View Planning and Refractive Modeling for Structure and Motion

Haner, Sebastian

2015

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

View Planning and Refractive Modeling for Structure and Motion

Sebastian Haner

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Doctoral Theses in Mathematical Sciences 2015:2
ISSN 1404-0034

ISBN 978-91-7623-233-0 (print)
ISBN 978-91-7623-234-7 (digital)
LUTFMA-1053-2015

Printed in Sweden by MediaTryck, Lund 2015

Preface

This thesis presents contributions to structure-and-motion estimation. In particular, the problem of view planning is considered, and continuous and discrete optimization-based algorithms are given for how to plan the path of a sensor to its destination, while balancing the competing goals of path length and reconstruction accuracy. The same concepts are then applied to the problem of sequential 3D reconstruction from unordered image sequences. Using covariance propagation and image order selection for view planning, significant gains in robustness and computational efficiency are achieved. A second topic is refractive structure-and-motion, specifically the problem of absolute pose estimation when the camera and structure are separated by an optically refracting plane. Using methods from algebraic geometry for solving multivariate polynomial systems, efficient minimal and near-minimal solvers are constructed. Finally, a practical method for calibrating a set of cameras under refraction is given, including an algorithm for efficient refractive bundle adjustment.

The content of the thesis is based on the following papers:

- Sebastian Haner and Anders Heyden "Optimal View Path Planning for Visual SLAM", *Scandinavian Conference on Image Analysis*, Ystad, 2011.

- Sebastian Haner and Anders Heyden "Covariance Propagation and Next Best View Planning for 3D Reconstruction", *European Conference on Computer Vision*, Florence, 2012.

- Sebastian Haner and Anders Heyden "Discrete Optimal View Path Planning", *International Conference on Computer Vision Theory and Applications*, Berlin, 2015.

- Sebastian Haner and Kalle Åström "Absolute Pose for Cameras Under Flat Refractive Interfaces", *International Conference on Computer Vision and Pattern Recognition*, Boston, 2015.
- Sebastian Haner, Linus Svärm, Erik Ask and Anders Heyden "Joint Under and Over Water Calibration of a Swimmer Tracking System", *International Conference on Pattern Recognition Applications and Methods*, Lisbon, 2015.

Papers not included in this thesis:

- Sebastian Haner and Anders Heyden "A Step Towards Self-calibration in SLAM: Weakly Calibrated On-line Structure and Motion Estimation", *Workshop on Mobile Vision*, San Francisco, 2010.
- Sebastian Haner and Anders Heyden "On-line Structure and Motion Estimation based on a Novel Parametrized Extended Kalman Filter", *International Conference on Pattern Recognition*, Istanbul, 2010.
- Sebastian Haner and Irene Y.-H. Gu "Combining Foreground/Background Feature Points and Anisotropic Mean Shift for Enhanced Visual Object Tracking", *International Conference on Pattern Recognition*, Istanbul, 2010.
- Pedro Piniés, Lina Maria Paz, Sebastian Haner and Anders Heyden "Decomposable Bundle Adjustment using a Junction Tree", *International Conference on Robotics and Automation*, Minneapolis, 2012.

Acknowledgements

I would like to thank all my colleagues at the Centre for Mathematical Sciences for making my time in Lund such a worthwhile experience and for providing a warm and welcoming working environment. In particular, I want to thank the members of the Mathematical Imaging Group and all the PhD students for their help in all matters computer vision, programming, mathematics and life, for their great company at lunch and at conferences, and for always making time for a chat when I wandered the halls of the department. I am very grateful for the support given by my supervisor Anders Heyden and co-supervisor Kalle Åström, and the generally important person Carl Olsson. Outside of work, my fellow badminton players have helped take the edge of the stressful days and have provided a different perspective on things. Finally, I want to thank my friends and family for always being there for me.

Lisbon, January 2015

Funding

Funding for this work has been provided by the Swedish Research Council projects “On-line Structure and Motion Estimation based on Hybrid Constraints and Non-linear Adaptive Observers” (grant no. 621-2008-4557) and “On-line Structure and Motion Estimation: Visual SLAM and Optimal Path Planning” (grant no. 621-2011-6150).

Contents

Preface iii

1 Introduction 1

2 Structure and Motion Preliminaries 5
- 2.1 Pinhole Camera Model 5
- 2.2 Maximum Likelihood Estimation 7
- 2.3 Feature Matching 11
- 2.4 RANSAC 11
- 2.5 Triangulation 13
- 2.6 Camera Resectioning 14
- 2.7 Epipolar Geometry 16
- 2.8 Bundle Adjustment 17
- 2.9 The Known Rotation Method 19

I View Planning

3 View Planning Preliminaries 23
- 3.1 Covariance Estimation 24
- 3.2 Additivity of Information 27

4 View Planning using Continuous Optimization 29
- 4.1 Related Work 30
- 4.2 Problem Formulation 31
- 4.3 Cost Function 32
- 4.4 Proposed Algorithm 35
- 4.5 Experiments 37
- 4.6 Discussion 37
- 4.7 Conclusion 41

5 View Planning using Discrete Optimization 43
5.1 Problem Description 43
5.2 Problem Formulation 44
5.3 Approximate Solution 48
5.4 Stratified Solution Strategy 53
5.5 Experiments 54
5.6 Conclusions 66
Appendix
5.A Semidefinite Programming 69
5.B Transforming the Objective 71

6 Reconstruction from Unordered Image Sequences 73
6.1 Estimation from Uncertain Geometry 74
6.2 Covariance Propagation 77
6.3 Next Best View Planning 79
6.4 Reconstruction Pipeline 80
6.5 Experiments 81
6.6 Conclusion 88

II Refractive Structure and Motion

7 Preliminaries on Refraction 91
7.1 Snell's Law 91
7.2 Solving Polynomial Equation Systems 93

8 Absolute Pose under Refraction 103
8.1 Snell's Law 104
8.2 Unknown Camera Translation 106
8.3 The 2D Case 109
8.4 Known Rotation Axis 110
8.5 Absolute Pose with Five Points 112
8.6 Unknown Focal Length with Six Points 114
8.7 Experiments 115
8.8 Conclusions 117

9 Refractive Camera Calibration 123
9.1 Camera Setup 124
9.2 Calibration 124
9.3 Stitching 135
9.4 Conclusion 137

Bibliography 141

Chapter 1

Introduction

A central problem in the field of geometric computer vision is reconstructing the world in three dimensions, given only two-dimensional images. The loss of depth information in an image projection can be compensated for by moving the camera and viewing the scene from different perspectives. Given two or more images of an object, and information on from what angle they were taken, the object's 3D coordinates can be calculated. As it turns out, it is even possible to compute both the cameras' viewpoints and the object coordinates using only image information, and this problem is known as *structure-and-motion* reconstruction. One of two major topics in the thesis is improving the accuracy of reconstructions using planning, either actively or passively. Using passive planning we tackle the extensively studied problem of reconstructing a scene given only an unordered collection of images of the same area or object, often crowd-sourced off the Internet. By selecting the order in which images are used in a sequential algorithm, and keeping track of the uncertainties involved in triangulation and pose estimation, we see notable improvements. In situations where the reconstruction algorithm is "in the loop", such as when part of a mobile robot's navigation system, the acquisition of images can be actively influenced. This can be exploited, achieving higher estimation accuracy by selecting the camera viewpoints used while respecting the robot's navigational aims.

The second topic of the thesis is reconstruction under refraction. With the advent of more advanced and autonomous unmanned underwater vehicles, underwater imaging has received increased attention in the last few years. When viewing an object under water, we are almost always required to do so through a window. At the interfaces between water and glass and glass and air, the bending of light poses problems for standard reconstruction algorithms which are built on the assumption that light travels in straight lines. Modeling the refractions using Snell's law, we study the problem of determining the camera's viewpoint when observing a known object through a refractive interface. We also consider the problem of jointly calibrating a set of cameras imaging under refraction, with the goal of generating visually pleasing synthetic panoramas of a swimming pool, and accurately measuring the position of a swimmer therein.

Thesis Overview

The thesis has been split into two parts, the first dealing with view planning and the second with refraction. Each has an introductory chapter with background material specific to that part.

Chapter 2 This chapter provides a general introduction to standard structure-and-motion concepts and algorithms, which are used later in the thesis.

Chapter 3 An introduction to view planning is given along with some theoretical results which are used in the subsequent chapters.

Chapter 4 A continuous model for optimal view path planning is given. With a fixed start and destination, the path of a camera sensor is optimized with respect to path length and reconstruction covariance.

Chapter 5 The problem from Chapter 4 is cast as a discrete optimization problem, and shown to be *NP*-hard. Semidefinite programming relaxations are used to obtain lower bounds and starting guesses for a genetic algorithm minimizer.

Chapter 6 The planning concepts from Chapter 4 and 5 are applied to reconstruction from unordered image collections.

Chapter 7 This chapter gives an introduction to refraction and Snell's law, and to polynomial equation solving using algebraic geometry.

Chapter 8 The problem of determining absolute camera pose under refraction is discussed, and efficient solvers based on the techniques introduced in Chapter 7 are derived.

Chapter 9 The practical calibration of a swimmer tracking system is described. An efficient algorithm for the forward projection of a scene point through flat refractive interfaces is derived and applied to refractive bundle adjustment.

Author Contributions

The per-paper contributions of the author are as follows:

- Sebastian Haner and Anders Heyden "Optimal View Path Planning for Visual SLAM" *Scandinavian Conference on Image Analysis*, 2011.

  Anders came up with the main idea, I developed and implemented the algorithm and wrote most of the paper.

- Sebastian Haner and Anders Heyden "Covariance Propagation and Next Best View Planning for 3D Reconstruction" *European Conference on Computer Vision*, 2012.

  I came up with the idea, developed the theory and algorithms and ran the experiments, and wrote most of the paper.

- Sebastian Haner and Anders Heyden "Discrete Optimal View Path Planning" *International Conference on Computer Vision Theory and Applications*, 2015.

  I developed the idea, implemented the algorithms and wrote the paper.

- Sebastian Haner and Kalle Åström "Absolute Pose for Cameras Under Flat Refractive Interfaces" *In submission*.

  The idea to investigate this problem came from Kalle, and we collaborated on the theory. I implemented the solvers, ran the experiments and wrote the paper.

- Sebastian Haner, Linus Svärm, Erik Ask and Anders Heyden "Joint Under and Over Water Calibration of a Swimmer Tracking System" *International Conference on Pattern Recognition Applications and Methods*, 2015.

  Linus designed and built the calibration object and assisted in data collection together with Erik and Anders. Erik developed the marker detection algorithm and wrote the corresponding part of the paper. I developed and implemented all the calibration algorithms and wrote the remainder of the paper.

Code

Code which can be used to reproduce some of the experiments in this thesis is available at http://www.github.com/sebhaner.

Chapter 2

Structure and Motion Preliminaries

This chapter gives a brief, and certainly not complete, introduction to multiple-view geometry and structure-and-motion reconstruction, and covers some of the standard concepts and algorithms from the field which are used in the thesis. More specific background material has been deferred to the separate part introductions in Chapters 3 and 7.

Notation Throughout the thesis, vectors, vector-valued functions and matrices are typeset in bold. The notation $\boldsymbol{P}_{i,:}$ and $\boldsymbol{P}_{:,i}$ is used to indicate the i:th row or column of the matrix $\boldsymbol{P}$, respectively.

2.1 Pinhole Camera Model

In this work, as in most of the computer vision literature, the camera is modeled as an old-fashioned pinhole camera, where light from the scene passes through a focal point (the camera center) and falls on the image plane, forming an inverted image. Geometrically, undoing this inversion is equivalent to placing the image plane in front of the camera center (see Figure 2.1). If the camera center is at the origin and the optical axis is aligned with the z-axis of the coordinate system, the image projection of the point $\boldsymbol{X}$ can be computed as $\boldsymbol{x} = (fX_1/X_3, fX_2/X_3)^\top$. In the general case, with the camera centered at $\boldsymbol{C}$ and its orientation given by a rotation matrix $\boldsymbol{R}$, the scene point must first be transformed into the camera coordinate system, $\boldsymbol{X}^\mathrm{c} = \boldsymbol{R}(\boldsymbol{X} - \boldsymbol{C})$. It is convenient to express $\boldsymbol{x}$ and $\boldsymbol{X}$ in *homogeneous coordinates*, where an extra dimension is added to the representation, so that e.g. $\boldsymbol{X} = (X_1, X_2, X_3, 1)^\top$. Each point is now represented by an equivalence class where $\boldsymbol{X} \sim \bar{\boldsymbol{X}}$ if

$\boldsymbol{X} = \lambda\bar{\boldsymbol{X}}$ for some $\lambda \neq 0$, and similarly for $\boldsymbol{x}$. With this convention, point projection can be described by the linear equation

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \boldsymbol{KPX} = \begin{pmatrix} f & \alpha & p_x \\ 0 & \gamma f & p_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & -\boldsymbol{RC} \end{pmatrix} \boldsymbol{X}. \tag{2.1}$$

The actual projection is then obtained from the representative of $\boldsymbol{x}$ with third coordinate equal to one, i.e. $(x_1/x_3, x_2/x_3, 1)^\top$. The matrix $\boldsymbol{K}$ encodes the *intrinsic* parameters of the camera that map the projected point to actual pixel coordinates. The aspect-ratio parameter γ allows for non-square pixels, the skew parameter α for non-perpendicular image coordinate axes, and the offset (p_x, p_y) accounts for the fact that the pixel coordinate system origin is at a corner of the image and not at the principal point. In most applications we can safely assume that $\gamma = 1$ and $\alpha = 0$.

The pinhole model does not fully capture the behavior of real cameras, which have lens systems exhibiting varying degrees of geometric distortion. This can be an unintended side-effect of the optics or a way to compress a large field of view onto a small imaging area, as is the case of a fish-eye lens. Many different models for the distortion have been proposed (e.g. Kilpelä 1981; Devernay and Faugeras 2001; Fitzgibbon 2001), but the most commonly used is a radially symmetric polynomial distortion model (Heikkilä and Silvén 1997)

$$\boldsymbol{x}_{\text{dist}} = \boldsymbol{x}(1 + k_1 r^2 + k_2 r^4 + \dots), \tag{2.2}$$

where $r = \|\boldsymbol{x}\|$. Here it is assumed that the image coordinate system has been shifted so that the center of distortion, taken to coincide with the principal point, is at the origin. Tangential distortion terms can be included but are usually ignored, and for most applications two or three radial terms provide sufficient accuracy. The dependence on the principal point means that in cameras with significant amounts of distortion, its location must be carefully calibrated, but otherwise it can be assumed to lie at the center of the image.

In the work that follows, unless otherwise stated it will be assumed that the intrinsic parameters f, γ, α, p_x and p_y are known and that lens distortion has been compensated for in the input to the algorithms. When this is the case, we say the camera is *calibrated*, and it is convenient to work with the normalized image coordinates $\boldsymbol{K}^{-1}\boldsymbol{x}$. We shall then simply take the projection model to be $\boldsymbol{x} = \boldsymbol{PX}$. For more background on camera models and computer vision geometry, see the standard textbook by Hartley and Zisserman (2003).

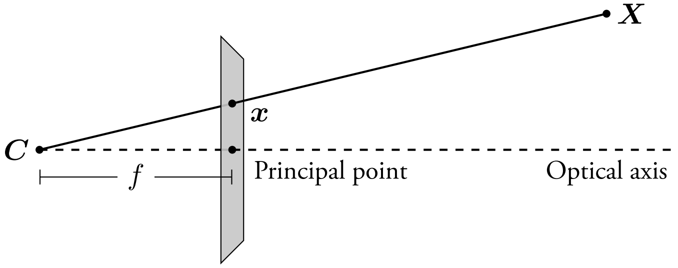


Figure 2.1: Pinhole camera model. The optical axis is perpendicular to the image plane and intersects it at the principal point at a distance f from the camera center, corresponding to the focal length.

2.1.1 Non-Central Cameras

In the pinhole camera, all light rays which fall on the image plane pass through a common point, the camera center; equivalently, one can say that each point on the image plane back-projects to a ray in space, and these all intersect in a single point. Virtually all cameras operate on this principle, but sometimes it is useful to extend the notion of "camera" to include external equipment such as mirrors through which the world is observed. Curved mirrors in front of the lens are often used to capture panoramic images, and the assembly as a whole does not act as a central camera, i.e. the back-projected image rays leaving the mirror surface do not in general intersect in a common central point. The image projection model adopted above must then be modified. Other examples of non-central (or *generalized*) cameras include assemblies of several pinhole cameras mounted together, as is common in robotics applications, or a camera observing a scene through a refractive interface, such as when mounted in a waterproof housing and submerged.

2.2 Maximum Likelihood Estimation

Solving structure-and-motion is an inverse problem where the loss of depth information in projections from 3D to 2D needs to be overcome. The problem is complicated further by the fact that image measurements are never exact, but exhibit a certain degree of noise. Given a probabilistic model of this noise, it can be argued that the best solution to the problem is the structure and motion which maximizes the probability of observing the measured data, the *maximum* *likelihood estimate.* Encoding the structure and motion in a parameter vector $\boldsymbol{\theta}$, the solution we seek is

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}), \tag{2.3}$$

where

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta} \mid \tilde{\boldsymbol{x}}) = p(\tilde{\boldsymbol{x}} \mid \boldsymbol{\theta}) \tag{2.4}$$

is the likelihood function and $\tilde{\boldsymbol{x}}$ a vector of measured image projections. If the camera model function $\boldsymbol{f}(\boldsymbol{\theta})$ gives the expected noise-free projections given camera and point parameters in $\boldsymbol{\theta}$, the *reprojection error* may be defined as $\boldsymbol{r}(\boldsymbol{\theta}) = \boldsymbol{f}(\boldsymbol{\theta}) - \tilde{\boldsymbol{x}}$ and $p(\tilde{\boldsymbol{x}} \mid \boldsymbol{\theta}) = D\big(\boldsymbol{r}(\boldsymbol{\theta})\big)$ where D is the probability density function of the noise distribution. In the case of zero-mean Gaussian noise, $D = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, the likelihood function becomes

$$\mathcal{L}(\boldsymbol{\theta}) \propto e^{-\frac{1}{2}\boldsymbol{r}(\boldsymbol{\theta})^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{r}(\boldsymbol{\theta})}, \tag{2.5}$$

and equivalently, taking the logarithm,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ \boldsymbol{r}(\boldsymbol{\theta})^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{r}(\boldsymbol{\theta}). \tag{2.6}$$

Under the common assumption $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$, this reduces further to just minimizing the sum of squares of the reprojection errors, i.e. a quadratic penalty. For other noise distributions, corresponding cost functions are similarly derived.

Unfortunately, the resulting cost functions are almost always non-convex and difficult to optimize. Given a decent guess for the optimal structure and motion parameters $\boldsymbol{\theta}^*$, however, it is often possible to find good solutions using iterative local optimization methods.

2.2.1 Least-squares Optimization

As we have just seen, finding the maximum likelihood solution to structure and motion problems with Gaussian image noise is equivalent to minimizing the sum of the squares of the reprojection errors, and in fact many problems have non-linear objective functions which are sums of squared terms.

The Gauss-Newton Algorithm

Given a differentiable vector-valued function $\boldsymbol{r}\colon \mathbb{R}^p \mapsto \mathbb{R}^m$, the Gauss-Newton algorithm attempts to minimize the squared Euclidean norm $s(\boldsymbol{\theta}) = \|\boldsymbol{r}(\boldsymbol{\theta})\|^2 = \sum_{i=1}^m r_i(\boldsymbol{\theta})^2$. Taylor expansion of the objective function gives

$$s(\boldsymbol{\theta} + \boldsymbol{\delta\theta}) \approx s(\boldsymbol{\theta}) + (\nabla s)^\top \boldsymbol{\delta\theta} + \frac{1}{2}\boldsymbol{\delta\theta}^\top \boldsymbol{H} \boldsymbol{\delta\theta}, \tag{2.7}$$

a quadratic approximation of the squared norm around the point $\boldsymbol{\theta} \in \mathbb{R}^p$. The gradient has the special form

$$\nabla s = 2\left(\sum_{k=1}^m \frac{\partial r_k}{\partial \theta_1} r_k, \, \dots, \sum_{k=1}^m \frac{\partial r_k}{\partial \theta_p} r_k\right) = 2\boldsymbol{r}^\top \boldsymbol{J}, \tag{2.8}$$

where $\boldsymbol{J}$ is the Jacobian matrix of $\boldsymbol{r}(\boldsymbol{\theta})$. Similarly, the Hessian matrix $\boldsymbol{H}$ has elements

$$H_{ij} = 2\sum_{k=1}^m \frac{\partial r_k}{\partial \theta_i} \cdot \frac{\partial r_k}{\partial \theta_j} + 2\sum_{k=1}^m \frac{\partial^2 r_k}{\partial \theta_i \partial \theta_j} \cdot r_k. \tag{2.9}$$

Assuming that the second order derivatives are bounded and the errors r_k small near the minimum, the second term is dropped and $\boldsymbol{H} \approx 2\boldsymbol{J}^\top \boldsymbol{J}$. Given a starting point $\boldsymbol{\theta}$, the quadratic approximation (2.7) of $s(\boldsymbol{\theta})$ is minimized by differentiating with respect to $\boldsymbol{\delta\theta}$ and equating to zero, giving the linear system $\boldsymbol{J}^\top \boldsymbol{J} \boldsymbol{\delta\theta} = -\boldsymbol{J}^\top \boldsymbol{r}$ for the update step. By iteratively moving to the minimizer of the approximation, $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{\delta\theta}_k$, the minimum of s will eventually be reached, given that the starting guess was close enough and the function is reasonably well-behaved.

Levenberg-Marquardt

The Levenberg-Marquardt (LM) algorithm (Levenberg 1944) is a small modification addressing some shortfalls of Gauss-Newton iteration. Far from the minimum, the quadratic approximation of Gauss-Newton may be a poor fit to the actual cost surface, and moving to its minimum could actually increase the objective function value. LM changes the update step equation system to

$$(\boldsymbol{J}^\top \boldsymbol{J} + \lambda \boldsymbol{I})\boldsymbol{\delta\theta} = -\boldsymbol{J}^\top \boldsymbol{r}, \tag{2.10}$$

where $\boldsymbol{I}$ is the identity matrix. For large values of λ, the step solution is close to gradient descent i.e. a safer linear approximation of the objective function instead of a quadratic. If a step does not decrease the objective value, λ is increased and a new step computed. As one approaches the minimum, λ may be decreased to take advantage of the quadratic convergence speed of Gauss-Newton. Augmenting the diagonal of the Hessian matrix also ensures it is positive definite so that a unique solution for the step exists.

The convergence speed of LM can sometimes be improved by using line search. Given the step direction $\boldsymbol{\delta\theta}_k$, the update is given by $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \tau_k \boldsymbol{\delta\theta}_k$ where the largest reduction in objective value is obtained by finding the step length

$$\tau_k = \arg\min_{\tau} s(\boldsymbol{\theta}_k + \tau\boldsymbol{\delta\theta}_k). \tag{2.11}$$

The solution to (2.11) is usually only found approximately using a few iterations of e.g. golden section search (Kiefer 1953) or Armijo's rule (Armijo 1966).

2.2.2 Initialization

The Levenberg-Marquardt algorithm is very general and can be applied to most differentiable cost functions, but as a local optimization method there is no guarantee it will find the global optimum, unless the problem has specific properties such as convexity. Its performance depends on the quality of the provided initial solution, which must be found by other means.

In general, finding the maximum likelihood solution to structure-and-motion problems is very difficult, and in practice it must be solved approximately as a sequence of subproblems. The basic building blocks of most structure-and-motion reconstruction systems are *triangulation* and *camera resectioning*, which provide initial solutions to maximum likelihood estimation, called *bundle adjustment*. However, the very first step is to analyze the input images and establish correspondences between objects seen in more than one view, known as *feature matching*. Related problems such as loop closure and pose graph optimization are important parts of many practical systems, especially in robotics applications, but are not the focus of this thesis.

2.3 Feature Matching

In this work, physical objects will be exclusively represented by collections of points in 3D space. Other representations are possible, such as lines and surfaces, but their projections are more difficult to extract from images. Any reconstruction algorithm starts by identifying points in different images corresponding to the same point in space. This is usually implemented in a three-stage process:

1. Identify salient image features
2. Compute descriptors for the image patches surrounding the features
3. Match descriptors from different images.

Salient features are areas of the image which stand out in some way, such as corners or high-contrast blobs. A large number of *feature detectors* have been proposed for this purpose, e.g. the Harris corner detector (Harris and Stephens 1988), FAST (Rosten and Drummond 2006), MSER (Matas, Chum, et al. 2002) and CenSurE (M. Agrawal et al. 2008). After a feature has been identified, a patch of the image around it is encoded by a *feature descriptor* which attempts to capture the appearance of the object point in a manner invariant to imaging conditions such as changes in the point of view and illumination. Feature descriptor algorithms often come with their own tailored feature detector, and are an active area of research. Popular algorithms include SIFT (Lowe 2004), SURF (Bay et al. 2008), ORB (Rublee et al. 2011), GLOH (Mikolajczyk and Schmid 2005), HOG (Dalal and Triggs 2005), DAISY (Tola et al. 2010), BRIEF (Calonder et al. 2010), BRISK (Leutenegger et al. 2011), FREAK (Alahi et al. 2012) and other increasingly amusing acronyms. To identify image features corresponding to the same 3D point, descriptors are compared and matched across images. An example is shown in Figure 2.2.

2.4 RANSAC

Feature detection and matching algorithms are not foolproof, and there will almost always be a certain number of incorrect matches, or *outliers*. These can seriously harm the performance of reconstruction algorithms and need to be dealt with. One approach is to use robust cost functions to mitigate the impact of erroneous measurements, but better still is to remove them completely. Usually the correct matches are known to adhere to some geometrical model;

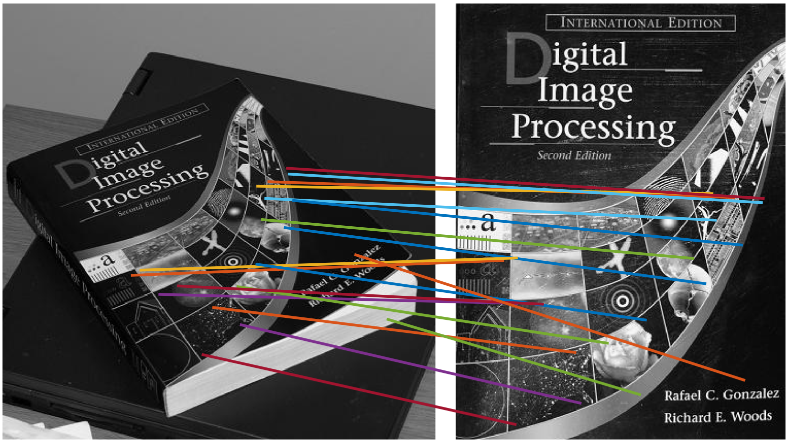


Figure 2.2: Feature matching example using the SURF feature detector and descriptor. The descriptors are 64-element vectors and each feature in the left image is assigned its closest match in the right as measured by the Euclidean distance in $\mathbb{R}^{64}$. Note that there is one incorrect match which can cause problems for reconstruction algorithms.

in the example in Figure 2.3 all points are known to lie on a line. Outliers stand out by not adhering to the model, so if the model parameters can be reliably estimated, outliers can be detected. But finding the model parameters which describe the correct matches well enough can be difficult when the data is contaminated. *Random sample consensus*, or RANSAC (Fischler and Bolles 1981), attempts to solve this problem by fitting the model to a randomly selected minimal set of points. If this set does not contain an outlier, one may assume the inliers will be close to the fitted model and the outliers not, allowing classification. If the minimal set contains an outlier, fewer points are expected to agree with the model so this can be detected. By repeating the process the probability of finding a minimal set without outliers can be made arbitrarily high. The relationship between the probability p of finding such a set (and thus presumably a solution) and the number of iterations i is easily shown to be

$$i = \frac{\log(1-p)}{\log(1-r^n)}\,, \tag{2.12}$$

where r is the ratio of inlier points to the total number of feature points, and n the number of points in the minimal set. For example, with $r = 0.5$ and $n = 3$, 34 iterations are required to reach a 99% confidence level, while for

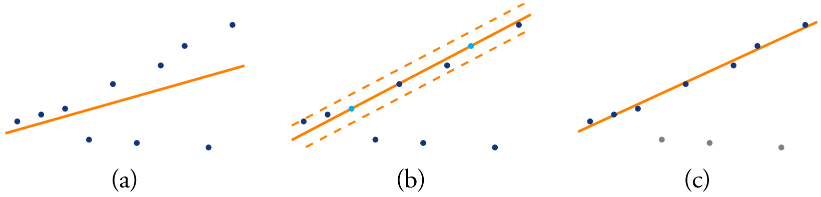


Figure 2.3: (a) Outliers skew the least-squares line fit. (b) RANSAC selects two random points and fits a line, all points close enough are designated inliers. (c) The outliers are discarded allowing a better fit.

$n = 4$ it is 71. For each extra point required to estimate the model parameters, the average number of iterations required to find a solution increases roughly by a factor of $1/r$. This is one of the reasons why efficient so-called *minimal solvers* are important in computer vision, i.e. algorithms which quickly fit a particular geometric model to the minimal number of data points. While many outlier rejection schemes have been proposed, *hypothesize-and-test* frameworks like RANSAC or derivatives (e.g. Torr 2002; Nistér 2003; Matas and Chum 2004) appear to still be the most popular.

2.5 Triangulation

Triangulation is the problem of finding the scene point $\boldsymbol{X}$ given its projection in two or more images and the corresponding camera poses. If the camera parameters and image projections are known precisely, the rays passing through camera centers $\boldsymbol{C}^i$ and corresponding image points $\boldsymbol{x}^i$ intersect in a single point. If there is noise in the image measurements, the rays will most likely not intersect, but we may try to find a point $\boldsymbol{X}$ that best explains the observations, a maximum likelihood estimate. As shown in Section 2.2, this involves minimizing the distance between the computed projection of $\boldsymbol{X}$ in the images and the measured values $\boldsymbol{x}^i$, a non-linear function with many local optima. In general this can only be accomplished using iterative optimization techniques. To find an initial estimate to start the iterative algorithm, or in the noiseless case, one may use a simple linear solution. In homogeneous coordinates we have $\boldsymbol{x}^i = \boldsymbol{P}^i\boldsymbol{X}$, $i = 1, \ldots, n$, which translates to the projections

$$\begin{pmatrix} x_1^i \\ x_2^i \end{pmatrix} = \frac{1}{\boldsymbol{P}^i_{3,:}\boldsymbol{X}} \begin{pmatrix} \boldsymbol{P}^i_{1,:}\boldsymbol{X} \\ \boldsymbol{P}^i_{2,:}\boldsymbol{X} \end{pmatrix} \tag{2.13}$$

where the subscript indicates the row of the vector or matrix. Each image gives two linear equations in $\boldsymbol{X}$,

$$\begin{pmatrix} x_1^i \boldsymbol{P}_{3,:}^i - \boldsymbol{P}_{1,:}^i \\ x_2^i \boldsymbol{P}_{3,:}^i - \boldsymbol{P}_{2,:}^i \end{pmatrix} \boldsymbol{X} = \boldsymbol{0}, \tag{2.14}$$

which when stacked together may be solved in a least-squares sense using the singular value decomposition. This method is fast and handles an arbitrary number of views, but it minimizes an algebraic error rather than the geometric reprojection error, which may produce poor results. For the case of only two or three views there exist methods which can compute the maximum likelihood triangulation under the assumption of Gaussian measurement noise, essentially in closed form (Hartley and P. Sturm 1997; Kanatani et al. 2008; Byröd et al. 2007). We will generally use the linear method followed by non-linear local optimization of the reprojection error using the LM algorithm.

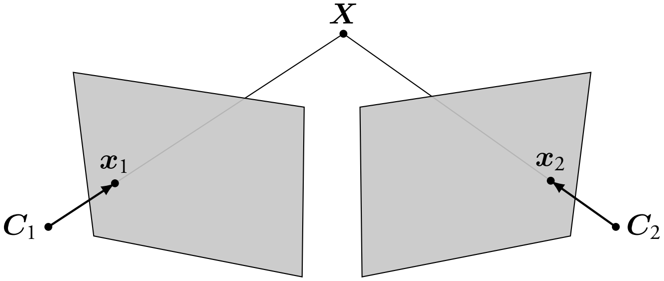


Figure 2.4: The principle of triangulation.

2.6 Camera Resectioning

Camera resectioning is the problem of determining the camera parameters given scene points $\boldsymbol{X}^i$ and corresponding image projections $\boldsymbol{x}^i$. If the intrinsic parameters are known, this is often referred to as *pose estimation* since only the camera translation and orientation need to be recovered. In the most general case, we seek the 3-by-4 camera matrix $\boldsymbol{M}$ best explaining the observations. Using (2.13), each projection $\boldsymbol{x}^i$ gives two linear equations in the components of $\boldsymbol{M}$,

$$\begin{pmatrix} \boldsymbol{X}^{i\top} & \boldsymbol{0} & -x_1^i \boldsymbol{X}^{i\top} \\ \boldsymbol{0} & \boldsymbol{X}^{i\top} & -x_2^i \boldsymbol{X}^{i\top} \end{pmatrix} \begin{pmatrix} M_{11} \\ M_{12} \\ \vdots \\ M_{34} \end{pmatrix} = \boldsymbol{0}. \tag{2.15}$$

With eleven or more equations, i.e. at least six point projections, the camera matrix may be uniquely determined (up to scale). It may then be factored into $\boldsymbol{M} = \boldsymbol{K}\boldsymbol{P}$ using RQ or Cholesky factorization, separating the intrinsic and the pose parameters. If the intrinsic parameters are known, methods exist (e.g. Gao et al. 2003) that allow pose estimation from three point projections, with up to four possible solutions; the correct one can be chosen by considering additional points. If only the focal length and a radial distortion parameter are unknown, four point are sufficient to find a solution (see e.g. Bujnak et al. 2010). In comparison, the linear method is again fast and simple, but minimizes an algebraic rather than a physically meaningful error. It also does not enforce the physical constraints that the observed scene points should all be in front of the camera, a problem which does arise in practice using this method.

As shown in Section 2.2, the assumption of Gaussian measurement noise implies that the maximum likelihood solution is obtained by minimizing the sum of squared reprojection errors. If we instead content ourselves with minimizing the maximum error, camera resectioning may be solved as a sequence of linear feasibility problems (Kahl and Hartley 2008). Choose an $\epsilon > 0$, and require the vertical and horizontal components of all reprojection errors to be smaller,

$$\left| x_1^i - \frac{\boldsymbol{M}_{1,:}\boldsymbol{X}^i}{\boldsymbol{M}_{3,:}\boldsymbol{X}^i} \right| \leq \epsilon, \tag{2.16}$$

$$\left| x_2^i - \frac{\boldsymbol{M}_{2,:}\boldsymbol{X}^i}{\boldsymbol{M}_{3,:}\boldsymbol{X}^i} \right| \leq \epsilon. \tag{2.17}$$

These constraints may be formulated as

$$\begin{pmatrix} -\boldsymbol{X}^{i\top} & \boldsymbol{0} & (x_1^i - \epsilon)\boldsymbol{X}^{i\top} \\ \boldsymbol{X}^{i\top} & \boldsymbol{0} & -(x_1^i + \epsilon)\boldsymbol{X}^{i\top} \\ \boldsymbol{0} & -\boldsymbol{X}^{i\top} & (x_2^i - \epsilon)\boldsymbol{X}^{i\top} \\ \boldsymbol{0} & \boldsymbol{X}^{i\top} & -(x_2^i + \epsilon)\boldsymbol{X}^{i\top} \end{pmatrix} \begin{pmatrix} M_{11} \\ M_{12} \\ \vdots \\ M_{34} \end{pmatrix} \leq \boldsymbol{0}. \tag{2.18}$$

To ensure that all points end up in front of the camera, the additional constraints $\boldsymbol{M}_{3,:}\boldsymbol{X}^i > 0$ are added. The task is then to find the smallest ϵ admitting a solution, which can be accomplished through bisection. Note that this formulation minimizes the maximum component-wise reprojection error. To use the Euclidean error instead, the linear inequalities can be replaced by second order cone constraints, at increased computational cost. Since a number of convex programs must be solved, the above approach is more costly than the direct linear algorithm, but usually gives better results since it minimizes a geometrically meaningful error. The exception is when there are outliers in the input data, i.e. grossly incorrect image measurements, which the max norm is highly sensitive to. The output of the above algorithms is usually used as the starting point for non-linear optimization to find a maximum likelihood estimate.

2.7 Epipolar Geometry

Point triangulation requires knowledge of the camera poses and resectioning requires knowledge of the 3D structure, so there seems to be a Catch 22; how do you start? One way to bootstrap the process is to simply guess depths of the points in one image and then refine the depth estimates as more images are considered, as is done in Haner and Heyden (2010) and other filter-based approaches. Epipolar geometry, on the other hand, allows us to compute the relative pose of two cameras without knowing the scene structure. Consider Figure 2.5. We may assume that the coordinate system has been chosen so that the first camera is given by $\boldsymbol{P}_1 = \begin{pmatrix}\boldsymbol{I}_{3\times 3} & \boldsymbol{0}\end{pmatrix}$ and the second by $\boldsymbol{P}_2 = \begin{pmatrix}\boldsymbol{R} & -\boldsymbol{R}\boldsymbol{C}\end{pmatrix}$. It is clear that the vectors $\boldsymbol{x}_1$, $\boldsymbol{R}^\top\boldsymbol{x}_2$ and $\boldsymbol{C}$ all lie in the same plane, and therefore their triple-product must be zero, $\boldsymbol{x}_1 \cdot \big(\boldsymbol{C} \times (\boldsymbol{R}^\top\boldsymbol{x}_2)\big) = 0$. Using the matrix representation $[\cdot]_\times$ of the cross product, defined so that $[\boldsymbol{a}]_\times\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$, the relation may be expressed as

$$\boldsymbol{x}_1^\top[\boldsymbol{C}]_\times\boldsymbol{R}^\top\boldsymbol{x}_2 = \boldsymbol{x}_1^\top\boldsymbol{E}\boldsymbol{x}_2 = 0. \tag{2.19}$$

The matrix $\boldsymbol{E} = [\boldsymbol{C}]_\times\boldsymbol{R}^\top$ is known as the *essential matrix* and encodes the relative pose between the cameras. From (2.19) the elements of the matrix can be deduced using a minimum of five matching pairs of image points (Nistér 2003a) or linearly using eight (Hartley 1997), and then factored into a rotation and a translation (see e.g. Hartley and Zisserman 2003). Only the direction of

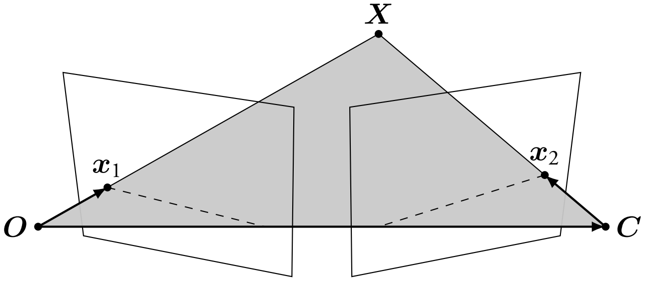


Figure 2.5: Epipolar geometry. The homogeneous image coordinates are transformed into the scene coordinate system using the respective camera's rotation and translation, leaving $\boldsymbol{x}_1$ unchanged and taking $\boldsymbol{x}_2$ to $\boldsymbol{R}^\top \boldsymbol{x}_2 + \boldsymbol{C}$.

the translation can be recovered, as there is no way to tell the absolute scale of a scene from the image projections alone.

2.8 Bundle Adjustment

To get the best reconstruction possible for the given input data, the statistically optimal maximum likelihood solution is preferred. As shown in Section 2.2, this comes down to minimizing the reprojection errors, now over all images and all camera and point parameters simultaneously. As already mentioned this is a difficult problem, and while small problems can be solved optimally using, for example, methods from Kahl and Henrion (2007), the high computational cost means finding a globally optimal solution is in general intractable. Given a good enough initial estimate, however, local optimization methods often find solutions near the global optimum. Optimization over all parameters is known as *bundle adjustment*, and the most popular method by far is Levenberg-Marquardt iteration (see Triggs et al. (2000) for an overview of bundle adjustment algorithms).

