *Int. J. Robot. Automation*, 17:154–162, 2002.

[82] N. Grzywacz and A. Yuille. A model for the estimate of local velocity. In *Proc. European Conf. on Computer Vision*, pages 331–335, 1990.

[83] N. Gupta and L. Kanal. Gradient based image motion estimation without computing gradients. *Int. J. Comput. Vision*, 22(1):81–101, 1997.

[84] F. Hampel. Beyond location parameters: robust concepts and methods. *Bulletin of the Int. Statist. Inst.*, 46:375–382, 1975.

[85] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.

[86] R. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proc. European Conference on Computer Vision*, 1992.

[87] R. Hartley. In defence of the 8-point algorithm. In *Proceedings of the 5th International Conference on Computer Vision*, pages 1064–1070, 1995.

[88] R. Hartley, E. Hayman, L. de Agapito, and I. Reid. Camera calibration and the search for infinity. In *Proceedings of the IEEE International Conference on Computer Vision*, 1999.

[89] R. Hartley and F. Kahl. Global optimization through searching rotation space and optimal estimation of the essential matrix. In *Proc. International Conference on Computer Vision*, 2007.

[90] R. Hartley and F. Kahl. Global optimization through rotation space search. *International Journal of Computer Vision*, 82(1):64–79, 2009.

[91] R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 504–509, 2004.

[92] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[93] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[94] D. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1:279–302, 1988.

[95] J. Heel. Direct estimation of structure and motion for multiple frames. In *Technical Report 1190, MIT AI Lab*, 1989.

[96] J. Heel. Direct dynamic motion vision. In *Proc. IEEE Conference on Robotics and Automation*, 1990.

[97] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[98] H. Helmholtz. *Handbuch der Physiologischen Optik*. Leipzig: Voss, 1867. (Treatise on Physiological Optics, translated by J.P.C. Southall, New York: Dover, 1962).

[99] E. Hildreth. The computation of the velocity field. *Proc. of the Royal Society of London B*, 221:189–220, 1984.

[100] E. Hildreth. Recovering heading for visually-guided navigation. *Vision Research*, 32(6):1177–1192, 1992.

[101] J. Hong, X. Tan, B. Pinette, R. Weiss, and E. Riseman. Image-based homing. *IEEE Control Systems Magazine*, 12:38–45, 1992.

[102] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[103] B. Horn and E. Weldon. Direct methods for recovering motion. *Int. Journal on Computer Vision*, 2(1):51–76, 1988.

[104] P. Hough. Method and means for recognizing complex patterns. *US Patent 3,069,654*, December 1962.

[105] C. Hu and L. Cheong. Linear ego-motion recovery algorithm based on quasi-parallax. In *International Conference on Image Processing*, pages 233–236, 2008.

[106] C. Hu and L. Cheong. Linear quasi-parallax SFM using laterally-placed eyes. *International Journal on Computer Vision*, 84(1):21–39, 2009.

[107] P. Huber. *Robust Statistics*. Wiley, 2004.

[108] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. Robotics and Automation*, 12(5):651–670, 1996.

[109] J. Illingworth and J. V. Kittler. The adaptive Hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):690–698, 1987.

[110] J. Illingworth and J. V. Kittler. A survey of the Hough transform. *Computer Vision, Graphics and Image Processing*, 44(1):87–116, 1988.

[111] Intel Open Source Computer Vision Library. available from: <http://sourceforge.net/projects/opencv>, 2007.

[112] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using region alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):268–272, 1997.

[113] B. Jahne. Motion determination in space-time images. In *Proc. European Conf. on Computer Vision*, pages 161–173, 1990.

[114] A. Jepson and D. Heeger. Subspace methods for recovering rigid motion I: algorithm and implementation. *International Journal of Computer Vision*, 7(2), 1992.

[115] A. Jepson and D. Heeger. Linear subspace methods for recovering translation direction. In *Spatial Vision in Humans and Robots*. Cambridge University Press, 1993.

