

EFFICIENT ALGORITHMS FOR ROBUST ESTIMATION

Rahul Raguram

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2013

Approved by:

Jan-Michael Frahm

Marc Pollefeys

Gary Bishop

Fabian Monrose

Jiří Matas

© 2013
Rahul Raguram
ALL RIGHTS RESERVED

ABSTRACT

RAHUL RAGURAM: Efficient Algorithms for Robust Estimation
(Under the direction of Jan-Michael Frahm and Marc Pollefeys)

One of the most commonly encountered tasks in computer vision is the estimation of model parameters from image measurements. This scenario arises in a variety of applications – for instance, in the estimation of geometric entities, such as camera pose parameters, from feature matches between images. The main challenge in this task is to handle the problem of *outliers* – in other words, data points that do not conform to the model being estimated. It is well known that if these outliers are not properly accounted for, even a single outlier in the data can result in arbitrarily bad model estimates. Due to the widespread prevalence of problems of this nature, the field of *robust estimation* has been well studied over the years, both in the statistics community as well as in computer vision, leading to the development of popular algorithms like Random Sample Consensus (RANSAC). While recent years have seen exciting advances in this area, a number of important issues still remain open. In this dissertation, we aim to address some of these challenges.

The main goal of this dissertation is to advance the state of the art in robust estimation techniques by developing algorithms capable of *efficiently* and *accurately* delivering model parameter estimates in the face of noise and outliers. To this end, the first contribution of this work is in the development of a coherent framework for the analysis of RANSAC-based robust estimators, which consolidates various improvements made over the years. In turn, this analysis leads naturally to the development of new techniques that combine the strengths of existing methods, and yields high-performance robust estimation algorithms, including for real-time applications.

A second contribution of this dissertation is the development of an algorithm that explicitly characterizes the effects of estimation uncertainty in RANSAC. This uncertainty arises from small-scale measurement noise that affects the data points, and consequently, impacts the accuracy of model parameters. We show that knowledge of this measurement noise can be leveraged to develop an inlier classification scheme that is dependent on the model uncertainty, as opposed to a fixed inlier

threshold, as in RANSAC. This has the advantage that, given a model with associated uncertainty, we can immediately identify a set of points that support this solution, which in turn leads to an improvement in computational efficiency.

Finally, we have also developed an approach to address the issue of the inlier threshold, which is a user-supplied parameter that can vary depending on the estimation problem and the data being processed. Our technique is based on the intuition that the residual errors for “good” models are in some way consistent with each other, while “bad” models do not exhibit this consistency. In other words, looking at the relationship between *subsets* of models can reveal useful information about the validity of the models themselves. We show that it is possible to efficiently identify this consistent behaviour by exploiting residual ordering information coupled with simple non-parametric statistical tests, which leads to an effective algorithm for threshold-free robust estimation.

To my parents, and to Anu.

For everything.

ACKNOWLEDGMENTS

This thesis, and graduate school in general, owe much to a number of wonderful people, who I shall now endeavour to thank. I'd first like to thank Jan-Michael Frahm and Marc Pollefeys, my amazing advisors. Jan and I started off at UNC at approximately the same time, and since my first day on campus, he has been incredibly generous in every respect - offering his time, ideas, advice and optimism on matters both academic and otherwise. I've truly enjoyed our brainstorming sessions; it was a rare week when Jan wouldn't drop by my office with some new idea or angle on a research problem. Seeing the kernels of these ideas slowly develop into complete research projects has been extremely rewarding, and I thank Jan for all the opportunities and the unwavering support he has given me. Marc and I have been on separate continents for most of my time at UNC, but his contribution to my research has been no less significant. Indeed, a single discussion I would have with him during his visits to UNC would provide enough material and insight to keep me thinking for the better part of a year. His perspective on research, and his insight on the important problems to solve has been invaluable.

I've been fortunate to have a wonderful thesis committee - Fabian Monrose, Gary Bishop, and Jiří Matas. I'd like to thank Fabian for being one of the most fun collaborators I've had - it truly was a pleasure to work on the iSpy project. I also thank Fabian for all the advice regarding research, presentations, and a professional career, and for relentlessly selling my work (and me!) to prospective employers. I thank Gary for his input and advice on this thesis; much of the material presented in Chapter 4 has its origin in a question that he asked elsewhere, at another student's oral exam. I'd like to thank Jiri for all the insight and input, and his focus on precision and rigor, which have gone a long way in making this thesis better. I'd also like to thank Jiri, and Ondřej Chum, for inviting me to visit Prague to collaborate on a paper.