The main computational costs of LM are computing the partial derivatives in $\boldsymbol{J}$ and solving the system (2.10). Due to the special structure of bundle adjustment problems, the Hessian has regular and easily predictable sparsity patterns which can be exploited allowing the solution of very large problems with millions of parameters.

If the variables are partitioned into camera and point parameters so that $\boldsymbol{\theta} = \left(\boldsymbol{\theta}_c^\top, \boldsymbol{\theta}_p^\top\right)^\top$, then $\boldsymbol{J} = \begin{pmatrix}\boldsymbol{J}_c & \boldsymbol{J}_p\end{pmatrix}$ and $\boldsymbol{r} = \left(\boldsymbol{r}_c^\top, \boldsymbol{r}_p^\top\right)^\top$ are similarly partitioned and the LM system to solve can be written

$$\begin{pmatrix} \boldsymbol{U} & \boldsymbol{W} \\ \boldsymbol{W}^\top & \boldsymbol{V} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta\theta}_c \\ \boldsymbol{\delta\theta}_p \end{pmatrix} = \begin{pmatrix} -\boldsymbol{J}_c^\top \boldsymbol{r}_c \\ -\boldsymbol{J}_p^\top \boldsymbol{r}_p \end{pmatrix} \tag{2.20}$$

where $\boldsymbol{U} = \boldsymbol{J}_c^\top \boldsymbol{J}_c$ and $\boldsymbol{V} = \boldsymbol{J}_p^\top \boldsymbol{J}_p$ are block diagonal with one block per camera and point respectively. Block $\boldsymbol{W}_{ij}$ is only non-zero if camera i observes point j. Eliminating $\boldsymbol{\theta}_p$ from the top row using the Schur complement of $\boldsymbol{V}$ gives the system

$$\begin{pmatrix} \boldsymbol{U} - \boldsymbol{W}\boldsymbol{V}^{-1}\boldsymbol{W}^\top & \boldsymbol{0} \\ \boldsymbol{W}^\top & \boldsymbol{V} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta\theta}_c \\ \boldsymbol{\delta\theta}_p \end{pmatrix} = \begin{pmatrix} -\boldsymbol{J}_c^\top \boldsymbol{r}_c + \boldsymbol{W}\boldsymbol{V}^{-1}\boldsymbol{J}_p^\top \boldsymbol{r}_p \\ -\boldsymbol{J}_p^\top \boldsymbol{r}_p \end{pmatrix} \tag{2.21}$$

which allows us to solve for the camera parameters separately. After $\boldsymbol{\delta\theta}_c$ is found, the point parameters can be solved for using back substitution. The dominating cost is forming the Schur complement $\boldsymbol{S} = \boldsymbol{U} - \boldsymbol{W}\boldsymbol{V}^{-1}\boldsymbol{W}^\top$ and solving for the camera parameters. Since $\boldsymbol{V}$ is block diagonal it can be inverted in time linear in the number of points, and the back substitution operation is orders of magnitude cheaper. $\boldsymbol{S}$ has a particular block sparsity pattern, where block $\boldsymbol{S}_{ij}$ is non-zero only if camera i and j observe a common point. Thus, depending on the scene geometry, the matrix exhibits varying degrees of sparsity. Real datasets are typically sparse enough that it is much more efficient solving the system using sparse Cholesky factorization (Y. Chen et al. 2008) or conjugate gradient algorithms (Byröd and Åström 2010; Agarwal, Snavely, Seitz, et al. 2010; Kushal and Agarwal 2012) than by dense factorization. Nevertheless, bundle adjustment is still a computationally heavy part of the reconstruction pipeline, and Chapter 6 explores a method to reduce the need to run bundle adjustment.

2.8.1 Robust Cost Functions

Since the LM algorithm minimizes the sum of squared reprojection errors, it maximizes the reconstruction likelihood assuming normally distributed errors. Gaussian noise is a good approximation for the small measurement errors introduced by the limited resolution of the input images and imprecise feature detection algorithms. It is, however, not a good model for the large errors which sometimes occur due to incorrect matching of features between images. While RANSAC usually does a good job of detecting outliers, it may not catch them all. Remaining outlier data will skew the results because of the high cost they incur under the quadratic penalty function. To mitigate the impact, robust cost functions giving less importance to large errors can be used. A popular choice is the Huber function, defined as

$$C(x) = \begin{cases} x^2 & \text{if } |x| < \beta \\ 2\beta|x| - \beta^2 & \text{otherwise}\,. \end{cases} \tag{2.22}$$

It assigns quadratic cost to small errors and linear to large, and is convex which ensures it does not give rise to new local minima. The influence of outliers is greatly reduced, so that the result of the bundle adjustment comes close to the ML estimate for the outlier-free data. The outliers may then be detected and removed. To incorporate a robust cost function C into the LM algorithm, iterative reweighting is usually employed. By scaling each component of the vector $\boldsymbol{r}$ by the weight $w_i = \sqrt{C(r_i)}/|r_i|$, the sum of squared errors $s = \sum_{i=1}^{m}(w_i r_i)^2 = \sum_{i=1}^{m} C(r_i)$ assumes the desired value. The step equation (2.10) is modified to

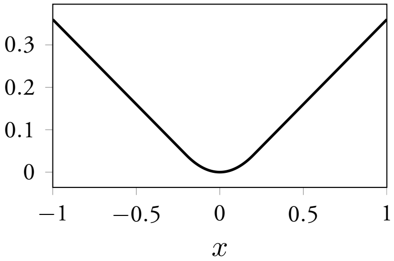


Figure 2.6: Huber cost function, $\beta = 0.2$.

$$(\boldsymbol{J}^\top \boldsymbol{W} \boldsymbol{J} + \lambda \boldsymbol{I})\boldsymbol{\delta\theta} = -\boldsymbol{J}^\top \boldsymbol{W} \boldsymbol{r}, \tag{2.23}$$

where $\boldsymbol{W} = \operatorname{diag}(w_1^2, \ldots, w_m^2)$. The weights are recomputed each iteration as the errors r_i change. In Chapter 6, iterative reweighting is used both for robust bundle adjustment, and as a way of implementing robust cost functions based on structure and camera covariance for triangulation and pose estimation.

2.9 The Known Rotation Method

A rather different approach to structure-and-motion was hinted at above, namely using the L_∞ norm and formulating the problem as a convex program. If the camera matrix is given by $\boldsymbol{P} = \begin{pmatrix}\boldsymbol{R} & \boldsymbol{a}\end{pmatrix}$ the projection in non-homogeneous coordinates may be written

$$\boldsymbol{x} = \frac{1}{\boldsymbol{R}_{3,:}\boldsymbol{X} + a_3}\begin{pmatrix}\boldsymbol{R}_{1,:}\boldsymbol{X} + a_1\\ \boldsymbol{R}_{2,:}\boldsymbol{X} + a_2\end{pmatrix}, \tag{2.24}$$

and the reprojection error constraints may be formulated as

$$\left|x_1 - \frac{\boldsymbol{R}_{1,:}\boldsymbol{X} + a_1}{\boldsymbol{R}_{3,:}\boldsymbol{X} + a_3}\right| \leq \epsilon, \tag{2.25}$$

$$\left|x_2 - \frac{\boldsymbol{R}_{2,:}\boldsymbol{X} + a_2}{\boldsymbol{R}_{3,:}\boldsymbol{X} + a_3}\right| \leq \epsilon. \tag{2.26}$$

For a point $\boldsymbol{X}^i$ seen in camera $\boldsymbol{P}^j$ these may be expressed as the inequality

$$\begin{pmatrix} (x_1^i - \epsilon)\boldsymbol{R}_{3,:}^j - \boldsymbol{R}_{1,:}^j & -1 & 0 & (x_1^i - \epsilon)\\ -(x_1^i + \epsilon)\boldsymbol{R}_{3,:}^j + \boldsymbol{R}_{1,:}^j & 1 & 0 & -(x_1^i + \epsilon)\\ (x_2^i - \epsilon)\boldsymbol{R}_{3,:}^j - \boldsymbol{R}_{2,:}^j & 0 & -1 & (x_2^i - \epsilon)\\ -(x_2^i + \epsilon)\boldsymbol{R}_{3,:}^j + \boldsymbol{R}_{2,:}^j & 0 & 1 & -(x_2^i + \epsilon)\end{pmatrix}\begin{pmatrix}\boldsymbol{X}^i\\ \boldsymbol{a}^j\end{pmatrix} \leq \boldsymbol{0}. \tag{2.27}$$

If the rotation matrices $\boldsymbol{R}^j$ are known for all cameras, the left-hand matrix is known and (2.27) is a linear inequality. The scene structure $\boldsymbol{X}^i$ and camera translations $\boldsymbol{a}^j$ can then be sought as the feasible solution to a linear program for which ϵ is as small as possible.

To find the camera orientations, the relative rotations between all pairs of cameras observing common 3D points can be computed using epipolar geometry. A process known as *rotation averaging* can then be used to find a set of absolute orientations consistent with the relative measurements. Many different algorithms for this have been proposed, see Hartley, Trumpf, et al. (2013) for an overview.

Part I

View Planning

Chapter 3

View Planning Preliminaries

Historically, structure-and-motion research has mainly focused on recovering the best reconstruction possible, given the input data. In other words, the algorithm acts as a passive consumer, and has no influence on what images or other sensor readings are collected. But in many cases, there is the possibility of steering the acquisition process. In the field of photogrammetry, for example, the goal is to measure the 3D coordinates of a scene or object as accurately as possible using visual information. Ideally, one would like to take as many images from as many angles as possible to cover the scene and get well-conditioned triangulation problems. Given constraints on resources such as the maximum number of images to capture, or physical constraints on possible placement of the camera, selecting a good set of imaging locations becomes an optimization problem known as *camera network design*. Traditionally, the process of setting up these "camera networks" was done by hand, placing cameras intuitively around an approximately known scene and then evaluating the resulting reconstruction using simulated measurement data (Fraser 1984). This was because automatic optimization is rather difficult; quite different camera configurations may give similar reconstruction errors, resulting in a very multi-modal, non-convex cost surface (cf. Mason 1997). Consequently, more recent efforts have resorted to stochastic optimization techniques for camera placement problems.

Of course, any such planning and optimization must be based on a *prediction of what will be seen* at a given camera location. Such predictions need some initial information about the scene structure, and also about other imaging conditions such as possible occlusions and the availability of surface texture for feature detection. In the photogrammetric setting this is usually not a problem since the site can be roughly surveyed using other means. In other applications, such initial information may not be easy to obtain and therefore planning the whole acquisition process beforehand is not possible, or even desired. In vision metrology, where often images of objects are captured by a camera on a robotic arm, there are also strong constraints on the possible movements of the camera and requirements of efficiency. The sequential nature of the acquisition then leads to the concept of *next best view* planning, where the next imaging location is planned based on the actual reconstruction obtained thus far, allowing for more flexibility with regard to unknown object shape and occlusions.

While such short-term planning is "safer" and easier to optimize, it only gives locally optimal results and struggles to handle long-term constraints on the acquisition process such as the maximum distance travelled by the sensor, for example. Finding the balance between a longer planning horizon, which may yield better results, and the risk of the unpredictability of what will be seen, is often referred to as the *exploration vs. exploitation* problem.

In the next two chapters, we will consider view planning in the context of *simultaneous localization and mapping*, the robotics term for structure-and-motion estimation. Given a target location, we ask how an autonomous robot should move to attain its goal while at the same time reconstructing its surroundings as accurately as possible. In Chapter 6, similar planning concepts are used on *given* image data, where only the order in which it is used can be chosen.

Below, some results are given explaining how the covariance of a reconstruction can be estimated, which is the basis of most planning algorithms. It may be beneficial to first read Chapter 4 to get some context, where we will see how the results are used to evaluate proposed paths of the camera.

3.1 Covariance Estimation

Let $\boldsymbol{r}(\boldsymbol{\theta}) = \boldsymbol{f}(\boldsymbol{\theta}) - \tilde{\boldsymbol{x}}$ be a vector collecting the reprojection errors of a structure-and-motion problem with projection function $\boldsymbol{f}$ and observations $\tilde{\boldsymbol{x}}$. Let $\boldsymbol{J} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\theta}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}$ be the Jacobian matrix of $\boldsymbol{r}$ evaluated at the maximum likelihood estimate $\boldsymbol{\theta}^*$, and $\boldsymbol{\Sigma}$ the image measurement covariance matrix. Under assumptions of Gaussian noise, the expression $(\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J})^{-1}$ then provides an estimate of the covariance matrix of the estimated structure and motion. A simple and intuitive proof of this is given in Hartley and Zisserman (2003), on which the following is based. The result is exact for affine projection functions; in the non-linear case, the function can be approximated by its tangent plane and an approximate estimate is obtained.

Lemma 3.1 *Let $\boldsymbol{v}$ be a random vector in $\mathbb{R}^m$ with mean $\bar{\boldsymbol{v}}$ and covariance matrix*

$\boldsymbol{\Sigma}$, and $\boldsymbol{f}\colon \mathbb{R}^m \mapsto \mathbb{R}^p$ an affine mapping defined by $\boldsymbol{f}(\boldsymbol{v}) = \boldsymbol{f}(\bar{\boldsymbol{v}}) + \boldsymbol{A}(\boldsymbol{v} - \bar{\boldsymbol{v}})$. Then $\boldsymbol{f}(\boldsymbol{v})$ is a random variable with mean $\boldsymbol{f}(\bar{\boldsymbol{v}})$ and covariance matrix $\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top$.

This follows easily from the definitions of mean and covariance, and is needed in the proof of the following theorem:

Theorem 3.2 (Result 5.9, Hartley and Zisserman 2003) *Let $\boldsymbol{f}\colon \mathbb{R}^p \mapsto \mathbb{R}^m$ be an affine mapping of the form $\boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{f}(\bar{\boldsymbol{\theta}}) + \boldsymbol{J}(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})$, where $\boldsymbol{J}$ has rank p. Let $\boldsymbol{x} \in \mathbb{R}^m$ be a Gaussian random variable with mean $\bar{\boldsymbol{x}} = \boldsymbol{f}(\bar{\boldsymbol{\theta}})$ and covariance matrix $\boldsymbol{\Sigma}$. Let $\boldsymbol{g}\colon \mathbb{R}^m \mapsto \mathbb{R}^p$ be the mapping taking a measurement $\boldsymbol{x}$ to the maximum likelihood parameter estimate $\boldsymbol{\theta}^*$. Then $\boldsymbol{\theta}^* = \boldsymbol{g}(\boldsymbol{x})$ is a random variable with mean $\bar{\boldsymbol{\theta}}$ and covariance matrix $(\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J})^{-1}$.*

Proof. The measurement model $\boldsymbol{f}\colon \mathbb{R}^p \mapsto S \subset \mathbb{R}^m$ takes parameters to expected measurements. The surface S is the space of all possible noise-free measurements. Since $\boldsymbol{J}$ has full rank, $\boldsymbol{f}$ is one-to-one and invertible. Define $\boldsymbol{\eta}\colon \mathbb{R}^m \mapsto S$ as the function mapping any measurement $\boldsymbol{x}$ to the unique closest point on S, in the Mahalanobis norm $\|\boldsymbol{x}\|_{\boldsymbol{\Sigma}} = \sqrt{\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{x}}$. The composition $\boldsymbol{g} = \boldsymbol{f}^{-1} \circ \boldsymbol{\eta}$ thus represents the maximum likelihood estimator, and may also be expressed as $\boldsymbol{g}(\boldsymbol{x}) = \operatorname{argmin}_{\boldsymbol{\theta}} \|\boldsymbol{x} - \boldsymbol{f}(\boldsymbol{\theta})\|_{\boldsymbol{\Sigma}} = \operatorname{argmin}_{\boldsymbol{\theta}} \|\boldsymbol{x} - \boldsymbol{f}(\bar{\boldsymbol{\theta}}) - \boldsymbol{J}(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})\|_{\boldsymbol{\Sigma}}$. Evaluating $\boldsymbol{g}$ is a weighted linear least-squares problem, and the normal equations give the solution $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{\theta}^* = (\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J})^{-1} \boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \big(\boldsymbol{x} - \boldsymbol{f}(\bar{\boldsymbol{\theta}})\big) + \bar{\boldsymbol{\theta}}$. This shows that $\boldsymbol{g}$ is an affine function of $\boldsymbol{x}$. Since $\boldsymbol{f}(\bar{\boldsymbol{\theta}}) = \bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{\theta}} = \boldsymbol{f}^{-1}(\bar{\boldsymbol{x}}) = \boldsymbol{g}(\bar{\boldsymbol{x}})$, Lemma 3.1 applies and gives the covariance matrix of $\boldsymbol{\theta}^*$ as $(\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J})^{-1} \boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma} \boldsymbol{\Sigma}^{-1} \boldsymbol{J} (\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J})^{-1} = (\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J})^{-1}$. □

This result may also be obtained using standard estimation theory. It can be shown that the maximum likelihood estimator under Gaussian noise, to first order, is unbiased and achieves the Cramér-Rao lower bound, i.e. that the covariance is given by the inverse of the Fisher information matrix, which is indeed $\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J}$ (see e.g. Morris 2001).

The condition that $\boldsymbol{f}$ is invertible means that no two points in the parameter space may give rise to the same measurements. This is the case for triangulation and pose estimation, where the cameras or points, respectively, are fixed in the coordinate system. However, when computing the covariance of a whole structure-and-motion system, the problem of gauge freedom appears. Any collection of scene points and cameras may be jointly translated, scaled and rotated without affecting the resulting image projections, thus an entire family of parameters are ML estimates of the same input data. Since for each degree of gauge freedom the rank of the information matrix drops by one, this ambiguity must be eliminated by choosing a minimal parametrization before the above result can be applied. To accomplish this, typically one camera is considered fixed, eliminating six degrees of freedom, and the distance to a second camera constrained, fixing the scale of the reconstruction.

Alternatively, one may take the Moore-Penrose pseudo-inverse of the rank-deficient information matrix instead, as the following results show.

Theorem 3.3 (Result 5.11, Hartley and Zisserman 2003) *Let $\boldsymbol{f}\colon \mathbb{R}^p \mapsto \mathbb{R}^m$ be a differentiable mapping taking parameters $\boldsymbol{\theta}$ to measurements $\boldsymbol{x}$. Let S be a smooth manifold of dimension d embedded in $\mathbb{R}^p$ passing through point $\boldsymbol{\theta}$, and such that $\boldsymbol{f}$ is one-to-one on S in a neighborhood of $\boldsymbol{\theta}$, mapping S to a manifold $\boldsymbol{f}(S) \subset \mathbb{R}^m$. Denote by $\boldsymbol{f}^{-1}$ the local inverse of $\boldsymbol{f}$ restricted to the surface $\boldsymbol{f}(S)$ in a neighborhood of $\boldsymbol{x}$. Let a Gaussian distribution be defined on $\mathbb{R}^m$ with mean $\boldsymbol{x}$ and covariance matrix $\boldsymbol{\Sigma}$ and let $\boldsymbol{\eta}\colon \mathbb{R}^m \mapsto \boldsymbol{f}(S)$ be the mapping taking a point in $\mathbb{R}^m$ to the closest point on $\boldsymbol{f}(S)$ in the Mahalanobis norm $\|\cdot\|_{\boldsymbol{\Sigma}}$. Via $\boldsymbol{f}^{-1} \circ \boldsymbol{\eta}$ the probability distribution on $\mathbb{R}^m$ induces a distribution on $\mathbb{R}^p$ with covariance matrix equal to first order to $(\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J})^{+\boldsymbol{A}} \equiv \boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J} \boldsymbol{A})^{-1} \boldsymbol{A}^\top$, where $\boldsymbol{A}$ is any matrix whose columns span the tangent space to S at $\boldsymbol{\theta}$.*

Proof. Let $\boldsymbol{g}\colon \mathbb{R}^d \mapsto \mathbb{R}^p$ map an open neighborhood $U \subset \mathbb{R}^d$ to an open subset of S containing $\boldsymbol{\theta}$. Then $\boldsymbol{f} \circ \boldsymbol{g}\colon \mathbb{R}^d \mapsto \mathbb{R}^m$ is one-to-one on U. Let $\boldsymbol{J}$ and $\boldsymbol{A}$ be the Jacobian matrices of $\boldsymbol{f}$ and $\boldsymbol{g}$, respectively. The Jacobian matrix of $\boldsymbol{f} \circ \boldsymbol{g}$ is then $\boldsymbol{J}\boldsymbol{A}$. Theorem 3.2 can now be applied to the first-order expansion of $\boldsymbol{f} \circ \boldsymbol{g}$ about $\boldsymbol{\theta}$, transporting the covariance backwards to covariance matrix $(\boldsymbol{A}^\top \boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J}\boldsymbol{A})^{-1}$. This matrix has rank and dimension d and so is invertible. Propagating this covariance forward through $\boldsymbol{g}$ into the higher-dimensional space using Lemma 3.1 gives the desired result, $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J}\boldsymbol{A})^{-1} \boldsymbol{A}^\top$. Substituting $\boldsymbol{A}\boldsymbol{B}$ for $\boldsymbol{A}$ leaves this expression unchanged if $\boldsymbol{B}$ is invertible, so $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ only depends on the column span of $\boldsymbol{A}$. □

Note that we are free to choose the manifold S (and thus $\boldsymbol{g}$ and $\boldsymbol{A}$). The following lemma simplifies the result for a particular choice:

Lemma 3.4 *Let $\boldsymbol{M}$ be a symmetric matrix, and let $\boldsymbol{M}^+$ be its Moore-Penrose pseudo-inverse. Then $\boldsymbol{M}^+ = \boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{M}\boldsymbol{A})^{-1}\boldsymbol{A}^\top = \boldsymbol{M}^{+\boldsymbol{A}}$ if $\boldsymbol{A}^\top \boldsymbol{M}\boldsymbol{A}$ is invertible and span$(\boldsymbol{A})$ = span$(\boldsymbol{M})$.*

Proof. Let $\boldsymbol{M} = \boldsymbol{UDU}^\top$ be the singular value decomposition of $\boldsymbol{M}$. If $\boldsymbol{M}$ has rank r, $\boldsymbol{U}$ may be partitioned into $\boldsymbol{U} = \begin{pmatrix}\boldsymbol{U}' & \boldsymbol{U}''\end{pmatrix}$ where $\boldsymbol{U}'$ are the first r columns and $\operatorname{span}(\boldsymbol{U}') = \operatorname{span}(\boldsymbol{M})$. We may write $\boldsymbol{M} = \boldsymbol{U}'\boldsymbol{D}'\boldsymbol{U}'^\top$ where $\boldsymbol{D}' = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$. The pseudo-inverse can then be computed as $\boldsymbol{M}^+ = \boldsymbol{U}'(\boldsymbol{D}')^{-1}\boldsymbol{U}'^\top$. As remarked above, $\boldsymbol{M}^{+\boldsymbol{A}} = \boldsymbol{M}^{+\boldsymbol{AB}}$ for any invertible $\boldsymbol{B}$. Since by assumption $\boldsymbol{M}$ and $\boldsymbol{A}$ span the same space, there is an invertible $\boldsymbol{B}$ such that $\boldsymbol{AB} = \boldsymbol{U}'$. Now $\boldsymbol{M}^{+\boldsymbol{A}} = \boldsymbol{M}^{+\boldsymbol{U}'} = \boldsymbol{U}'(\boldsymbol{U}'^\top\boldsymbol{M}\boldsymbol{U}')^{-1}\boldsymbol{U}'^\top = \boldsymbol{U}'(\boldsymbol{U}'^\top\boldsymbol{U}'\boldsymbol{D}'\boldsymbol{U}'^\top\boldsymbol{U}')^{-1}\boldsymbol{U}'^\top = \boldsymbol{U}'(\boldsymbol{D}')^{-1}\boldsymbol{U}'^\top = \boldsymbol{M}^+$. □

By taking the pseudo-inverse of the Fisher information matrix, we are thus choosing $\operatorname{span}(\boldsymbol{A}) = \operatorname{span}(\boldsymbol{J}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{J}) = \operatorname{span}(\boldsymbol{J}^\top)$. This corresponds to the restricted parameter manifold S being locally orthogonal to the null-space of the Jacobian $\boldsymbol{J}$. This is natural, since moving in this space does not change the measurements, but only explores the various gauge freedoms of the parametrization. In a sense, the pseudo-inverse allows us to compute covariances of structure-and-motion reconstructions independently of the coordinate system; if the gauge is locked by fixing parameters, the computed covariances will be expressed relative to these. On the other hand, the cost of computing the pseudo-inverse may be prohibitive in practice, while the ordinary inverse is not. As discussed in Chapter 6, different regularizing constraints may then be added to the reprojection function making $\boldsymbol{J}$ full rank.

3.2 Additivity of Information

As above, let $\boldsymbol{r}(\boldsymbol{\theta})$ be the collected reprojection errors of a structure-and-motion problem. We may partition $\boldsymbol{r}$ into separate observations, for example

$$\boldsymbol{r} = (\boldsymbol{r}_1^\top, \ldots, \boldsymbol{r}_n^\top)^\top, \tag{3.1}$$

where $\boldsymbol{r}_i$ contains the errors from image i. The rows of the Jacobian matrix $\boldsymbol{J}$ will be correspondingly partitioned, and assuming that the observations are independent so that the measurement covariance matrix $\boldsymbol{\Sigma}$ is block diagonal, the expression for the total information matrix is seen to be

$$\boldsymbol{I} = \boldsymbol{J}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{J} = \sum_{i=1}^{n} \boldsymbol{J}_i^\top\boldsymbol{\Sigma}_i^{-1}\boldsymbol{J}_i, \tag{3.2}$$

that is the sum of the individual information matrices from each image (note that we will use $\boldsymbol{I}$ to denote both information and identity matrices; the meaning will be clear from the context). The goal of view planning is to select the terms of the sum (3.2), i.e. the camera viewpoints, so that some scalar function $F(\boldsymbol{I}^{-1})$ of the covariance matrix is minimized, possibly in competition with other objectives. A reasonable criterion on F is that it should decrease monotonically as more information becomes available, and two possible choices are the trace and maximum eigenvalue functions. Note that the information matrices are always positive semidefinite by construction. The following theorems show that the eigenvalues of a sum of positive semidefinite matrices are always larger than the eigenvalues of the terms. As a consequence, since $\mathrm{tr}(\boldsymbol{I}^{-1})$ and $\lambda_{\mathrm{max}}(\boldsymbol{I}^{-1})$ are monotonically decreasing functions of the eigenvalues of $\boldsymbol{I}$, their value must decrease as terms are added to the sum.

Theorem 3.5 (Courant-Fischer) *Let $\boldsymbol{A}$ be a $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \leq \ldots \leq \lambda_k \leq \ldots \leq \lambda_n$. Then*

$$\lambda_k = \min_{S_k} \max_{\substack{\boldsymbol{x} \in S_k \\ \|\boldsymbol{x}\|=1}} \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x},$$

where S_k is any k-dimensional subspace of$\mathbb{R}^n$.

Theorem 3.6 *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be positive semidefinite $n \times n$ matrices, and let $\lambda_1 \leq \ldots \leq \lambda_k \leq \ldots \leq \lambda_n$ and $\mu_1 \leq \ldots \leq \mu_k \leq \ldots \leq \mu_n$ be the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{A}+\boldsymbol{B}$ respectively. Then $\mu_k \geq \lambda_k$ for $1 \leq k \leq n$.*

Proof. Let ν_1 be the smallest eigenvalue of $\boldsymbol{B}$. Then we have that for all $\boldsymbol{x}$ s.t. $\|\boldsymbol{x}\| = 1$,

$$\begin{aligned} \boldsymbol{x}^\top \boldsymbol{B} \boldsymbol{x} &\geq \nu_1 \\ \Rightarrow\ \boldsymbol{x}^\top (\boldsymbol{A}+\boldsymbol{B}) \boldsymbol{x} &\geq \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} + \nu_1. \end{aligned}$$

By Theorem 3.5,

$$\mu_k = \min_{S_k} \max_{\substack{\boldsymbol{x} \in S_k \\ \|\boldsymbol{x}\|=1}} \boldsymbol{x}^\top (\boldsymbol{A}+\boldsymbol{B}) \boldsymbol{x} \geq \min_{S_k} \max_{\substack{\boldsymbol{x} \in S_k \\ \|\boldsymbol{x}\|=1}} \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} + \nu_1 = \lambda_k + \nu_1.$$

Since $\nu_1 \geq 0$, $\mu_k \geq \lambda_k$. □

Chapter 4

View Planning using Continuous Optimization

Structure-and-motion estimation, or simultaneous localization and mapping (SLAM), is a chicken-and-egg type problem; given the map, localization is relatively easy and given the camera positions, map triangulation is straightforward. Accomplishing both at once is at the heart of the SLAM problem, which has received a lot of attention in both the robotics and vision research communities. Much effort is spent improving the robustness and accuracy of algorithms, particularly with respect to error accumulation, drift and loop closing (see e.g. Oskarsson and Åström 2000; Guilbert et al. 2004; Botterill et al. 2010; Piniés et al. 2010). As mentioned in the introduction, a less studied problem is how to make efficient use of the information collected in active SLAM systems, that is systems where the motion of the sensor can be controlled. In this chapter we consider the problem of maximizing the useful information gained from a fixed number of images by active planning of the vision sensor movement. Specifically, we consider the task of finding a camera trajectory between two predetermined locations such that the reconstruction accuracy of observed geometry is maximized while the path length is minimized. The envisaged application is robot path planning, where the accuracy usually is a secondary objective, so the focus is on providing the best reconstruction given time or distance constraints.

In this work we only consider the geometric aspects of the problem and do not account for availability of texture or object occlusion, which are of course issues in a real system relying on feature tracking. We further assume the following:

- An initial maximum likelihood estimate of the structure is available, based on observations up to that point.

- All cameras along the trajectory are oriented towards a particular point of interest, e.g. the centroid of the features to be estimated.
- The camera can be positioned with such relative accuracy that its pose and location is fully known at each observation.

These assumptions may be relaxed, as discussed in Section 4.6.2. Finally, the robot path is represented by a sequence of camera locations, and the number of cameras on the path must be chosen in advance.

4.1 Related Work

Both camera network design and the next best view problem are hard due mainly to the non-convex, multi-modal costs arising, but also to the sometimes high computational burden of evaluating the cost function. Recent research has mostly focused on the latter problem of accurately and efficiently evaluating the expected information gain of a potential sensor configuration (Low and Lastra 2006; Vasquez-Gomez et al. 2013; Foix et al. 2012) and on achieving coverage of the scene (Blaer and Allen 2007), while other works tackle minimizing the resulting cost functions to find one or a series of optimal sensor configurations. The myopic planning horizon of next best view planning is often adopted due to these difficulties, and sensor planning problems of this type have mainly been addressed using stochastic optimization algorithms or by solving a relaxed, easier version. For example, in Dunn, Olague, et al. (2006) the camera network problem is solved using a genetic optimization algorithm searching the high-dimensional parameter space of camera placements. In Wenhardt et al. (2006) the authors reconstruct objects using a camera mounted on a robotic arm. The object geometry is estimated using a Kalman filter, and the next imaging location is determined by searching a discrete parameter space and evaluating the expected information gain in the filter at each position. A different approach is taken in Trummer et al. (2010) where the next imaging location is decided based only on the single currently least well-determined feature, allowing a simple closed form solution. In Dunn, Berg, et al. (2009) the path of a robot moving in the plane is planned based on the expected reconstruction accuracy of an observed object. An approximation of the geometry is given and the expected information gain from observing the object from a particular vantage point is determined on a discrete grid of camera locations. Each grid cell is assigned a cost proportional to the inverse of the information gain, and a minimum cost path is found between the starting point and the global minimum grid cell. This does not take into account the fact that an observation may be more or less valuable depending on what other observations are available, and thus does not really optimize the desired objective. For recent general surveys of the sensor planning field see e.g. LaValle (2006); Shengyong Chen et al. (2008).

In contrast to the above, we will use a continuous optimization approach and a planning horizon of several steps along with a path length regularizer.

4.2 Problem Formulation

The planner takes as input an initial estimate of the structure, the current location of the sensor and the desired destination. The output is a path, represented by a discrete set of sensor locations, connecting these points. The number of locations on the path can be set explicitly or deduced from e.g. the robot's speed and sample rate and the distance to be traveled. For the experiments in this chapter, the sensor is assumed to be a single fully calibrated camera, although extension to stereo and multi-camera systems is straightforward. The standard pinhole camera model is used, so that the relation between a world point $\boldsymbol{X}$ and its projection $\boldsymbol{x}$ is given by

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{P}, \boldsymbol{X}) = \left(\frac{\boldsymbol{R}_{1,:}(\boldsymbol{X} - \boldsymbol{t})}{\boldsymbol{R}_{3,:}(\boldsymbol{X} - \boldsymbol{t})}, \frac{\boldsymbol{R}_{2,:}(\boldsymbol{X} - \boldsymbol{t})}{\boldsymbol{R}_{3,:}(\boldsymbol{X} - \boldsymbol{t})} \right)^{\top}, \tag{4.1}$$

where $\boldsymbol{R}$ and $\boldsymbol{t}$ are the camera rotation and translation. However, any differentiable projection function $\boldsymbol{f}(\boldsymbol{P}, \boldsymbol{X})$ may be substituted, collecting the camera parameters in the vector $\boldsymbol{P}$.

In the interest of reducing the parameter space dimension, each camera is parametrized only by its position and is automatically oriented toward a point of interest, typically chosen as the centroid of the structure under consideration. Features are deemed visible if they fall within the camera's field of view; possible occlusion by other objects is not considered. The measurement uncertainty of features is also considered fixed.

We define the optimization problem as follows: *minimize the reconstruction uncertainty of observed geometry and the distance traveled by the sensor between imaging locations*. These are conflicting objectives, which are combined in a cost function defined below.

4.3 Cost Function

Lacking ground truth data or other *a priori* information, the quality of a reconstruction can only be judged by the statistical uncertainty of the estimate. Condensing a probability distribution into a scalar quality measure is not entirely straight-forward, however, and choices must be made depending on the intended application. Also, in most situations only estimates of the probability distribution are available, e.g. the mean and covariance. In the experimental design literature, many summary statistics have been proposed and are usually functions of the eigenvalues of the covariance matrix, e.g. the trace and determinant, cf. Montgomery (2000). In the structure-from-motion problem, the eigenvalues have a direct geometric interpretation which we consider below.

If we assume the position and orientation of the camera is fully known when an observation is made, the structure estimates corresponding to individual features are independent of each other, and the covariance matrix is block diagonal with 3-by-3 blocks (assuming point features). The eigenvalues of each block correspond to the semi-axes of the ellipsoid representing the variance of the feature location. We would like these ellipsoids to be as small as possible, but in what sense? If we minimize the volume, i.e. the determinant, we admit solutions where a point may be very well-determined in two directions but with a large uncertainty in the third (typically the depth). Minimizing the determinant of the entire covariance matrix (the so-called D-optimality criterion) could favor solutions where one point is very well determined while others are much less certain. For navigation and mapping purposes, we would like all, or at least the majority of features to be reconstructed to reasonable accuracy. Minimizing the largest eigenvalue (E-optimality) would achieve this, but results in a non-smooth objective function. We choose to minimize the sum of the eigenvalues (A-optimality), i.e. the trace of the covariance matrix, which provides a good trade-off with the added computational benefit of not having to calculate individual eigenvalues.

Before introducing the cost function, we discuss how to compute the trace given a set of measurements.

4.3.1 Calculating Covariance

In many recent SLAM systems (e.g. Klein and Murray 2007; Strasdat et al. 2010; Mouragnon et al. 2009) maximum likelihood estimates obtained via bundle adjustment are available. We assume the structure estimate is optimal in the ML sense with respect to the observations; as shown in Section 3.1, the information matrix is then given to first order by $\boldsymbol{I} = \boldsymbol{J}^\top \boldsymbol{R}^{-1} \boldsymbol{J}$ where $\boldsymbol{J}$ is the Jacobian of the reprojection error evaluated at the minimum, and $\boldsymbol{R}$ the measurement noise covariance. Also, the (pseudo-)inverse of $\boldsymbol{I}$ gives an approximation of the covariance matrix. Since information is additive, including new observations in the estimate amounts to summing the individual information matrices. In other words, to calculate the effect of new observations on the structure estimate, we compute the Jacobian of each observation and add the corresponding information matrices to the initial one. New observations may of course shift the ML estimate, invalidating the approximation, but this is avoided in a natural way as discussed in Section 4.4.

Given a world point $\boldsymbol{X}$ and a camera $\boldsymbol{P}$, let $\hat{\boldsymbol{x}}$ be the measured image coordinate, and $\boldsymbol{f}(\boldsymbol{P}, \boldsymbol{X})$ the projection function mapping $\boldsymbol{X}$ to the expected image coordinate $\boldsymbol{x}$. Define the reprojection error as $\boldsymbol{E}_{\boldsymbol{X}}(\boldsymbol{P}, \boldsymbol{X}, \hat{\boldsymbol{x}}) = \boldsymbol{f}(\boldsymbol{P}, \boldsymbol{X}) - \hat{\boldsymbol{x}}$ with Jacobian

$$\boldsymbol{J}_{\boldsymbol{X}} = \frac{d\boldsymbol{E}_{\boldsymbol{X}}}{d\boldsymbol{X}} = \begin{pmatrix} \frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} & \frac{\partial f_1}{\partial X_3} \\ \frac{\partial f_2}{\partial X_1} & \frac{\partial f_2}{\partial X_2} & \frac{\partial f_2}{\partial X_3} \end{pmatrix}. \tag{4.2}$$

If several points $\boldsymbol{X}^{1,\dots,N}$ are observed simultaneously, let

$$\boldsymbol{E}(\boldsymbol{P}, \boldsymbol{X}^{1:N}, \hat{\boldsymbol{x}}^{1:N}) = \begin{pmatrix} E_{\boldsymbol{X}^1} \\ \vdots \\ E_{\boldsymbol{X}^N} \end{pmatrix} \tag{4.3}$$

with block diagonal Jacobian

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{J}_{\boldsymbol{X}^1} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{J}_{\boldsymbol{X}^N} \end{pmatrix}. \tag{4.4}$$

The information matrix for a single image is then given by

$$\boldsymbol{I}(\boldsymbol{P}, \boldsymbol{X}^{1:N}) = \begin{pmatrix} \boldsymbol{J}_{\boldsymbol{X}^1}^\top \boldsymbol{R}_1^{-1} \boldsymbol{J}_{\boldsymbol{X}^1} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{J}_{\boldsymbol{X}^N}^\top \boldsymbol{R}_N^{-1} \boldsymbol{J}_{\boldsymbol{X}^N} \end{pmatrix}, \tag{4.5}$$

where usually

$$\boldsymbol{R}_i = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}. \tag{4.6}$$

The final information matrix given the initial information $\boldsymbol{I}_0$ and images from camera positions $\boldsymbol{P}^{1,\dots,M}$ is now

$$\boldsymbol{I}_M = \boldsymbol{I}_0 + \sum_{j=1}^{M} \boldsymbol{I}(\boldsymbol{P}^j, \boldsymbol{X}^{1:N}). \tag{4.7}$$

Note that the computation is linear in the number of observed features and the number of images, and that the covariance of the estimate is the inverse, $\boldsymbol{\Sigma}_{\boldsymbol{P}^{1:M},\boldsymbol{X}^{1:N}} = \boldsymbol{I}_M^{-1}$. For notational convenience, for the remainder of the chapter let $\boldsymbol{P}$ denote the set $\boldsymbol{P}^{1:M}$ of camera poses along a path, and $\boldsymbol{X} = \boldsymbol{X}^{1:N}$ the estimated structure.

4.3.2 Cost Function

We propose the following cost function:

$$\begin{aligned} C(\boldsymbol{P}, \boldsymbol{X}) &= \frac{1}{N}\operatorname{tr}(\boldsymbol{\Sigma}_{\boldsymbol{P},\boldsymbol{X}}) + \frac{\alpha}{(M-1)^{1-q}} \sum_{j=1}^{M-1} \|\boldsymbol{P}^{j+1}_{\text{pos}} - \boldsymbol{P}^{j}_{\text{pos}}\|^q \\ &= U(\boldsymbol{P}, \boldsymbol{X}) + \alpha D(\boldsymbol{P}), \end{aligned} \tag{4.8}$$

i.e. the uncertainty measure plus a function of the camera path, weighted by a constant factor $\alpha > 0$, where $q \geq 1$. The normalization constants N^{-1} and $(M-1)^{q-1}$ are designed to make the cost approximately invariant with respect to the number of observed features and camera positions on the path. Note that by choosing $q > 1$, $D(\boldsymbol{P})$ will favor solutions with equidistant spacing between the camera positions, and introducing an offset d, $D(\boldsymbol{P}) = \sum_{j=1}^{M-1}(\|\boldsymbol{P}^{j+1}_{\text{pos}} - \boldsymbol{P}^{j}_{\text{pos}}\| - d)^q$, we can impose the soft constraint that the path length be $d(M-1)$, if desired.