[116] H. Jin, P. Favaro, and S. Soatto. Real-time 3-D motion and structure from point features: a front end system for vision-based control and interaction. In *Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 778–779, 2000.

[117] F. Kahl. Multiple view geometry and the L_∞-norm. In *Proc. International Conference on Computer Vision*, pages 1002–1009, 2005.

[118] K. Kanatani. 3-D interpretation of optical flow by renormalization. *International Journal of Computer Vision*, 11(3), 1992.

[119] K. Kanatani. Comparing optimal 3-D reconstruction for finite motion and optical flow. *Journal of Electronic Imaging*, 12(3):478–488, 2003.

[120] K. Kanatani, Y. Shimizu, N. Ohta, M. Brooks, W. Chojnacki, and A. van den Hengel. Fundamental matrix from optical flow: optimal computation and reliability evaluation. *Journal of Electronic Imaging*, 9(2):194–202, April 2000.

[121] S. Kang. Catadioptric self-calibration. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 201–207, 2000.

[122] J. Kannala and S. Brandt. A generic camera model and calibration method for conventional, wide-angle and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1335–1340, 2006.

[123] S. Karabernou and F. Terranti. Real-time FPGA implementation of Hough transform using gradient and CORDIC algorithm. *Image and Vision Computing*, 23(11):1009–1017, 2005.

[124] J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–396, 1984.

[125] J. Koenderink and A. van Doorn. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica Acta*, 22(9):773–791, 1975.

[126] J. Koenderink and A. van Doorn. Local structure of movement parallax of the plane. *Jour. of the Optical Society of America*, 66:717–723, 1976.

[127] P. D. Kovesi. MATLAB and Octave Functions for Computer Vision and Image Processing, School of Computer Science & Software Engineering, The University of Western Australia, available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.

[128] B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, 1991.

[129] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to the 5-pt and 6-pt relative pose problems. In *Proc. British Machine Vision Conference (BMVC)*, 2008.

[130] R. Kumar, A. Tirumalai, and R. C. Jain. A non-linear optimization algorithm for the estimation of structure and motion parameters. In *Proc.*

IEEE Conf. on Computer Vision and Pattern Recognition, pages 136–143, June 1989.

[131] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Biomimetic Robots, Robotics and Autonomous Systems*, 30:39–64, 2000.

[132] D. Lawton and W. Carter. Qualitative spatial understanding and the control of mobile robots. In *IEEE Conf. on Decision and Control*, pages 1493–1497, 1990.

[133] J. Lee, B. Wood, and T. Newman. Very fast ellipse detection using GPU-based RHT. In *International Conference on Pattern Recognition*, 2008.

[134] H. Li. Two-view motion segmentation from linear programming relaxation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2007.

[135] H. Li. Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *Proc. International Conf. Computer Vision*, 2009.

[136] H. Li and R. Hartley. Five-point motion estimation made easy. In *18th International Conference on Pattern Recognition (ICPR'06)*, pages 630–633, 2006.

[137] Z. Li, B. Yao, and F. Tong. A linear generalized Hough transform and its parallel implementation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1991.

[138] W. Lin, G. Tan, and L. Cheong. When discrete meets differential- assessing the stability of structure from small motion. *International Journal of Computer Vision*, June 2009.

[139] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):224–270, 1994.

[140] S. Loizou and V. Kumar. Biologically inspired bearing-only navigation and tracking. In *IEEE Conf. on Decision and Control*, 2007.

[141] H. Longuet-Higgins. A computer algorithm for reconstruction of a scene from two projections. *Nature*, 293:133–135, 1981.

[142] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 208(1173):385–397, 1980.

[143] D. Lowe. Sift implementation available from (accessed 2009): <http://www.cs.ubc.ca/∼lowe/keypoints/>.

[144] D. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999.