I owe much to the fantastic students and post-docs I've had the pleasure of collaborating with at UNC: David Gallup and Brian Clipp, for being excellent office-mates and inspirational colleagues, and thanks to Dave for also being a wonderful mentor during my internship at Google; Enrique Dunn, Pierre Georgel and Christopher Zach, for providing wisdom and sage advice from the post-doc point of view; Changchang Wu, Joe Tighe, Tim Johnson, Andrew White, Enliang Zheng, Dibyendu Goswami and Yi Xu for being great co-authors, and for the many discussions and collaborations over the years; Megha Pandey, Ram Krishan Kumar, Jared Heinly, Yilin Wang, Li Guan, Seon-Joo Kim, Sudipta Sinha, Alex Keng and Yunchao Gong for their advice, input and friendship. To the members of the 3D Vision group – I'll miss our weekly lunches at Bandidos! Thanks also to Svetlana Lazebnik for her excellent courses in computer vision and machine learning, and for being a fantastic collaborator.

Thanks to Alexei Skurikhin, who hosted me on a great internship at Los Alamos, and to Steve Seitz and the Lightfield Team, for an excellent and fun summer internship at Google Seattle. Many thanks to the administrative staff at UNC, particularly Jodie Turnbull, who put up admirably (and surprisingly cheerily) with my horribly delayed paperwork.

Outside my academic group at UNC, I'm grateful to Kobus Barnard at the University of Arizona, for an excellent course in vision, which further helped cement my interest in the subject. Thanks to Michael Marcellin and the SPACL group at the University of Arizona, for showing me the ropes during my masters studies; and to the Cog Sci Brown Bag Seminars at the UofA, and the awesome series of speakers they had in 2005-2006, which finally convinced me to pursue a PhD in vision.

I'm immensely grateful to my family and friends, for being there, despite my sometimes woe-ful (in)ability to keep in touch. The Country Matters/CG/RV bunch for much needed diversions; Krishna, for introducing me to computer vision and Wagner; Akks and Ax, for being the siblings I never had; the rest of my wonderful extended family, old and new, for their unwavering support, and for listening patiently while I tried to explain the arcane things I was working on. And of course my wife Anu, who has supported me and been by my side through it all. She's kept me motivated, sane (she might disagree on this one), and entertained during the entire time I've known her.

And finally, I thank my parents who raised me right, loved me, and gave me a happy childhood which continues to this present day.

TABLE OF CONTENTS

1	Introduction	1
1.1	Thesis Statement	7
1.2	Contributions	8
1.3	Organization	9
2	Background and Related Work	11
2.1	Problem Definition	11
2.2	Random Sample Consensus	12
2.2.1	Objective function	13
2.2.2	Subset size	13
2.2.3	Stopping criterion	14
2.2.4	Threshold selection	15
2.2.5	RANSAC improvements	17
2.3	Other robust estimators	18
2.3.1	M-estimators	19
2.3.2	Least Median of Squares	20
2.3.3	M-estimator Sample Consensus (MSAC)	20
2.3.4	Maximum Likelihood Estimation Sample Consensus (MLE SAC)	21
2.4	Summary	22
3	A Comparative Analysis of RANSAC Techniques	23
3.1	Overview	23
3.2	A Universal Framework for Random Sample Consensus (USAC)	24
3.2.1	Prefiltering (Stage 0)	25

3.2.2	Sample minimal subset (Stage 1)	25
3.2.2.1	NAPSAC	27
3.2.2.2	Guided sampling for MLESAC:	27
3.2.2.3	PROSAC	28
3.2.2.4	GroupSAC	30
3.2.3	Generate minimal sample model(s) (Stage 2)	31
3.2.4	Is the model interesting? (Stage 3)	32
3.2.4.1	The $T_{d,d}$ Test	34
3.2.4.2	Bail-Out Test	34
3.2.4.3	SPRT test	35
3.2.4.4	Preemptive verification	36
3.2.4.5	Model Selection	39
3.2.4.6	DEGENSAC	40
3.2.4.7	QDEGSAC	41
3.2.5	Model refinement (Stage 4)	41
3.2.5.1	Local Optimization	42
3.2.6	USAC Implementation	43
3.2.6.1	Sample minimal subset	43
3.2.6.2	Generate minimal sample model(s)	44
3.2.6.3	Is the model interesting?	45
3.2.6.4	Generate non-minimal sample model	45
3.2.6.5	Confidence in solution achieved?	45
3.2.7	Evaluation	47
3.2.7.1	Homography	50
3.2.7.2	Fundamental matrix	54
3.2.7.3	Essential matrix	57
3.2.7.4	Motion segmentation	59
3.2.8	Summary	61