4.3.3 Cost Function Properties

The multi-modality of the objective functions normally used in next best view planning makes optimization difficult. The proposed cost function is no exception, but due to the somewhat local nature of the sought solution there are obvious bounds on the cost and geometry of the path.

Proposition 4.1 *$U(\boldsymbol{P}^{1:M}, \boldsymbol{X})$ is a non-negative decreasing function of the number of observations M.*

This is a direct consequence of Theorem 3.6. We may now derive a simple bound on the optimal path length.

Theorem 4.2 *The length of the path at the minimum $\boldsymbol{P}^*$ is bounded.*

Proof. Given any initial estimate $\hat{\boldsymbol{P}}$ of the path, we have

$$\begin{aligned}\alpha D(\boldsymbol{P}^*) &\leq U(\hat{\boldsymbol{P}}, \boldsymbol{X}) + \alpha D(\hat{\boldsymbol{P}}) - U(\boldsymbol{P}^*, \boldsymbol{X}) \\ &\leq U(\hat{\boldsymbol{P}}, \boldsymbol{X}) + \alpha D(\hat{\boldsymbol{P}}) \\ &\leq U_{\text{initial}} + \alpha D(\hat{\boldsymbol{P}}),\end{aligned}$$

where $U_{\text{initial}} = \frac{1}{N}\operatorname{tr}(\boldsymbol{\Sigma}_0)$ and $\boldsymbol{\Sigma}_0$ the covariance of the current structure estimate. Since $\|\boldsymbol{P}^{j+1}_{\text{pos}} - \boldsymbol{P}^{j}_{\text{pos}}\| < \|\boldsymbol{P}^{j+1}_{\text{pos}} - \boldsymbol{P}^{j}_{\text{pos}}\|^q + 1$, the length of $\boldsymbol{P}^*$ is bounded from above by $(M-1)^{1-q}\big(\alpha^{-1}U_{\text{initial}} + D(\hat{\boldsymbol{P}})\big) + M - 1$. □

We see that the path must be contained inside an ellipsoid with foci at the (fixed) first and last camera positions, and that the bound can be computed easily in advance. As expected, the optimal path approaches the line segment between the foci as α grows.

This result suggests that we may attempt to find and compare several local minima by optimizing with varying initial paths sampled from within the feasible ellipsoid.

4.4 Proposed Algorithm

As noted in the introduction, the next best view problem is known to suffer from multiple local minima; this is true for all reasonable choices of U. Finding the global minimum is a difficult problem, and the prevailing approach in the literature seems to be more or less exhaustive search over a discretized parameter space (Wenhardt et al. 2006; Dunn, Berg, et al. 2009) or stochastic optimization methods (S.Y. Chen and Y. Li 2004; Dunn, Olague, et al. 2006). In the interest of speed, however, we adopt a gradient based optimization scheme, using the well-known Levenberg-Marquardt (LM) method. LM minimizes the Euclidean norm of a residual vector $\boldsymbol{r}$, which we construct as

$$\boldsymbol{r} = \left(\frac{\operatorname{tr}(\boldsymbol{\Sigma}_{\boldsymbol{P},\boldsymbol{X}^1})}{N}, \ldots, \frac{\operatorname{tr}(\boldsymbol{\Sigma}_{\boldsymbol{P},\boldsymbol{X}^N})}{N}, \right. \\ \left. \frac{\alpha\|\boldsymbol{P}^2_{\text{pos}} - \boldsymbol{P}^1_{\text{pos}}\|^q}{(M-1)^{1-q}}, \ldots, \frac{\alpha\|\boldsymbol{P}^M_{\text{pos}} - \boldsymbol{P}^{M-1}_{\text{pos}}\|^q}{(M-1)^{1-q}} \right)^{\frac{1}{2}} \tag{4.9}$$

(the exponent indicates element-wise square root) so that $\|\boldsymbol{r}\|^2 = C(\boldsymbol{P}, \boldsymbol{X})$. The parameter space is the $M-2$ intermediate camera positions; the camera orientation is determined by its position and the interest point.

The final hurdle is how to evaluate the cost function *before* any observations are made. The best we can do is predict what the camera will see at a particular location given the current best estimate of the structure. Assuming that measurements are corrupted with zero-mean noise, the expected observation is simply the projection $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{P}_i, \boldsymbol{X})$. Such an observation has zero reprojection error, and so does not affect the ML estimate.

The optimization is applied within the following framework:

1. Given an initial estimate of the structure, calculate its centroid and let this be the camera's point of interest. Select a target location for the camera, i.e. select the endpoint of the path.

2. Generate an initial path by linear interpolation between the first and last camera locations. The number of discrete camera locations along the path could be selected to match the image sampling rate and speed of the robot, but this would normally result in far too many locations and a very high-dimensional search space. However, it stands to reason that more images taken from approximately the same vantage point do not contribute qualitatively to the reconstruction, so a relatively sparse distribution of camera locations is sufficient.

3. Find a minimum of the cost function with respect to $\boldsymbol{P}_{\text{pos}}$ using the LM algorithm. For improved convergence, an optimal step length may be selected through line search.

4. Move the camera to the next location along the path and make an actual observation. Update the structure estimate with this new information,

and update the camera interest point location and path endpoint, if needed.

Repeat steps 3 and 4, each time using the previous path estimate as an initial guess. While the endpoint has been kept fixed in the experiments below, in real use this is expected to be continually updated by the robot's higher-level planning functions; it is not anticipated that this sort of view planning will be performed with horizons longer than required to traverse a room, for example.

4.5 Experiments

We first apply the above algorithm to the scenario of a robot trying to pass through a doorway. The doorway is represented by a rectangular array of point features which are optimally triangulated from the first two views, see Figure 4.1a. In all experiments we assume an image measurement noise σ equivalent to about one pixel. The target location is placed in front of the doorway, and the path is discretized with four waypoints in between. The optimization is run until convergence and the robot is moved to the next prescribed location along the path, where a new image is acquired and the structure estimate is updated using bundle adjustment.

The influence of the parameter α is illustrated in Figure 4.2 and Table 4.1. The robot passes by a point cloud, and to get a closer look it must make a detour. A large α penalizes long paths at the expense of reconstruction accuracy.

4.6 Discussion

4.6.1 Computational Complexity

As noted in Section 4.3.1, the cost function can be evaluated in $\mathcal{O}(MN)$ time. The LM algorithm requires the computation of the Jacobian of the residual vector $\boldsymbol{r}$ each iteration. The analytic expression may be very complicated and expensive to evaluate, so automatic differentiation or a finite difference approximation is preferred. The cost function must be differentiated with respect to $3(M-2)$ parameters, requiring $3(M-2)+1$ function evaluations to compute the Jacobian. But the covariance matrix is a function of a sum of individual information matrices, where only one term changes as the camera parameters are perturbed one at a time. By careful bookkeeping of the information matrices

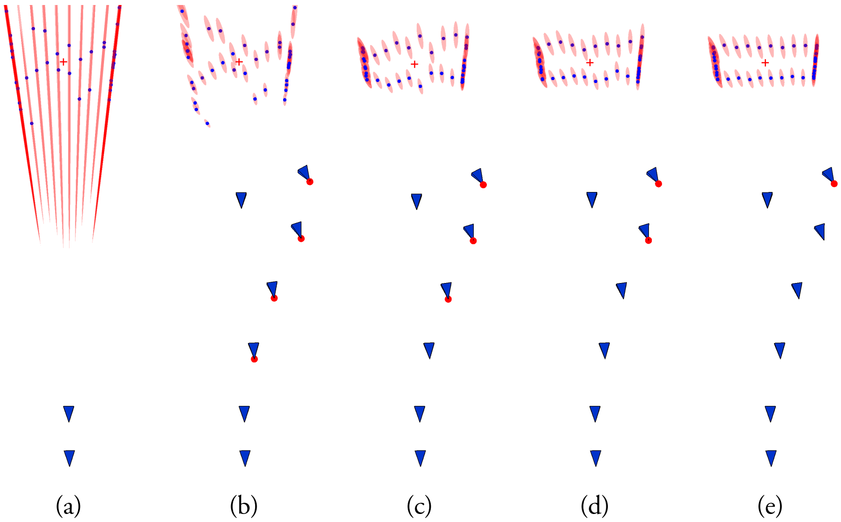


Figure 4.1: Doorway scenario. The robot wishes to approach the passage while determining its geometry as accurately as possible. The first two cameras on the path represent the last two images the robot has acquired and provide the initial optimal triangulation of the geometry. Red dots indicate which cameras are free to move, the red cross is the point of interest. In this case subsequent observations do not visibly change the initially planned path. The uncertainty ellipsoids represent 5σ in (a) and 50σ in (b)-(e). Note that in the latter cases the *expected* uncertainties, given all observations along the path, are displayed. The values $q = 3$ and $\alpha = 4.5 \cdot 10^{-7}$ were used.

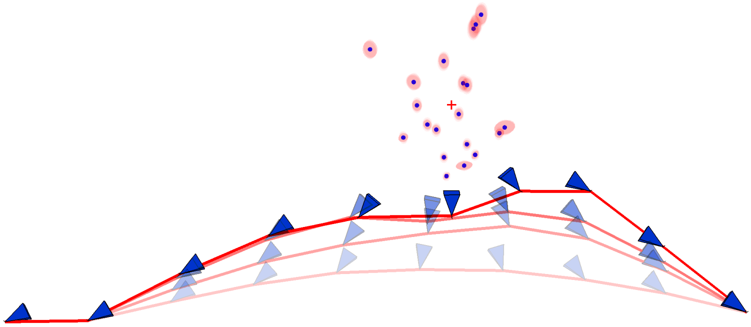


Figure 4.2: Here the robot passes (from left to right) by a point cloud and makes a detour to get as close to the features as possible; this is natural, since the closer the feature, the higher its angular resolution. Four cases are plotted, fading out with increasing values of α.

	Optimized path		Straight path	
α	Rel. err.	Rec. err.	Rel. err	Rec. err
$1.0 \cdot 10^{-7}$	$1.64 \cdot 10^{-3}$	$8.32 \cdot 10^{-4}$	$2.02 \cdot 10^{-3}$	$1.03 \cdot 10^{-3}$
$0.5 \cdot 10^{-7}$	$1.25 \cdot 10^{-3}$	$7.15 \cdot 10^{-4}$	”	”
$0.2 \cdot 10^{-7}$	$5.36 \cdot 10^{-4}$	$4.53 \cdot 10^{-4}$	”	”

Table 4.1: Relative error $U(\boldsymbol{P}, \boldsymbol{X})/U(\boldsymbol{P}^{1:2}, \boldsymbol{X})$ and absolute reconstruction error $\frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{X}^i - \boldsymbol{X}^i_{\text{true}}\|$, where $\boldsymbol{X}^{1:N}_{\text{true}}$ is the ground truth structure being observed, computed for different values of α in the scenario of Figure 4.2. The relative error represents the expected decrease in uncertainty from the initial estimate given by the first two images, the reconstruction error the actual error after all observations have been made. As α is decreased, the optimized path deviates more from the straight line between the first and last camera position, and the reconstruction error is decreased.

only four instances need to be computed for each camera instead of all $3(M-2)+1$ of a naïve implementation. This lowers the complexity of computing the Jacobian from $\mathcal{O}(M^2N)$ to $\mathcal{O}(MN)$. Nevertheless, in real-time applications computing the path should take a few seconds at most, and recent SLAM systems track hundreds or thousands of features. It may therefore be necessary to restrict attention to a subset of reconstructed features, e.g. those with the largest uncertainty, when evaluating the cost.

Furthermore, due to the iterative nature of the optimization, the path computation may be aborted before convergence but still yield a good approximation, depending on available time and computational resources.

4.6.2 Extensions

The assumptions in Section 4 can of course be relaxed. If an initial ML structure estimate is not available, we can either choose to ignore any prior information and initialize the algorithm using optimal triangulation from the most recent images, or simply substitute a non-ML estimate (e.g. from an EKF). If the estimate is good enough, the inverse of the covariance matrix will still be a good approximation to the Fisher information. Even if it's a poor approximation we would expect the optimized paths to yield better reconstruction accuracy than a straight or random one.

The requirement that the camera be oriented toward a particular point is only intended to reduce the dimension of the parameter space. Optimization over the orientations, or other rules for selecting orientation based on camera position and estimated structure could easily be incorporated.

It is also assumed that the camera position and orientation are known to high accuracy when acquiring images. Obviously, this is rarely true in a practical SLAM system, where there may be considerable uncertainty in the robot location. However, the location is usually well-determined relative to nearby, recently observed features, so for short-term local path planning this is a fair approximation. Nevertheless, incorporating the camera uncertainty in the covariance estimation would be straightforward, but would also introduce correlations between features. The information and covariance matrices would no longer be block diagonal, raising the computational load considerably, and the cost function would possibly have to be modified to include the camera location uncertainty. The practical gain of including such information is less clear.

The nature of the optimization scheme makes it easy to incorporate different constraints. For example, in the basic formulation (4.8) points behind the camera contribute the same information as if they were in the corresponding position in front of the camera, resulting in a physically incorrect model of

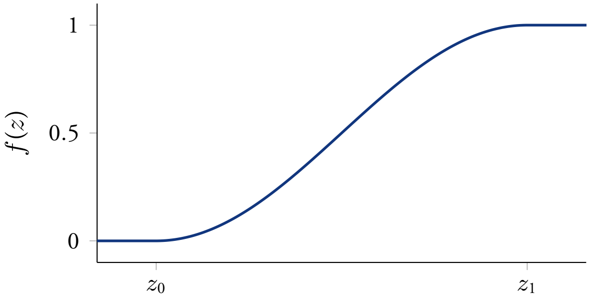


Figure 4.3: Example of a differentiable weight function, $f(z) = \max(s(z)\Theta(z_0), \Theta(z_1))$ where $s(z) = \sin\big((2(z - z_0)/(z_1 - z_0) - 1)\pi) + 1\big)/2$.

image acquisition. This can be rectified by weighting the information gain from each observation by a function of the point's depth. A suitable function can be seen in Figure 4.3, which smoothly diminishes the influence of points as they come too close to the camera. Similar weighting must also be employed to encourage the camera to keep the scene structure in its limited field of view, where points near or outside the image borders are downweighted. The effects of this weighting can be seen in Figure 4.2, where the path taken for small values of α would otherwise have passed right through the point cloud.

It is also possible to include penalty constraints on the path curvature, and obstacles in the robot's path can be modeled as a potential field added to the cost function, see Figure 4.4.

4.7 Conclusion

This chapter has presented a continuous optimization approach to certain instances of the next best view planning problem, aimed toward application in SLAM systems. Unlike previous algorithms the next best view is chosen with consideration of several expected future observations. Experiments show that reconstruction accuracy is improved, at a computational cost linear in the number of cameras and features. Nevertheless, the algorithm only produces locally optimal solutions, and relies on the cost functions being smooth. In the next chapter, we treat the same problem using a discrete optimization framework in order to overcome some of these difficulties.

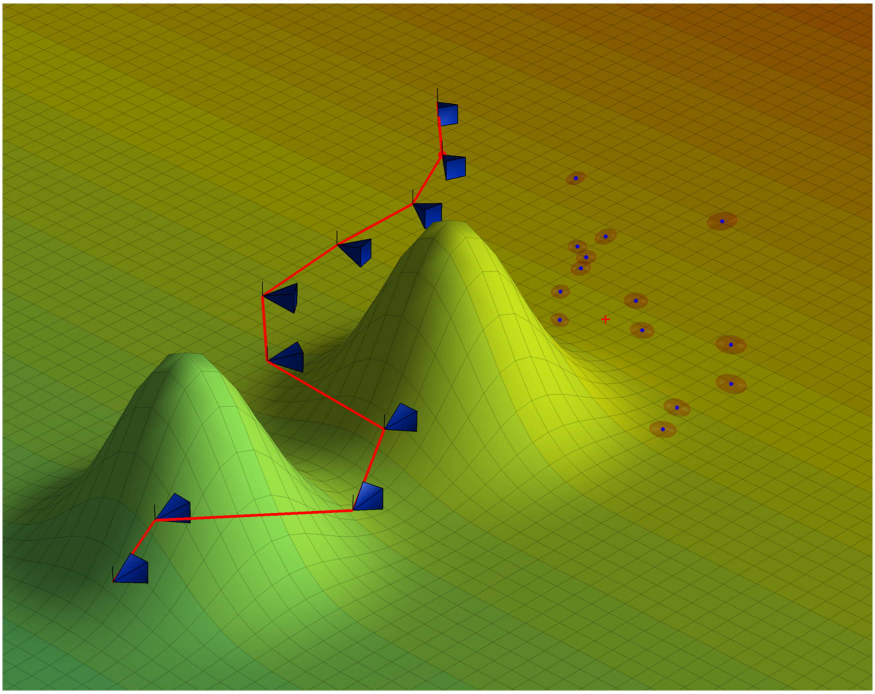


Figure 4.4: Obstacle avoidance may be accomplished by adding a smooth potential to the cost function, here illustrated as a height field.

Chapter 5

View Planning using Discrete Optimization

In this chapter, we formulate discrete analogs of the continuous planning problem formulation of the previous chapter and cast them as integer semidefinite programs (SDPs). The relaxations to continuous SDPs may be used in a branch-and-bound scheme to find optimal solutions, or as input to a stochastic optimization algorithm proposed below in Section 5.3.

Related discrete approaches include Englot and Hover (2010) where a shortest path linear program formulation similar to this work is used, but only considers view coverage and not uncertainty. Hollinger et al. (2012) uses a two-stage approach where good views are selected based on uncertainty, and then connected by solving a traveling salesman problem.

In A. Singh et al. (2009); Golovin and Krause (2010) approximation algorithms for the constrained path problem using greedy strategies are shown to provide solutions within a constant factor of the optimum, given that the underlying cost function is submodular. Unfortunately, the maximum eigenvalue metric proposed below is not submodular and such guarantees cannot be given; however, an optimality gap can always be computed.

5.1 Problem Description

Assume that the goal of a moving sensor is to reach a predefined target destination, while simultaneously reconstructing its surroundings as accurately as possible, based on observations taken along the path to the destination. There is a trade-off between reaching the destination quickly, and reducing the reconstruction uncertainty; for a bearing-only sensor such as a camera, a longer path can accommodate more observations with greater parallax, thus improving triangulation accuracy. Given a trade-off preference, or a bound on the path length, an optimal path can be found by solving a discrete optimization problem. The space between and around the start and destination positions is discretized into a finite number of possible sensor positions, and these positions constitute the nodes of a graph. The edges of the graph encode a neighborhood connectivity, i.e. the possible motions between the fixed positions. Thus a path in the graph corresponds to a physical path. With each node is also associated an information matrix encoding how much information about the environment we can expect to gain, if performing a measurement at that node's location.

The problem formulations given are agnostic to the graph geometry and topology, and to how the information matrices are generated. Thus there are no restrictions such as continuity or smoothness on the function used to evaluate the information content of a proposed sensor configuration, but which are typically required by continuous optimization approaches. Furthermore, the information of each view can be computed in parallel to leverage modern multi-core processors and GPU:s.

5.2 Problem Formulation

Define a directed graph $G = (V, E)$ and a set of positive semi-definite information matrices $\{\boldsymbol{I}_i \in S^n_+ \mid i = 0, \dots, |V|\}$. For a given trade-off parameter λ, define the optimization problem

$$\min_{p \in P} \quad \text{length}(p) + \frac{1}{\lambda} F\Big(\big(\boldsymbol{I}_0 + \sum_{i \in p} \boldsymbol{I}_i\big)^{-1} \Big), \tag{P1}$$

where P is the set of all simple paths in G from the source node to the destination node. $\boldsymbol{I}_0$ is the initially available information matrix of the environment structure, and $\boldsymbol{I}_i$ the expected information to be gained at node i. The inverse of the information matrix is the covariance matrix of the reconstructed structure, so the second term measures the variance using the scalarizing function F. This function is typically the trace or maximum eigenvalue, as discussed in Section 4.3. For these choices of F, we note that the second term is always decreasing as a function of the number of nodes on the path (this follows from Theorem 3.6). We now make two observations: if λ is large enough, the problem is equivalent to finding the shortest path through the graph, and may be solved efficiently using standard algorithms. If λ is sufficiently small, it is optimal to include all (non-zero) information matrices in the sum, while still minimizing the distance traveled, thus the problem is equivalent to the traveling salesman problem (TSP). Since TSP is known to be *NP*-hard, an efficient exact algorithm for the general case is out of reach. Also, the recognition version of TSP (*"Is there a tour of length less than L?"*) is *NP*-complete, so we should not expect even to be able to verify if a given solution is optimal. This is true even for graphs with nodes of degree at most four, e.g. planar grids (Papadimitriou and Steiglitz 1998).

In the above formulation (P1) the parameter λ is used to control the trade-off between a short path and a more accurate reconstruction of the surroundings. It is however not obvious how to select this parameter, or even its suitable range, without some trial-and-error. In fact, another problem formulation may be more natural: given a time or distance budget, what is the best reconstruction obtainable? In other words, given an upper bound on the length of the path traveled, minimize the reconstruction error, i.e.

$$\begin{aligned} \text{min.} \quad & F\Big(\big(\boldsymbol{I}_0 + \sum_{i \in p} \boldsymbol{I}_i\big)^{-1}\Big) \\ \text{s.t.} \quad & \text{length}(p) \leq L\,. \end{aligned} \tag{P2}$$

Note that with this formulation, as the allowed path length grows we no longer approach TSP. Instead, for L large enough, any Hamiltonian path on the graph will be optimal, and for the types of graphs considered here, these are usually easily generated. Unfortunately, the problem still appears difficult for length limits of practical interest. There are several other variations on the problem formulation, for example one could minimize the path length under the constraint that the covariance is reduced by a certain amount. However, all of them appear equally hard to solve.

Below, we give convex relaxations of (P1) and (P2) and show how these can be used to solve the original problems in a branch-and-bound scheme, or more practically as guides for more local optimization methods. The convex relaxation and optimization methods presented are easily adapted to alternative problem formulations.

5.2.1 Shortest Path as a Linear Program

The problem of finding the shortest path between two nodes in a graph with positive edge weights is often solved using Dijkstra's algorithm. It can be shown that this algorithm is equivalent to applying a primal-dual solver to the following linear program (Papadimitriou and Steiglitz 1998):

$$\begin{aligned}
\text{min.} \quad & \sum_{(i,j)\in E} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j\colon (i,j)\in E} x_{ij} = \sum_{j\colon (i,j)\in E} x_{ji}, \quad i \in \{1,\dots,|V| \setminus s, t\} \\
& \sum_{j\colon (s,j)\in E} x_{sj} = 1, \qquad \sum_{j\colon (j,t)\in E} x_{jt} = 1 \\
& 0 \le x_{ij} \le 1\,.
\end{aligned} \tag{LP}$$

Here x_{ij} is a variable indicating if the edge between node i and j is part of the path or not, and c_{ij} the associated non-negative edge weight. The constraints express flow conservation, so that the number of edges incident on a node equal the number exiting, except for the source (s) and terminal (t) nodes which have one outgoing and one incident edge respectively. These constraints can be summarized into $\boldsymbol{A}_G\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A}_G$ is the $|V|$-by-$|E|$ *edge incidence matrix* of G with entries

$$A_{ij} = \begin{cases} -1 \text{ if edge } j \text{ leaves node } i \\ +1 \text{ if edge } j \text{ enters node } i \\ \;\;0 \text{ otherwise} \end{cases}, \tag{5.1}$$

and $\boldsymbol{x}$ are the edge indicator variables suitably stacked. It is easily shown that (LP) must have an integer optimal solution. Note that this formulation does not explicitly forbid solutions consisting of a path between the source and terminal, plus any number of closed loops; these are only eliminated by virtue of not being optimal.

5.2.2 View Planning as a Semidefinite Program

We adapt the shortest path problem formulation above to the planning problem (P1). For convenience, introduce binary variables α_i for each node of the graph, indicating whether that node is on the path or not. We form the relaxed optimization problem

$$\begin{aligned}
\text{min.} \quad & \sum_{(i,j)\in E} c_{ij} x_{ij} + \frac{1}{\lambda} F\Big(\big(\boldsymbol{I}_0 + \sum_{i=1}^{|V|} \alpha_i \boldsymbol{I}_i \big)^{-1} \Big) \\
\text{s.t.} \quad & \boldsymbol{A}_G \boldsymbol{x} = \boldsymbol{b} \\
& \alpha_i = \sum_{j\colon\, (j,i)\in E} x_{ji}, \quad i \neq s \\
& \alpha_s = 1 \\
& 0 \le \alpha_i \le 1\,,
\end{aligned} \tag{P3}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{x}$ are not required to be binary. The cost functions used in next best view planning are generally non-smooth and multi-modal, and difficult to optimize. However, due to the discretization, the argument to the second term of the objective above is affine in $\boldsymbol{\alpha}$. Both the trace-of-inverse and maximum eigenvalue-of-inverse functions are convex, and using the epigraph trick the second term may be formulated as a convex semidefinite constraint (see e.g. Boyd and Vandenberghe (2004) or the chapter appendix). As one would expect, this semidefinite program no longer has all the desirable properties of the linear program; integrality of $\boldsymbol{x}$ or $\boldsymbol{\alpha}$ is no longer guaranteed, and a solution with disjoint loops may in fact be optimal.

The corresponding convex relaxation of (P2) is the same as (P3), except that the first term of the objective is transformed into a linear inequality:

$$\begin{aligned}
\text{min.} \quad & F\Big(\big(\boldsymbol{I}_0 + \sum_{i=1}^{|V|} \alpha_i \boldsymbol{I}_i \big)^{-1} \Big) \\
\text{s.t.} \quad & \boldsymbol{A}_G \boldsymbol{x} = \boldsymbol{b} \\
& \sum_{(i,j)\in E} c_{ij} x_{ij} \le L \\
& \alpha_i = \sum_{j\colon\, (j,i)\in E} x_{ji}, \quad i \neq s \\
& \alpha_s = 1 \\
& 0 \le \alpha_i \le 1\,.
\end{aligned} \tag{P4}$$

Selecting L is more intuitive than choosing λ; one must only be careful not to produce an infeasible problem by selecting L too low, but the lower limit is readily obtained using a standard shortest path algorithm.

It is possible to find an optimal integer solution to (P3) or (P4) using a standard branch-and-bound search, but this is also known to be *NP*-hard and may take a large number of iterations, each involving solving a potentially quite large SDP. If a solution is found, it may also contain unwanted disjoint loops. While it is easy to include linear constraints forbidding any *particular* loop in the SDP, since there are exponentially many possible loops in the graph, adding constraints against them all at the outset is infeasible. But, they can be added on an as-needed basis; if loops are present in the solution, add constraints against them and solve again until no loops remain. As it turns out, 2-cycles are quite common in the solutions, and as their number is typically linear in the number of nodes, it is feasible to remove them at the outset which may lead to faster convergence to a loop-free solution.

Obviously, the above procedure may be very time consuming or completely intractable for all but the smallest problem instances. However, we also noted above that depending on the trade-off parameter λ, the original problem (P1) should vary in difficulty between simple shortest path (typically $\mathcal{O}(|E| \log|V|)$ for Dijkstra's) up to exponential complexity. Empirically, it turns out that many instances are in fact "easy", in that very few steps of branch-and-bound are required and few or no loops are included in the solution. Yet, many other instances are indeed difficult and not amenable to this approach.

5.3 Approximate Solution

Despite the problems of tractability in finding optimal solutions described above, it can be noted that the relaxed SDP formulations (P3) and (P4) provide lower bounds on the optimal objective values of (P1) and (P2). This may be used to verify the performance of approximation algorithms. Also, if the problem instance at hand is "easy enough", the relaxed solution $\boldsymbol{x}^*$ may be quite close to a valid integer, loop-free solution. In these cases it is possible to construct a valid solution to (P1) using a simple shortest path search on the graph G with edge weights $c_{ij} = 1 - x^*_{ij}$. This solution may be good enough, or can serve as initialization for local or stochastic optimization algorithms.

To obtain a feasible solution to (P2) from a fractional SDP solution $\boldsymbol{x}^*$ of (P4), this method cannot be used as the path is not guaranteed to be shorter than L. Instead, we solve the integer linear program

$$\begin{aligned} \text{max.} \quad & \sum_{(i,j)\in E} x^*_{ij} x_{ij} \\ \text{s.t.} \quad & \boldsymbol{A}_G \boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{A}^+_G \boldsymbol{x} \leq \mathbf{1} \\ & \sum_{(i,j)\in E} c_{ij} x_{ij} \leq L \\ & x_{ij} \in \{0, 1\}, \end{aligned} \tag{5.2}$$

where $\boldsymbol{A}^+_G = \max(\boldsymbol{A}_G, 0)$, so that the second constraint ensures that at most one edge is incident on every node. This heuristic attempts to cover as much of the weight of the SDP solution as possible, and is guaranteed to return a feasible path. However, as with the SDP, the solution may also contain disjoint loops. In cases where the bulk of the SDP solution is covered by these unwanted loops, the returned path is likely to be a poor solution to the original problem. In this case, loop constraints can again be added and the program solved again as described in Section 5.2.2. The ILP (5.2) does not depend on the scene information $\boldsymbol{I}_i$ and can benefit from specialized branch-and-bound solvers, thus it is typically orders of magnitude faster to solve than (P4).

5.3.1 A Simplified Formulation

Given the hardness of (P1) and (P2), it is natural to seek a simplified problem formulation which might admit faster solution algorithms. Part of the difficulty is the nonlinearity of the interaction between the information matrices when taking the inverse to obtain the covariance; the value of any particular contribution to the information depends on all the others. Forgoing this interaction, instead of minimizing the covariance, one can maximize the trace of the information matrix, yielding the problem

$$\min_{p\in P} \quad \text{length}(p) - \frac{1}{\lambda} \operatorname{tr}\Big(\boldsymbol{I}_0 + \sum_{i\in p} \boldsymbol{I}_i\Big). \tag{P5}$$

Since the trace is linear, this results in a shortest path problem on G with modified weights (subtract $\operatorname{tr}(\boldsymbol{I}_i)/\lambda$ from each edge incident on node i). As long as this does not result in any negative cycles, this may be efficiently solved using e.g. the Bellman-Ford algorithm, or even Dijkstra's if all weights are non-negative. If negative cycles are present, the problem again becomes much more difficult. In the extreme where all edge weights are negative, the problem is equivalent to the longest simple path problem on $-G$, which is known to be *NP*-hard (Schrijver 2004). Unfortunately, for many scenarios and reasonable choices of λ negative cycles will be present, and in these cases (P5) can be formulated as (LP) but with binary constraints on $\boldsymbol{x}$. While this ILP may be significantly faster to solve than (P3), the complexity is the same and no-loop constraints must also be introduced incrementally. However, in some scenarios it may be reasonable to restrict the graph G to be acyclic, and then the shortest path problem can always be solved in linear time. A general graph may be reduced to a directed acyclic graph by ordering the nodes by decreasing distance to the target node, and only keeping edges reducing the distance, thus forcing the sensor to move monotonically towards the destination. This will of course not work for purely exploratory scenarios where the start and end points may be near.

Even with this significantly simplified formulation sacrificing the interdependence of measurements, the problem is still not easy in general. We therefore introduce a stochastic genetic algorithm applicable to all problem formulations.

5.3.2 A Genetic Algorithm

Genetic algorithms (GA) are a class of evolutionary optimization algorithms which emulate the process of natural selection. A population of candidate solutions is maintained, and in every iteration of the algorithm, a new population is generated by mutation and crossings of individuals of the previous generation. The chance of an individual producing offspring in the next generation is proportional to that individual's fitness, calculated from the corresponding value of the function being minimized. Genetic algorithms have been found to be quite efficient in providing good solutions to many combinatorial optimization problems, including TSP (Choi et al. 2003; Schmitt and Amini 1998) and path planning (Davoodi et al. 2013), which motivates the use here.

To use a genetic algorithm, one must choose a representation for a candidate solution, and define unary mutation and binary crossover operators. In this work, each individual is described simply as a sequence of vertices constituting the path (the encoding is usually called a *chromosome*, in keeping with the evolutionary theme). Algorithm 1 shows the basic operation of the proposed genetic algorithm. The different steps and operators used are described below.

Algorithm 1: Proposed genetic path planning algorithm

```
Initialize a population $P$
$B \leftarrow -\infty$
best $\leftarrow \emptyset$
for $iter \leftarrow 1$ to $maxiter$ do
    for $i \in P$ do
        $f_i \leftarrow$ evaluateFitness($i$)
    end
    if $max(f_i) > B$ then
        $B \leftarrow \max(f_i)$
        best $\leftarrow \arg\max(f_i)$
    end
    $\bar{P} \leftarrow \emptyset$
    for $n \leftarrow 1$ to $|P|$ do
        With probability $\propto f_j, f_k$, select individuals $j, k \in P$
        $i \leftarrow j$
        With probability $p_{\text{cross}}$, $i \leftarrow$ crossChromosomes($j$,$k$)
        With probability $p^1_{\text{mutate}}$, $i \leftarrow$ mutateChromosome($i$)
        ...
        With probability $p^m_{\text{mutate}}$, $i \leftarrow$ mutateChromosome($i$)
        $\bar{P} \leftarrow \bar{P} \cup \{i\}$
    end
    best $\leftarrow$ locallyRefine(best)
    $P \leftarrow \bar{P} \cup \{\text{best}\}$
    if no change in $B$ for $K$ iterations then
        break
    end
end
return best
```

Initialization The first step is to generate candidate solutions, in this case paths in G from the source to the terminal node. Unless we have some a priori information on the characteristics of the optimal solution, these should be spread out uniformly across the space of all feasible paths. Unfortunately, truly uniform sampling of simple paths on a general graph appears to be a difficult problem. Reasonably random paths, however, may be obtained using random order depth-first search (DFS), *loop-erased random walk* (Lawler 1980), or for undirected graphs by computing the minimum cost spanning tree with randomized edge weights, and extracting the unique path in the tree. Starting at the source, loop-erased random walk implies walking on the graph, choosing every edge with equal probability until the terminal is reached. Any loops created along the way are then cut out to form a simple path.

If candidate solutions have been obtained using any of the heuristic methods based on the SDP relaxation, these can be included in the initial population and will then be refined.

Mutation Operators A mutation operator should introduce "noise" or randomness into an existing chromosome, while preserving the main features of the encoded path. In practice, several mutation operators are often employed, exploiting problem-specific heuristics. To modify a path, just randomly replacing vertices is not possible, since not every sequence of vertices is a valid path in the graph G. Instead, our first operator selects two random cut points along the path, and replaces the path in between with a random one generated using either randomized DFS or loop-erased random walk. A second operator instead replaces the section with the shortest path between the cut points. This is motivated by the fact that optimal paths are often quite regular, so it makes sense to smooth out kinks. Both these operators are comparatively slow, so we also use a much faster but more local operator which simply selects a random vertex on the path, and replaces it with one picked from the intersection of nodes reachable from the preceding node with those with outgoing edges incident on the next node on the path.

Crossover Operator The crossover operator takes two existing paths as input and produces a mixed path, containing parts of both, assuming they cross at some point. This is accomplished by selecting a random simple path on the graph obtained from G consisting only of the edges on the two paths. See Figure 5.1 for an illustration.

Local refinement To speed up convergence to a locally optimal solution, chromosomes may be optimized by systematically applying the fast local mutation operator described above in a deterministic fashion. Each node on the path,

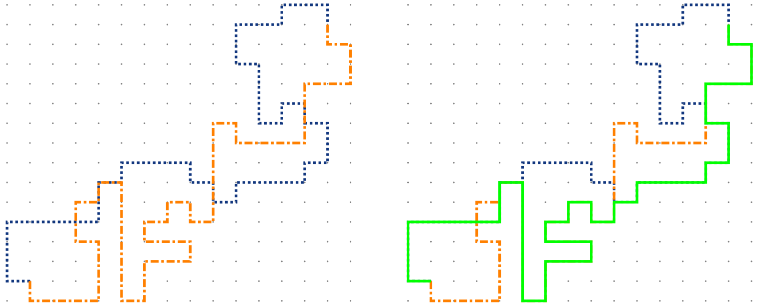


Figure 5.1: Illustration of the crossover operator. At each intersection either path is chosen with equal probability.

visited in random order, is replaced with the neighbor which minimizes the objective function.

With the formulation (P2), we run the risk of generating infeasible paths in the course of the genetic algorithm. A simple solution is to reject any infeasible path obtained and repeat the mutation or crossover operation until a feasible realization is produced. It is easy to verify that if the inputs are feasible, the operators defined above will eventually produce feasible output. However, depending on how close L is to the lower bound of feasibility, this may take an unreasonable amount of time. The very simplest solution is to relax the length constraint to a barrier penalty term in the objective, e.g.

$$\text{min.} \quad F\Big(\big(\boldsymbol{I}_0 + \sum_{i \in p} \boldsymbol{I}_i\big)^{-1}\Big) + C \max\big(0, \text{length}(p) - L\big)^2, \tag{5.3}$$

where C is a suitably large constant. Since the genetic algorithm does not require smoothness the barrier could simply be defined as the indicator function of the feasible set, but it is advantageous to allow infeasible individuals in the population as it provides more diversity.

5.4 Stratified Solution Strategy

The genetic algorithm will quickly find good solutions if the search space is not too large. For large grids with many hundreds of nodes and large neighborhood connectivities, the algorithm risks getting stuck in local optima, often producing implausible-looking paths. We therefore propose to reduce the search space by substituting a smaller graph, based on the solution of the SDP relaxation of the problem, to obtain a good initialization which can then be refined on the full graph.

5.4.1 Reducing the Graph

The idea is to keep only a subset of the N most important nodes, as indicated by the fractional SDP solution $\boldsymbol{\alpha}^*$. Interpreting these values as probabilities, we draw N nodes without replacement, selecting nodes in proportion to their $\boldsymbol{\alpha}^*$-value. The reduction in the covariance achieved using only these nodes is computed and maximized through repeated random sampling of the subset.

Once a subset has been chosen, a new fully connected graph is formed, where edges between the nodes represent the shortest path between them in the original graph. This allows the mapping of paths on the reduced graph to the full graph where they can be evaluated. The genetic algorithm can now be run without modification on the reduced graph, where the parameter N can be chosen to trade fidelity for convergence speed.

5.5 Experiments

Due to the general formulation of the basic problem, many different scenarios can be accommodated by adapting the graph G and edge weights c_{ij}, which do not need to fulfill geometric constraints such as the triangle inequality. For example, each node can represent a camera position and an orientation, and the connectivity between poses can be defined so as to constrain angular velocity on the path. Purely exploratory behavior can be achieved by selecting start and/or destination nodes as "super-nodes" connected to every other node with zero weight, thus effectively permitting arbitrary start and destination points. Typically, we know less about the scene further away from the starting point, so the predictions of what will be seen, or what obstacles lay ahead, may be incorrect. Therefore one should plan with caution; by weighting the information matrices based on distance to the starting node, such behavior can be incorporated.

In the synthetic experiments below, each node (except super-nodes) has an associated camera pose defining position and orientation. The environment structure is represented by 3D points, each considered independently estimated such that the initial information matrix $\boldsymbol{I}_0$ is block diagonal (in fact we let it be a multiple of the identity matrix). To compute the information gained from acquiring an image at a certain pose, the standard pinhole camera model is used; let the projection $\hat{\boldsymbol{x}}$ of $\boldsymbol{X}$ be given by $\hat{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{P}, \boldsymbol{X})$. Given a point $\bar{\boldsymbol{X}}$, the corresponding block of $\boldsymbol{I}_i$ is computed as $\boldsymbol{J}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{J}$, where $\boldsymbol{J} = \frac{d\boldsymbol{f}}{d\boldsymbol{X}}\Big|_{(\boldsymbol{P}_i, \bar{\boldsymbol{X}})}$ is the projection Jacobian and $\boldsymbol{\Sigma}$ the assumed measurement error covariance on the image plane. However, if $\bar{\boldsymbol{X}}$ is out of the camera's field of view or too far away, the block is set to zero. In Figures 5.2–5.9 different experiments are shown; the setups and results are described in the figure captions for easier reference.