[145] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[146] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[147] Q.-T. Luong, R. Deriche, O. Faugeras, and T. Papadopoulo. On determining the fundamental matrix: Analysis of different methods and experimental results. *Technical Report RR-1894, INRIA*, 1993.

[148] Y. Ma, J. Kosecká, and S. Sastry. Motion recovery from image sequences: Discrete viewpoint vs. differential viewpoint. In *European Conference on Computer Vision*, 1998.

[149] Y. Ma, J. Kosecka, and S. Sastry. Linear differential algorithm for motion recovery: a geometric approach. *International Journal on Computer Vision*, 36(1), 2000.

[150] W. Maclean, A. Jepson, and R. Frecker. Recovery of egomotion and segmentation of independent object motion using the EM algorithm. In *British Machine Vision Conference*, pages 13–16, 1994.

[151] M. J. Magee and J. K. Aggarwal. Determining vanishing points from perspective images. *Computer Vision, Graphics and Image Processing*, 26:256–267, 1984.

[152] M. Mainberger, A. Bruhn, and J. Weickert. Is dense optic flow useful to compute the fundamental matrix? In *Proc. of the 5th Int. Conf. on Image Analysis and Recognition*, pages 630–639, 2008.

[153] A. Makadia and K. Daniilidis. Rotation estimation from spherical images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1170–1175, 2006.

[154] A. Makadia, C. Geyer, S. Sastry, and K. Daniilidis. Radon-based structure from motion without correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2005.

[155] J. Matas and O. Chum. Randomized RANSAC with $t_{d,d}$ test. *Image and Vision Computing*, 22(10):837–842, 2004.

[156] J. Matas and O. Chum. Optimal randomized RANSAC. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1472–1482, 2008.

[157] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, pages 384–393, 2002.

[158] L. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithm for estimating depth from image sequences. In *Proc. Conference on Computer Vision and Pattern Recognition*, June 1988.

[159] S. Maybank. The angular velocity associated with the optical flow field arising from motion through a rigid environment. *Proceedings of the Royal Society of London*, 410:317–326, 1985.

[160] S. J. Maybank and O. Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8:123–151, 1992.

[161] B. Micusik and T. Pajdla. Estimation of omnidirectional camera model from epipolar geometry. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 485–490, 2003.

[162] B. Micusik and T. Pajdla. Autocalibration and 3D reconstruction with non-central catadioptric cameras. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[163] B. Micusik and T. Pajdla. Structure from motion with wide circular field of view cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1135–1149, 2006.

[164] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142, 2002.

[165] R. Möller. Insect visual homing strategies in a robot with analog processing. *Biological Cybernetics: special issue in Navigation in Biological and Artificial Systems*, 83(3):231–243, 2000.

[166] H. Moravec. Rover visual obstacle avoidance. In *International Joint Conference on Artificial Intelligence*, pages 785–789, 1981.

[167] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real time structure from motion. In *Proceedings of the 18th British Machine Vision Conference*, 2007.

[168] M. Muhlich and R. Mester. The role of total least squares in motion analysis. In *Proc. European Conf. Computer Vision*, 1998.

[169] D. Murray. Recovering range using virtual multicamera stereo. *Computer Vision and Image Understanding*, 61(2):285–291, 1995.

[170] H.-H. Nagel. On the estimation of optic flow: Relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324, 1987.

[171] S. Nayar. Catadioptric omnidirectional camera. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 482–488, 1997.

[172] S. Negahdaripour and B. Horn. Direct passive navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):168–176, 1987.

[173] S. Negahdaripour and B. Horn. A direct method for locating the focus of expansion. *Computer Vision, Graphics and Image Processing*, 46:303–326, 1989.

[174] S. Negahdaripour and C. H. Yu. A generalized brightness change model for computing optical flow. In *Proc. Int'l Conf. Computer Vision*, pages 2–11, 1993.