3.3	Adaptive Real-Time Random Sample Consensus (ARRSAC)	61
3.3.1	The approach	63
3.3.2	Algorithm Description	65
3.3.2.1	Generating the initial hypothesis set:	66
3.3.2.2	Preemptive evaluation:	67
3.3.3	Experimental evaluation	67
3.3.4	Summary	72
3.4	Large Scale Applications	72
3.4.1	Real-Time Robust Estimation with ARRSAC	73
3.4.2	Improved Geometric Verification with USAC	73
3.4.2.1	Related Work	73
3.4.2.2	Approach	74
3.4.2.3	Results	77
3.4.3	Summary	79
3.5	Discussion	79
4	Exploiting Uncertainty in Random Sample Consensus	81
4.1	Introduction	81
4.2	Related Work	84
4.3	Uncertainty Analysis	85
4.3.1	Uncertainty for homography estimation	85
4.3.2	Uncertainty for fundamental matrix estimation	89
4.4	Incorporating uncertainty into RANSAC	91
4.4.1	Computing (uncertain) support:	91
4.4.2	Applying the test:	93
4.4.3	Algorithm	94
4.5	Results	97
4.5.1	Homography estimation	97
4.5.2	Fundamental matrix estimation	98

4.6	Discussion	99
5	Scale Adaptive Robust Estimation via Residual Consensus	102
5.1	Introduction	102
5.2	Related Work	103
5.3	Approach	106
5.3.1	Residual Consensus	106
5.3.1.1	Uncontaminated models	106
5.3.1.2	Contaminated models	107
5.3.2	Efficient Residual Consensus	111
5.3.3	RECON: Algorithm Description	114
5.4	Experiments	115
5.4.1	Synthetic data:	116
5.4.2	Real data	119
5.5	Discussion	122
6	Conclusion	124
6.1	Summary	124
6.2	Future Work	125
	Appendices	128
	Appendix A	128
A.1	List of image pairs used	128

LIST OF FIGURES

1.1	Example estimation problems in computer vision.	4
1.2	3D reconstruction from video.	6
1.3	3D reconstruction from internet image collections.	7
3.1	The Universal RANSAC framework.	26
3.2	Randomized model verification methods (verifications per model vs. inlier ratio).	38
3.3	Randomized model verification methods (number of models vs. inlier ratio).	39
3.4	Universal RANSAC implementation (USAC-1.0)	44
3.5	Fraction of inliers returned by different RANSAC approaches.	51
3.6	Histogram of inlier counts over 500 runs, for RANSAC vs. USAC-1.0	52
3.7	Visualization of the stability of the estimation results	53
3.8	Image pairs for (quasi-)degenerate scenes.	57
3.9	Preemptive RANSAC as a function of the inlier ratio.	68
3.10	Number of hypotheses scored in preemptive RANSAC and ARRSAC.	69
3.11	Filtering features based on geometric verification.	75
3.12	Fundamental matrix estimation problem with low inlier ratio.	79
4.1	Effect of noise on RANSAC results.	82
4.2	Inlier counts and minimal samples in RANSAC.	83
4.3	Using the covariance matrix of the homography in point transfer.	88
4.4	Using the covariance matrix of the fundamental matrix in point transfer.	90
4.5	Using the covariance test in homography estimation.	99
5.1	Merging sparse 3D submodels obtained via structure-from-motion.	103
5.2	Overlap between inlier sets for uncontaminated vs. contaminated models.	108

5.3	Behaviour of the normalized partial overlap on synthetic data.	113
5.4	Plots for homography estimation with $\varepsilon = 0.4$	118
5.5	Plots for homography estimation with $\varepsilon = 0.4$ and $\sigma = 0.5$	118
5.6	Merged 3D models obtained using RECON.	121

LIST OF TABLES

3.1	USAC: Results for homography estimation.	48
3.2	USAC: Results for homography estimation (contd.)	49
3.3	USAC: Results for fundamental matrix estimation.	55
3.4	USAC: Results for fundamental matrix estimation (contd.)	56
3.5	USAC: Results for essential matrix estimation.	58
3.6	USAC: Results for motion segmentation.	60
3.7	ARRSAC: Evaluation results for epipolar geometry estimation.	71
3.8	Large-scale experiment 1 (BerlinDome-10k)	78
3.9	Large-scale experiment 2 (Berlin-2.77M)	78
4.1	cov-RANSAC: Results for homography estimation.	98
4.2	cov-RANSAC: Results for fundamental matrix estimation.	100
5.1	RECON: Results on synthetic data.	116
5.2	RECON: Results on real data.	120

Chapter 1

Introduction

Stated simply, one of the primary goals in computer vision is to extract useful information from images. This information may take on various forms: geometric information, such as the 3-dimensional structure of a scene and the location and orientation of the cameras that captured the images; photometric information, such as the colour of objects and surfaces; and semantic information, such as the identity of objects or events.