5.5.1 Practical Considerations

The choice of scalarizing function F can have a large impact on the solution time of the SDP, depending on the dimension of the information matrices. To minimize the trace of the covariance matrix, one variable per eigenvalue is required, while the maximum eigenvalue cost only needs one. On the other hand, evaluating the trace cost function is typically faster. Furthermore, the maximum eigenvalue is vulnerable to outliers e.g. features which are not seen in any or too few views. If such features are not removed in a preprocessing step, the cost function can never decrease below the initial uncertainty.

The algorithms were implemented in Matlab with core functions in MEX C++. SDPs were set up using YALMIP (Löfberg 2004) and solved using the MOSEK interior-point optimizer (Dahl 2012). For the experiment in Figure 5.6, solving the SDP with the trace cost took 8.7 s as opposed to 4.2 s for the maximum eigenvalue, while the genetic algorithm (on the full problem) runs at about 10 iterations per second on the same Core 2 Duo 3.0 GHz computer, with a population of 60 individuals. With parallel processing of individuals, speed can likely be increased manyfold.

5.5.2 Tighter Relaxations

The convex relaxations proposed simply replace the binary constraints $\boldsymbol{x} \in \{0, 1\}^n$ by $\boldsymbol{x} \in [0, 1]^n$. Theoretically, the best relaxation possible would be the convex hull of the non-convex feasible set, and it is possible to approach this ideal using additional semidefinite constraints. Introducing $\boldsymbol{y} = 2\boldsymbol{x} - 1$ and a symmetric matrix $\boldsymbol{H} \in S^n$ of *lifting variables*, we may replace the binary

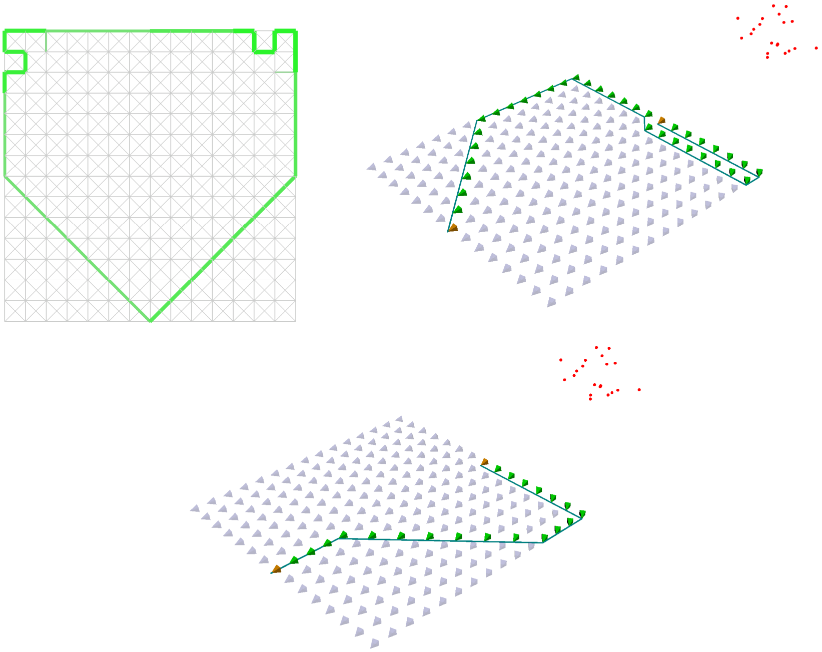


Figure 5.2: Cameras looking at a point cloud are placed on an 8-connected unit grid. On the left, the greenness and thickness of edges is proportional to x^*_{ij}, the corresponding variable of the solution to the relaxed SDP (P3), using the maximum eigenvalue scalarizing function and $\lambda = 2$. Due to the symmetry of the problem, both a left and right path seem to be given equal consideration. The lower bound obtained is 61.1. On the right, the same problem with the path (green) found by the genetic algorithm among possible positions (blue), with a cost of 67.3. Below, the same problem but with $\lambda = 40$ and final cost 32.6, as compared to the lower bound of 23.0. Depending on problem characteristics, bounds may be more or less tight.

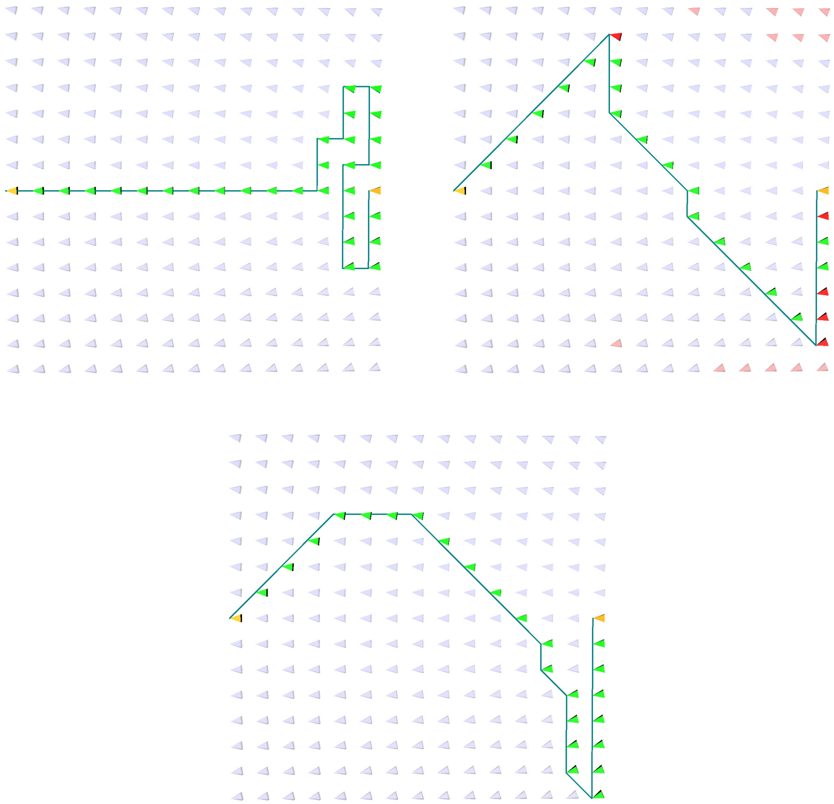


Figure 5.3: The same situation as in Figure 5.2, now solving problem (P2) with $L = 30$. On the left, the solution obtained by solving the analog of the simpler problem (P5), giving a cost of 92.3 compared to the SDP lower bound of 67.7. On the right, the red nodes indicate the reduced graph obtained by sampling the SDP solution with $N = 20$, and applying the GA gives a path with cost 88.7. Below, the GA run on the full graph with cost 86.36. The simplified formulation is qualitatively different from the others and focuses on getting as close to the structure as possible while neglecting the parallax effects. It is therefore not a suitable approximation in many situations.

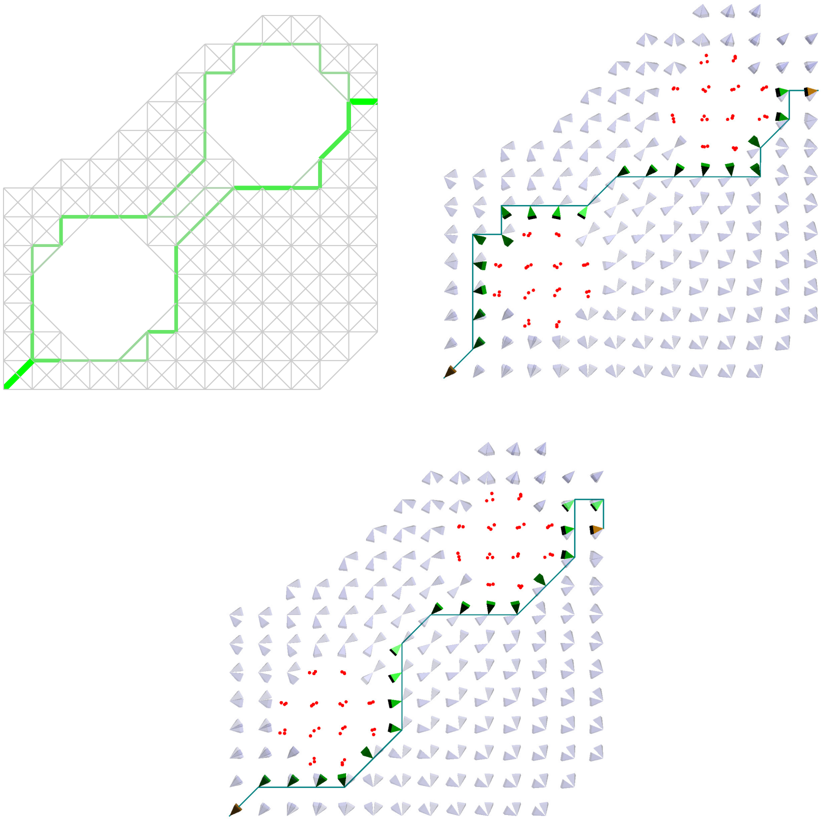


Figure 5.4: In this example, two graph nodes each are placed at every point of an 8-connected unit grid. The two nodes represent a camera looking at either of two objects/obstacles. On the left, the solution to the relaxed SDP (P4), using the trace scalarizing function and $L = 22$. Nodes of the original square grid whose combined shortest distance to the start and destination nodes is greater than L have been removed, since they cannot be part of a feasible solution. As in Figure 5.2, there appears to be two competing paths, with edge values $x^*_{ij} \approx 1/2$. On the right, the path obtained from the simplified formulation (P5), which works reasonably for this problem instance. Below, the solution obtained using the proposed genetic algorithm. The corresponding objective values are 192.1, 271.8 and 262.1. The gap between the final objective and the lower bound given by the SDP is relatively large, but the path obtained directly from the SDP solution is still quite reasonable.

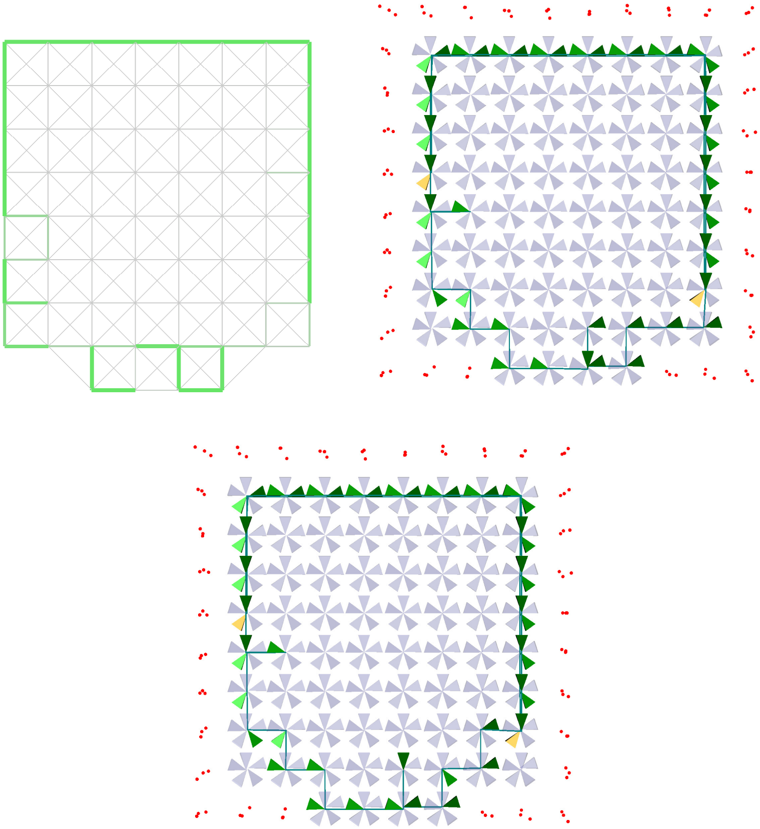


Figure 5.5: Here, a room is to be explored, with free start and end points. L was set to 50. At each spatial location there are five different orientations to choose from, and graph connectivity was defined so that angular velocity is limited to 72° per step. On the left, the relaxed SDP solution giving the lower bound 382.2. On the right, the path obtained by iterating (5.2) while removing loops, finally giving a path cost of 412.2. Below, the refined path obtained from the GA initialized with the path on the right, and with cost 399.6. Note that the room is circled twice with the camera in two different predominant orientations; this is most easily seen by viewing the embedded 3D models on-screen, where the different orientations have been spatially separated for illustration purposes. Interestingly, no diagonal edges are included in any of the solutions.

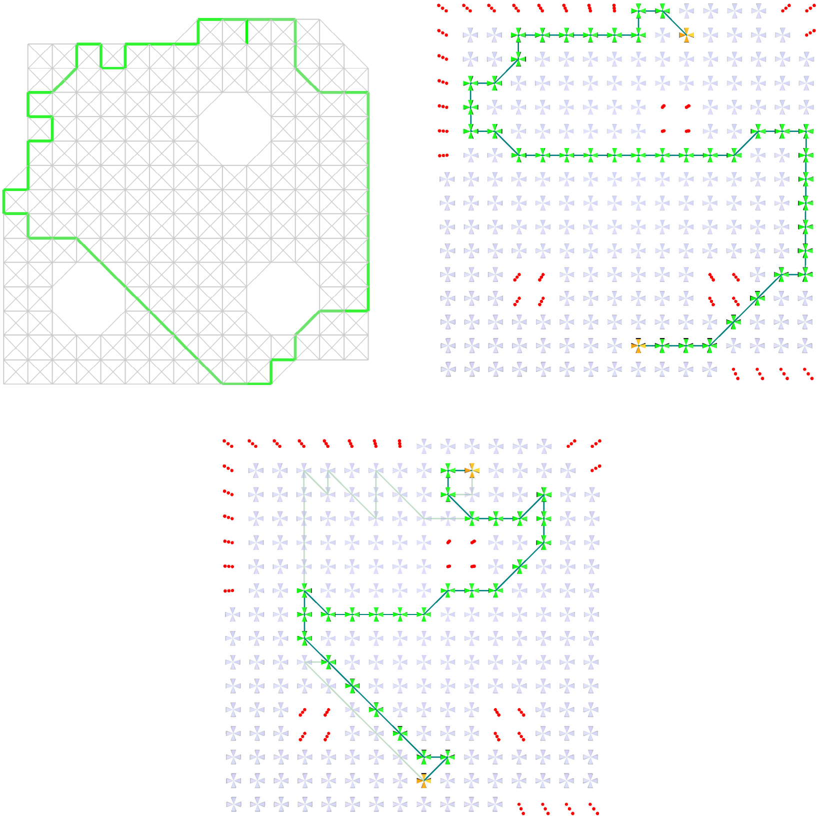


Figure 5.6: In this experiment, we simulate an omnidirectional camera by adding the information matrices of four cameras at each location. Each cluster thus corresponds to only one node of the graph. On the left, the SDP (P4) with $L = 43$ gives a lower bound at 4.11. On the right, the solution using the stratified approach, with cost 5.47. Below, the graph has been reduced to a DAG such that the observer must move towards the target at every step. This limits the search space making branch-and-bound tractable, and the optimal solution with cost 17.98 is shown next to the solution of the simplified problem (P5) (pale green) with cost 30.81.

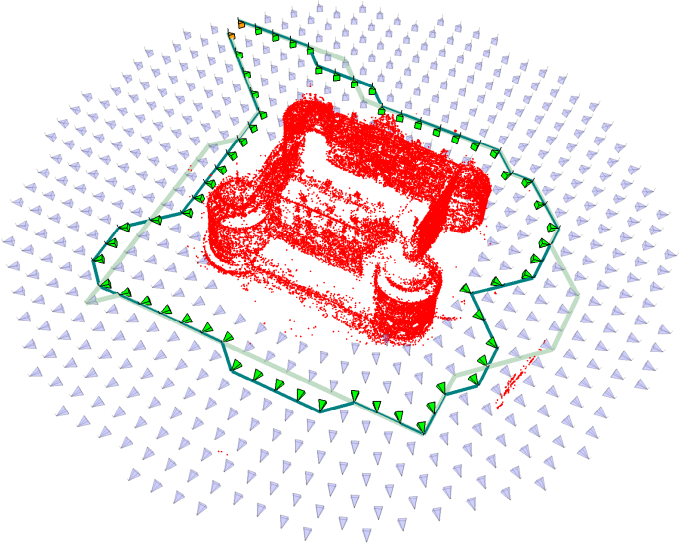


Figure 5.7: Stratified algorithm run on the Örebro castle dataset, reduced to one hundred representative points using random sampling.

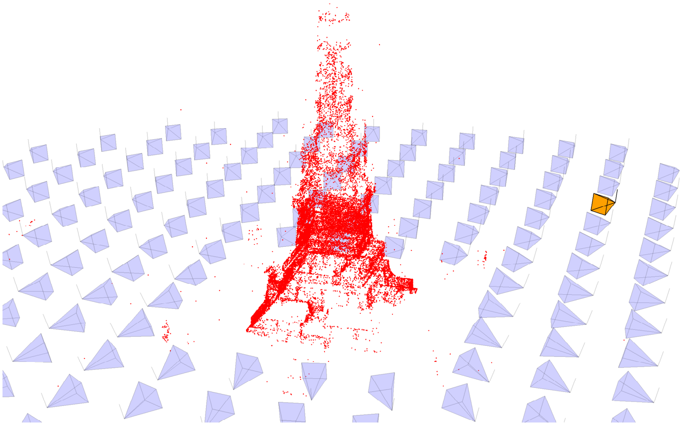


Figure 5.8: The Battle of Lund monument dataset and problem setup.

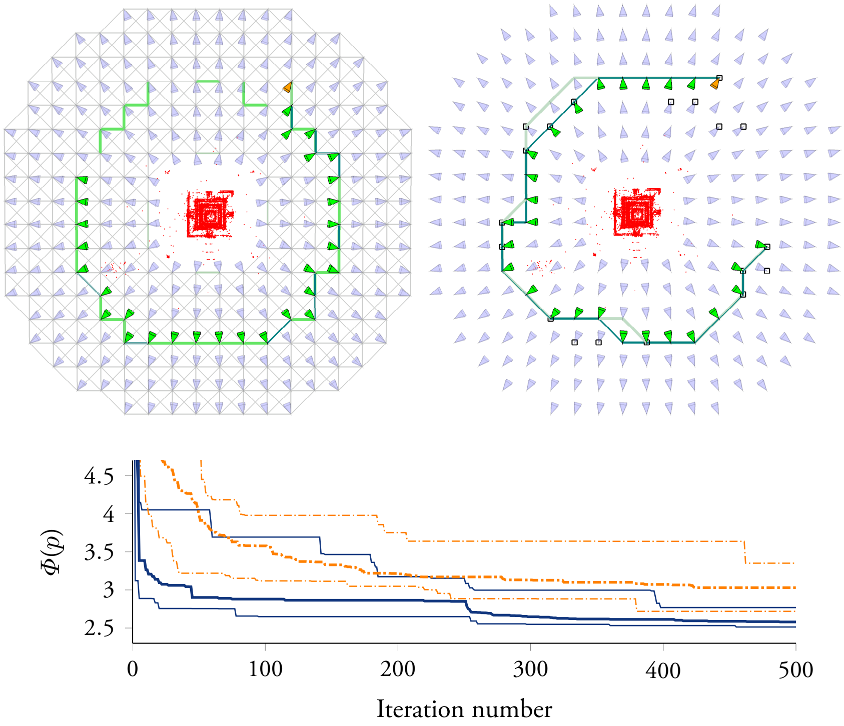


Figure 5.9: Exploratory scenario with fixed start and free end point. The point cloud of the Lund monument has been reduced to a few hundred representative points (by random sampling) to constrain the information matrix dimension. Top left, the SDP solution giving lower bound 1.76 along with the "filled in" path found as the shortest path on the graph with weights $1 - x^*$ (see Section 5.3), having cost 3.18. Right, the GA solution on the reduced graph obtained by sampling the SDP solution, with nodes marked with black squares, shown in light green with cost 2.78. The dark green path is the result of the GA on the full graph, seeded with the reduced solution, with cost 2.57. Achieving similar cost using random initialization takes significantly longer; the plot shows the progression of the objective value (Φ) over iterations of the proposed genetic algorithms. The orange dashed curves show the maximum, minimum and median over 20 runs of the GA on the full graph with random initialization. The green curves show the same for the stratified approach, first running 250 iterations on the reduced graph, then switching to refinement on the full graph. It is clear that the stratified scheme converges much faster.

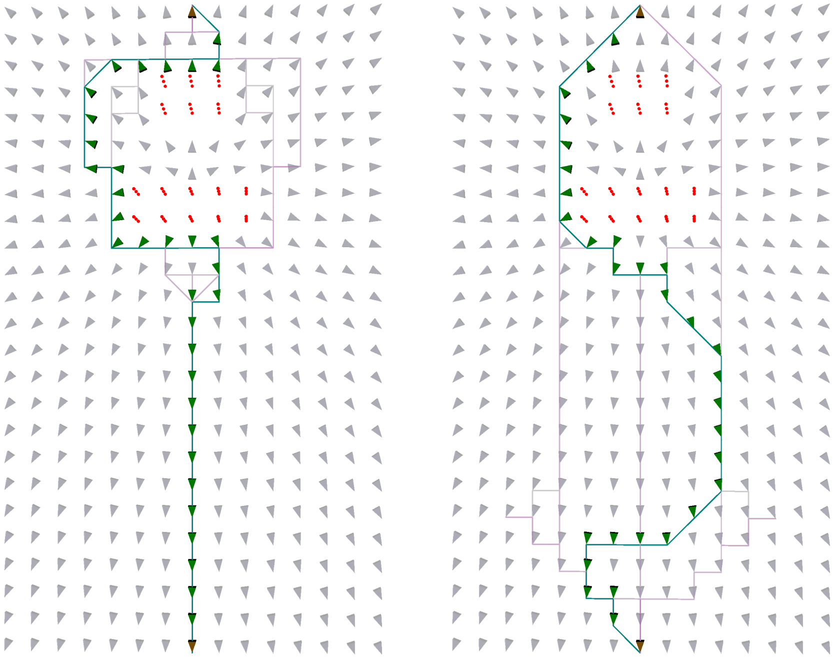


Figure 5.10: Effect of reduced confidence in future measurements. The plots show the same scenario but on the right the information matrix at each node has been down-weighted by the distance from the start node. The pale purple lines indicate the SDP solution, the green path the GA solution.

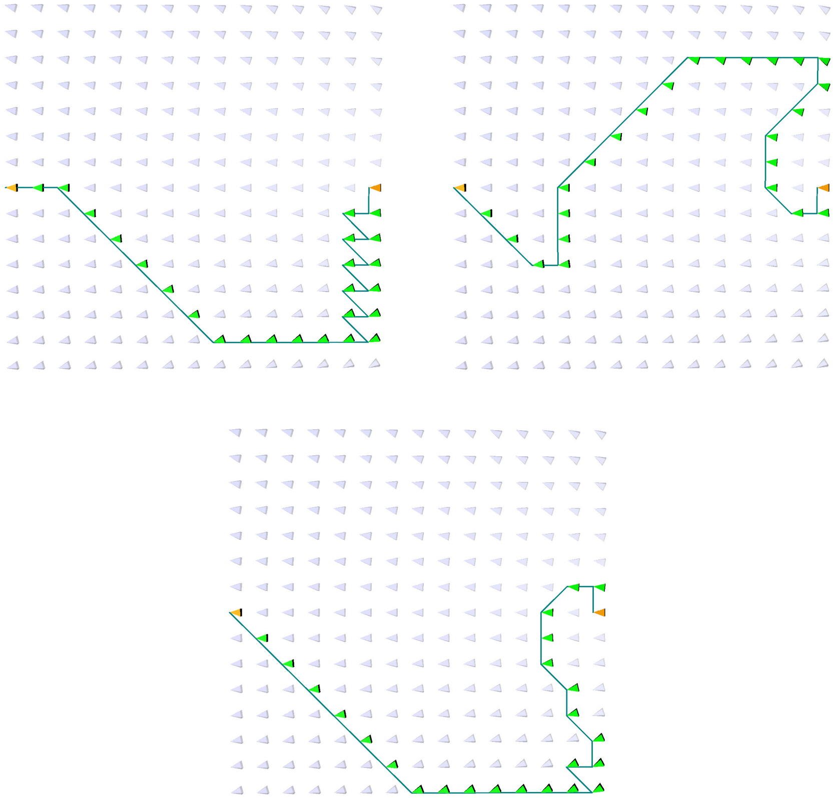


Figure 5.11: In this example the graph has been reduced to a DAG, only allowing movement strictly decreasing the distance to the target node. Left: the simplified formulation solution with cost 80.4. Right: GA solution after 200 iterations with cost 76.6. Below: the optimal solution obtained using branch-and-bound, with cost 69.6. The lower bound obtained from the relaxed SDP is only 46.9 in this case.

constraints of the original problem with

$$\begin{pmatrix} 1 & \boldsymbol{y}^\top \\ \boldsymbol{y} & \boldsymbol{H} \end{pmatrix} \succeq 0, \quad \operatorname{diag}(\boldsymbol{H}) = \boldsymbol{1}, \quad \operatorname{rank}(\boldsymbol{H}) = 1. \tag{5.4}$$

By removing the non-convex rank constraint we obtain Shor's semidefinite relaxation (Shor 1987). It has been shown that dropping the rank requirement is equivalent to Lagrangian relaxation, i.e. solving the dual of the original optimization problem. In theory, one could go further and apply even tighter relaxations from Lasserre's hierarchy (Lasserre 2000) of which Shor's is the first level. Unfortunately, the added computational complexity of even this first level with n^2 extra variables makes it impractical for all but the smallest problems. Crucially, in our experiments the tightening of the lower bound turns out to be insignificant, at the cost of a three orders of magnitude increase in running time.

5.5.3 Receding Horizon Control

In practice, the proposed problem formulations are foreseen to be solved in a "receding horizon" manner, where the destination is continually updated as new navigational objectives are received from higher-level planning functions. If the path endpoint moves frequently or unpredictably, trying to find the optimal path from start to finish may be pointless and wasteful. We therefore propose a more local planning model, where the path is only optimized over the next k number of steps, while still attempting to respect the longer-term path length constraints.

Given the robot's position, the graph is trimmed so that only nodes reachable within k steps, and the destination node, remain. New edges between the "horizon" nodes (at precisely k steps away) and the destination node are introduced with weights equal to the shortest path distance between them in the original graph. The information at the horizon nodes is set to the accumulated information expected along those shortest paths. The planning problem can then be solved on the new graph using the GA optimizer or optimally using branch-and-bound, if k is small enough (say ≤ 4 for reasonable running times). After moving to the last horizon node on the path, the traveled distance is subtracted from the length limit L and the process is repeated. This ensures that the total path length upon reaching the destination does not exceed the original limit, but also leads to the "parallax budget" being used up in the early

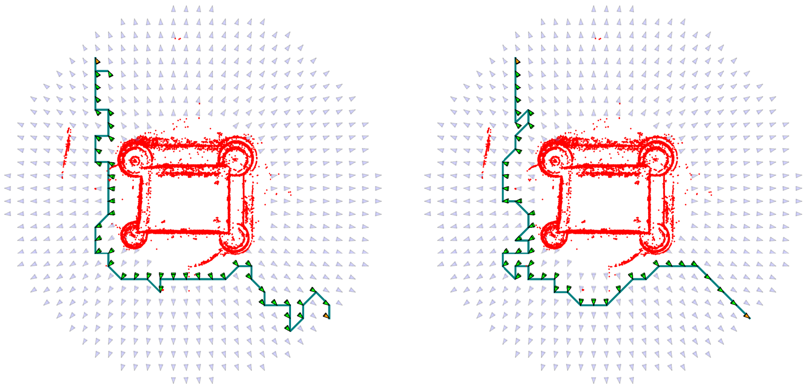


Figure 5.12: Örebro castle dataset with the trace scalarizing function and $L = 50$. On the left, the short-term planning ($k = 3$) solution using branch-and-bound with final objective value 262.0 and path length 49.3. On the right, the GA solution with objective value 253.9 and length 49.8. The SDP lower bound was 244.3.

stages, leaving only the shortest path solution available later on. This effect can be alleviated by augmenting the length limit somewhat at each stage. An example of this approach is shown in Figure 5.12.

5.6 Conclusions

While the general problems considered in this chapter are demonstrably hard, satisfactory solutions can be found sometimes directly from the SDP relaxation, and often by the proposed genetic algorithm. In many scenarios, the SDP solution gives hints as to what a good path might look like, while in others it consists of seemingly random, disconnected edges only. In those cases the lower bound obtained is usually not very tight and it is difficult to draw any conclusions about the optimality of any path. This is of course to be expected given the hardness of the problem. Nevertheless, the SDP solution can always be used to seed the GA optimizer in the proposed three-stage stratified algorithm.

The linearized approximation (P5) sometimes gives reasonable solutions, as in Figure 5.4, but most often does not show any proper long-term planning behavior, as in Figure 5.3. On a directed acyclic graph (see Figure 5.6) it does have the advantage of being extremely fast compared to the other methods.

For computational tractability, structure points must be considered independent, and real point cloud data need to be subsampled. How to best subsample while preserving data characteristics has not yet been considered. As the graph size and connectivity increases, computational complexity also rises and the quality of solutions attainable in reasonable time drops. This limits the resolution of the discretization, particularly in the orientation space, which means local, continuous refinement may be a desirable second step, using e.g. the optimization algorithm from the previous chapter.

Appendix

5.A Semidefinite Programming

Semidefinite programs are a class of convex conic optimization problems where a linear objective function is minimized over the cone defined by semidefinite matrices. On standard form the problem may be stated as

$$\begin{aligned} \text{min.} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{F}_0 + x_1 \boldsymbol{F}_1 + \ldots + x_n \boldsymbol{F}_n \preceq \boldsymbol{0}, \end{aligned} \tag{5.5}$$

where $\boldsymbol{c}$ is a vector of coefficients, the $\boldsymbol{F}_i$ are symmetric matrices and $\preceq$ indicates that the sum should be negative semidefinite. Linear, quadratic and second order cone programs are all special cases of semidefinite programs. The optimum of an SDP can be found to within ϵ accuracy in time polynomial in ϵ and the problem size, most commonly using interior point methods, see for example Nesterov and Nemirovskii (1994).

Branch and Bound

Imposing integral or binary constraints on the variables of a convex program instantly makes it much more difficult. Many hard combinatorial problems, such as the traveling salesman problem, may be cast as integer linear programs which must therefore be *NP*-complete (see Papadimitriou and Steiglitz 1998). Given an SDP with added binary constraints on all or some of the variables, a solution can be found using a branch-and-bound strategy.

First the problem is solved with the binary constraints relaxed to be convex, replacing $x_i \in \{0, 1\}$ with $x_i \in [0, 1]$. If some of the variables designated to be binary are fractional in the solution, one is chosen and two new relaxed problems are posed. In one the chosen variable is set to one, in the other to zero.

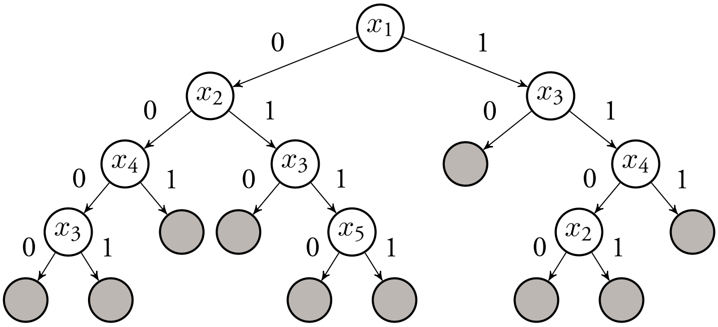


Figure 5.13: Example branch-and-bound search tree. Branching has been terminated at the shaded nodes either because an integer solution was found, the problem was infeasible or the lower bound computed was higher than the value of the current incumbent solution.

The two new subproblems are solved, and if any of the remaining variables are still fractional, one is chosen and the problem is split once again. Recursively applied, this procedure gives rise to a binary search tree, see Figure 5.13 for an example. Branching stops whenever a subproblem is infeasible or returns a binary solution. After solving a subproblem, a fractional solution is rounded and checked for feasibility. If it is, and the objective value is lower than that of the best solution found so far (if one has been found), the solution becomes the new *incumbent*, and provides an upper bound on the optimal objective value. Since every node of the search tree solves a relaxed version of the original problem, the objective value computed is a lower bound on the optimal value. Therefore, branching may also be terminated whenever a node computes a lower bound which is higher than the current incumbent upper bound. When the lowest lower bound of any leaf node matches the incumbent upper bound, the optimal solution has been found, at least in theory. However, if the SDP solver used is not accurate enough, it can be difficult to tell whether a node should be closed or not. In our experiments, we have found that MOSEK and SeDuMi (J. F. Sturm 1998) may give different results, with MOSEK producing slightly lower objective values.

Because of the bounding mechanism, the order in which nodes are visited becomes important. Different strategies have been proposed and they may lead to drastically different running times. Unfortunately, one cannot know beforehand which will work best for a given problem, and the number of nodes to visit will still generally be exponential in the number of variables.

Another angle of attack is to generate feasible binary solutions from fractional ones more intelligently than just by rounding the values, in an attempt to reduce the upper bound as quickly as possible. In our problem formulations, rounding is unlikely to produce a connected path in the graph. Instead one could explicitly generate feasible solutions from fractional ones as described in Section 5.3. Unfortunately, this strategy did not improve the convergence speed in our experiments, mainly because the convex relaxations are often not very tight and few of the nodes ever produce lower bounds higher than the optimal value, except for near the very end of the process.

5.B Transforming the Objective

The objective functions used in the planning problems can be transformed to epigraph form with linear objectives under semidefinite constraints.

Minimizing the Trace

The problem

$$\text{min.} \quad \operatorname{tr}(\boldsymbol{A}^{-1}) \tag{5.6}$$

is equivalent to the SDP

$$\begin{aligned} \text{min.} \quad & \sum_{i=1}^{n} t_i \\ \text{s.t.} \quad & \begin{pmatrix} \boldsymbol{A} & \boldsymbol{e}_i \\ \boldsymbol{e}_i^\top & t_i \end{pmatrix} \succeq \mathbf{0}, \quad i = 1, \ldots, n\,, \end{aligned} \tag{5.7}$$

where $\boldsymbol{e}_i$ is the i:th column of the identity matrix, and n the dimension of $\boldsymbol{A}$. This holds because a matrix is positive semidefinite if and only if its Schur complement is, and in the constraints above, the complements are $t_i - \boldsymbol{e}_i^\top \boldsymbol{A}^{-1} \boldsymbol{e}_i$ so they are equivalent to $(\boldsymbol{A}^{-1})_{ii} \leq t_i$. Minimizing $\sum t_i$ thus minimizes the sum of the diagonal elements of $\boldsymbol{A}^{-1}$, i.e. the trace. Note that this requires n extra variables in the formulation.

Minimizing the Maximum Eigenvalue

The problem

$$\text{min.} \quad \lambda_{\text{max}}(\boldsymbol{A}^{-1}) \tag{5.8}$$

is equivalent to

$$\begin{aligned} \text{min.} \quad & t \\ \text{s.t.} \quad & \boldsymbol{A} + t\boldsymbol{I} \succeq \mathbf{0}, \end{aligned} \tag{5.9}$$

where $\boldsymbol{I}$ is the identity matrix. Note that if the matrix $\boldsymbol{A}$ has eigenvalues $\lambda_i(\boldsymbol{A})$ then $\boldsymbol{A} + t\boldsymbol{I}$ has eigenvalues $\lambda_i(\boldsymbol{A}) + t$. The semidefinite constraint is thus equivalent to $\lambda_{\text{min}}(\boldsymbol{A}) + t \geq 0$ which in turn is equivalent to $\lambda_{\text{max}}(\boldsymbol{A}^{-1}) \leq -1/t$ (note that $t < 0$ at the optimal point since we assume $\boldsymbol{A}$ is positive definite). Minimizing $-1/t$ is then equivalent to minimizing t and the result follows. The objective value will however not equal the maximum eigenvalue, which is required if we want to add additional terms to the objective (such as the path length). In this case, we introduce a new variable u and add the constraint $-1/t \leq u$ which can be formulated as a Schur complement, yielding the SDP

$$\begin{aligned} \text{min.} \quad & u \\ \text{s.t.} \quad & \boldsymbol{A} + t\boldsymbol{I} \succeq \mathbf{0} \\ & \begin{pmatrix} -t & 1 \\ 1 & u \end{pmatrix} \succeq \mathbf{0}. \end{aligned} \tag{5.10}$$

In this formulation only one (or two) extra variables are required and the SDP is faster to solve than when minimizing the trace.

Chapter 6

Reconstruction from Unordered Image Sequences

Three-dimensional reconstruction from unordered image sequences is a well-studied problem in the computer vision literature, see for example Snavely et al. (2007); Agarwal, Snavely, Simon, et al. (2009); Frahm et al. (2010); Olsson and Enqvist (2011); Crandall et al. (2011). "Unordered" refers to the assumption that nothing is known about the relative location of the cameras or time of capture of the images. This is in contrast to the SLAM problem formulation where consecutive images are known to be close in space and time which can be exploited for tracking, matching and reconstruction. Unordered image sequences can comprise anything from carefully planned image series taken for the specific purpose of reconstruction, to completely uncurated image collections downloaded from the Internet, where the images have only been tagged with the name of the location we wish to reconstruct. In the most general case, one must therefore assume that little is known at the outset in terms of scene coverage, image quality or camera calibration. The main difficulty is usually image feature matching, which may be very challenging given widely varying viewpoints and illumination conditions. In this chapter, we will nevertheless assume that some matching has been performed, and focus on the geometric reconstruction phase. We apply concepts of active view planning seen previously to the unordered image set reconstruction problem, using a sequential algorithm (in contrast to more holistic methods such as known rotation algorithms). Although we can no longer choose the viewpoint freely, in a sequential algorithm there is usually a choice between a subset of the images at every step. Our strategy will be to choose the image giving the smallest error, judged by the expected covariance of the reconstruction. To be able to determine the covariance, it is necessary to know the uncertainty of the observed geometry. In the following, it is shown how this is achieved by propagating covariances when resectioning cameras and triangulating points, and how as a side effect the algorithms gain robustness and better approximate the maximum likelihood estimate.

6.1 Estimation from Uncertain Geometry

The cornerstones of sequential structure-and-motion are triangulation and camera pose estimation. Usually, one attempts to find the maximum likelihood solution for the point or camera given noisy image measurements, but assuming that all other parameters are known exactly. This is of course rarely the case, since points and cameras are triangulated and resectioned using noisy data. Below, we derive algorithms that also take the uncertainty of the 3D structure and camera parameters into account.