[175] R. Nelson and J. Aloimonos. Finding motion parameters from spherical flow fields (or the advantages of having eyes in the back of your head). *Biological Cybernetics*, 58:261–273, 1988.

[176] J. Neumann. *Computer Vision in the Space of Light Rays: Plenoptic Video Geometry and Polydioptric Camera Design*. 2003. PhD Thesis.

[177] J. Neumann, C. Fermüller, and Y. Aloimonos. Eyes from eyes: New cameras for structure from motion. In *IEEE Workshop on Omnidirectional Vision 2002*, pages 19–26, June 2002.

[178] J. Neumann, C. Fermüller, and Y. Aloimonos. Polydioptric camera design and 3D motion estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 294–301, 2003.

[179] D. Nistér. An efficient solution to the five-point relative pose problem. In *Proc. Conference on Computer Vision and Pattern Recognition*, 2003.

[180] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *Proc. IEEE International Conference on Computer Vision (ICCV 2003)*, pages 199–206, April 2003.

[181] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 756–770, 2004.

[182] M. Ogata and T. Sato. Motion detection model with two-stages: spatiotemporal filtering and feature matching. *Journal of the Optical Society of America A*, 9:377–387, 1992.

[183] Omnitech Robotics. <http://www.omnitech.com/>, 2009.

[184] T. Pajdla, T. Svoboda, and V. Hlaváč. *Panoramic Vision*, chapter : Epipolar Geometry of Central Panoramic Catadioptric Cameras, pages 73–102. Springer, 2001. (editors: R. Benosman and S.B. Kang).

[185] J. Patel and A. N. Choudhary. *Parallel Architectures and Parallel Algorithms for Integrated Vision Systems*. Kluwer Academic Pub, 1990.

[186] R. Pless. Using many cameras as one. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[187] Point Grey Research. <http://www.ptgrey.com>, 2007.

[188] M. Pollefeys, R. Koch, and L. V. Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters.

In *Proceedings of the IEEE International Conference on Computer Vision*, pages 90–95, 1998.

[189] I. Powell. Panoramic fish-eye imaging system. In *US Patent 5631778*, 1997.

[190] K. Prazdny. Egomotion and relative depth map from optical flow. *Biological Cybernetics*, 36:87–102, 1980.

[191] K. Prazdny. On the information in optical flows. *Computer Graphics and Image Processing*, 22:239–259, 1983.

[192] J. Prince and E. McVeigh. Motion estimation from tagged MR image sequences. *IEEE Trans. on Medical Imaging*, 11:238–249, 1992.

[193] M. Pupilli and A. Calway. Real-time camera tracking using a particle filter. In *Proc. British Machine Vision Conference (BMVC'05)*, pages 519–528, September 2005.

[194] G. Qian and R. Chellappa. Structure from motion using sequential Monte Carlo methods. In *Proceedings of Eighth IEEE International Conference on Computer Vision*, pages 614–621, July 2001.

[195] N. Quadrianto, T. Caetano, J. Lim, and D. Schuurmans. Convex relaxation of mixture regression with efficient algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[196] L. Quan and R. Mohr. Determining perspective structures using hierarchical Hough transform. *Pattern Recognition Letters*, 9(4):279–286, 1989.

[197] R. Raguram, J. M. Frahm, and M. Pollefeys. A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision*, pages 500–513, 2008.

[198] S. Ramalingam, P. Sturm, and S. Lodha. Towards complete generic camera calibration. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1093–1098, 2005.

[199] J. Rieger and D. Lawton. Processing differential image motion. *Jour. of the Optical Society of America A*, 2(2):354–359, 1985.

[200] J. W. Roach and J. K. Aggarwal. Determining the movement of objects from a sequence of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):55–62, 1980.

[201] C. Rother and S. Carlsson. Linear multi-view reconstruction and camera recovery. In *Proc. International Conf. Computer Vision*, pages 42–50, 2001.