6.1.1 Pose Estimation

Consider the problem of camera pose estimation given N 3D point coordinates $\boldsymbol{X}$ and their measured projections in one image, $\tilde{\boldsymbol{x}}$. Assuming there are errors in the image measurements, the problem is to find the maximum likelihood solution, i.e. the camera parameters $\boldsymbol{\theta}^*$ satisfying

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}), \tag{6.1}$$

where

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta} \mid \tilde{\boldsymbol{x}}, \boldsymbol{X}) = p(\tilde{\boldsymbol{x}} \mid \boldsymbol{\theta}, \boldsymbol{X}) \tag{6.2}$$

is the likelihood function. In this formulation it is assumed that the structure parameters $\boldsymbol{X}$ are precisely known. More generally, given a probability distribution of $\boldsymbol{X}$, the problem is to maximize

$$\mathcal{L}(\boldsymbol{\theta}) = \int_{\mathbb{R}^{3N}} p(\tilde{\boldsymbol{x}} \mid \boldsymbol{\theta}, \boldsymbol{X}) p(\boldsymbol{X}) \, d\boldsymbol{X}. \tag{6.3}$$

We restrict our attention to the case of Gaussian distributions. Then we have

$$\mathcal{L}(\boldsymbol{\theta}) \propto \int_{\mathbb{R}^{3N}} e^{-\|\tilde{\boldsymbol{x}} - \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\theta})\|^2_{\boldsymbol{R}}} \cdot e^{-\|\boldsymbol{X} - \bar{\boldsymbol{X}}\|^2_{\boldsymbol{Q}}} \, d\boldsymbol{X}, \tag{6.4}$$

where $\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\theta})$ is the projection of the points $\boldsymbol{X}$ using camera parameters $\boldsymbol{\theta}$, $\boldsymbol{R}$ the measurement error covariance, $\boldsymbol{Q}$ and $\bar{\boldsymbol{X}}$ are the covariance matrix and mean of the distribution of $\boldsymbol{X}$ and $\|\boldsymbol{y}\|^2_{\boldsymbol{\Sigma}} = \boldsymbol{y}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{y}$ the squared Mahalanobis distance. Next, we project the distribution of $\boldsymbol{X}$ onto the image plane, by integrating along the light rays. Formally, for a given $\boldsymbol{\theta}$ we parametrize each 3D point by its image projection $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{X},\boldsymbol{\theta})$ and depth ρ, so that

$$\mathcal{L}(\boldsymbol{\theta}) \propto \int_{\mathbb{R}^{2N}} e^{-\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|^2_{\boldsymbol{R}}} \left(\int_{\mathbb{R}^{N}} e^{-\|(\boldsymbol{x},\rho)-\bar{\boldsymbol{X}}\|^2_{\boldsymbol{Q}}}\, d\rho \right) d\boldsymbol{x}. \tag{6.5}$$

The right-hand factor is a distribution on the $2N$-dimensional generalized image plane, and may be seen as the projection of a Gaussian random variable, i.e. $\boldsymbol{f}\big(\mathcal{N}(\bar{\boldsymbol{X}},\boldsymbol{Q}),\boldsymbol{\theta}\big)$. By Taylor expansion about $\bar{\boldsymbol{X}}$, $\boldsymbol{f}$ can be approximated by $\tilde{\boldsymbol{f}}(\boldsymbol{X},\boldsymbol{\theta}) = \boldsymbol{f}(\bar{\boldsymbol{X}},\boldsymbol{\theta}) + \boldsymbol{J}(\boldsymbol{X}-\bar{\boldsymbol{X}})$, and for affine functions

$$\tilde{\boldsymbol{f}}\big(\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}),\boldsymbol{\theta}\big) = \mathcal{N}\big(\tilde{\boldsymbol{f}}(\boldsymbol{\mu},\boldsymbol{\theta}), \boldsymbol{J}_{\boldsymbol{X}}\boldsymbol{\Sigma}\boldsymbol{J}_{\boldsymbol{X}}^\top\big) \tag{6.6}$$

with $\boldsymbol{J}_{\boldsymbol{X}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{X}}\Big|_{\boldsymbol{\theta}}$. We now have

$$\mathcal{L}(\boldsymbol{\theta}) \underset{\sim}{\propto} \int_{\mathbb{R}^{2N}} e^{-\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|^2_{\boldsymbol{R}}} \cdot e^{-\|\boldsymbol{f}(\bar{\boldsymbol{X}},\boldsymbol{\theta})-\boldsymbol{x}\|^2_{\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^\top}}\, d\boldsymbol{x}, \tag{6.7}$$

which may be seen as the convolution $(u * v)(\boldsymbol{\tau}) = \int u(\boldsymbol{x})v(\boldsymbol{\tau}-\boldsymbol{x})\,d\boldsymbol{x}$ of the Gaussians $u(\boldsymbol{x}) = e^{-\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|^2_{\boldsymbol{R}}}$ and $v(\boldsymbol{x}) = e^{-\|\boldsymbol{x}-\boldsymbol{0}\|^2_{\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^\top}}$, with $\boldsymbol{\tau} = \boldsymbol{f}(\bar{\boldsymbol{X}},\boldsymbol{\theta})$. The convolution of Gaussians is particularly simple,

$$\mathcal{N}(\tilde{\boldsymbol{x}},\boldsymbol{R}) * \mathcal{N}(\boldsymbol{0},\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^\top) = \mathcal{N}(\tilde{\boldsymbol{x}},\boldsymbol{R}+\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^\top), \tag{6.8}$$

giving

$$\mathcal{L}(\boldsymbol{\theta}) \underset{\sim}{\propto} e^{-\|\tilde{\boldsymbol{x}}-\boldsymbol{f}(\bar{\boldsymbol{X}},\boldsymbol{\theta})\|^2_{\boldsymbol{R}+\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^\top}}. \tag{6.9}$$

Taking the logarithm, maximizing the likelihood is equivalent to minimizing

$$-\log \mathcal{L}(\boldsymbol{\theta}) \underset{\sim}{\propto} \|\tilde{\boldsymbol{x}} - \boldsymbol{f}(\bar{\boldsymbol{X}},\boldsymbol{\theta})\|^2_{\boldsymbol{R}+\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^\top}, \tag{6.10}$$

which can be solved using an iteratively reweighted nonlinear least-squares algorithm. In fact, only a minor modification to a standard algorithm for minimizing the reprojection error is required. For example, a Levenberg-Marquardt optimization loop would be modified to

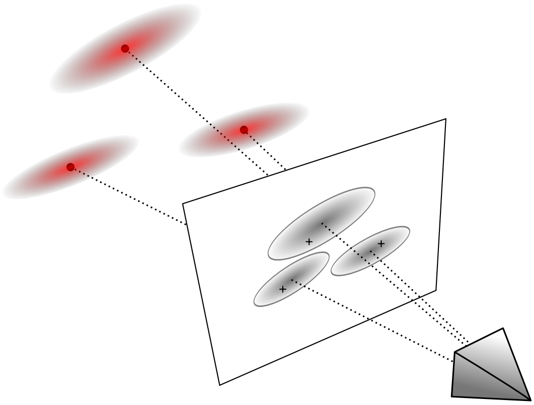


Figure 6.1: For camera resectioning, the uncertainties of the 3D points are projected onto the image plane and convolved with the image measurement uncertainty giving the reprojection error metric. Note that the projections are not necessarily independent; however, in this work inter-point covariances are discarded for computational reasons.

```
while not converged do
    ...
    $\boldsymbol{W} \leftarrow (\boldsymbol{R} + \boldsymbol{J_X} \boldsymbol{Q} \boldsymbol{J_X}^\top)^{-1}$
    $\boldsymbol{\delta\theta} \leftarrow (\boldsymbol{J_\theta}^\top \boldsymbol{W} \boldsymbol{J_\theta} + \lambda \boldsymbol{I})^{-1} \boldsymbol{J_\theta}^\top \boldsymbol{W} \boldsymbol{r}$
    ...
end while
```

where $\boldsymbol{J_\theta} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}}$ and $\boldsymbol{J_X}$ as above (cf. equation 2.10). By Theorem 3.2, the covariance matrix of the recovered camera parameters $\boldsymbol{\theta}^*$ can be estimated by the inverse of the Hessian matrix evaluated at the minimum,

$$\boldsymbol{\Sigma_\theta} \approx (\boldsymbol{J_{\theta^*}} \boldsymbol{W}^* \boldsymbol{J_{\theta^*}^\top})^{-1}. \tag{6.11}$$

Of course, a good initial guess is required to start the iterative algorithm, and can be obtained using standard minimal or linear solvers. The general effect of taking the distribution of $\boldsymbol{X}$ into account is to give more weight to well-determined 3D points than uncertain ones when finding the camera pose.

6.1.2 Triangulation

Handling uncertainty in camera parameters when triangulating 3D structure is completely analogous to the pose estimation case. The linearized problem formulation is to find

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|\tilde{\boldsymbol{x}} - \boldsymbol{f}(\boldsymbol{\theta}, \bar{\boldsymbol{P}})\|^2_{\boldsymbol{R}+\boldsymbol{J}\boldsymbol{S}\boldsymbol{J}^\top} , \tag{6.12}$$

where $\boldsymbol{\theta}$ now represents the 3D structure, and $\bar{\boldsymbol{P}}$ is the mean of the distribution of the cameras with covariance $\boldsymbol{S}$ and $\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{P}}\Big|_{\boldsymbol{\theta}}$.

6.1.3 Complexity

The introduction of the weight matrix $\boldsymbol{W}$ in the algorithms above inevitably incurs extra computational costs. In particular, if the input variables are correlated, $\boldsymbol{W}$ will be a full matrix and the natural sparsity of the problems is lost. To mitigate this, we will assume no correlation between pairs of cameras or points, so that $\boldsymbol{W}$ is block diagonal. Such simplification is also necessary since the full covariance matrix of even a moderately sized reconstruction problem would occupy hundreds of gigabytes of memory. Furthermore, it may not be necessary to recompute $\boldsymbol{W}$ every iteration, since the projection is not expected to change significantly given a good initialization. In our experiments, it is recomputed every third iteration and after the last, to ensure that the covariance estimate given by (6.11) is as accurate as possible.

6.2 Covariance Propagation

The proposed algorithms open the possibility of covariance propagation throughout the reconstruction process. Uncertainties in 3D points are transferred to uncertainty in resectioned cameras, which in turn transfer uncertainty to triangulated points, and so on. In this manner, a rough estimate of the covariances is available at any time and can be used, for example, to improve reconstruction accuracy and for next best view planning, which we exploit to reduce error accumulation.

Below we detail a system for 3D reconstruction from unordered image sequences and show the benefits that can be gained.

6.2.1 Selecting the Seed

In choosing the set of images on which to initialize the reconstruction, we strive for the following: the initial reconstruction should be stable, contain many structure points and it should be near the center of the camera graph (the graph with each camera a vertex and edges between cameras observing common features). The latter is motivated by the fact that error accumulation is a function of the distance from the seed; if the "degrees of separation" from the seed is kept low, error accumulation can be minimized. We therefore wish to minimize the distance of every camera to the seed. For our purposes we define the center as any vertex of the camera connectivity graph with minimal *farness*, the sum of shortest distances from the node to all others. We define the edge weights of the graph as $1/\max(0, n_c - 4)$, where n_c is the number of observed points common to both cameras. This heuristic, while ignoring the actual two-view geometry, is based on the assumption that cameras sharing many observed points are well-determined relative to each other. The maximum imposes a five point overlap threshold, needed to determine relative motion between views. Now, all shortest paths in the graph can be computed and summed for each camera, the k lowest scoring yielding a set of candidate images. For each candidate, an adjacent view with a balance between many common image points and good parallax is selected as in Snavely et al. (2007), i.e. each pairing is scored according to the proportion of outliers to a homography fit. The top-scoring pair is selected, and standard two-view reconstruction is performed, followed by bundle adjustment.

6.2.2 Fixing the Gauge

Reconstruction from image measurements only is subject to global translation, rotation and scale ambiguity. Unlike Snavely et al. (2008), which measured pairwise covariances in local coordinate systems, we need globally referenced covariances and so must compute these for the seed reconstruction. For the covariances to be defined we must fix the gauge, especially the scale, since the dimension of the nullspace of the Hessian matches the number of degrees of freedom of the system. From a theoretical standpoint, taking the pseudoinverse of the unconstrained Hessian is the most satisfying solution (Hartley and Zisserman 2003; Morris 2001), however it can be computationally very expensive if the seed views share many points (i.e. > 1000). An alternative approach is to constrain the parameters of the system by adding penalties to the cost function, making the Hessian full rank so it can be inverted without finding an SVD. Different constraints lead to somewhat different estimates of the covariance; one way is to lock the first camera and impose a distance constraint on the mean of the structure points, as was done in Snavely et al. (2008), or one can simply fix the distance between the first and second camera. The first prior gives results closer to the pseudoinverse, but also destroys the sparsity of the Hessian matrix making inversion more expensive. In cases where the pseudoinverse is too expensive we choose the second option which preserves sparsity.

After fixing the scale, there is still a difficulty in quantifying just how large an uncertainty is, since it must be put in relation to the overall size of the reconstruction. The scale is unknown in the beginning, since there is no guarantee that the distance between the seed cameras is representative of the whole scene. This has implications for the various outlier rejection thresholds used in the reconstruction pipeline.

6.3 Next Best View Planning

View planning in a sequential reconstruction process aims to actively choose the most favorable sensor configuration (camera position and orientation) to achieve a certain goal, in this case geometric accuracy. In each iteration, we can choose which camera to resection among those observing a sufficient number of triangulated points. Usually, the camera observing the largest number of triangulated points is chosen first. However, if the geometry is such that the pose is poorly determined, triangulations using the image will have larger errors, propagating to subsequently resectioned cameras, etc. It therefore makes sense to minimize the error accumulation in every step. To this end, we propose to select the camera with lowest estimated reconstruction error, by exhaustive search among candidate images. The covariance is computed by first resectioning the camera using a linear or minimal solver and taking the inverse of the Hessian, $\boldsymbol{\Sigma}_{\text{cam}} \approx (\boldsymbol{J}_{\boldsymbol{\theta}} \boldsymbol{W} \boldsymbol{J}_{\boldsymbol{\theta}}^{\top})^{-1}$ as defined in Section 6.2. As a scalar measure of reconstruction error we use $\text{tr}(\boldsymbol{\Sigma}_{\text{cam}}) \cdot \epsilon_{\text{rp}}$, where ϵ_{rp} is the mean reprojection error. This turns out to give better results than the covariance alone; a small estimated covariance does not necessarily imply a low reprojection error, and a well-determined camera should ideally have both. Note that the score can be cached for each camera between iterations and need only be recomputed if more points in the camera's view have been triangulated. While the number of views that need to be resectioned in each iteration is dependent on the particular data set and could theoretically grow with the number of triangulated points, in practice this number is found to be approximately constant throughout the reconstruction process and typically between 10 and 50.

6.4 Reconstruction Pipeline

We apply NBV planning and covariance propagation to the problem of reconstruction from unordered image collections. We will assume that matching and tracking of image features has been performed and is outlier-free. If not (as in the last experiment below), the proposed method is easily integrated with outlier detection schemes such as RANSAC. The algorithm is mainly standard fare:

1. Find initial seed views (Section 6.2.1).
2. Reconstruct and bundle adjust the seed geometry.
3. Compute the covariance of the seed (Section 6.2.2).
4. Choose a camera to resect following Section 6.3. Resect using a linear method; if it fails (i.e. large reprojection error or points behind the camera) try an L_∞ formulation (see Section 2.6) instead. If that also fails, choose another camera and try again. Else, refine the camera pose by minimizing (6.10) and store its covariance.
5. Triangulate all points seen by at least two resectioned cameras using a linear method. Compute an approximate uncertainty by evaluating the Hessian of the standard reprojection error and taking the trace of the inverse. Well-determined points, i.e. with low covariance and reprojection error, as specified by thresholds, are kept and further refined by minimizing (6.12). Store the covariance derived from this reprojection error.
6. (Optional) Bundle adjust over all or part of the reconstruction, and update covariances accordingly.
7. If possible, goto step 4 and find the next view, else terminate.

In the first experiments below, bundle adjustment is only performed on the seed to demonstrate the efficacy of the approach in reducing error accumulation. In real use, step 6 should performed at regular intervals. The cost of updating the covariances afterwards is a computational bottleneck, which needs to be addressed.

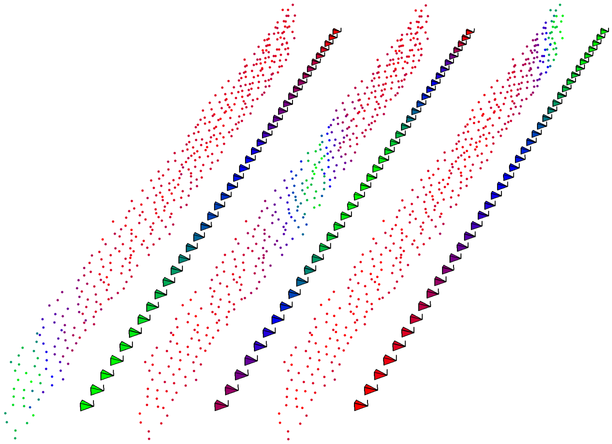


Figure 6.2: Toy example of camera array observing a wall illustrating the covariance estimation results depending on which seed is chosen (from left to right, cameras 1 and 2, 19 and 20, 39 and 40). The points and cameras are color coded by the trace of their covariance, with green through blue to red for increasing uncertainty. Choosing the seed in the middle reduces the maximum camera uncertainty with respect to the seed.

6.5 Experiments

Figure 6.2 shows a simple synthetic example of the dependence on the seed of the propagated covariances. Although the *relative* uncertainties between all cameras remain the same in our linearized Gaussian propagation model, in reality the reconstruction errors depend heavily on the path taken.

Next, the algorithm is applied to a dataset extracted from photos of the “Spilled blood” church in St. Petersburg. The reconstruction and a comparison with a standard method is shown in Figures 6.3 and 6.4. The comparison shows the mean standard reprojection error and the ground truth deviation, defined as the mean distance of each triangulated point from its ground truth position, after the two point clouds have been aligned using a Procrustes transformation. The “ground truth” in this case has been obtained from the system described in Olsson and Enqvist (2011), using the known-rotation L_∞ reconstruction method. The plain method, without covariance propagation or NBV planning, runs into trouble around iteration 300 and does not manage to resection all cameras, whereas the proposed algorithm does and is generally more robust and accurate. A similar comparison is made for the “Trafalgar” dataset of Agarwal,

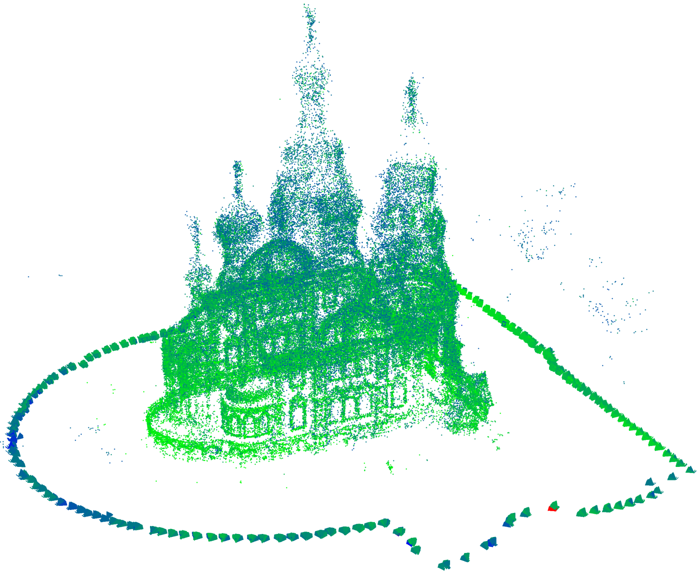


Figure 6.3: Resulting point cloud reconstruction of the “Spilled blood” dataset using the proposed algorithm, color coded by estimated covariance. No bundle adjustment has been performed. The dataset has 781 images, 162,521 points and 2,541,736 feature measurements.

Snavely, Seitz, et al. (2010) in Figure 6.5.

Finally, we compare three variants of the proposed algorithm on the Lund Cathedral dataset. The covariance propagation and next best view-planning can be used independently, i.e. the next image can be chosen by the maximum overlap principle while propagating covariances, or the next view can be chosen based on camera uncertainty calculated using zero point covariances, with no propagation. As Figure 6.7 shows, using NBV planning alone does not work well at all and the process breaks down, like the plain method. Propagation without planning works almost as well as both combined, and is probably the greatest contributing factor.

Next, we apply the proposed algorithm to a more realistic scenario. The “Örebro castle” dataset is corrupted by replacing 5% of matches with artificially generated outliers. Camera resectioning is preceded by RANSAC outlier detection, and we allow local bundle adjustment to be performed every 20 iterations,

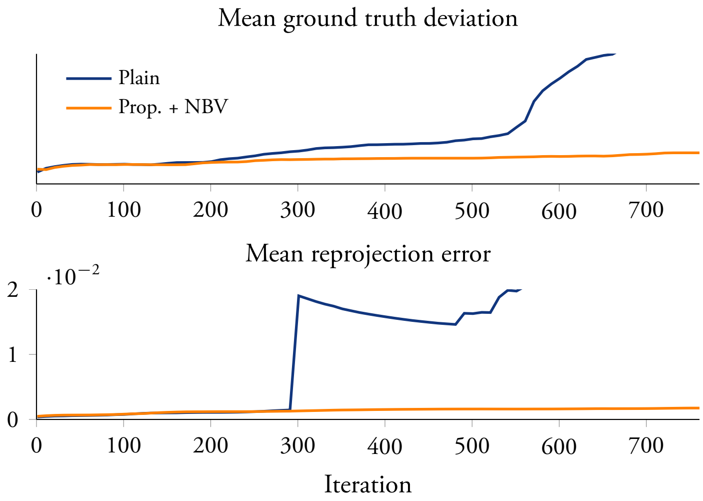


Figure 6.4: Comparison between the proposed algorithm and a baseline “plain” method on the “Spilled blood” dataset. In the plain method the standard reprojection error is minimized instead, and the next camera is chosen by the maximum overlap principle. Running times were 22 and 13 min respectively. There is no absolute scale on the top graph since it depends on the overall scale of the reconstruction, which is arbitrary.

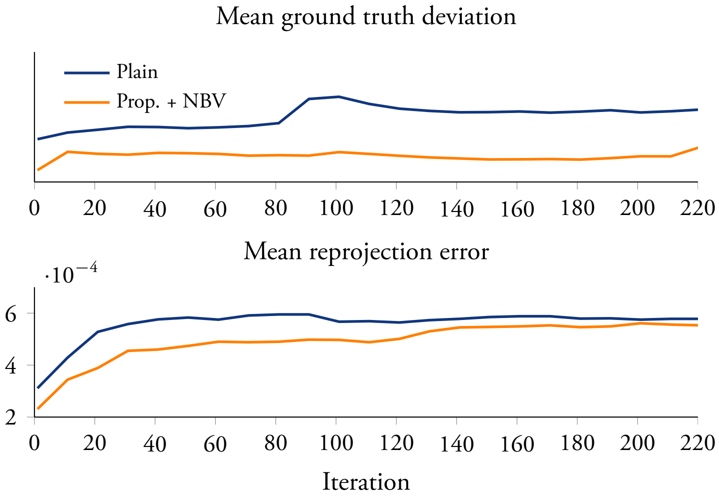


Figure 6.5: Results for the Trafalgar dataset (256 images). Running times were 83 and 138 s respectively.

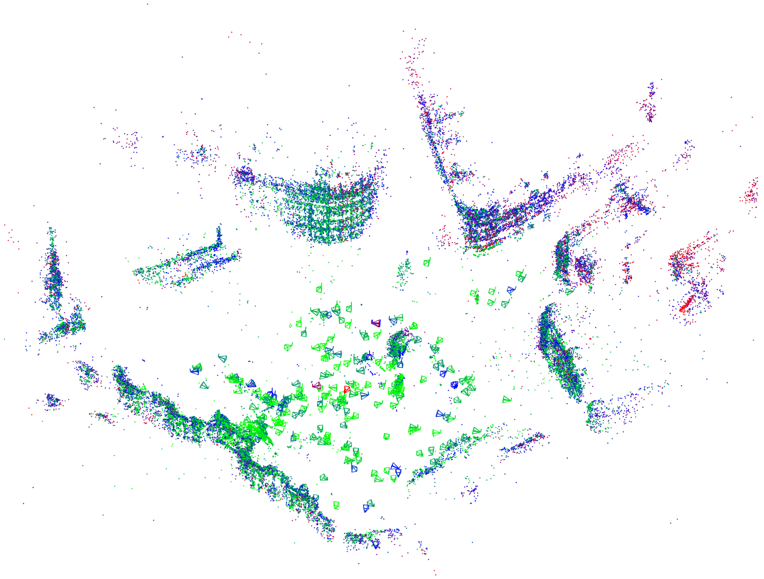
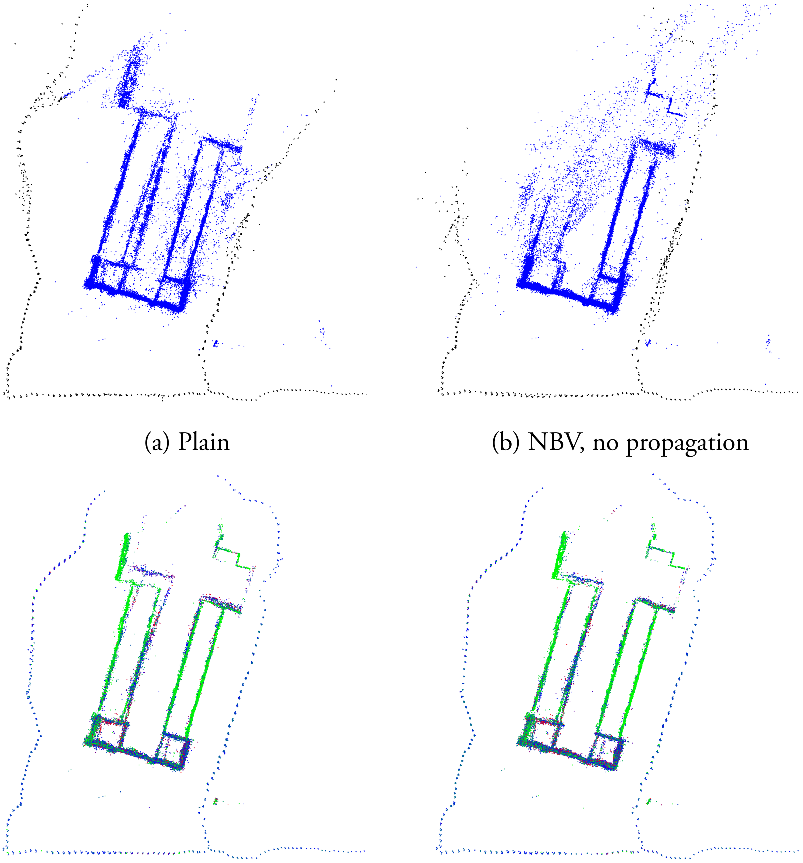


Figure 6.6: Resulting point cloud reconstruction of the Trafalgar Square dataset.

optimizing over the 20 last resectioned cameras and the triangulated points visible in these views. Measurements from other cameras where the points are visible are also included. A robust Huber cost function is used, and measurements with high reprojection error after convergence are deemed outliers and removed. The resulting point cloud reconstructions are shown in Figure 6.9. The proposed method produces higher quality output and, surprisingly, is faster in this case. Note that in this experiment the covariances are not updated after bundle adjustment, and still there is marked improvement.

After bundle adjustment, the estimated covariances of the affected parameters are no longer valid and need to be updated. As mentioned in Section 6.2.2, inverting the whole Hessian matrix of the LM system is infeasible for all but the smallest problems. However, since we only need the diagonal blocks of the covariance, a lot of work can be saved. If the covariance matrix corresponding to the camera parameters only is known, the individual point covariance blocks can be computed very efficiently. The Hessian of a typical bundle adjustment



(a) Plain

(b) NBV, no propagation

(c) Propagation only

(d) Propagation and NBV

Figure 6.7: Lund Cathedral dataset (1060 images, 45770 points, 408625 projections) reconstructed using the baseline algorithm, next best view-planning only without propagating covariances, propagating covariances but using the maximum overlap principle, and the proposed algorithm, using both NBV planning and propagation.

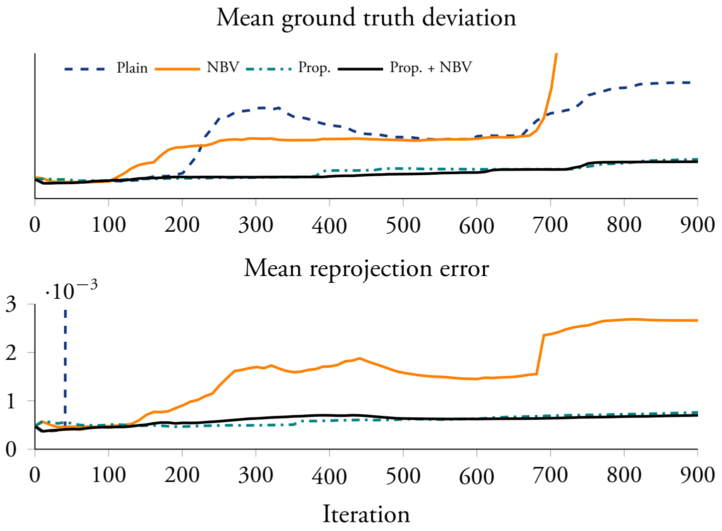
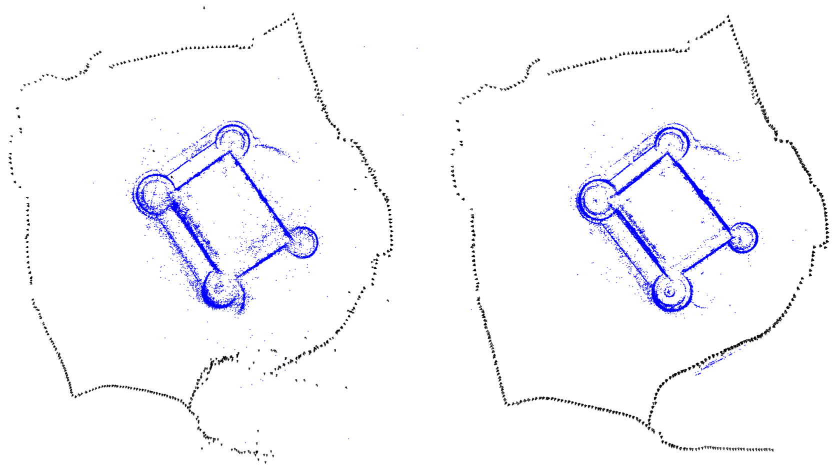


Figure 6.8: Error plots for the Lund Cathedral dataset.

problem has the particular structure

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A} & \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{B} \\ \boldsymbol{B}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A} & \boldsymbol{B}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{B} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U} & \boldsymbol{W} \\ \boldsymbol{W}^\top & \boldsymbol{V} \end{pmatrix}, \tag{6.13}$$

where $\boldsymbol{\Sigma}$ is the measurement noise covariance matrix and $\boldsymbol{U}$ and $\boldsymbol{V}$ are block diagonal with blocks corresponding to the cameras and points respectively. In the fixed-gauge case, the covariance of the camera parameters is then given by $\boldsymbol{\Sigma}_\mathrm{c} = (\boldsymbol{U} - \boldsymbol{W}\boldsymbol{V}^{-1}\boldsymbol{W}^\top)^{-1}$ and the covariance for point i is given by $\boldsymbol{\Sigma}_{\mathrm{p}_i} = \boldsymbol{V}_i^{-1}\boldsymbol{W}_i^\top \boldsymbol{\Sigma}_\mathrm{c} \boldsymbol{W}_i \boldsymbol{V}_i^{-1} + \boldsymbol{V}_i^{-1}$ where $\boldsymbol{V}_i$ is the corresponding block of $\boldsymbol{V}$, and $\boldsymbol{W}_i$ the corresponding rows of $\boldsymbol{W}$ (see Hartley and Zisserman (2003) for details). Exploiting the sparsity of $\boldsymbol{W}$, the product $\boldsymbol{W}_i^\top \boldsymbol{\Sigma}_\mathrm{c} \boldsymbol{W}_i$ can be evaluated in time proportional to the number of cameras observing point i. The dominating cost is in practice computing the camera covariance, i.e. forming and inverting the Schur complement, typically of cubic cost in the number of cameras. When this number is low, the covariance update is fast (on the order of a few seconds), but as the reconstruction grows the cost becomes prohibitive and the time would be better spent on bundle adjustment. However, as shown above, updating the covariances is not critical to the performance of the algorithm, and an approximation may be sufficient. For example, experiments have shown



(a) Standard method, 908 s.

(b) Propagation and next best view planning, 782 s.

Figure 6.9: Reconstructions of the Örebro castle dataset with 5% outliers and local bundle adjustment every 20 iterations. The difference in processing time is due to the plain method often failing to resection cameras using the fast linear method and falling back on the slower but more robust L_∞ solver, and more often failing and reattempting triangulation as new views are resectioned.

that only inverting the diagonal blocks corresponding to individual cameras of the Schur complement gives a reasonable approximation to the full inverse. Nevertheless, efficient updating of the covariances remains an open problem and is the subject of future work.

6.6 Conclusion

The proposed method increases robustness to errors such as poorly resectioned cameras and poorly triangulated points, reduces error accumulation and also provides estimates of reconstruction accuracy which could be further processed for outlier detection etc. This comes at a cost of up to a twofold increase in running time. However, this cost is practically linear in the problem size, whereas iterated bundle adjustment costs between $\mathcal{O}(n^3)$ and $\mathcal{O}(n^4)$, depending on problem structure. Thus, trading less frequent bundling for covariance propagation and next best view planning should pay off for large problems. In addition, the covariance propagation mechanism is a minor modification to standard algorithms and can therefore easily be incorporated into existing sequential reconstruction pipelines.

Part II

Refractive Structure and Motion

Chapter 7

Preliminaries on Refraction

The following chapters are mainly concerned with optical refraction effects as a complicating factor in structure-and-motion systems. When a ray of light passes from one optical medium into another, such as from air into water or glass, it bends. Thus, if a camera in air observes a scene under water, the usual pinhole camera model is no longer valid since it assumes that light travels along straight lines from the 3D point to the camera center. Not accounting for the refractive effects at the boundary between optical media can lead to significant errors in the reconstruction, so they need to be accurately modeled.

This chapter gives a short primer on modeling refraction using Snell's law, and an introduction to solving systems of polynomial equations, which is central to our implementation of efficient minimal solvers for camera pose estimation under refraction.

7.1 Snell's Law

Refraction of light at an optical medium boundary is described by Snell's law, named for Willebrord Snellius' discovery in 1621 but already described much earlier by Ibn Sahl in 984. The law states that

$$\rho_1 \sin\theta_1 = \rho_2 \sin\theta_2, \tag{7.1}$$

where $\rho_{1,2}$ are the refractive indices of the two media and $\theta_{1,2}$ the angles the incident and refracted ray make with the surface normal (see Figure 7.1). Since the angle of refraction only depends on the ratio of refractive indices, it is convenient to define $\mu \equiv \rho_1/\rho_2$. Furthermore, the incident ray with direction vector $\boldsymbol{u}$, the refracted ray direction $\boldsymbol{v}$ and the surface normal $\boldsymbol{n}$ must all lie in a common plane, and so $\boldsymbol{v}$ can be expressed as a linear combination of $\boldsymbol{u}$ and $\boldsymbol{n}$,

$$\boldsymbol{v} = a\boldsymbol{u} + b\boldsymbol{n}. \tag{7.2}$$

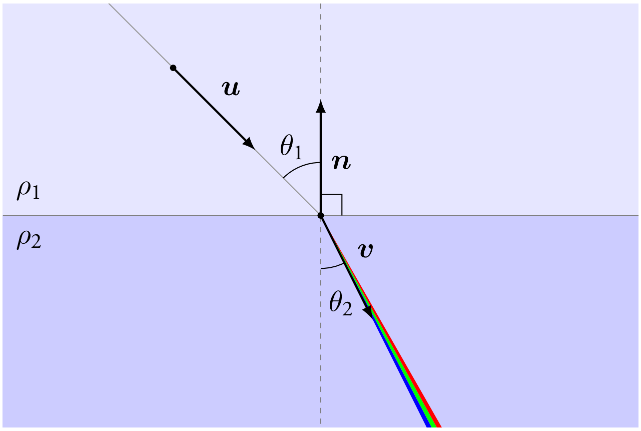


Figure 7.1: A light ray with direction vector $\boldsymbol{u}$ is refracted at the media boundary with surface normal $\boldsymbol{n}$, and emerges with new direction $\boldsymbol{v}$ according to Snell's law. Since the index of refraction is wavelength-dependent, white light disperses into its constituent colors.

Taking the cross product with $\boldsymbol{n}$ and the norm, we find $\|\boldsymbol{v} \times \boldsymbol{n}\| = |a|\|\boldsymbol{u} \times \boldsymbol{n}\|$ and assuming $\boldsymbol{u}$ and $\boldsymbol{n}$ are unit vectors (and $\theta_1 \neq 0$),

$$|a| = \frac{\|\boldsymbol{v} \times \boldsymbol{n}\|}{\|\boldsymbol{u} \times \boldsymbol{n}\|} = \frac{\sin\theta_2}{\sin\theta_1} = \frac{\rho_1}{\rho_2} = \mu. \tag{7.3}$$

It is clear from Figure 7.1 that $a \geq 0$, so $a = \mu$. Taking instead the scalar product of (7.2) with $\boldsymbol{n}$ gives

$$b = \boldsymbol{v} \cdot \boldsymbol{n} - a\boldsymbol{u} \cdot \boldsymbol{n} = -\cos\theta_2 + \mu\cos\theta_1. \tag{7.4}$$

Given the surface normal and the incident ray, the refracted ray direction may thus be computed as

$$\boldsymbol{v} = \mu\boldsymbol{u} + (\mu\cos\theta_1 - \cos\theta_2)\boldsymbol{n}, \tag{7.5}$$

where

$$\begin{aligned} \cos\theta_1 &= -\boldsymbol{n} \cdot \boldsymbol{u} \\ \cos\theta_2 &= \sqrt{1 - \mu^2(1 - \cos^2\theta_1)}\,. \end{aligned} \tag{7.6}$$

In the last equation we see that if $\mu > 1$ the argument to the square root might become negative. If this happens, it means that the ray undergoes total internal reflection at the boundary, and no light enters the second medium.

Note that by induction, if a ray traverses several flat parallel optical media interfaces (such as the two sides of a pane of glass), it will always stay in a single plane; this turns out to be a very useful fact.

The refractive indices are inversely proportional to the speed of light in the material, and are therefore also dependent on the wavelength. Light of different colors will therefore bend slightly differently, but this effect is quite weak and we will usually ignore it. However, in Chapter 9 we will see how treating the red, green and blue color channels of an image separately can reduce so-called *chromatic aberration* effects in underwater imagery.

7.2 Solving Polynomial Equation Systems

In the following chapter we will need to solve systems of multivariate polynomial equations derived from Snell's law, and to understand the methods employed some background on algebraic geometry is needed. We start with some definitions.

Definition 7.1 *A monomial in n variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is a product of the form*

$$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \tag{7.7}$$

where $\alpha_i \in \mathbb{N}_0$. The total degree *of a monomial is the sum $|\boldsymbol{\alpha}| = \alpha_1 + \ldots + \alpha_n$.*

Definition 7.2 *A polynomial f in $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is a finite sum of the form*

$$f(\boldsymbol{x}) = \sum_{\alpha} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}, \tag{7.8}$$

where the $c_\alpha \in \mathbb{C}$ are the coefficients. *The* degree *of a polynomial is defined as* $\max_{\boldsymbol{\alpha}} |\boldsymbol{\alpha}|$, *the largest total degree of any of its monomials.*

The set of all polynomials with complex coefficients is denoted $\mathbb{C}(\boldsymbol{x})$. The problem we wish to solve is the following:

Problem 7.3 *Given a set of polynomials $f_i(\boldsymbol{x}) \in \mathbb{C}(\boldsymbol{x})$, $i = 1, \ldots, m$, find all solutions to the system of equations*

$$\begin{aligned} f_1(\boldsymbol{x}) &= 0 \\ &\vdots \\ f_m(\boldsymbol{x}) &= 0\,. \end{aligned} \tag{7.9}$$

Definition 7.4 *The* variety $V(f_1, \ldots, f_m)$ *is the set of points $\boldsymbol{x}$ where all polynomials $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ vanish.*

A variety may be empty if no solution to the system exists, consist of a finite number of isolated points, or contain a continuum of solutions. We are only interested in systems with a finite non-zero number of solutions, i.e. with finite zero-dimensional varieties.

Definition 7.5 *The* ideal *generated by the polynomials $f_1, \ldots, f_m$ is the set*

$$I = \langle f_1, \ldots, f_m \rangle = \left\{ \sum_{i=1}^{m} h_i(\boldsymbol{x}) f_i(\boldsymbol{x}) \, : \, h_1, \ldots, h_m \in \mathbb{C}(\boldsymbol{x}) \right\}. \tag{7.10}$$

By the definition, all polynomials in I vanish on the corresponding variety V. If the ideal contains *all* polynomials which vanish on V, it is said to be *radical*. Two polynomials f and g are said to be equivalent with respect to I, or congruent modulo I, if $f - g \in I$, meaning that they attain the same values on the variety. With this equivalence relation we can define the quotient space $\mathbb{C}(\boldsymbol{x})/I$, the set of all equivalence classes modulo I.

It may be shown (see e.g. Cox et al. 2007) that for a radical ideal I, $\mathbb{C}(\boldsymbol{x})/I$ is isomorphic to $\mathbb{C}^r$ where $r = |V|$ is the number of solutions to the corresponding equation system; consequently, it is a linear vector space.

Next we need to define an order relation on the monomials. With the *lexicographical* ordering, a monomial $\boldsymbol{x}^{\boldsymbol{\alpha}} >_{\text{lex}} \boldsymbol{x}^{\boldsymbol{\beta}}$ if the first non-zero element of the vector difference $\boldsymbol{\alpha} - \boldsymbol{\beta}$ is positive. With the *graded lexicographical* ordering, $\boldsymbol{x}^{\boldsymbol{\alpha}} >_{\text{grlex}} \boldsymbol{x}^{\boldsymbol{\beta}}$ if the total degree $|\boldsymbol{\alpha}| > |\boldsymbol{\beta}|$, or if $|\boldsymbol{\alpha}| = |\boldsymbol{\beta}|$ and $\boldsymbol{x}^{\boldsymbol{\alpha}} >_{\text{lex}} \boldsymbol{x}^{\boldsymbol{\beta}}$. Finally, the *graded reverse lexicographic* ordering also orders by total degree, but breaks ties differently; $\boldsymbol{x}^{\boldsymbol{\alpha}} >_{\text{grevlex}} \boldsymbol{x}^{\boldsymbol{\beta}}$ if $|\boldsymbol{\alpha}| > |\boldsymbol{\beta}|$, or $|\boldsymbol{\alpha}| = |\boldsymbol{\beta}|$ and the last non-zero element of $\boldsymbol{\alpha} - \boldsymbol{\beta}$ is negative.

Definition 7.6 *The* leading term $LT(f)$ *of a polynomial f is the term with the 'largest' monomial with respect to a given monomial ordering.*

The leading terms $LT(I)$ of the polynomials in an ideal I themselves generate an ideal $\langle LT(I)\rangle$.

Proposition 7.7 *Given a monomial ordering on $\mathbb{C}(\boldsymbol{x})$ and an ideal $I \subset \mathbb{C}(\boldsymbol{x})$, every polynomial $f \in \mathbb{C}(\boldsymbol{x})$ is congruent modulo I to a unique polynomial which is a $\mathbb{C}$-linear combination of the monomials in the complement of $\langle LT(I)\rangle$. Also, the elements of $\{\boldsymbol{x}^{\boldsymbol{\alpha}} : \boldsymbol{x}^{\boldsymbol{\alpha}} \notin \langle LT(I)\rangle\}$ are linearly independent modulo I.*

Proof. See Cox et al. (2007) p. 230. □

Proposition 7.7 tells us that the monomials which are not in $\langle LT(I)\rangle$ constitute a basis for the vector space $\mathbb{C}(\boldsymbol{x})/I$, and that any polynomial function defined on the corresponding variety can be expressed as a linear combination of these basis monomials.

7.2.1 The Companion Matrix

Before trying to solve a multivariate system, we consider the univariate case. A common method to find the roots of a univariate polynomial

$$f(x) = x^n + c_{n-1}x^{n-1} + \ldots + c_1 x + c_0 \tag{7.11}$$

is to construct the so-called *companion matrix*. Note that if $f(x) = 0$ then

$$x^n = -c_{n-1}x^{n-1} - \ldots - c_1 x - c_0, \tag{7.12}$$

and this relation together with a trivial identity mapping can be expressed on matrix form as

$$\underbrace{\begin{pmatrix} -c_{n-1} & -c_{n-2} & \cdots & -c_1 & -c_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}}_{\boldsymbol{C}} \underbrace{\begin{pmatrix} x^{n-1} \\ x^{n-2} \\ \vdots \\ x \\ 1 \end{pmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{pmatrix} x^n \\ x^{n-1} \\ \vdots \\ x^2 \\ x \end{pmatrix}}_{x\boldsymbol{b}}. \tag{7.13}$$

Equation (7.13) shows that any root of f must be an eigenvalue of the companion matrix $\boldsymbol{C}$, and it may also be verified that the characteristic polynomial of $\boldsymbol{C}$ equals f. Furthermore, the eigenvectors equal (up to scale) the vector $\boldsymbol{b}$ evaluated at the corresponding root. The eigenvalues and eigenvectors can be computed stably and efficiently using methods from numerical linear algebra, making this a popular technique.

Note also that using any of the monomial orderings introduced above, $\langle LT(I(f))\rangle = \langle\{x^m : m \geq n\}\rangle$, and that $\boldsymbol{b}$ consists of its complement, thus forming a basis for $\mathbb{C}(x)/I(f)$ by Proposition 7.7. This corresponds to the fact that any function defined only on the n roots of f can be expressed as a polynomial of degree $n - 1$.

7.2.2 The Action Matrix

The companion matrix technique can be extended to the multivariate case, as first shown by Lazard (1981). Let $I = \langle f_1, \dots, f_m\rangle$ be an ideal generated by a polynomial system with a finite number of solutions $|V| = r$. Consider the linear operator on $\mathbb{C}(\boldsymbol{x})/I$

$$T_a \colon f(\boldsymbol{x}) \mapsto a(\boldsymbol{x})f(\boldsymbol{x}) \tag{7.14}$$

parametrized by some $a(\boldsymbol{x}) \in \mathbb{C}(\boldsymbol{x})$, called the *action polynomial.* Proposition 7.7 now implies that there exists a finite linear basis $\boldsymbol{b}$ for the quotient space, and therefore T_a can be represented by an r-by-r matrix $\boldsymbol{M}_a$, called the *action matrix*. Any member of the quotient space can be written $f = \boldsymbol{c}^\top \boldsymbol{b}$ where $\boldsymbol{c} \in \mathbb{C}^r$ is a vector of coefficients, and so $T_a(f) = (\boldsymbol{M}_a\boldsymbol{c})^\top \boldsymbol{b} = \boldsymbol{c}^\top \boldsymbol{M}_a^\top \boldsymbol{b}$. Since this holds for all $\boldsymbol{c}$, we must have

$$a(\boldsymbol{x})\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{M}_a^\top \boldsymbol{b}(\boldsymbol{x}) \tag{7.15}$$

for any $\boldsymbol{x} \in V$ which shows that the eigenvalues and eigenvectors of $\boldsymbol{M}_a^\top$ correspond to $a(\boldsymbol{x})$ and $\boldsymbol{b}(\boldsymbol{x})$ evaluated at the solutions, respectively. This is in analogy with the univariate case, where the companion matrix encodes the mapping T_x.

7.2.3 The Action Matrix Method

If we can find an action matrix $\boldsymbol{M}_a$ for some polynomial a, and the variables $\boldsymbol{x}$ can be easily extracted from the basis monomials $\boldsymbol{b}$, solving the system boils down to an eigenvalue problem. In practice, the action polynomial is chosen as a simple monomial, usually just one of the variables, and will henceforth be referred to as the action variable.

To find an action matrix for a given polynomial system with r solutions we use a technique called single elimination with basis selection (Faugère 2002; Byröd et al. 2008; Byröd et al. 2009). The first step is to expand the given set of equations by multiplying each polynomial with a set of monomials. Exactly which and how many monomials to choose is highly problem-specific and a bit of a dark art. The expanded system is written on matrix form as

$$\boldsymbol{C}\boldsymbol{M} = \mathbf{0}, \tag{7.16}$$

where $\boldsymbol{C}$ is a matrix of coefficients and $\boldsymbol{M}$ a vector of monomials. Given an action variable a, the monomials $\mathcal{M}$ in $\boldsymbol{M}$ are partitioned into three sets, $\mathcal{M} = \mathcal{E} \cup \mathcal{R} \cup \mathcal{P}$. The set $\mathcal{P} = \{\boldsymbol{x}^{\boldsymbol{\alpha}} \,:\, a\boldsymbol{x}^{\boldsymbol{\alpha}} \in \mathcal{M}\}$ contains the monomials which stay in $\mathcal{M}$ on multiplication with a, called the *permissible* set. The set $\mathcal{R} = a\mathcal{P} \setminus \mathcal{P}$ is called the *reducible* monomials, and $\mathcal{E} = \mathcal{M} \setminus (\mathcal{P} \cup \mathcal{R})$ the *excessive*. With this partition we can write (7.16) as

$$\begin{pmatrix} \boldsymbol{C}_{\mathcal{E}} & \boldsymbol{C}_{\mathcal{R}} & \boldsymbol{C}_{\mathcal{P}} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}_{\mathcal{E}} \\ \boldsymbol{M}_{\mathcal{R}} \\ \boldsymbol{M}_{\mathcal{P}} \end{pmatrix} = \mathbf{0}. \tag{7.17}$$

The goal is to find a subset $\mathcal{B} \subset \mathcal{P}$ containing a basis for the quotient space defined by the polynomial system, and a linear mapping expressing the corresponding reducible monomials in $\mathcal{R}$ in terms of $\mathcal{B}$.

Using QR factorization of the coefficient matrix, the system is put on the form

$$\begin{pmatrix} \boldsymbol{U}_{\mathcal{E}_1} & \boldsymbol{C}_{\mathcal{R}_1} & \boldsymbol{C}_{\mathcal{P}_1} \\ \mathbf{0} & \boldsymbol{U}_{\mathcal{R}_2} & \boldsymbol{C}_{\mathcal{P}_2} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}_{\mathcal{E}} \\ \boldsymbol{M}_{\mathcal{R}} \\ \boldsymbol{M}_{\mathcal{P}} \end{pmatrix} = \mathbf{0}, \tag{7.18}$$

where $\boldsymbol{U}_{\mathcal{E}_1}$ and $\boldsymbol{U}_{\mathcal{R}_2}$ are upper triangular. Having eliminated the dependence on the excessive monomials from some of the equations, the top rows of the matrix are discarded. This leaves the system

$$\boldsymbol{U}_{\mathcal{R}_2}\boldsymbol{M}_{\mathcal{R}} = -\boldsymbol{C}_{\mathcal{P}_2}\boldsymbol{M}_{\mathcal{P}}, \tag{7.19}$$

and if it can be solved for $\boldsymbol{M}_{\mathcal{R}}$ in terms of $\boldsymbol{M}_{\mathcal{P}}$ the action matrix can be easily constructed. In general, the set $\mathcal{P}$ is much larger than the basis dimension, so there is no need to find the mapping for *all* the monomials in $\mathcal{R}$. Indeed, $\boldsymbol{U}_{\mathcal{R}_2}$

is often not even of full rank. Using QR factorization with column pivoting (see e.g. Golub and Van Loan 1989), the columns of $\boldsymbol{U}_{\mathcal{R}_2}$ can be heuristically partitioned into a well-conditioned set $\boldsymbol{U}_{\mathcal{B}}$ and a set linearly dependent on the others. The monomials corresponding to the first k well-conditioned columns are selected as the set $\mathcal{B}$. Discarding the rest of $\mathcal{P}$, the reducible monomials $a\mathcal{B} \setminus \mathcal{B}$ can now be solved for in terms of $\mathcal{B}$.

For the above algorithm to succeed, the set $\mathcal{B}$ must contain a basis for the quotient space, but there are no theoretical results guaranteeing this will happen or even that it is possible to find r independent columns in $\boldsymbol{U}_{\mathcal{R}_2}$. In practice, however, given a sufficiently expanded equation set and the right size selected for $\mathcal{B}$, the method works. If $|\mathcal{B}| > r$, not all eigenvalues and vectors of the action matrix can correspond to solutions of the original polynomial system, but if $\mathcal{B}$ contains a basis the solutions are guaranteed to be a subset. The incorrect solutions are then filtered out by checking in the equations.

Speed

The action matrix size depends on k which for reasons of speed should be chosen as small as possible to give a smaller eigenvalue problem. However, the costliest step is often the first linear elimination, and the smaller expanded equation set one can get away with, the better. If the equation system exhibits certain kinds of symmetry, this can be used to reduce the problem size further, as shown in Ask (2014). For example, if a variable x only appears in even powers in the equations, an implicit substitution for x^2 can be used to cut the basis size in half. In addition, the monomial ordering used within the partitioned sets $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{E}$ can impact the amount of fill-in which occurs using sparse QR factorization. We have found that using the GREVLEX ordering usually gives less fill-in and faster solvers. More advanced algorithms like SuiteSparseQR (Davis 2011) can compute their own fill-reducing orderings, and then the original monomial ordering has little impact. Recently, sparse QR factorization has been implemented on graphics processors with reported ten-fold speedups compared to multicore CPU implementations (Yeralan et al. 2013), which promises even faster solvers.

7.2.4 Polynomial Eigenvalue Problems

A slightly less versatile, but sometimes faster and always simpler method for solving a polynomial system, is to formulate it as a *polynomial eigenvalue problem*

(PEP). Any system of polynomial equations $f_i(\boldsymbol{x}) = 0$ may be written

$$(\lambda^n \boldsymbol{A}_n + \lambda^{n-1}\boldsymbol{A}_{n-1} + \ldots + \lambda \boldsymbol{A}_1 + \boldsymbol{A}_0)\boldsymbol{M} = \boldsymbol{0}, \tag{7.20}$$

where the $\boldsymbol{A}_k$ are coefficient matrices, $\lambda = x_j$ is called the *hidden* variable and $\boldsymbol{M}$ consists of monomials in $x_i,\ i \neq j$. For example, choosing x as the hidden variable, the system

$$\begin{aligned} x^2 + yz &= 1 \\ xz &= 2 \\ xyz &= 3 \end{aligned} \tag{7.21}$$

can be written

$$\left[x^2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + x \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 0 \\ -3 & 0 & 0 \end{pmatrix}\right] \begin{pmatrix} 1 \\ z \\ yz \end{pmatrix} = \boldsymbol{0}.$$

Note that the matrices $\boldsymbol{A}_k$ are not in general square, but this can be achieved by generating additional equations by multiplying with monomials, as in the action matrix method. Then (7.20) can be written

$$\boldsymbol{A}\boldsymbol{y} = \lambda \boldsymbol{B}\boldsymbol{y} \tag{7.22}$$

where

$$\begin{aligned} \boldsymbol{A} &= \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\boldsymbol{A}_0 & -\boldsymbol{A}_1 & -\boldsymbol{A}_2 & \cdots & -\boldsymbol{A}_{n-1} \end{pmatrix} \\ \boldsymbol{B} &= \begin{pmatrix} \boldsymbol{I} & & & \\ & \ddots & & \\ & & \boldsymbol{I} & \\ & & & \boldsymbol{A}_n \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} \boldsymbol{M} \\ \lambda \boldsymbol{M} \\ \vdots \\ \lambda^{n-1}\boldsymbol{M} \end{pmatrix}. \end{aligned} \tag{7.23}$$

We recognize (7.22) as a standard generalized eigenvalue problem. If either $\boldsymbol{A}_0$ or $\boldsymbol{A}_n$ has full rank, the problem can be transformed to an ordinary eigenvalue problem, and the solutions extracted from the eigenvalues and vectors; if not, it is potentially ill-posed and solutions may not exist. The applicability of this method therefore depends on the ability to generate equations satisfying this condition, which is not always possible (at least in practice). For a more thorough introduction to this approach, see Kukelova et al. (2012).

A related technique, simply called the "hidden variable method", uses the fact that unless the system only has the trivial solution $\boldsymbol{x} = \mathbf{0}$, the sum $\lambda^n \boldsymbol{A}_n + \ldots + \boldsymbol{A}_0$ must be singular and its determinant zero. The determinant is a univariate polynomial in λ which can be solved for the hidden variable. Recursive application can then determine the values of the remaining variables. While this method has been applied successfully to some problems (see e.g. H. Li 2006), for large systems simply expanding the determinant can be infeasible and H. Li and Hartley (2006) had to resort to symbolic computations to successively simplify the expression, making the algorithm very slow. Also, the determinant may turn out to be identically zero and then the method does not apply.

Speed

The eigenvalue problems arising are potentially much larger than those in the action matrix method, since we need to form the block matrices in (7.23). On the other hand, it may not be necessary to generate as many new equations, and since there is no need for QR factorization the PEP formulation can in fact be an order of magnitude faster, as we shall see in the next chapter.

7.2.5 Finding the Number of Solutions

To successfully apply the action matrix method, we need to know how many solutions a given polynomial system has. Algebraic geometry software such as Macaulay2 (Grayson and Stillman) or Maple can be used to compute a basis for the quotient space, thus revealing its dimension. This is done by computing a Gröbner basis for the ideal using e.g. Buchberger's algorithm, but this is highly sensitive to round-off errors and therefore all computations are performed using exact rational arithmetic. If the coefficients of the polynomial system are given as inexact floating-point numbers, problem-specific correlations between the coefficients may not be detected and a larger basis than necessary is returned. In some cases, problem instances with integer coefficients can be constructed to avoid this, but with equations deriving from Snell's law this is difficult to achieve; the square root in (7.6) means coefficients are likely to be irrational. As a consequence, in all but the simplest cases we will only be able to give upper bounds on the number of solutions.

7.2.6 Other Methods

Action matrix and PEP-based algorithms have been the most successful when applied to computer vision problems, but there are many other methods for solving polynomial systems, see e.g. Dickenstein and Emiris (2010) and references therein. Some are fast but not very flexible and only apply to a limited class of systems, such as *extended linearization* (Courtois et al. 2000). Others are more powerful but too slow for use in a solver intended for real-time use, such as *homotopy continuation*, but with the continued development of efficient algorithms such as PHCpack (Verschelde 1999) this may be possible in the future.

Chapter 8

Absolute Pose under Refraction

Refractive structure-and-motion problems come in many varieties, depending on what relationships between scene structure, cameras and refractive planes are known. In many applications, such as in underwater photography where the camera views the world through a waterproof housing, the relationship between the camera and the glass can be pre-calibrated and assumed known. The back-projections of image points through and past the refractive interface can then be precomputed in the camera's coordinate system, and the whole assembly acts as an axial camera, as shown in A. Agrawal et al. (2012). Axial cameras are a special case of generalized cameras and algorithms previously developed for these can be used. For example, absolute pose for generalized cameras was solved minimally in Nistér (2004) using three points, and relative pose in Stewénius et al. (2005) using six points (as reported in Kim et al. (2010), that algorithm degenerates for some axial camera configurations, but not in this case). To our knowledge, optimal two-view triangulation under refraction has not been solved, but standard linear methods are of course applicable when the back-projected rays are known. These three components along with refractive bundle adjustment (Jordt-Sedlazeck and Koch 2013) are the building blocks of a structure-and-motion system, and for the practically important case of known camera–refractive plane pose the problem can be considered more or less solved.

In Chari and P. F. Sturm (2009) a theory is presented for the multiple-view geometry of cameras observing a scene through a common refractive interface. The existence of refractive projection, fundamental and homography matrices is shown and the relative pose problem is solved under very specific conditions. In Kang et al. (2012) the relative translation problem is solved optimally under the L_∞-norm, given the camera orientations.

In this chapter we consider the problem of determining absolute pose of a camera observing known scene structure through a single known refractive planar interface. A more general problem was solved by A. Agrawal et al. (2012)

in the context of calibration, where the relative poses of camera, scene structure and refractive plane are all unknown, and indeed even the ratio of refractive indices. However, the algorithm requires at least eight point correspondences and has unnecessary degrees of freedom if the relative pose between scene structure and refractive plane is in fact known. In Chang and T. Chen (2011) the absolute pose problem is solved minimally with two point correspondences given that the camera's vertical direction is known, as given by an accelerometer. However, it is also assumed that the refractive plane is horizontal, significantly simplifying the problem. We present a minimal solution for the same case but with arbitrary refractive plane, and analyze the closed-form solutions to the known-orientation case. We also present a non-minimal algorithm for the general case using five points, and extend it to the case of unknown camera focal length using six points.

8.1 Snell's Law

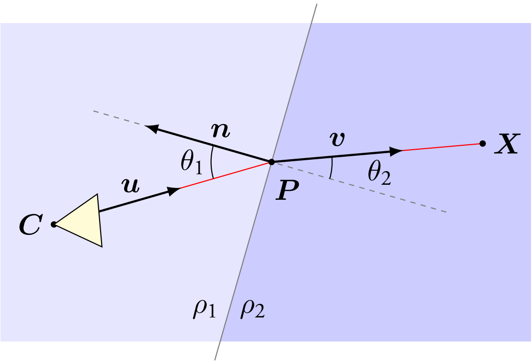


Figure 8.1: The image ray $(\boldsymbol{C}, \boldsymbol{u})$ from the camera center intersects the plane with normal $\boldsymbol{n}$ at $\boldsymbol{P}$ and is refracted into the ray $(\boldsymbol{P}, \boldsymbol{v})$ according to Snell's law.

As we saw in the introductory chapter, refraction of light at an optical medium boundary is described by Snell's law

$$\rho_1 \sin\theta_1 = \rho_2 \sin\theta_2, \tag{8.1}$$

where $\rho_{1,2}$ are the refractive indices of the two media and $\theta_{1,2}$ the angles the impinging and refracted ray make with the surface normal (see Figure 8.1). It was also noted that the impinging ray with direction vector $\boldsymbol{u}$, the refracted ray $\boldsymbol{v}$ and the plane normal $\boldsymbol{n}$ must all lie in the same plane. Using properties of the cross product, Snell's law may then be expressed on vector form as

$$\rho_1 \frac{\boldsymbol{u} \times \boldsymbol{n}}{\|\boldsymbol{u}\|\|\boldsymbol{n}\|} = \rho_2 \frac{\boldsymbol{v} \times \boldsymbol{n}}{\|\boldsymbol{v}\|\|\boldsymbol{n}\|} \tag{8.2}$$

or equivalently

$$\mu\|\boldsymbol{v}\|(\boldsymbol{u} \times \boldsymbol{n}) = \|\boldsymbol{u}\|(\boldsymbol{v} \times \boldsymbol{n}), \tag{8.3}$$

where $\mu = \rho_1/\rho_2$. Note that by squaring both sides component-wise we obtain three equations which are polynomial in all variables. However, since both sides of (8.3) are orthogonal to $\boldsymbol{n}$ only two of the equations can be independent. The co-planarity constraint on the rays and normal can also be written as $\boldsymbol{u} \times \boldsymbol{v} \cdot \boldsymbol{n} = 0$, independently of the refractive indices. It is obvious that the camera center $\boldsymbol{C}$ and scene point $\boldsymbol{X}$ must also lie in this plane, implying

$$\boldsymbol{u} \times (\boldsymbol{X} - \boldsymbol{C}) \cdot \boldsymbol{n} = 0. \tag{8.4}$$

In Section 7.1 we also derived an expression for the refracted ray direction given $\boldsymbol{u}$ and $\boldsymbol{n}$,

$$\boldsymbol{v} = \mu\boldsymbol{u} + \big(r\cos\theta_1 - \operatorname{sign}(\cos\theta_1)\cos\theta_2\big)\boldsymbol{n} \tag{8.5}$$

where

$$\begin{aligned} \cos\theta_1 &= -\boldsymbol{n} \cdot \boldsymbol{u} \\ \cos\theta_2 &= \sqrt{1 - \mu^2(1 - \cos^2\theta_1)}\,. \end{aligned} \tag{8.6}$$

The sign function in (8.5) is needed in case the normal $\boldsymbol{n}$ is given pointing away from the camera.

In what follows, we will usually assume that the intrinsic parameters of the camera are known so that the back-projected ray direction $\boldsymbol{u}$ of an image point can be computed in the world coordinate system, given only the camera's orientation and translation. However, in the six-point algorithm presented in Section 8.6, the camera focal length is assumed unknown.

Furthermore, for the minimal solvers we only consider the case of a single flat refractive interface, since explicitly modeling two refractions using Snell's law leads to significantly more complex equations. For example, as shown in

Glaeser and Schröcker (2000), computing the forward projection of a scene point into a camera through one refractive plane amounts to finding the roots of a fourth-degree polynomial, while A. Agrawal et al. (2012) show doing the same for two parallel planes gives a 12^{th} degree polynomial. The proposed non-minimal five- and six-point solvers only use the co-planarity constraints (8.4) and thus handle multiple refractions as long as all interfaces are parallel. In real applications there are usually two refractions, e.g. air–glass and glass–water, but if the glass is thin compared to the scene scale this is well-approximated by a single air–water refraction. This approximation is validated experimentally in Section 8.7.

8.2 Unknown Camera Translation

We start with the simpler problem of only finding the translation of the camera, with the orientation given. Snell's law (8.3) gives three equations per point, but since the projection only has two degrees of freedom, they are not independent and only give two constraints. To solve for the translation, one and a half points i.e. three coordinates, are thus needed; with two point matches, the extra constraint can be used to determine the refractive index ratio.

By coordinate transformation we may assume the refractive plane is described by $z = 0$, and that the image ray directions $\boldsymbol{u}$ have been normalized to unit length and rotated into the global coordinate frame using the camera's known orientation. The intersection of the ray from the camera center $\boldsymbol{C}$ in the direction $\boldsymbol{u}$ with the plane is then given by

$$\boldsymbol{P} = \boldsymbol{C} - \frac{C_z}{u_z}\boldsymbol{u} \tag{8.7}$$

and the refracted ray $\boldsymbol{v} = \boldsymbol{X} - \boldsymbol{P}$ where $\boldsymbol{X}$ is the corresponding known 3D point. Substituting this into (8.3) gives

$$\begin{aligned}\mu(\boldsymbol{u} \times \boldsymbol{n})\|\boldsymbol{X} - \boldsymbol{C} + (C_z/u_z)\boldsymbol{u}\| = \\ \big(\boldsymbol{X} - \boldsymbol{C} + (C_z/u_z)\boldsymbol{u}\big) \times \boldsymbol{n}.\end{aligned} \tag{8.8}$$

Multiplying through by u_z, squaring both sides component-wise and using that $\boldsymbol{n} = (0, 0, 1)^\top$ we obtain two polynomial equations (the z-component is identically zero) in the components of $\boldsymbol{C}$ and the squared ratio μ^2. The co-planarity constraint $(\boldsymbol{u} \times \boldsymbol{n}) \cdot (\boldsymbol{X} - \boldsymbol{C}) = 0$ provides two linear equations in the variables C_x and C_y which can be uniquely solved for as long as the two image rays $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are not parallel (i.e. the image points are distinct). Substituting these values into Snell's law (8.8) gives two independent equations, linear in μ^2 and quadratic in C_z. Eliminating μ we obtain a quartic equation in C_z. The four solutions thus differ in the perpendicular distance C_z to the refractive plane and the refractive index ratio. If the ratio μ is known, this can be directly substituted into one of the equations quadratic in C_z, yielding two solutions. Given perfect data, the two equations have one common root corresponding to the true solution, but the other roots might not agree. This is possible because not all solutions returned are physically correct.

Note that Snell's law as stated in (8.1) only specifies the angle the refracted ray makes with the normal, but not on which side (see Figure 8.2), nor that the two rays should be on different sides of the plane. There is thus an ambiguity in the equations leading to incorrect solutions; some cases are illustrated in Figure 8.3. In the known refractive index case, experiments indicate one of the solutions is always physically incorrect, while in the unknown index case there can be between one and three valid solutions. False solutions can be filtered by checking if the back-projection ray given by (8.5) intersects the scene point.

This example illustrates a major difficulty in designing minimal solvers for refractive problems, namely an abundance of solutions which grows with the number of point matches required. When the orientation is not given the polynomial equations become much more involved and closed-form solutions are not possible.

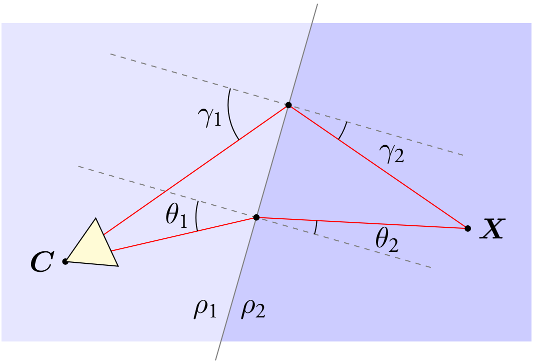


Figure 8.2: Ambiguity in Snell's law giving rise to false solutions. Both $\rho_1 \sin\theta_1 = \rho_2 \sin\theta_2$ and $\rho_1 \sin\gamma_1 = \rho_2 \sin\gamma_2$ are fulfilled.

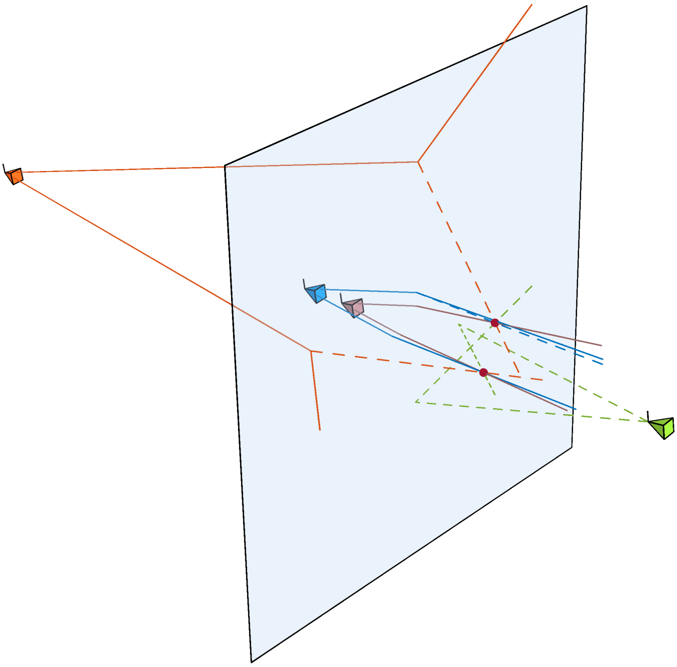


Figure 8.3: Four solutions to the known orientation, unknown refractive index case, three of which are incorrect due to ambiguities in the equations. Solid lines are the physical back-projections of the image points while dashed lines illustrate spurious optical paths consistent with (8.3) giving rise to false solutions.

8.2.1 Unknown Refractive Plane

The problem of determining the refractive plane given the camera pose and the scene structure is very similar to the known-orientation pose problem. Using the co-planarity constraints, the x- and y-components of the plane normal can be solved for linearly in terms of the z-component using two image correspondences. Using Snell's law, two equations of degree three in the plane translation and refractive ratio can then be obtained. This gives three solutions of which only one is physically correct.

8.3 The 2D Case

In two dimensions there is only one rotation angle for the camera orientation which greatly simplifies the equations. The coordinate system may be transformed so that the known refraction interface, now just a line, coincides with the y-axis. The intersection point $\boldsymbol{P}$ of an image ray $\boldsymbol{u}$ with the line is then given by

$$\boldsymbol{P} = \boldsymbol{C} - \frac{C_y}{\boldsymbol{R}_{:,2} \cdot \boldsymbol{u}} \boldsymbol{R}^\top \boldsymbol{u}, \tag{8.9}$$

where

$$\boldsymbol{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{8.10}$$

and $\boldsymbol{R}_{:,2}$ denotes the second column of the camera's rotation matrix. Embedding in 3D and plugging into Snell's law (8.3) now only provides one constraint per projection, so three points are needed. Multiplying with the denominator in $\boldsymbol{P}$ and squaring as before, we obtain three polynomials of total degree six in the variables C_x, C_y, $c = \cos\theta$ and $s = \sin\theta$, and add the constraint $c^2 + s^2 = 1$.

Analysis of the resulting system using algebraic geometry tools shows it may have up to 96 solutions. However, there is symmetry in the equations; s and c only occur in even powers meaning that if (C_x, C_y, c, s) is a solution, so is $(C_x, C_y, -c, -s)$. This corresponds to the fact that rotating the camera 180° results in the same image projections in 2D. This symmetry can be exploited when solving the system using the action matrix method (Ask et al. 2012), essentially allowing us to solve for only half of the solutions giving an effective basis size of 48. We multiply the four original equations with all monomials which are products of the "basis monomials" c^2, s^2, cs, C_x, C_y s.t. the total degree does not exceed 7 and the degrees of the variables c, s, C_x, C_y do not exceed 6, 5, 7 and 3 respectively (of course this assignment is symmetric in c and s and C_x and C_y). This results in 1272 equations in 1484 monomials. Using c^2 as the action monomial then allows us to solve for C_x, C_y, c^2, s^2 and cs. We also know that $c > 0$, since otherwise the optical axis is pointing away from the refractive plane, so the sign of s can be determined from cs.

Experiments indicate that there is rarely more than one or two physically correct solutions among the 48. An optimized Matlab implementation of the solver runs in 80 ms including Newton steps to refine the solutions. However, since the 2D case is of limited practical importance we will not focus on this solver any further.

8.4 Known Rotation Axis

The problem in 3D of known translation and one degree of rotational freedom is similar to the pure 2D pose problem, which is a special case. This problem formulation arises when e.g. the camera's elevation and roll angle can be determined using an accelerometer, such as in a mobile phone, but the azimuth is unknown. A special case was considered in Chang and T. Chen (2011) where it was assumed that the refractive plane is horizontal, i.e. that the normal is known.

We transform the coordinate system so that the unknown rotation axis is parallel with the y-axis. If the refractive plane is described by $\boldsymbol{n} \cdot \boldsymbol{X} + d = 0$ the intersection with an image ray is given by

$$\boldsymbol{P} = \boldsymbol{C} - \frac{\boldsymbol{n} \cdot \boldsymbol{C} + d}{\boldsymbol{n} \cdot (\boldsymbol{R}^\top \boldsymbol{u})} \boldsymbol{R}^\top \boldsymbol{u}, \tag{8.11}$$

where

$$\boldsymbol{R} = \boldsymbol{R}_{xz} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \tag{8.12}$$

is the camera rotation matrix, decomposed into the known elevation and roll and the unknown y-axis rotation. Four degrees of freedom means two projections are required to solve the minimal case, and as usual plugging into Snell's law gives four independent polynomial equations after multiplying with the denominator in $\boldsymbol{P}$ and squaring. Along with the constraint $\cos^2\theta + \sin^2\theta = 1$, these are enough to solve the problem using the action matrix technique, and analysis with Macaulay2 shows there could be up to 64 solutions. However, there is no longer symmetry in the rotation parameters that can be used to reduce the basis size, and it turns out the equation set has to be expanded to thousands of polynomials, yielding a slow solver ($\sim$1 s). Instead we change the rotation parametrization to that used in Kukelova et al. (2010) and Chang and T. Chen (2011), letting

$$q = \tan(\theta/2) \tag{8.13}$$

giving

$$\begin{aligned} \cos\theta &= (1-q^2)/(1+q^2) \\ \sin\theta &= 2q/(1+q^2)\,. \end{aligned} \tag{8.14}$$

The resulting system can now be solved as a polynomial eigenvalue problem. To transform it to PEP form the equation set has to be expanded, but we use a simpler strategy than the resultant-based method proposed in Kukelova et al. (2012). We expand the original system, consisting of all six equations from (8.3) and both co-planarity constraints (8.4), to 32 equations by multiplying with monomials C_x, C_y and C_z. It may seem unnecessary to include the co-planarity constraints since they are implicit in Snell's law, but in transforming (8.3) to polynomial form, some information is lost which is retained in (8.4), further constraining the solutions. Hiding the variable q we obtain a matrix polynomial equation

$$(q^8\boldsymbol{A}_8 + q^7\boldsymbol{A}_7 + \ldots + q\boldsymbol{A}_1 + \boldsymbol{A}_0)\boldsymbol{M} = \mathbf{0} \tag{8.15}$$

of degree eight where the $\boldsymbol{A}_i$ are 32-by-20 matrices and $\boldsymbol{M}$ a vector of 20 monomials in $C_{x,y,z}$. To convert this to a PEP the $\boldsymbol{A}_i$ must be made square without losing rank, and this is accomplished by left-multiplying (8.15) by a random 20-by-32 matrix or simply by $\boldsymbol{A}_1^\top$, similar to what was done in Fitzgibbon (2001). We could use any of the matrices $\boldsymbol{A}_{1,\ldots,7}$ but not $\boldsymbol{A}_0$ or $\boldsymbol{A}_8$ since these may be rank deficient for certain specific input data (though never simultaneously).

An upper bound on the number of solutions to the original system is found to be 112, while the PEP returns up to $8 \cdot 20 = 160$ solutions. Of these only a subset fulfill the equations and only a handful provide physically correct solutions.

8.4.1 Degeneracies

As noted in Section 8.3, the 2D case requires a minimum of three points. This means that if the scene points lie in the plane spanned by the camera up vector and the refractive plane normal, the translation cannot be solved for using only two points. Note that this is a degeneracy shared with the non-refractive case.

The solver itself also exhibits degeneracies even when the problem is well-posed; if the 3D points $\boldsymbol{X}_{1,2}$, the plane normal $\boldsymbol{n}$ and the camera center lie in the same plane, or if $\boldsymbol{X}_1 - \boldsymbol{X}_2$ is parallel with $\boldsymbol{n}$, the solver will fail. These conditions are however unlikely to be exactly fulfilled by real data. In addition, the parametrization chosen has a singularity at $\theta = 180^\circ$, which in most situations can be avoided by suitable rotation of the coordinate system around the y-axis.

Interestingly, as the true solution angle approaches the singularity, the equations become extremely sensitive. We have found that even for solution angles as far away as $\theta = 160^\circ$, accurately evaluating the polynomials is not possible. For an angle of 179°, a disturbance on the order of $\epsilon_{\text{mach}} \approx 10^{-16}$ changes the function value by orders of magnitude. This seems to be due entirely to floating-point cancellation effects, as the polynomial coefficients remain bounded and relatively small. Nevertheless, the proposed solver still returns accurate results, but they cannot be verified by checking in the equations. The solutions can still be filtered using (8.5), but for example directly refining the solutions using Newton's method is not possible in these cases.

8.5 Absolute Pose with Five Points

Three points are minimal for the general absolute pose problem, but using the same approach as above the equations become too difficult, with thousands of terms. As was noted in A. Agrawal et al. (2012), much information can be gained using only the co-planarity constraints (8.4), given enough point correspondences. In that paper, absolute pose and refractive plane parameters are solved for linearly using eleven points, and with eight points using a clever application of a minimal solver for the standard five-point relative pose problem. This is afforded by the relative simplicity of the co-planarity constraints compared to the full Snell's law. We therefore solve the absolute pose problem using only the co-planarity constraints, which requires five point correspondences. From these equations all parameters except for the perpendicular distance of the camera to the plane can be recovered.

To simplify the equations, we may assume that the refractive plane normal is parallel with the z-axis, and the co-planarity constraints then take the form

$$\boldsymbol{R}^\top \boldsymbol{u} \times (\boldsymbol{X} - \boldsymbol{C}) \cdot (0,0,1)^\top = 0. \tag{8.16}$$

Note that this equation does not contain C_z. We parametrize the camera rotation matrix $\boldsymbol{R}$ using quaternions $\boldsymbol{q} = (s, \boldsymbol{\omega}^\top)^\top$ so that

$$\boldsymbol{R}(\boldsymbol{q}) = 2(\boldsymbol{\omega}\boldsymbol{\omega}^\top - s[\boldsymbol{\omega}]_\times) + (s^2 - \boldsymbol{\omega}^\top\boldsymbol{\omega})\boldsymbol{I}_{3\times 3}\,. \tag{8.17}$$

$\boldsymbol{R}(\boldsymbol{q})$ is only orthonormal if $\boldsymbol{q}$ has unit length, but all matrix elements scale with $||\boldsymbol{q}||^2$. Since (8.16) is homogeneous in $\boldsymbol{R}$, the unit-length requirement can be dropped and we set the scalar component $s = 1$, as was done in Stewénius et al. (2005). Since both $\boldsymbol{q}$ and $-\boldsymbol{q}$ represent the same rotation, fixing the sign and magnitude in this way gives a minimal parametrization and halves the number of solutions, at the cost of introducing a singularity for all 180° rotations (for which $s = 0$). We now have five polynomial equations of total degree three in five unknowns, and analysis gives an upper bound on the number of solutions as 48. By multiplying with all monomials up to total degree three, 280 equations are obtained and the action matrix method can be applied. As it turns out, correlations in the coefficients of the system means there are in general only 16 solutions. This manifests itself in the action matrix algorithm as a rank deficiency of four when solving for the reduction monomials in terms of the basis monomials. This means there are four monomials which cannot be solved for and must be removed from the basis. Which monomials to remove depends on the data, and can be determined using column-pivoting QR factorization as discussed in Section 7.2.3.

The above solution works independently of the refractive indices or indeed how many refractive layers are traversed, as long as each ray stays in a single plane. Assuming there is only one interface given by $z = 0$ and that the refractive index ratio is known, we can plug the rotation and x- and y-translation into the full Snell's law for a single projection and obtain a quadratic equation for the z-translation. The two roots correspond to the situation in Figure 8.2 and it can be shown that the physical solution is the one with the camera closest to the plane. If there are several parallel refractive planes and/or the refractive indices are unknown, the method presented in A. Agrawal et al. (2012) can be used to solve for the translation, given enough point correspondences.