[202] P. Rousseeuw. Least median of squares regression. *Jour. Amer. Statist. Assoc*, 79:871–880, 1984.

[203] J. Salvi, X. Armangue, and J. Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, 35(7):1617–1635, July 2002.

[204] R. Satzoda, S. Suchitra, and T. Srikanthan. Parallelizing the Hough transform computation. *Signal Processing Letters*, 15:297–300, 2008.

[205] K. Schindler and D. Suter. Two-view multibody structure-and-motion with outliers through model selection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(6):983–995, 2006.

[206] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5):530–534, 1997.

[207] B. Schunck. Image flow: Fundamentals and future research. In *Proc. Int. Conf. on Pattern Recognition*, pages 560–571, 1985.

[208] O. Shakernia, R. Vidal, and S. Sastry. Omnidirectional vision-based egomotion estimation from backprojection flows. In *Proc. IEEE Workshop Omnidirectional Vision*, 2003.

[209] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[210] J. Shufelt. Performance evaluation and analysis of vanishing point detection techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3):282–288, 1999.

[211] C. Silva and J. Santos-Victor. Direct egomotion estimation. In *Proc. 13th International Conference on Pattern Recognition*, volume 1, pages 702–706, 1996.

[212] C. Silva and J. Santos-Victor. Robust egomotion estimation from the normal flow using search subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1026–1034, 1997.

[213] E. Simoncelli and E. Adelson. Computing optical flow distributions using spatio-temporal filters. Technical report, MIT Media Lab Vision and Modeling, Technical Report, 1991.

[214] P. Sobey and M. V. Srinivasan. Measurement of optical flow by a generalized gradient scheme. *Journal of the Optical Society of America A*, 8:1488–1498, 1991.

[215] M. Srinivasan. Generalized gradient schemes for the measurement of two-dimensional image motion. *Biological Cybernetics*, 63(6):421–431, 1990.

[216] M. Srinivasan, J. Chahl, K. Weber, S. Venkatesh, M. Nagle, and S. Zhang. Robot navigation inspired by principles of insect vision. *Robotics and Autonomous Systems*, 26:203–216, 1999.

[217] M. Srinivasan, M. Lehrer, W. Kirchner, and S. Zhang. Range perception through apparent image speed in freely-flying honeybees. *Visual Neuroscience*, 6:519–535, 1991.

[218] M. Srinivasan, S. Zhang, M. Lehrer, and T. Collett. Honeybee navigation enroute to the goal: visual flight control and odometry. *Journal of Experimental Biology*, 199:237–244, 1996.

[219] G. Stein, O. Mano, and A. Shashua. A robust method for computing vehicle ego-motion. In *IEEE Intelligent Vehicles Symposium (IV2000)*, October 2000.

[220] G. P. Stein and A. Shashua. Model-based brightness constraints: On direct estimation of structure and motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):992–1015, 2000.

[221] H. Stewénius, C. Engels, and D. Nistér. An efficient minimal solution for infinitesimal camera motion. In *Conference on Computer Vision and Pattern Recognition*, 2007.

[222] P. Sturm and S. Ramalingam. A generic camera calibration concept. In *European Conf. on Computer Vision (ECCV)*, 2004.

[223] W. Stürzl and J. Zeil. Depth, contrast and view-based homing in outdoor scenes. *Biological Cybernetics*, 96:519–531, 2007.

[224] J. Suhr, H. Jung, K. Bae, and J. Kim. Outlier rejection for cameras on intelligent vehicles. *Pattern Recognition Letters*, 29(6):828–840, April 2008.

[225] M. Sutton, W. Wolters, W. Peters, and S. McNiell. Determination of displacements using an improved digital correlation method. *Image and Vision Computing*, 1(3):133–139, 1983.

[226] T. Svoboda and T. Pajdla. Epipolar geometry for central catadioptric cameras. *International Journal of Computer Vision*, 49(1):23–37, 2002.