8.5.1 Degeneracies

If all points lie on a line parallel with the plane normal, the camera can be rotated around this line without changing the image projections (in fact there is even a three-dimensional family of solutions since the constraints are weaker). The solver will also fail if all points and the plane normal lie in a common plane. The singularity of the parametrization can be avoided with high probability by randomly rotating the coordinate system about the z-axis.

8.6 Unknown Focal Length with Six Points

The five-point formulation above is easily extended to the case of unknown camera focal length. Under the same assumptions the co-planarity constraints take the form

$$\boldsymbol{R}^\top \boldsymbol{K}^{-1}\boldsymbol{u} \times (\boldsymbol{X} - \boldsymbol{C}) \cdot (0,0,1)^\top = 0, \tag{8.18}$$

where

$$\boldsymbol{K}^{-1} = \mathrm{diag}(1,1,f) \tag{8.19}$$

and f is the focal length. The extra degree of freedom means six points are required, and analysis of the system gives an upper bound on the number of solutions as 104. Note however that a symmetry has been introduced; changing the sign of the focal length and rotating the camera 180° around the optical axis results in the same image projections. With the chosen parametrization, such a rotation is equivalent to flipping the signs of the first two vector components of the quaternion, and there is thus a two-fold symmetry in the variables f, ω_1 and ω_2.

We obtain 648 equations by multiplying the original six with all monomials which are products of the “basis monomials” f^2, ω_1^2, ω_2^2, $f\omega_1$, $f\omega_2$, $\omega_1\omega_2$, ω_3, C_x, C_y s.t. the degrees of the variables f, ω_1, ω_2, ω_3, C_x, C_y do not exceed 3, 3, 3, 1, 1 and 1 respectively. Choosing ω_3 as the action variable, a basis size of 52 is now sufficient to compute the solution using the same basis selection method as for the five-point solver. Since the monomials f^2, $f\omega_1$ and $f\omega_2$ are in the basis, the correct signs for the rotation components are easily deduced.

8.6.1 Degeneracies

While the five-point solver fails if the plane normal and scene points lie in a common plane, the equations (8.18) of the six-point problem are actually under-determined in this case. The problem as such is still well-posed, but the co-planarity conditions are not enough to constrain the solution.

8.7 Experiments

The solvers were implemented in Matlab, and all experiments run on a 3.0 GHz Core 2 Duo computer. The known-axis solver runs in around 60 ms, most of which is spent solving the rather large PEP problem using Matlab's `polyeig` command. In the action matrix-based solvers most of the time is spent in the elimination step reducing the expanded system, and when using sparse fill-reducing QR factorization the five- and six-point solvers run in around 10 and 20 ms respectively.

We test the solvers' numerical stability using randomly generated problem instances. As seen in Figure 8.4, the known-axis and five-point solvers perform very well with essentially no failure cases, while the six-point solver is somewhat more unstable. To see if the degeneracies inherent to the solvers is problematic, we also generate random degenerate configurations of plane and scene points, and disturb the points slightly by adding normally distributed noise of relative magnitude 10^{-5}. Figure 8.5 shows that the solvers still manage to find solutions in the majority of cases, indicating that the set of problematic problem instances is small and not of practical concern.

Figure 8.6 shows the distribution of the number of real solutions returned by the solvers. Among these, the five- and six-point solvers never returned more than one physically correct solution, and the known-axis and 2D solvers never more than three.

The known-axis solver was derived under the assumption that there is only one refractive interface. In situations where the two media are separated by e.g. a sheet of glass, this assumption introduces an error. To quantify this we conduct a synthetic experiment where a sheet of glass is placed roughly half-way between a camera in air and scene points in water, which are around ten length units apart. The refractive index of air is taken to be unity, water 1.333 and glass 1.5. Figure 8.7 shows the translational and angular error of the pose estimate for varying glass thicknesses and levels of image measurement noise. It is clear that the error introduced by the approximation is small and is dominated by the image measurement error.

8.7.1 Real Data

To validate our algorithms on real data, a Rubik's cube was submerged in a small rectangular acrylic plastic tank with clear sides. The cube was captured by an HTC Desire mobile phone while recording the accelerometer readings. The relative locations of the cube corners and the refractive plane (the tank wall) was measured by ruler, and the image correspondences marked by hand. While the five- and six-point solvers only return one physically valid solution given perfect synthetic data, since they are not minimal several plausible solutions with small reprojection errors may be found with noisy input. To determine the best camera pose from each solver a RANSAC-like approach was used, where minimal sets of corner matches were selected at random, and all valid solutions compared in terms of reprojection error, computed over all points. While the solvers neglect the effect of the tank wall, this is included when computing the reprojection errors. The index of refraction of water was taken as 1.333, and 1.49 for the plastic. In addition to the three proposed solvers, we also solve for the pose while ignoring the refraction effects, i.e. assuming all refractive indices are unity.

Figure 8.8 shows the reprojections of the different algorithms overlaid on two of the images, and their reprojection errors are summarized in Table 8.1. The average error in the focal length returned by the six-point solver was 67 pixels or 2.6% compared with the ground truth camera calibration. The reconstructed camera poses are shown in Figure 8.9. The five- and six-point solutions agree closely while the two-point solution shows translation errors in

Solver	2-pt known axis	5-pt	6-pt	No refraction	Iterative
Error	16.5	7.1	8.3	31.9	5.3

Table 8.1: Average reprojection error magnitude in pixels over several runs with different random seeds to the RANSAC procedure. Images were captured at 2592 × 1952 resolution. The iterative solution minimizes the reprojection error over all points by Levenberg-Marquardt iteration seeded with the five-point solution, and represents a lower bound on the error.

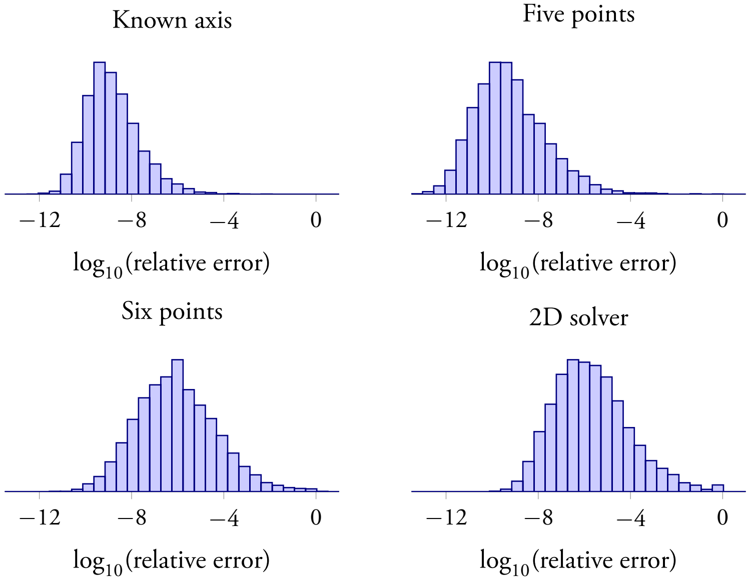


Figure 8.4: Distribution of solver error relative to ground truth, computed over 5000 random problem instances.

the vertical direction, probably due to noisy or biased accelerometer data. The no-refraction assumption clearly leads to large reprojection errors and skewed pose estimates.

8.8 Conclusions

We have presented efficient solutions to several variants of the refractive absolute pose problem. We have shown that the solvers are numerically stable and produce accurate results on real images. There is still room for improvement with regard to numerical stability, particularly for the six-point solver, and with regard to speed. For example, techniques from Kuang and Åström (2012) or Naroditsky and Daniilidis (2011) could be used to reduce the size of the expanded equation sets used in the action matrix method. Degeneracies for the problem setups have also not been thoroughly explored. While the goal of a truly minimal solver in the general case has not yet been reached, the presented algorithms are likely to be faster, enough to compensate for the higher number

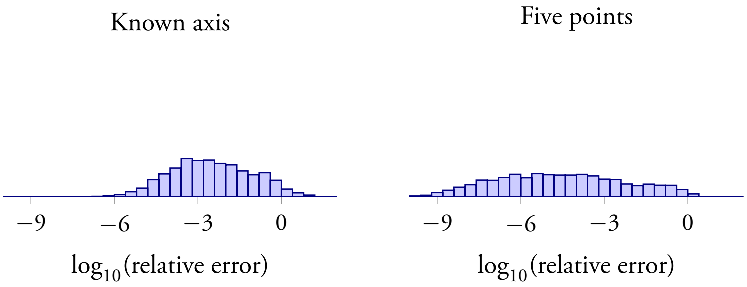


Figure 8.5: Distribution of solver error computed over 5000 random problem instances near degenerate configurations for the solvers.

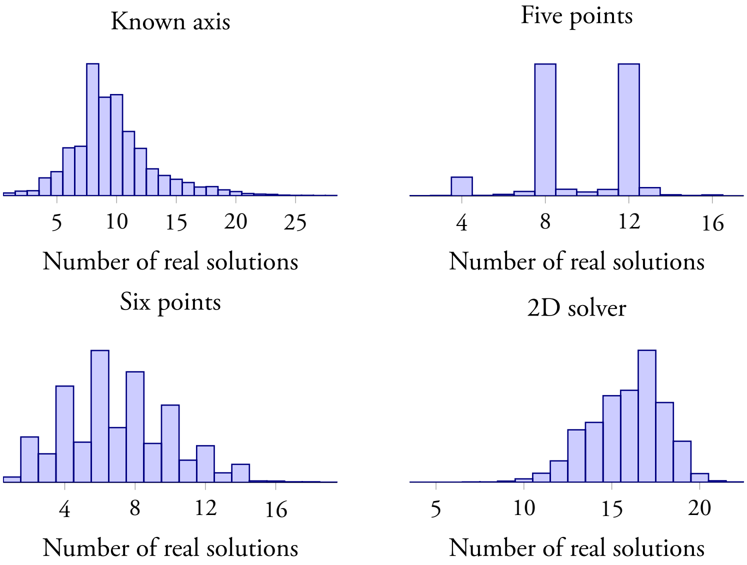


Figure 8.6: Distribution of the number of real solutions returned by the solvers, computed over 5000 random problem instances.

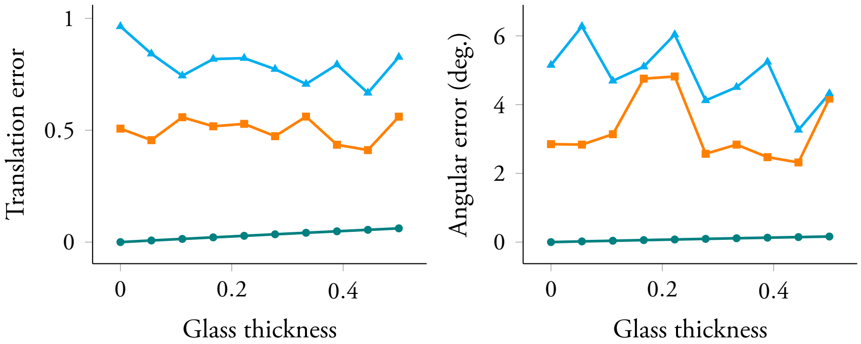


Figure 8.7: Pose error of the known rotation axis solver as a function of glass thickness and image noise, averaged over 100 random problem instances (scene scale approx. 10 units). Bottom, middle and top graphs correspond to zero, one and two pixel std. dev. Gaussian noise respectively.

of iterations required in a hypothesize-and-test framework. The unmanageable size of the polynomials derived from Snell's law in the general case suggests a new approach is needed, where the physical constraints can be enforced to limit the number of solutions. This appears to apply also to the relative pose problem under refraction, where we did not manage to derive polynomial systems of tractable sizes.

Using the techniques developed here it should nevertheless be feasible to extend the six-point solver to include more calibration parameters, such as radial distortion, and solve problems such as optimal two- and three-view triangulation. This is left as future work.

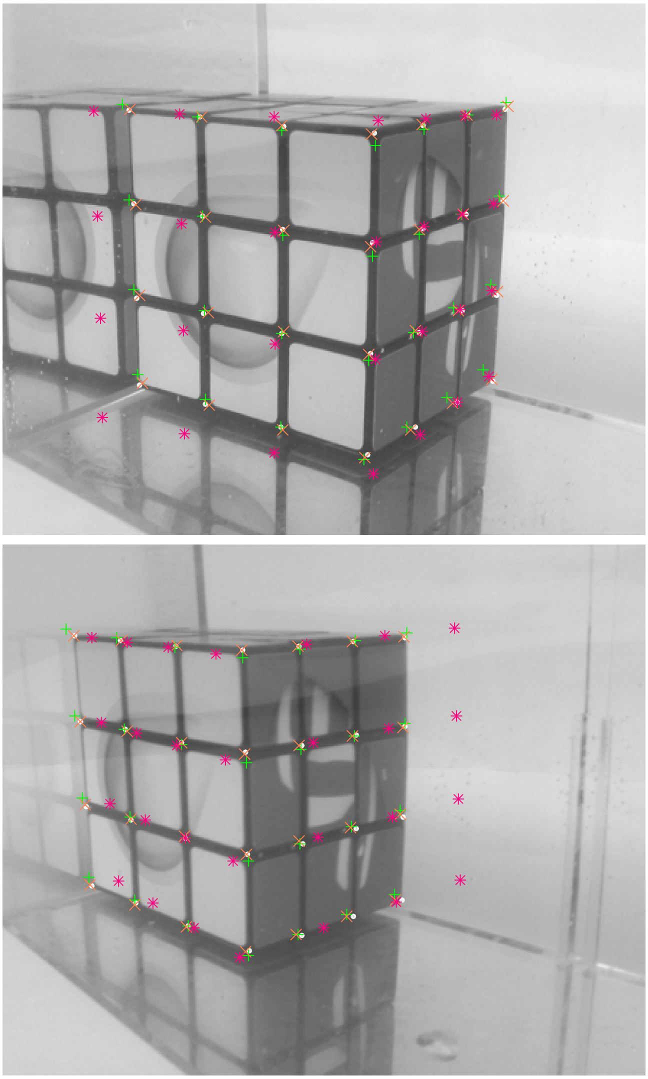


Figure 8.8: Visualization of the reprojection errors in the Rubik's cube experiment. Manual image measurements of the cube corners are shown as white dots. The known-axis solver reprojection is shown as green plus-signs, the five-point solver as orange crosses, and the reference non-refractive solution as magenta stars. The reprojections of the six-point solver are very similar to the five-point solution and are omitted for clarity.

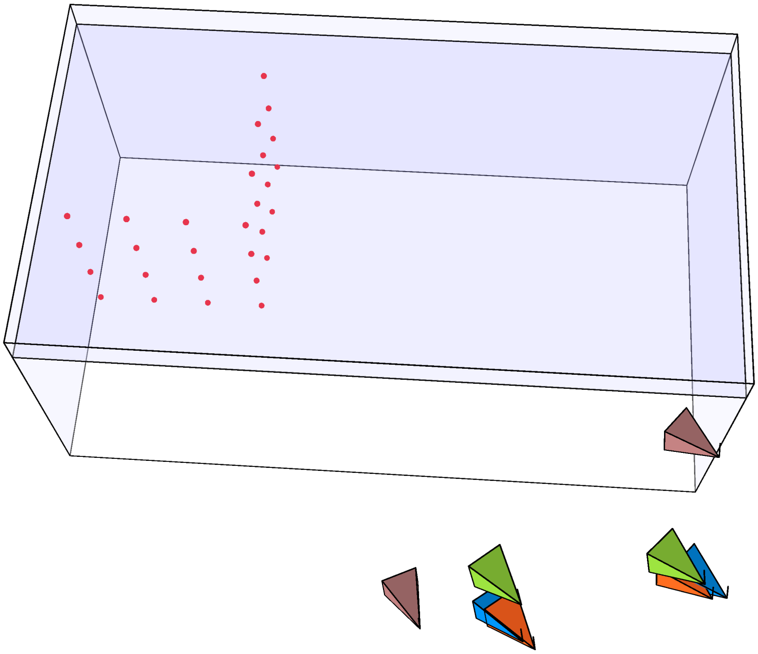


Figure 8.9: Reconstructed poses from two images of the Rubik's cube experiment. Green: known-axis, orange: 5-point, blue: 6-point, mauve: no-refraction solution.

Chapter 9

Refractive Camera Calibration

In competitive swimming, being able to measure and analyze a swimmer's movements during training can be very useful in identifying weaknesses and for tracking progress. In this chapter, we take a very practical approach to computer vision and describe parts of a real-time system designed for this purpose, with focus on the calibration of the camera rig and the problems of underwater imaging arising from refraction effects.

First we consider the problem of calibrating a combined over– and underwater camera setup. Calibration is necessary to be able to compute the swimmer's position from image projections, and to generate synthetic panning views following the swimmer using stationary cameras. In Section 9.2, we describe the process of intrinsic and extrinsic calibration, and how we deal with the problems arising from refraction and reflection at optical media boundaries.

The second problem we consider regards generating visually pleasing images from the cameras. To accomplish this, all geometric distortions must be neutralized along with lens vignetting, chromatic aberration and exposure variations. In Section 9.3, we present practical methods for achieving these goals, and for stitching together images from the stationary cameras allowing synthetic panning shots of the swimmer.

To illustrate the effectiveness of our approach we conclude with full-length stitched over– and underwater panoramas of the pool along with example output from the complete vision system.

Related Work

The literature on camera calibration is vast and spans many decades, but the calibration of cameras under refraction has received little attention until recently. As mentioned in the previous chapter, A. Agrawal et al. (2012) present a method for determining camera pose and refractive plane parameters from a single image of a known calibration object using eight point correspondences. Chang and T. Chen (2011) solve the relative and absolute pose problems of cameras observing structure through a common horizontal flat refractive surface, given that the gravity vector is known, while Jordt-Sedlazeck and Koch (2013) rely on iterative optimization for determining relative and absolute pose when the refractive planes are known relative to the cameras. Kang et al. (2012) solve relative and absolute pose optimally under the L_∞-norm given known rotations. However, in the application considered here, relative poses of the cameras and refractive plane parameters are known to a sufficient degree so that only the absolute pose of a calibration object needs to be computed before non-linear iterative optimization is applied. In Jordt-Sedlazeck and Koch (2013) efficient refractive bundle adjustment is performed using the Gauss-Helmert model; in contrast, our method of computing the forward projection through refractive media allows bundle adjustment using standard non-linear least-squares solvers which typically do not implement equality constraints.

9.1 Camera Setup

The system consists of two rows of cameras mounted along the long edge of a 50 m swimming pool; one row looking down at the surface from above, and one row below the water line observing the pool through glass windows (see Figure 9.1). The cameras are oriented to observe one of the lanes in the pool. All cameras are synchronized so that matches of moving calibration markers between cameras are known to correspond to the same spatial location.

9.2 Calibration

The aim of calibration is to determine the intrinsic and extrinsic parameters of all the cameras in a joint coordinate system. The system at hand presents two major difficulties: the underwater cameras experience refraction effects at the air–glass and glass–water interfaces, and total internal reflection at the water surface means no objects above the surface are visible to the underwater cameras. Likewise, unpredictable refraction effects due to surface waves means observations of underwater objects from above the pool are unreliable.

The calibration process consists of the following steps:

1. Intrinsic in-air calibration of all cameras

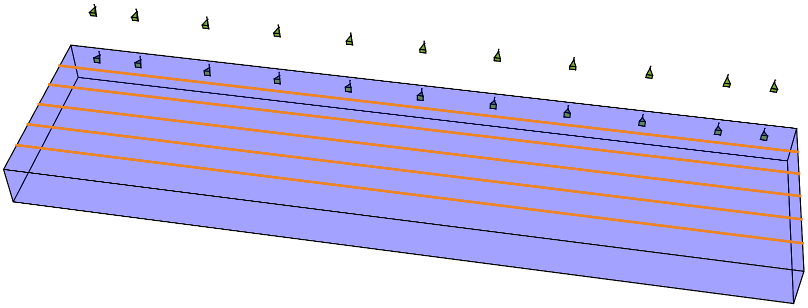


Figure 9.1: The system setup with two rows of cameras along the pool observing the middle lane.

2. Capture and detection of calibration object markers

3. Initialization of calibration object pose and camera extrinsic parameters

4. Parameter refinement using bundle adjustment.

Below, we describe these in more detail.

9.2.1 Intrinsic Calibration

All cameras are calibrated in air, i.e. without the refractive interface, recovering focal length, principal point and lens distortion parameters prior to full system calibration. We use the standard method of Zhang (1999) and the distortion model of Heikkilä and Silvén (1997). The wide-angle lenses used also exhibit a significant degree of vignetting which must be compensated for to produce seamless stitching of the images. We model each color channel of the vignetted image as $I_{\text{vig}}(r, \theta) = I(r, \theta)(1 + c_1 r^2 + c_2 r^4 + c_3 r^6)$ where the origin is taken to be the recovered principal point. The parameters c_i can be estimated independently for each channel using linear least-squares fitting to images taken of evenly lit single-color flat surfaces. For the underwater cameras, we use images of the opposite pool wall for this purpose, taken after the cameras have been mounted. This ensures the vignetting effect produced by Fresnel reflection at the glass interfaces is also accounted for, although it is quite weak. Figure 9.2 shows a result of the method.

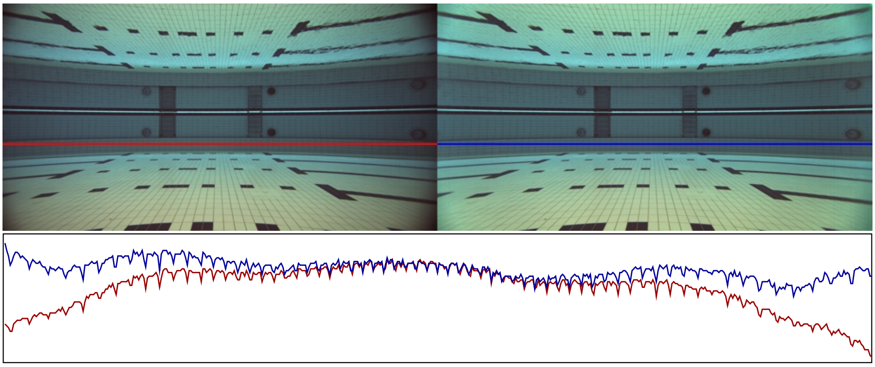


Figure 9.2: Devignetting result. On the left the original image, on the right the devignetted and below the intensity profiles of the indicated lines. The vignetting effect is severe near the corners of the image and cannot be corrected with the adopted model. However, these parts of the image are not used in the panorama generation due to the horizontal overlap between cameras.

9.2.2 The Calibration Object

As mentioned above, total internal reflection and surface waves means no single point can be observed simultaneously by both an underwater and a wall-mounted camera. The solution is to use a semi-submersed known rigid calibration object, different parts of which can be observed simultaneously by the two sets of cameras. The object was chosen as a vertical straight rod with easily recognizable markings at known intervals; we used eight bright-yellow balls, four mounted below and four above a polystyrene foam flotation device. A more elaborate rig with two– or three-dimensional structure would give additional calibration constraints, but also be more unwieldy and difficult to construct and use. The floating rig is towed around the pool while capturing images making sure to cover each camera's field of view.

9.2.3 Marker Detection

The marker detection and localization is performed in three steps. First all pixels are classified based on color content using a support vector machine (SVM) on a transformed color space, then the SVM response map is thresholded and connected components are identified. The obtained regions are then filtered with respect to shape to obtain the final potential marker locations used for

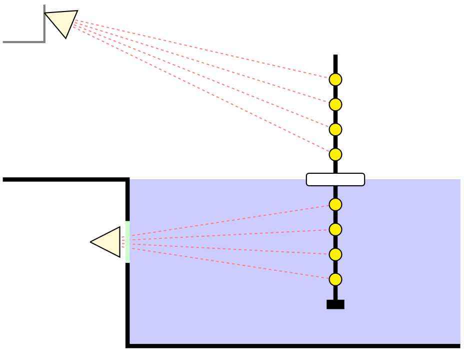


Figure 9.3: The cameras mounted below the waterline only see the bottom half of the calibration object, and the wall-mounted cameras only the top half.

solving the pose problem (see Section 9.2.4).

SVM and Color Transformation

The markers have uniform and distinct color. However, due to light absorption in water, the color in the underwater cameras varies with distance. The available fluorescent lighting above the pool has a fairly narrow spectrum which also makes color differentiation more difficult. In addition, specularities and reflections in the water surface may also appear yellow. For these reasons a linear SVM classifier based only on RGB channels proved insufficient for segmenting the markers. To mitigate the issues of varying lighting conditions, and improve the detection of yellow, the images are converted to the CMYK color space with the non-linear transformation

$$\begin{aligned} K &= \min(1-R, 1-G, 1-B) \\ C &= \frac{1-R-K}{1-K} \\ M &= \frac{1-G-K}{1-K} \\ Y &= \frac{1-B-K}{1-K}\ . \end{aligned} \tag{9.1}$$

To further augment the input data to the SVM, all second order combinations of the chromatic components are added to the feature vector. For each pixel k

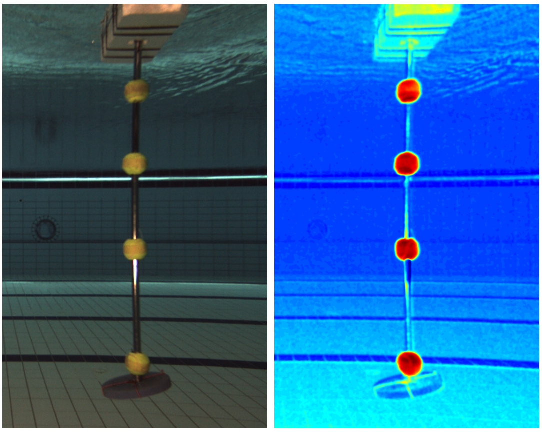


Figure 9.4: Example of SVM performance. The calibration tool and the SVM response. The SVM was trained on a different sample image.

the feature $\boldsymbol{v}_k$ is taken as

$$\boldsymbol{v}_k = \left(C, M, Y, C^2, M^2, Y^2, CM, CY, MY, K\right)_k^\top . \tag{9.2}$$

Separate classifiers are constructed for over and underwater cameras using the standard linear soft margin SVM formulation (Cortes and Vapnik 1995), solving the convex quadratic program

$$\begin{aligned} \min_{\boldsymbol{w},b,\zeta} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_k \zeta_k \\ \text{s.t.} \quad & t_k\left(\boldsymbol{w}^\top \boldsymbol{v}_k + b\right) \geq 1 - \zeta_k \\ & \zeta_k \geq 0, \end{aligned} \tag{9.3}$$

where $\boldsymbol{w}$, b are the sought SVM parameters, ζ_k slack variables allowing features being placed on the wrong side of the hyperplane, and $t_k \in \{-1, 1\}$ a class indicator. Positive and negative samples are extracted from a single representative image for each of the classifiers. The penalty weight C is selected to be as large as possible while still giving feasible solutions.

A representative example of the detection using the trained SVM is shown in Figure 9.4.

Region Properties

As the markers are spherical their image projections should be conics. A reasonable simplification is to search for approximately circular regions. A simple confidence measure of how circular a region is was devised as the following:

1. Discard all regions whose area is smaller than a disc of radius 5 pixels or larger than a disc of radius 50.
2. Estimate two radii (r_{min}, r_{max}) based on the regions' second order moments, i.e. do ellipse fitting.
3. Create two circular regions around the center with the sizes r_{min} and $3r_{\text{min}}$.
4. Score based on the ratio of region inside the inner circle to region in the larger circle.

Regions are then culled based on relative response where regions whose confidence is below a ratio of 20% of the maximum confidence are discarded. The centroids of the remaining regions are used as candidates for the markers when solving for the calibration stick pose.

9.2.4 Solving for the Calibration Object Pose

We model the calibration rod markers as points on a line. The pose of the stick relative to a perspective camera can be uniquely determined (up to rotation around its own axis) from the projection of three markers in a single image, given their absolute positions on the stick. Unlike the general three-point pose problem, where the points are not collinear, this can be solved easily in closed form using simple trigonometry (it is in fact a special case of the Snellius-Pothenot problem, see Wildberger 2010). Consider Figure 9.5, where $\boldsymbol{C}$ represents the known camera center and $D_{1,2}$ the known distances between the markers. If we can compute two of the depths x, y or z, the pose relative to the camera can be inferred. From the law of sines we have

$$\frac{\sin\varepsilon}{y} = \frac{\sin\delta}{x} = \frac{\sin\alpha}{D_1 + D_2}, \quad \frac{\sin\varepsilon}{z} = \frac{\sin\gamma}{D_1}, \quad \frac{\sin\delta}{z} = \frac{\sin\beta}{D_2}, \tag{9.4}$$

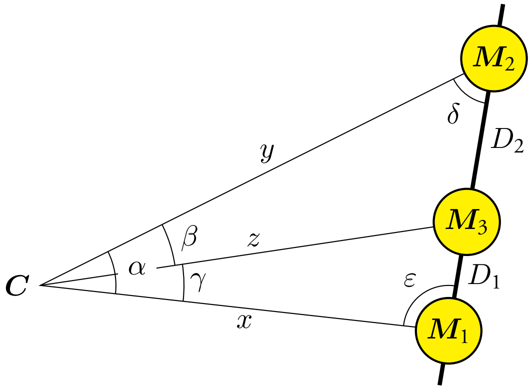


Figure 9.5: Three markers on the calibration rod viewed by a camera with center $\boldsymbol{C}$.

giving

$$\begin{aligned} \frac{z}{x} &= \frac{\sin\delta/x}{\sin\delta/z} = \frac{D_2 \sin\alpha}{(D_1 + D_2)\sin\beta} \equiv K_x \\ \frac{z}{y} &= \frac{\sin\varepsilon/y}{\sin\varepsilon/z} = \frac{D_1 \sin\alpha}{(D_1 + D_2)\sin\gamma} \equiv K_y\,. \end{aligned} \tag{9.5}$$

From the law of cosines, $D_1^2 = x^2 + z^2 - 2xz\cos\gamma$, and by substituting $z = xK_x$ we find

$$\begin{aligned} x &= D_1/\sqrt{K_x^2 - 2K_x\cos\gamma + 1} \\ y &= xK_x/K_y\,. \end{aligned} \tag{9.6}$$

Note that the argument to the square root is always non-negative, guaranteeing a physical solution. Given the normalized image projections $\boldsymbol{m}_{1,2,3}$ of the markers $\boldsymbol{M}_{1,2,3}$ in homogeneous coordinates, scaled so that $\|\boldsymbol{m}_{1,2,3}\| = 1$, we can compute $\cos\gamma = \boldsymbol{m}_1 \cdot \boldsymbol{m}_3$, $\sin\alpha = \|\boldsymbol{m}_1 \times \boldsymbol{m}_2\|$, $\sin\beta = \|\boldsymbol{m}_2 \times \boldsymbol{m}_3\|$ and $\sin\gamma = \|\boldsymbol{m}_1 \times \boldsymbol{m}_3\|$, and thus x and y.

The fourth marker provides redundancy and is used to verify the marker detection in a RANSAC scheme. By ordering candidate marker locations vertically, potential correspondences can be established and tested against the reprojection error. All markers are included in a subsequent non-linear refinement step where the reprojection error in the image is minimized over rigid motions of the calibration object.

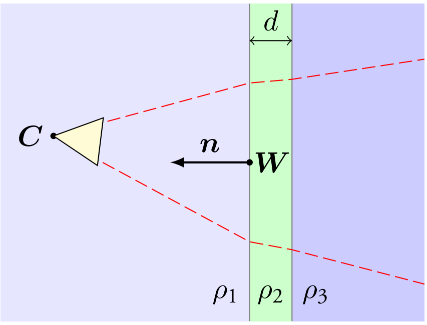


Figure 9.6: Each underwater camera $\boldsymbol{C}$ views the pool through a glass pane of thickness d mounted at $\boldsymbol{W}$ and with normal vector $\boldsymbol{n}$.

However, before this procedure can be applied to the underwater images, refraction effects in the windows through which the object is viewed must be taken into account.

9.2.5 Refraction

The underwater cameras observe the pool through glass windows (see Figure 9.6). To obtain a physically accurate model of the imaging system, the refraction effects at both optical medium interfaces must be taken into account. We turn again to Snell's law

$$\mu \sin\theta_1 = \sin\theta_2, \tag{9.7}$$

where $\theta_{1,2}$ are the angles of incidence and $\mu = \rho_1/\rho_2$ the respective indices of refraction of the two media. Given a ray passing through a point $\boldsymbol{P}_1$ in the direction $\boldsymbol{u}$ and an interface plane passing through the point $\boldsymbol{W}$ with normal vector $\boldsymbol{n}$, the refracted ray $(\boldsymbol{P}_2, \boldsymbol{v})$ may be computed as

$$\begin{aligned} \boldsymbol{P}_2 &= \boldsymbol{P}_1 - \frac{\boldsymbol{n}\cdot(\boldsymbol{P}_1 - \boldsymbol{W})}{\boldsymbol{n}\cdot\boldsymbol{u}}\boldsymbol{u} \\ \boldsymbol{v} &= \mu\boldsymbol{u} + \big(\mu\cos\theta_1 - \operatorname{sign}(\cos\theta_1)\cos\theta_2\big)\boldsymbol{n} \\ \text{where} \quad \cos\theta_1 &= -\boldsymbol{n}\cdot\boldsymbol{u} \\ \cos\theta_2 &= \sqrt{1-\mu^2(1-\cos^2\theta_1)}\,, \end{aligned} \tag{9.8}$$

given that $\boldsymbol{n}$ and $\boldsymbol{u}$ have been normalized to unit length (see Figure 9.7 for an illustration). Note that no trigonometric functions need to be evaluated,

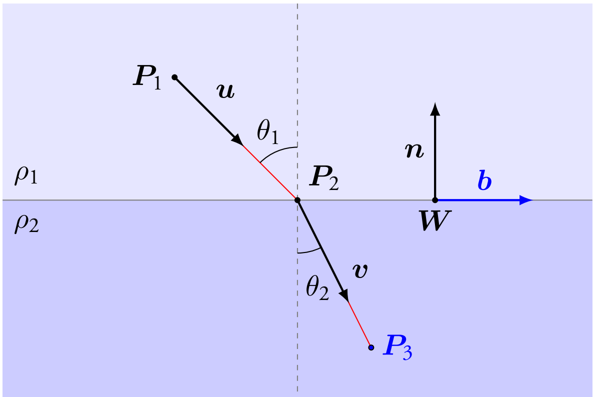


Figure 9.7: Refraction of ray $(\boldsymbol{P}_1, \boldsymbol{u})$ into $(\boldsymbol{P}_2, \boldsymbol{v})$ under Snell's law.

making it fast to compute. The above formula only works in the backward direction, i.e. given the ray corresponding to an image point, we can follow it into the second medium. However, we are mainly interested in going the other way, computing the projection of a world point into the camera. This is more difficult; given the world point $\boldsymbol{P}_3$ on the ray $(\boldsymbol{P}_2, \boldsymbol{v})$ and camera center $\boldsymbol{P}_1$, determine $\boldsymbol{P}_2$ so that Snell's law is satisfied (the image projection of $\boldsymbol{P}_3$ is then given by the in-air camera model projection of $\boldsymbol{P}_2$). In previous works this has either been avoided (Jordt-Sedlazeck and Koch 2012), or solved by finding the roots of a 12th degree polynomial (A. Agrawal et al. 2012) or by numerical optimization of the back-projection (Kunz and H. Singh 2008; Yau et al. 2013). We will use a variant of the latter.

Computing the Forward Projection

The ray directions $\boldsymbol{u}$ and $\boldsymbol{v}$ must lie in the same plane as the interface normal $\boldsymbol{n}$; call the normal vector of this plane $\boldsymbol{w} = \boldsymbol{n} \times (\boldsymbol{P}_3 - \boldsymbol{P}_1)$. This restricts the possible locations of $\boldsymbol{P}_2$ to the line

$$\boldsymbol{P}_2(t) = \boldsymbol{P}_2^0 + t\boldsymbol{b} \tag{9.9}$$

where $\boldsymbol{b} = \boldsymbol{n} \times \boldsymbol{w}$. Given $\boldsymbol{P}_2(t)$ for some t, we can refract and trace the ray $(\boldsymbol{P}_1, \boldsymbol{P}_2(t) - \boldsymbol{P}_1)$ into the second medium. At the optimal t, the ray $\big(\boldsymbol{P}_2(t), \boldsymbol{v}(t)\big)$ will pass through $\boldsymbol{P}_3$. Define the signed orthogonal distance between $\boldsymbol{P}_3$ and the ray as

$$d(t) = \boldsymbol{w} \cdot \big(\boldsymbol{P}_3 - \boldsymbol{P}_2(t)\big) \times \boldsymbol{v}(t). \tag{9.10}$$

Unlike the absolute distance, this signed distance function is differentiable at its root, which means it can be minimized using Newton's method. If $\rho_1 \leq \rho_2$, the intersection of the interface plane and the straight line between $\boldsymbol{P}_1$ and $\boldsymbol{P}_3$ can be used as an initial guess for $\boldsymbol{P}_2$, otherwise a safer starting point is the orthogonal projection of $\boldsymbol{P}_1$ onto the plane, to avoid encountering total internal reflection.

For our underwater cameras, equations (9.8) are applied twice in the backward direction, first for the air–glass and then for the glass–water transition. Since the two interfaces are parallel, all rays still lie in the same plane and the search remains one-dimensional. Yau et al. (2013) take a similar approach minimizing the back-projection error, but use bisection with inferior convergence properties.

On average over a typical range of angles, a precision of 10^{-6} (corresponding to a reprojection error on the order of 0.01 pixels) is reached in four iterations using forward finite difference derivatives. While the method is general and does not require the imaging plane to be parallel with the interface(s) as in Treibitz et al. (2012), it is still fast and can compute the forward projection of two million points per second on a Core 2 Duo E7500 3.0 GHz computer in a multi-threaded C++ implementation. It is thus well-suited for use in large-scale bundle adjustment algorithms minimizing the true image reprojection error, and the algorithm is simple to implement. It is also easily and efficiently parallelized on graphics hardware, since given a fixed number of iterations there are no branch points, unlike in the bisection approach. It may also be extended to the case of multiple non-parallel interfaces, which would require a two-dimensional search for $\boldsymbol{P}_2$, at some additional computational cost.

9.2.6 Structure Initialization

Once we can compute the projection of any given point into each camera, all extrinsic parameters may be optimized through bundle adjustment if a good initialization is available. In the swimming pool case, the positions of the cameras are easily measured by hand or from blue-prints, and we assume these to be known, except for the exact distance of the underwater cameras to the glass pane as this number significantly influences the refraction effects. The thicknesses of the window panes are also considered known, and the panes are initially assumed to be mounted exactly flush with the pool wall.

The initial pose of the calibration object in every frame is determined relative to the camera with the "best" view (i.e. with the markers closest to the image center), using the single-view solver above. If the best view is an underwater camera the image will be distorted by refraction, and solving while only accounting for intrinsic camera parameters is likely to give inaccurate results. Since the refraction effects are actually three-dimensional in nature, the coordinates cannot be exactly normalized without knowing the depth of the markers beforehand. We settle for an approximation where the image rays are traced into the pool (using the initial camera and window parameters), and their intersections with a plane parallel to the image plane at the expected mean depth of the calibration target are computed. The intersection points are then projected back into the images, now assuming there are no refraction effects, producing the measurements we would have obtained had there been no water or windows. This approximation, which assumes the markers are at a known depth halfway into the pool, is quite accurate as the depth dependence of the correction is relatively weak, and is certainly sufficient for initialization purposes.

As was noted in Jordt-Sedlazeck and Koch (2013), and argued in the beginning of Chapter 8, it is possible to solve exactly for the depth of three points also in the refractive case, assuming the camera's pose relative to the refractive plane is known. The camera and glass then form a generalized camera (in fact, an axial camera), where the back-projected image rays do not intersect in a common point but rather a common axis. The generalized three-point pose solver (Nistér 2004) can then be applied, which produces up to eight solutions. However, it does not exploit the fact that the points on our calibration object are co-linear, and we have found in experiments that our simpler solver together with the approximation produces stabler and more accurate results under image measurement noise.

9.2.7 Non-linear Refinement

The bundle adjustment problem is formulated and solved using the Ceres non-linear least-squares solver (Agarwal, Mierle, et al. 2012). Due to the relatively complex projection algorithm, numerical finite difference derivatives are used, although automatic differentiation could possibly be applied. We allow the glass pane normals and underwater camera distance from the glass to vary, along with all camera orientations and calibration stick poses. While the calibration thus obtained is quite accurate, the human eye is very sensitive to discrepancies at image seams which becomes obvious when rendering stitched panoramic views. In particular, horizontal lines on the pool wall (see Figure 9.11) need to match to the pixel. To this end, we mark points in the images along these lines, and require their back-projection intersections with the pool wall to be co-linear. This may be achieved by introducing a new variable $\bar{y}_k$ for each line into the optimization, and adding the terms $||y_{k,i} - \bar{y}_k||^2$ to the bundle adjustment cost function, where $y_{k,i}$ are the vertical components of the back-projected points lying on line k.

9.3 Stitching

One goal of the calibration is to be able to stitch together images to form a panorama of the pool, or equivalently, panning shots of the swimmer. To render such a view, we define an image plane in the world coordinate system, typically parallel to the long side of the pool. Output pixels are sampled in a grid on this plane, and projected into the devignetted camera images to determine their color. Where images overlap, blending is applied to smooth out the transition. For speed, projection maps for each camera can be precomputed. Since the projection depends on the depth of the rendering plane, separate maps are computed for a discrete set of depths and then interpolated between to match the current depth of the swimmer. This can be efficiently implemented on graphics hardware and allows us to generate full HD panning views in real time at over 100 frames per second.

9.3.1 Exposure Correction

While all cameras are set to the same white balance, exposure and gain, differences between individual cameras are sometimes visible, particularly near the transition edges. To minimize visual discrepancies post-capture, we assume that the pixel value of a point visible in two cameras simultaneously is described by the relation $\gamma_i I_i = \gamma_j I_j$ where γ_k is the gain correction for camera k and I_k the intensity in the captured image. To achieve even lighting of the stitched image, points are sampled on the render plane and projected into low-pass filtered images to obtain the I_k. Each relation above contributes a row to the

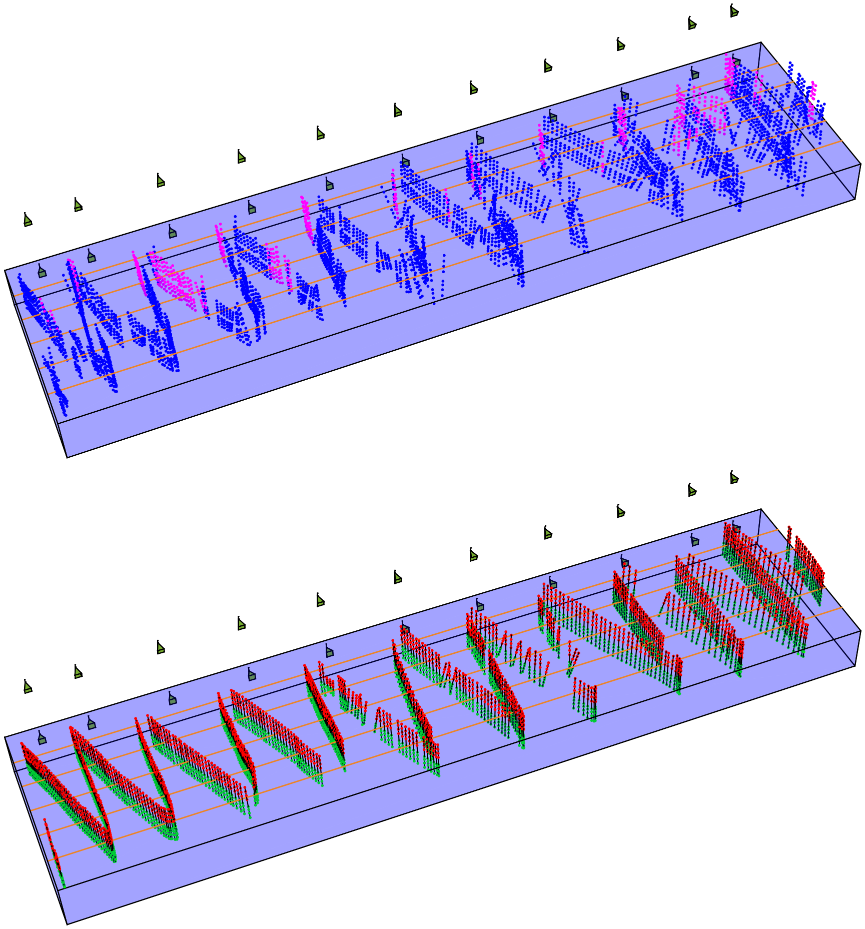


Figure 9.8: Top: the initialization to the bundle adjustment problem, where the pose of the calibration rod in each frame has been determined from one view only (blue indicates an underwater image was used, magenta above water). Bottom: the result of optimizing over calibration object pose, window pane normals and camera orientations. Bundle adjustment over the 700 poses and 1900 images took 32 seconds at 7 iterations/s on a Core 2 Duo 3.0 GHz computer.

linear system

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & I_i & \mathbf{0} & -I_j & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_n \end{pmatrix} = \mathbf{0} \tag{9.11}$$

which can be solved for the γ in a least-squares sense using singular value decomposition. After scaling the gain coefficients to have unit mean, they are multiplied with the raw images before stitching. See Figure 9.10 for an illustration.

9.3.2 Chromatic Aberration Correction

An often ignored fact is that the indices of refraction used in the ray-tracing equations (9.8) are wavelength-dependent. However, the resulting chromatic aberration is quite pronounced even to the naked eye observing e.g. the bottom tiles of the pool through refraction at the surface. The effect is also apparent near the edges of our underwater images in areas of high contrast. By using (empirically determined) separate indices of refraction for the red, green and blue color channels, the aberration can be almost completely neutralized, improving the visual quality of rendered images (see Figure 9.9).

The final stitched panoramas are shown in Figure 9.11, along with the raw images captured by each camera.

9.4 Conclusion

We have developed an effective and practical procedure for combined refractive and non-refractive camera calibration. We have also presented an efficient method of computing the forward projection through refractive media, and shown how visually pleasing stitched panoramas may be generated. The calibration data and images generated can then be used for tracking and analyzing a swimmer's movements in and above the water. An example of the output of the full system is shown in Figure 9.12.

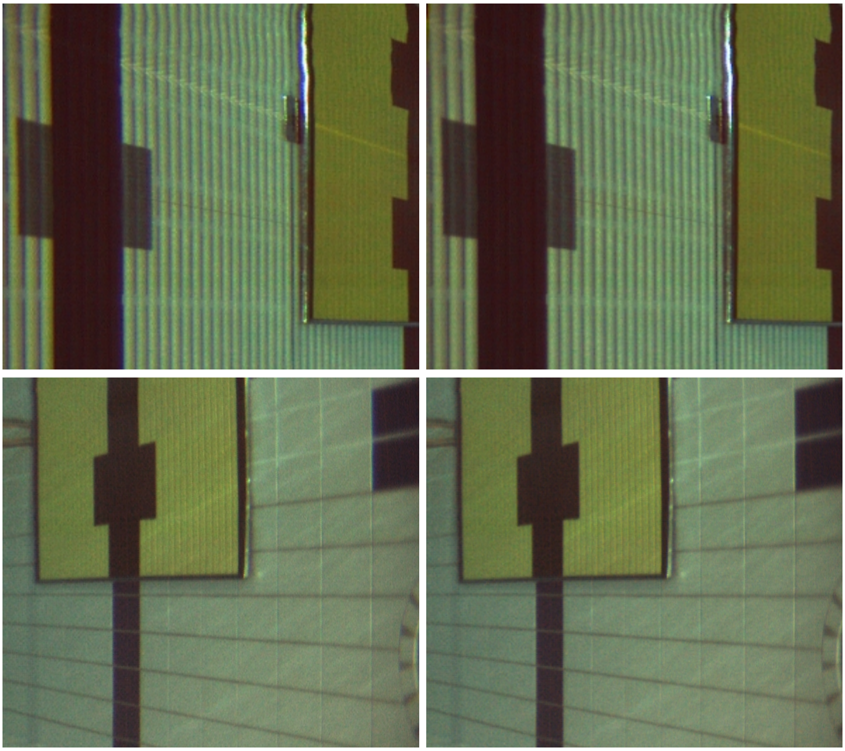


Figure 9.9: Chromatic aberration correction. On the left, the images were rendered using the same index of refraction of water for all channels ($n = 1.333$); on the right, $n_{\text{red}} = 1.333$, $n_{\text{green}} = 1.3338$, $n_{\text{blue}} = 1.3365$. Notice the reduced rainbow effect around the edges of the black bars (the differences are subtle so these images are best viewed on-screen in the electronic version of the thesis).

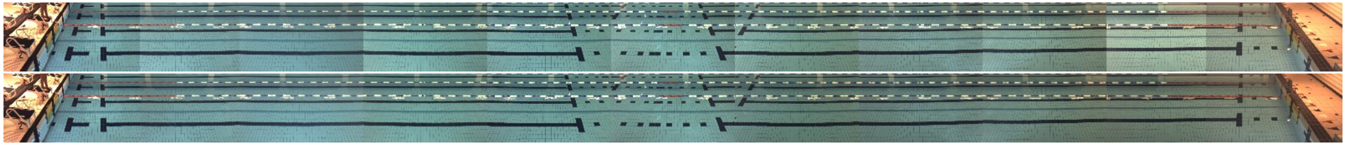


Figure 9.10: Exposure/gain correction. On top the uncorrected images, below each image has been multiplied with the corresponding correction factor derived from equation (9.11). No blending has been applied at the seams.

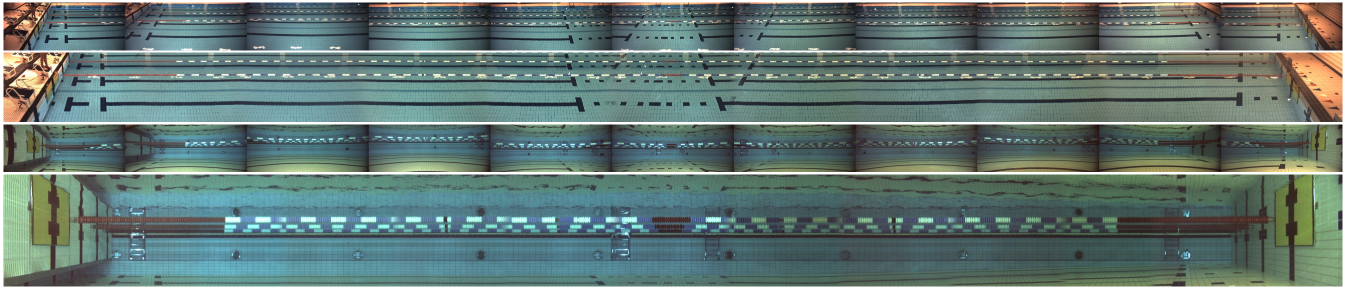
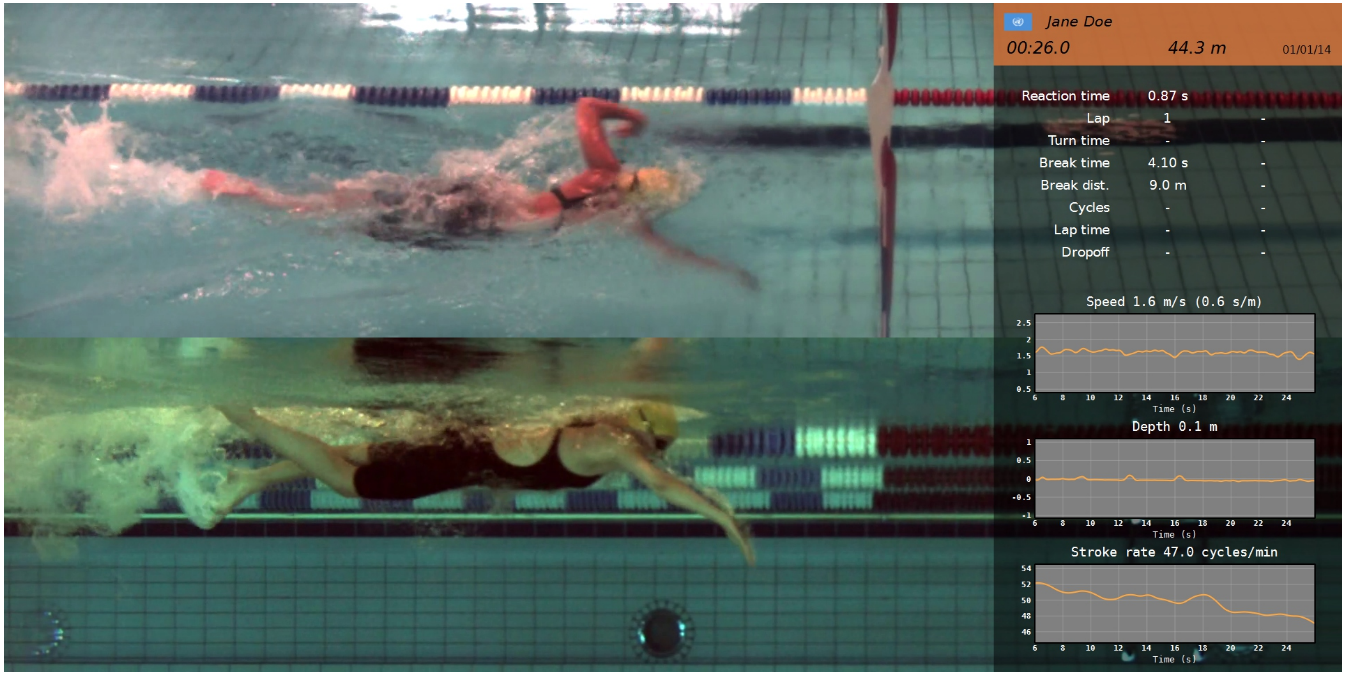


Figure 9.11: The raw images captured by the camera setup along with the stitched panoramas. The lenses used are rectilinear, the distortion effect seen in the underwater views is entirely due to refraction in the windows.



Jane Doe

00:26.0 44.3 m 01/01/14

Reaction time	0.87 s	
Lap	1	-
Turn time	-	-
Break time	4.10 s	-
Break dist.	9.0 m	-
Cycles	-	-
Lap time	-	-
Dropoff	-	-

Speed 1.6 m/s (0.6 s/m)

Depth 0.1 m

Stroke rate 47.0 cycles/min

Figure 9.12: Still image from the final output of the swimmer tracking system. Synthetic panning views of the swimmer are accompanied by performance statistics automatically extracted from the images.

Bibliography

Agarwal, Sameer, Noah Snavely, Steven M. Seitz, and Richard Szeliski (2010). "Bundle Adjustment in the Large." *European Conference on Computer Vision*. Cited on pages 18 and 81.

Agarwal, Sameer, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski (2009). "Building Rome in a Day." *International Conference on Computer Vision*. Cited on page 73.

Agarwal, Sameer, Keir Mierle, et al. (2012). *Ceres Solver*. Available at http://ceres-solver.org. Cited on page 134.

Agrawal, Amit, Srikumar Ramalingam, Yuichi Taguchi, and Visesh Chari (2012). "A Theory of Multi-layer Flat Refractive Geometry." *Conference on Computer Vision and Pattern Recognition*. Cited on pages 103, 106, 112, 114, 123, and 132.

Agrawal, Motilal, Kurt Konolige, and Morten Rufus Blas (2008). "CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching." *European Conference on Computer Vision*. Cited on page 11.

Alahi, Alexandre, Raphael Ortiz, and Pierre Vandergheynst (2012). "FREAK: Fast Retina Keypoint." *Conference on Computer Vision and Pattern Recognition*. Cited on page 11.

Armijo, Larry (1966). "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives." In *Pacific Journal of Mathematics*. Cited on page 10.

Ask, Erik (2014). "Methods for Optimal Model Fitting and Sensor Calibration." PhD thesis. Lund University. Cited on page 98.

Ask, Erik, Yubin Kuang, and Kalle Åström (2012). "Exploiting P-fold Symmetries for Faster Polynomial Equation Solving." *International Conference on Pattern Recognition*. Cited on page 109.

Bay, Herbert, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool (2008). "SURF: Speeded Up Robust Features." In *Computer Vision and Image Understanding*. Cited on page 11.

Blaer, Paul S. and Peter K. Allen (2007). "Data Acquisition and View Planning for 3-D Modeling Tasks." *International Conference on Intelligent Robots and Systems*. Cited on page 30.

Botterill, Tom, Steven Mills, and Richard Green (2010). "Bag-of-Words-driven, Single-camera Simultaneous Localization and Mapping." In *Journal of Field Robotics*. Cited on page 29.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. New York, NY, USA, Cambridge University Press. Cited on page 47.

Bujnak, Martin, Zuzana Kukelova, and Tomás Pajdla (2010). "New Efficient Solution to the Absolute Pose Problem for Camera with Unknown Focal Length and Radial Distortion." *Asian Conference on Computer Vision*. Cited on page 15.

Byröd, Martin and Kalle Åström (2010). "Conjugate Gradient Bundle Adjustment." *European Conference on Computer Vision*. Cited on page 18.

Byröd, Martin, Klas Josephson, and Kalle Åström (2007). "Fast Optimal Three View Triangulation." *Asian Conference on Computer Vision*. Cited on page 14.

Byröd, Martin, Klas Josephson, and Kalle Åström (2008). "A Column-Pivoting Based Strategy for Monomial Ordering in Numerical Gröbner Basis Calculations." *European Conference on Computer Vision*. Cited on page 97.

Byröd, Martin, Klas Josephson, and Kalle Åström (2009). "Fast and Stable Polynomial Equation Solving and Its Application to Computer Vision." In *International Journal of Computer Vision*. Cited on page 97.

Calonder, Michael, Vincent Lepetit, Christoph Strecha, and Pascal Fua (2010). "BRIEF: Binary Robust Independent Elementary Features." *European Conference on Computer Vision*. Cited on page 11.

Chang, Yao-jen and Tsuhan Chen (2011). “Multi-View 3D Reconstruction for Scenes under the Refractive Plane with Known Vertical Direction.” *International Conference on Computer Vision*. Cited on pages 104, 110, 111, and 124.

Chari, Visesh and Peter F. Sturm (2009). “Multi-View Geometry of the Refractive Plane.” *British Machine Vision Conference*. Cited on page 103.

Chen, Shengyong, Y. F. Li, Jianwei Zhang, and Wanliang Wang (2008). *Active Sensor Planning for Multiview Vision Tasks*. Springer Publishing Company, Incorporated. Cited on page 31.

Chen, S.Y. and Y.F. Li (2004). “Automatic Sensor Placement for Model-based Robot Vision.” In *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. Cited on page 35.

Chen, Yanqing, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam (2008). “Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate.” In *ACM Transactions on Mathematical Software*. Cited on page 18.

Choi, In-Chan, Seong-In Kim, and Hak-Soo Kim (2003). “A Genetic Algorithm with a Mixed Region Search for the Asymmetric Traveling Salesman Problem.” In *Computers & Operations Research*. Cited on page 50.

Cortes, Corinna and Vladimir Vapnik (1995). “Support-Vector Networks.” In *Machine Learning*. Cited on page 128.

Courtois, Nicolas, Alexander Klimov, Jacques Patarin, and Adi Shamir (2000). “Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations.” *International Conference on the Theory and Applications of Cryptographic Techniques*. Cited on page 101.

Cox, David A., John Little, and Donal O’Shea (2007). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3rd ed.* Springer-Verlag New York, Inc. Cited on pages 94 and 95.

Crandall, David, Andrew Owens, Noah Snavely, and Daniel P. Huttenlocher (2011). "Discrete-Continuous Optimization for Large-Scale Structure from Motion." *Conference on Computer Vision and Pattern Recognition*. Cited on page 73.

Dahl, Joachim (2012). "Semidefinite Optimization using MOSEK." *21st International Symposium on Mathematical Programming*. URL: http://www.mosek.com. Cited on page 55.

Dalal, Navneet and Bill Triggs (2005). "Histograms of Oriented Gradients for Human Detection". Cited on page 11.

Davis, Timothy A. (2011). "Algorithm 915, SuiteSparseQR: Multifrontal Multithreaded Rank-revealing Sparse QR Factorization." In *ACM Transactions on Mathematical Software*. Cited on page 98.

Davoodi, Mansoor, Fatemeh Panahi, Ali Mohades, and Seyed Naser Hashemi (2013). "Multi-objective Path Planning in Discrete Space." In *Applied Soft Computing*. Cited on page 50.

Devernay, Frederic and Olivier D. Faugeras (2001). "Straight Lines Have to be Straight." In *Machine Vision and Applications*. Cited on page 6.

Dickenstein, Alicia and Ioannis Z. Emiris (2010). *Solving Polynomial Equations: Foundations, Algorithms, and Applications*. Springer Publishing Company, Incorporated. Cited on page 101.

Dunn, Enrique, Jur van den Berg, and Jan-Michael Frahm (2009). "Developing Visual Sensing Strategies through Next Best View Planning." *International Conference on Intelligent Robots and Systems*. Cited on pages 30 and 35.

Dunn, Enrique, Gustavo Olague, and Evelyne Lutton (2006). "Parisian Camera Placement for Vision Metrology." In *Pattern Recognition Letters*. Cited on pages 30 and 35.

Englot, Brendan and Franz Hover (2010). "Inspection Planning for Sensor Coverage of 3D Marine Structures." *International Conference on Intelligent Robots and Systems*. Cited on page 43.

Faugère, Jean-Charles (2002). "A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero." *International Symposium on Symbolic and Algebraic Computation*. Cited on page 97.

Fischler, Martin A. and Robert C. Bolles (1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." In *Communications of the Association for Computing Machinery*. Cited on page 12.

Fitzgibbon, Andrew W. (2001). "Simultaneous Linear Estimation of Multiple View Geometry and Lens Distortion." *Conference on Computer Vision and Pattern Recognition*. Cited on pages 6 and 111.

Foix, Sergi, Simon Kriegel, Stefan Fuchs, Guillem Alenyà, and Carme Torras (2012). "Information-Gain View Planning for Free-Form Object Reconstruction with a 3D ToF Camera." In *Advanced Concepts for Intelligent Vision Systems*. Lecture Notes in Computer Science. Springer Berlin Heidelberg. Cited on page 30.

Frahm, Jan-Michael et al. (2010). "Building Rome on a Cloudless Day." *European Conference on Computer Vision*. Cited on page 73.

Fraser, Clive S. (1984). "Network Design Considerations for Non-Topographic Photogrammetry." In *Photogrammetric Engineering and Remote Sensing*. Cited on page 23.

Gao, Xiao-Shan, Xiaorong Hou, Jianliang Tang, and Hang-Fei Cheng (2003). "Complete Solution Classification for the Perspective-Three-Point Problem." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Cited on page 15.

Glaeser, Georg and Hans-Peter Schröcker (2000). "Reflections on Refractions." In *Journal of Geometry and Graphics*. Cited on page 106.

Golovin, Daniel and Andreas Krause (2010). "Adaptive Submodularity: A New Approach to Active Learning and Stochastic Optimization." *23rd Annual Conference on Learning Theory*. Cited on page 43.

Golub, Gene H. and Charles F. Van Loan (1989). *Matrix Computations*. 2nd ed. Baltimore, Johns Hopkins University Press. Cited on page 98.

Grayson, Daniel R. and Michael E. Stillman. *Macaulay2, a Software System for Research in Algebraic Geometry*. Available at http://www.math.uiuc.edu/Macaulay2/. Cited on page 100.

Guilbert, Nicolas et al. (2004). “Constraint Enforcement in Structure and Motion Applied to Closing and Open Sequence.” *Asian Conference on Computer Vision*. Cited on page 29.

Haner, Sebastian and Anders Heyden (2010). “A Step Towards Self-calibration in SLAM: Weakly Calibrated On-line Structure and Motion Estimation.” *Workshop on Mobile Vision*. Cited on page 16.

Harris, Chris and Mike Stephens (1988). “A Combined Corner and Edge Detector.” *4th Alvey Vision Conference*. Cited on page 11.

Hartley, Richard (1997). “In Defense of the Eight-Point Algorithm.” In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Cited on page 16.

Hartley, Richard and Peter Sturm (1997). “Triangulation.” In *Computer Vision and Image Understanding*. Cited on page 14.

Hartley, Richard, Jochen Trumpf, Yuchao Dai, and Hongdong Li (2013). “Rotation Averaging.” In *International Journal of Computer Vision*. Cited on page 20.

Hartley, Richard and Andrew Zisserman (2003). *Multiple View Geometry*. Cambridge University Press. Cited on pages 6, 16, 24, 25, 26, 78, and 86.

Heikkilä, Janne and Olli Silvén (1997). “A Four-step Camera Calibration Procedure with Implicit Image Correction.” *Conference on Computer Vision and Pattern Recognition*. Cited on pages 6 and 125.

Hollinger, Geoffrey A., Brendan Englot, Franz Hover, Urbashi Mitra, and Gaurav S. Sukhatme (2012). “Uncertainty-driven View Planning for Underwater Inspection.” *International Conference on Robotics and Automation*. Cited on page 43.

Jordt-Sedlazeck, Anne and Reinhard Koch (2012). “Refractive Calibration of Underwater Cameras.” *European Conference on Computer Vision*. Cited on page 132.

Jordt-Sedlazeck, Anne and Reinhard Koch (2013). “Refractive Structure-from-Motion on Underwater Images.” *International Conference on Computer Vision*. Cited on pages 103, 124, and 134.

Kahl, Fredrik and Richard Hartley (2008). “Multiple View Geometry Under the L_∞ Norm.” In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Cited on page 15.

Kahl, Fredrik and Didier Henrion (2007). “Globally Optimal Estimates for Geometric Reconstruction Problems.” In *International Journal of Computer Vision*. Cited on page 17.

Kanatani, Kenichi, Yasuyuki Sugaya, and Hirotaka Niitsuma (2008). “Triangulation from Two Views Revisited: Hartley-Sturm vs. Optimal Correction.” *British Machine Vision Conference*. Cited on page 14.

Kang, Lai, Lingda Wu, and Yee-Hong Yang (2012). “Two-View Underwater Structure and Motion for Cameras under Flat Refractive Interfaces.” *European Conference on Computer Vision*. Cited on pages 103 and 124.

Kiefer, Jack (1953). “Sequential Minimax Search for a Maximum.” In *Proceedings of The American Mathematical Society*. Cited on page 10.

Kilpelä, Einari (1981). “Compensation of Systematic Errors of Image and Model Coordinates.” In *Photogrammetria*. Cited on page 6.

Kim, Jae-Hak, Hongdong Li, and Richard I. Hartley (2010). “Motion Estimation for Nonoverlapping Multicamera Rigs: Linear Algebraic and L_∞ Geometric Solutions.” In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Cited on page 103.

Klein, Georg and David Murray (2007). “Parallel Tracking and Mapping for Small AR Workspaces.” *International Symposium on Mixed and Augmented Reality*. Cited on page 32.

Kuang, Yubin and Kalle Åström (2012). “Numerically Stable Optimization of Polynomial Solvers for Minimal Problems.” *European Conference on Computer Vision*. Cited on page 117.

Kukelova, Zuzana, Martin Bujnak, and Tomás Pajdla (2010). “Closed-Form Solutions to Minimal Absolute Pose Problems with Known Vertical Direction.” *Asian Conference on Computer Vision*. Cited on page 111.

Kukelova, Zuzana, Martin Bujnak, and Tomás Pajdla (2012). "Polynomial Eigenvalue Solutions to Minimal Problems in Computer Vision." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Cited on pages 100 and 111.

Kunz, Clayton and Hanumant Singh (2008). "Hemispherical Refraction and Camera Calibration in Underwater Vision." *OCEANS*. IEEE. Cited on page 132.

Kushal, Avanish and Sameer Agarwal (2012). "Visibility Based Preconditioning for Bundle Adjustment." *Conference on Computer Vision and Pattern Recognition*. Cited on page 18.

Lasserre, Jean (2000). *Convergent LMI Relaxations for Nonconvex Quadratic Programs*. Cited on page 65.

LaValle, Steven M. (2006). *Planning Algorithms*. Cambridge University Press. Cited on page 31.

Lawler, Gregory F. (1980). "A Self-avoiding Walk." In *Duke Mathematical Journal*. Cited on page 52.

Lazard, Daniel (1981). "Résolution des Systèmes d'Equations Algébriques." In *Theoretical Computer Science*. Cited on page 96.

Leutenegger, Stefan, Margarita Chli, and Roland Siegwart (2011). "BRISK: Binary Robust Invariant Scalable Keypoints." *International Conference on Computer Vision*. Cited on page 11.

Levenberg, Kenneth (1944). "A Method for the Solution of Certain Non-linear Problems in Least Squares." In *Quarterly Journal of Applied Mathmatics*. Cited on page 9.

Li, Hongdong (2006). "A Simple Solution to the Six-point Two-view Focal Length Problem." *European Conference on Computer Vision*. Cited on page 100.

Li, Hongdong and Richard Hartley (2006). "Five-point Motion Estimation Made Easy." *International Conference on Pattern Recognition*. Cited on page 100.

Löfberg, Johan (2004). "YALMIP: A Toolbox for Modeling and Optimization in MATLAB." *Conference on Computer Aided Control Systems Design*. URL: http://users.isy.liu.se/johanl/yalmip. Cited on page 55.

Low, Kok-Lim and Anselmo Lastra (2006). "An Adaptive Hierarchical Next-Best-View Algorithm for 3D Reconstruction of Indoor Scenes." *14th Pacific Conference on Computer Graphics and Applications*. Cited on page 30.

Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints." In *International Journal of Computer Vision*. Cited on page 11.

Mason, Scott (1997). "Heuristic Reasoning Strategy for Automated Sensor Placement." In *Photogrammetric Engineering and Remote Sensing*. Cited on page 23.

Matas, Jiri and Ondrej Chum (2004). "Randomized RANSAC with $\mathrm{T}_{d,d}$ test." In *Image and Vision Computing*. Cited on page 13.

Matas, Jiri, Ondrej Chum, Martin Urban, and Tomás Pajdla (2002). "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions." *British Machine Vision Conference*. Cited on page 11.

Mikolajczyk, Krystian and Cordelia Schmid (2005). "A Performance Evaluation of Local Descriptors." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Cited on page 11.

Montgomery, Douglas C. (2000). *Design and Analysis of Experiments*. 5th ed. John Wiley & Sons Canada, Ltd. Cited on page 32.

Morris, Daniel D. (2001). "Gauge Freedoms and Uncertainty Modeling for 3D Computer Vision." PhD thesis. Carnegie Mellon University. Cited on pages 25 and 78.

Mouragnon, Etienne, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd (2009). "Generic and Real-time Structure from Motion using Local Bundle Adjustment." In *Image and Vision Computing*. Cited on page 32.

Naroditsky, Oleg and Kostas Daniilidis (2011). "Optimizing Polynomial Solvers for Minimal Geometry Problems." *International Conference on Computer Vision*. Cited on page 117.

Nesterov, Yurii and Arkadii Nemirovskii (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics. Cited on page 69.

Nistér, David (2003a). "An Efficient Solution to the Five-point Relative Pose Problem." *Conference on Computer Vision and Pattern Recognition*. Cited on page 16.

Nistér, David (2003b). "Preemptive RANSAC for Live Structure and Motion Estimation." *International Conference on Computer Vision*. Cited on page 13.

Nistér, David (2004). "A Minimal Solution to the Generalised 3-Point Pose Problem." *Conference on Computer Vision and Pattern Recognition*. Cited on pages 103 and 134.

Olsson, Carl and Olof Enqvist (2011). "Stable Structure from Motion for Unordered Image Collections." *Scandinavian Conference on Image Analysis*. Cited on pages 73 and 81.

Oskarsson, Magnus and Kalle Åström (2000). "Automatic Geometric Reasoning in Structure and Motion Estimation." In *Pattern Recognition Letters*. Cited on page 29.

Papadimitriou, Christos H. and Kenneth Steiglitz (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications. Cited on pages 45, 46, and 69.

Piniés, Pedro, Lina María Paz, Dorian Gálvez-López, and Juan D. Tardós (2010). "CI-Graph Simultaneous Localization and Mapping for Three-dimensional Reconstruction of Large and Complex Environments using a Multicamera System." In *Journal of Field Robotics*. Cited on page 29.

Rosten, Edward and Tom Drummond (2006). "Machine Learning for High-Speed Corner Detection." *European Conference on Computer Vision*. Cited on page 11.

Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski (2011). "ORB: An Efficient Alternative to SIFT or SURF." *International Conference on Computer Vision*. Cited on page 11.

Sahl, Ibn (984). *On Burning Mirrors and Lenses*. Cited on page 91.

Schmitt, Lawrence J. and Mohammad M. Amini (1998). “Performance Characteristics of Alternative Genetic Algorithmic Approaches to the Traveling Salesman Problem using Path Representation: An Empirical Study.” In *European Journal of Operational Research*. Cited on page 50.

Schrijver, Alexander (2004). *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics. Springer. Cited on page 50.

Shor, Naum Z. (1987). “Quadratic Optimization Problems.” In *Tekhnicheskaya Kibernetika*. Cited on page 65.

Singh, Amarjeet, Andreas Krause, Carlos Guestrin, and William J. Kaiser (2009). “Efficient Informative Sensing using Multiple Robots.” In *Journal of Artificial Intelligence Research*. Cited on page 43.

Snavely, Noah, Steven M. Seitz, and Richard Szeliski (2007). “Modeling the World from Internet Photo Collections.” In *International Journal of Computer Vision*. Cited on pages 73 and 78.

Snavely, Noah, Steven M. Seitz, and Richard Szeliski (2008). “Skeletal Graphs for Efficient Structure from Motion.” *Conference on Computer Vision and Pattern Recognition*. Cited on page 78.

Stewénius, Henrik, David Nistér, Magnus Oskarsson, and Kalle Åström (2005). “Solutions to Minimal Generalized Relative Pose Problems.” *Workshop on Omnidirectional Vision*. Cited on pages 103 and 113.

Strasdat, Hauke, J. M. M. Montiel, and Andrew J. Davison (2010). “Scale Drift-Aware Large Scale Monocular SLAM.” In *Robotics: Science and Systems VI*. Cited on page 32.

Sturm, Jos F. (1998). *Using SeDuMi 1.02, a MATLAB Toolbox for Optimization over Symmetric Cones*. Available at http://sedumi.ie.lehigh.edu/. Cited on page 70.

Tola, Engin, Vincent Lepetit, and Pascal Fua (2010). “DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo.” In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Cited on page 11.

Torr, Philip H.S. (2002). “Bayesian Model Estimation and Selection for Epipolar Geometry and Generic Manifold Fitting.” In *International Journal of Computer Vision*. Cited on page 13.

Treibitz, Tali, Yoav Y. Schechner, Clayton Kunz, and Hanumant Singh (2012). "Flat Refractive Geometry." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Cited on page 133.

Triggs, Bill, Philip Mclauchlan, Richard Hartley, and Andrew Fitzgibbon (2000). "Bundle Adjustment: a Modern Synthesis." *Vision Algorithms: Theory and Practice*. Cited on page 17.

Trummer, Michael, Christoph Munkelt, and Joachim Denzler (2010). "Online Next-Best-View Planning for Accuracy Optimization Using an Extended E-Criterion." *International Conference on Pattern Recognition*. Cited on page 30.

Vasquez-Gomez, J. Irving, L. Enrique Sucar, and Rafael Murrieta-Cid (2013). "Hierarchical Ray Tracing for Fast Volumetric Next-Best-View Planning." *International Conference on Computer and Robot Vision*. Cited on page 30.

Verschelde, Jan (1999). "Algorithm 795: PHCpack: A General-purpose Solver for Polynomial Systems by Homotopy Continuation." In *ACM Transactions on Mathematical Software*. URL: https://github.com/janverschelde/PHCpack. Cited on page 101.

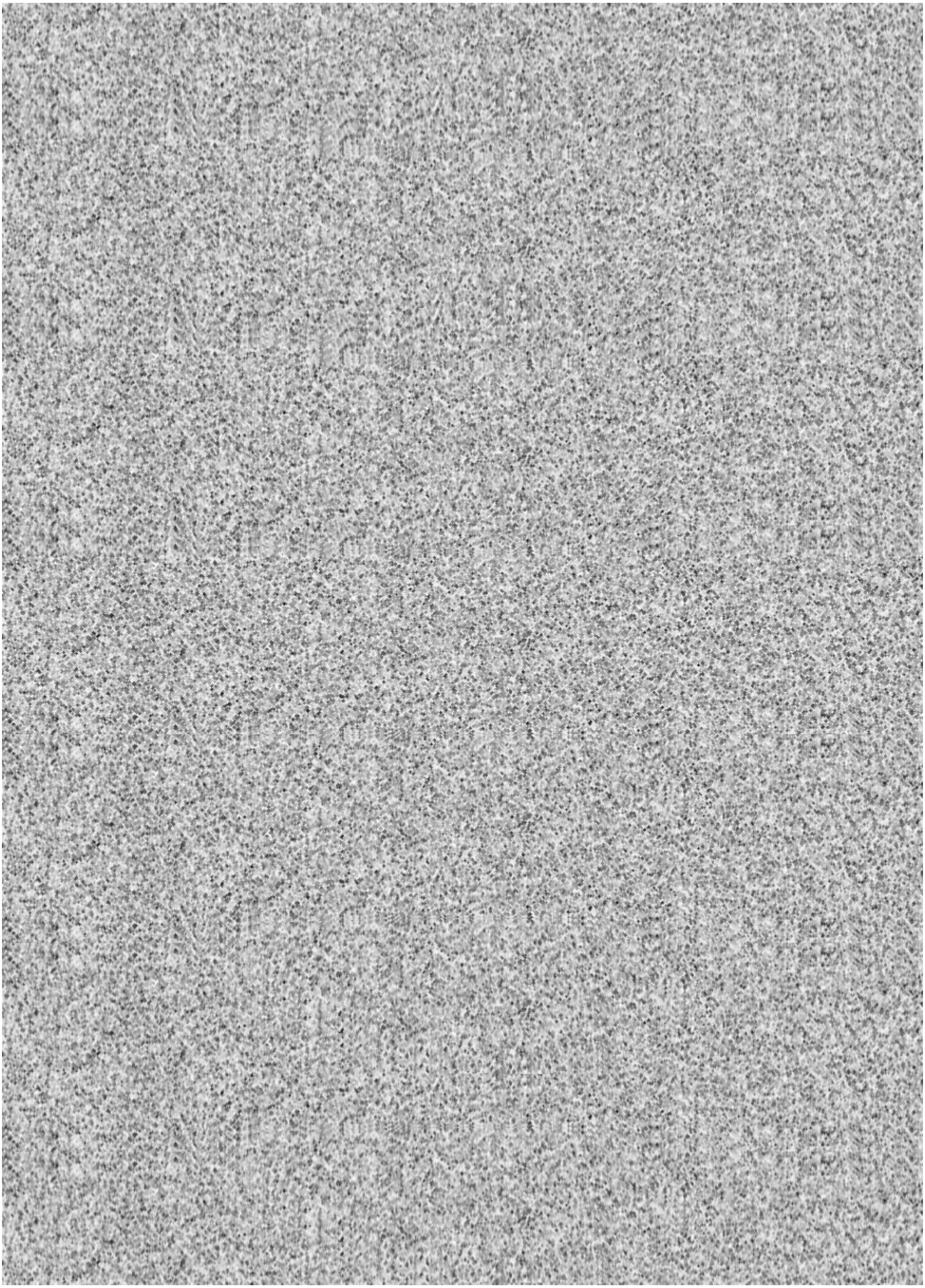
Wenhardt, Stefan, Benjamin Deutsch, Joachim Hornegger, Heinrich Niemann, and Joachim Denzler (2006). "An Information Theoretic Approach for Next Best View Planning in 3-D Reconstruction." *International Conference on Pattern Recognition*. Cited on pages 30 and 35.

Wildberger, Norman J. (2010). "Greek Geometry, Rational Trigonometry, and the Snellius-Pothenot Surveying Problem." In *Chamchuri Journal of Mathematics*. Cited on page 129.

Yau, Timothy, Minglun Gong, and Yee-Hong Yang (2013). "Underwater Camera Calibration Using Wavelength Triangulation." *Conference on Computer Vision and Pattern Recognition*. Cited on pages 132 and 133.

Yeralan, Sencer N., Timothy A. Davis, and Sanjay Ranka (2013). *Sparse QR Factorization on GPU Architectures*. Technical report. Cited on page 98.

Zhang, Zhengyou (1999). "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations." *International Conference on Computer Vision*. Cited on page 125.



Ceci n'est pas une pomme.