[227] T. Svoboda, T. Pajdla, and V. Hlávac. Motion estimation using central panoramic cameras. In *IEEE International Conf. on Intelligent Vehicles*, pages 335–340, June 1998.

[228] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using non-linear least squares. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 752–753, June 1993.

[229] M. Taalebinezhaad. Direct recovery of motion and shape in the general case by fixation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14:847–853, 1992.

[230] S. Tagzout, K. Achourb, and O. Djekouneb. Hough transform algorithm for FPGA implementation. *Signal Processing*, 81(6):1295–1301, 2001.

[231] N. Thakoor and J. Gao. Branch-and-bound hypothesis selection for two-view multiple structure and motion segmentation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2008.

[232] T. Thanh, H. Nagahara, R. Sagawa, Y. Mukaigawa, M. Yachida, and Y. Yagi. Robust and real-time rotation estimation of compound omnidirectional sensor. In *Proc. IEEE International Conference on Robotics and Automation*, April 2007.

[233] I. Thomas and E. Simoncelli. Linear structure from motion. Technical report, IRCS, University of Pennsylvania, 1994.

[234] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[235] T. Tian, C. Tomasi, and D. Heeger. Comparison of approaches to egomotion computation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 315–320, 1996.

[236] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *Conference on Computer Vision and Pattern Recognition*, 2008.

[237] C. Tomasi and J. Shi. Direction of heading from image deformations. In *Conference on Computer Vision and Pattern Recognition*, pages 422–427, 1993.

[238] A. Torii and A. Imiya. The randomized-Hough-transform-based method for great-circle detection on a sphere. *Pattern Recognition Letters*, 28(10):1186–1192, 2007.

[239] P. Torr. Geometric motion segmentation and model selection. *Philosophical Trans. of the Royal Society of London*, 356(1740):1321–1340, 1998.

[240] P. Torr and A. Zisserman. Robust computation and parametrization of multiple view relations. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 727–732, 1998.

[241] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Journal of Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[242] B. Triggs. Autocalibration and the absolute quadric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 609–614, 1997.

[243] B. Triggs. Differential matching constraints. In *Proc. International Conf. Computer Vision*, 1999.

[244] B. Triggs. Plane + parallax, tensors and factorization. In *Proc. European Conf. Computer Vision*, pages 522–538, 2000.

[245] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms '99*, pages 298–372, 1999.

[246] R. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[247] R. Tsai and T. Huang. Uniqueness and estimation of 3-D motion parameters of rigid bodies with curved surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:13–27, 1984.

[248] T. Tuytelaars and L. V. Gool. Wide baseline stereo based on local, affinely invariant regions. In *British Machine Vision Conference*, pages 412–422, 2000.

[249] O. Tuzel, R. Subbarao, and P. Meer. Simultaneous multiple 3D motion estimation via mode finding on Lie groups. In *10th IEEE Int. Conference on Computer Vision*, pages 18–25, 2005.

[250] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London*, pages 405–426, 1979.

[251] A. Vardy and R. Möller. Biologically plausible visual homing methods based on optical flow techniques. *Connection Science*, 17(1):47–89, 2005.

[252] A. Verri and T. Poggio. Motion field and optical flow: Qualitative properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):490–498, 1989.

[253] R. Vidal, Y. Ma, S. Soatto, and S. Sastry. Two-view multibody structure from motion. *International Journal on Computer Vision (IJCV)*, 68(1):7–25, 2006.

[254] R. Vidal and S. Sastry. Optimal segmentation of dynamic scenes from two perspective views. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2003.

[255] T. Viéville and O. D. Faugeras. The first order expansion of motion equations in the uncalibrated case. *Computer Vision and Image Understanding*, 64(1):128–146, 1996.

[256] S. Šegvic, G. Schweighofer, and A. Pinz. Influence of numerical conditioning on the accuracy of relative orientation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[257] N. Wade and S. Finger. The eye as an optical instrument: from camera obscura to Helmholtz's perspective. *Perception*, 30:1157–1177, 2001.

[258] A. Watson and A. Ahumada. Model of human visual motion sensing. *Journal of the Optical Society of America A*, 2:322–341, 1985.

[259] A. Waxman and S. Ullman. Surface structure and three-dimensional motion from image flow kinematics. *International Journal of Robotics Research*, 4:72–94, 1985.

[260] J. Weber, J. Malik, S. Devadas, and P. Michel. Robust computation of optical flow in a multi-scale differential framework. *International Journal of Computer Vision*, 14:12–20, 1994.

[261] K. Weber, S. Venkatesh, and M. Srinivasan. Insect-inspired robotic homing. *Adaptive Behavior*, 7(1):65–97, 1999.

[262] J. Weng, T. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11:451–476, 1989.

[263] C. Wheatstone. Contributions to the physiology of vision. Part the first: On some remarkable, and hitherto unobserved, phenomena of binocular vision. *Phil. Trans. of the Royal Society of London*, 128:371–394, 1838.

[264] A. Witkin. Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

[265] K. Wohn, J. Wu, and R. Brockett. A contour-based recovery of image flow: Iterative transformation method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(8):746–760, 1991.

[266] W. Wolfe, D. Mathis, C. Sklair, and M. Magee. The perspective view of 3 points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, 1991.

[267] WowWee. <http://www.wowwee.com/en/support/flytech-dragonfly>, 2009.

[268] C. Wu. SiftGPU: A GPU Implementation of SIFT (accessed: 26 May 2009) http://www.cs.unc.edu/∼ccwu/siftgpu/.

[269] F. Wu, F. Duan, Z. Hu, and Y. Wu. A new linear algorithm for calibrating central catadioptric cameras. *Pattern Recognition*, 41(10):3166–3172, October 2008.

[270] T. Xiang and L. Cheong. Understanding the behavior of SFM algorithms: a geometric approach. *International Journal on Computer Vision*, 51(2):111–137, 2003.

[271] L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized Hough transform. *Pattern Recognition Letters*, 11(5):331–338, 1990.

[272] B. L. Yen and T. S. Huang. Determining 3-D motion and structure of a rigid body using the spherical projection. In *Computer Vision, Graphics and Image Processing*, volume 21, pages 21–32, 1983.

[273] X. Ying and Z. Hu. Catadioptric camera calibration using geometric invariants. *Pattern Analysis and Machine Intelligence*, 26(10):1260–1271, October 2004.

[274] X. Ying and H. Zha. Linear approaches to camera calibration from sphere images or active intrinsic calibration using vanishing points. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 596–603, 2005.

[275] X. Ying and H. Zha. Identical projective geometric properties of central catadioptric line images and sphere images with applications to calibration. *Int. J. on Computer Vision (IJCV)*, 78(1):89–105, June 2008.

[276] J. Zeil, M. Hoffmann, and J. Chahl. Catchment areas of panoramic images in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469, 2003.

[277] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *Technical Report RR-2927, INRIA*, 1996.

[278] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, 2000.

[279] Z. Zhang. Camera calibration. In *Emerging Topics in Computer Vision*, pages 4–43. Prentice Hall, 2004. (editors: G. Medioni and S.B. Kang).

[280] Z. Zhang, R. Deriche, O. Faugeras, and Q. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78:87–119, 1995.

[281] X. Zhuang, T. Huang, N. Ahuja, and R. Haralick. A simplified linear optic flow motion algorithn. *Computer Vision, Graphics, and Image Processing*, pages 334–344, 1988.

[282] J.-C. Zufferey, A. Klaptocz, A. Beyeler, J.-D. Nicoud, and D. Floreano. A 10-gram vision-based flying robot. *Advanced Robotics*, 21(14):1671–1684, 2007.