Indeed, the computer vision problem, as stated above, might appear simple at first glance, especially given the ease with which humans perform these same tasks. However, decades of research have revealed the subtle difficulties inherent in designing automated systems that are capable of emulating their human counterparts. Any computer vision system that aims to recover information from images must deal with a number of challenges: variations in viewpoint, illumination and scale, occlusions and clutter, and scene and camera motion, to name but a few. While these challenges ensure that computer vision is an incredibly hard problem to solve in general, the problem is compounded by the fact that the input to computer vision systems is itself typically unreliable. Each pixel in an input image is subject to small-scale random deviations, or noise, caused by the processes of sensing and digitization. In turn, this noise impacts any subsequent processing; for instance, any low-level image features extracted from the input images are also subject to noise and imprecision.

One of the most fundamental tasks in computer vision is the estimation of model parameters (e.g., geometric primitives such as lines or curves, motion models, camera pose parameters) from image measurements (e.g., raw pixels, image corners, matched features between images). It is

typically the case that the number of image measurements is often orders of magnitude larger than the number of parameters to be determined – for instance, a typical image today consists of millions of pixels, and often hundreds or thousands of low-level image features. This volume of data implies that model parameter estimation in computer vision is often an over-constrained problem, and can be efficiently solved using a standard technique like least-squares estimation (Strang, 1988). Moreover, if it is assumed that the image measurements are corrupted by independent Gaussian noise, then the least squares solution can be shown to be the maximum likelihood estimate (MLE) of the model parameters (von Mises, 1964).

Unfortunately, the problem is not always this simple. In addition to small-scale random measurement noise, we are often faced with data that does not conform to the model being estimated. The points, called *outliers*, represent gross errors; for instance, specular highlights or saturated sensor measurements in images, or errors introduced by the feature detection and matching algorithms. These errors, which can be arbitrarily large, cannot be simply “averaged out” as in least squares. Indeed, due to the non-robust quadratic error function in least squares, even a single outlier in the data can result in arbitrarily bad model estimates. This fact was recognized very early on; indeed, in the very first publication on least squares (Legendre, 1805):

If among these errors are some which appear too large to be admissible, then those observations which produced these errors will be rejected, as coming from too faulty experiments, and the unknowns will be determined by means of the other observations, which will then give much smaller errors.

As a consequence of this observation, much effort has been directed towards the problem of automatically identifying and compensating for outliers. The field of *robust estimation* has been well studied over the years in the statistics community (Tukey, 1977; Huber, 1981; Siegel, 1982; Rousseeuw, 1984). However, while the problem of outliers is a general one, arising in virtually any domain where model parameters are estimated from real-world data measurements, it is also arguably the case that this problem is of particular interest in computer vision and image analysis applications, where noisy and erroneous image measurements are inevitable. It is perhaps due to this reason that a rich variety of robust estimation algorithms have been developed in parallel in the computer vision community (Hough, 1962; Fischler and Bolles, 1981; Illingworth and Kittler,

1988). Today, virtually every computer vision system that aims to operate reliably on real-world data has, at its core, some robust estimation algorithm that is capable of dealing with the problem of outliers.

Figure 1.1 illustrates a sampling of such problems, where a variety of models are estimated from different kinds of data measurements. Consider Figure 1.1(a), for example, which illustrates the problem of estimating the relative pose between a pair of cameras that capture photographs of the same scene. A typical way to estimate this relation is by first extracting low-level image features, such as corners, in each of the images, which are then matched across the image pair to identify putative correspondences. Given this set of putative correspondences, standard techniques in multi-view geometry (Hartley and Zisserman, 2000; Hartley, 1997a; Nistér, 2004) can then be used to estimate the relative pose between the cameras that captured the images. While the procedure outlined above would indeed yield acceptable results given a set of perfectly matched correspondences, the situation in practice is typically far more challenging: not only are the individual feature measurements in each image subject to small-scale random variations (i.e., noise), there also exist mismatched features (i.e., gross errors) within the set of putative correspondences. If these mismatches are not properly accounted for, even a single mismatch in the set of putative correspondences can lead to arbitrarily bad estimates of the relative pose.

Over the past four decades, a wide variety of robust estimation algorithms have been proposed, both in the statistics and computer vision literature. These algorithms vary in the degree of robustness that they provide to outliers, the assumptions they make about the data, their computational complexity, and the level of generality, amongst other aspects. Of these many robust estimators, perhaps the one that is used most widely, particularly in computer vision, is Random Sample Consensus, or RANSAC (Fischler and Bolles, 1981), which forms the focus of the work in this dissertation.

The RANSAC algorithm is a remarkably simple, yet powerful, technique. In addition to its simplicity, one compelling reason for its widespread adoption is the ability of the algorithm to tolerate a tremendous level of contamination, providing reliable parameter estimates even when well over half the data consists of outliers. However, while robust, the basic RANSAC algorithm has its drawbacks: for instance, it can be computationally expensive as the level of data contamination and/or

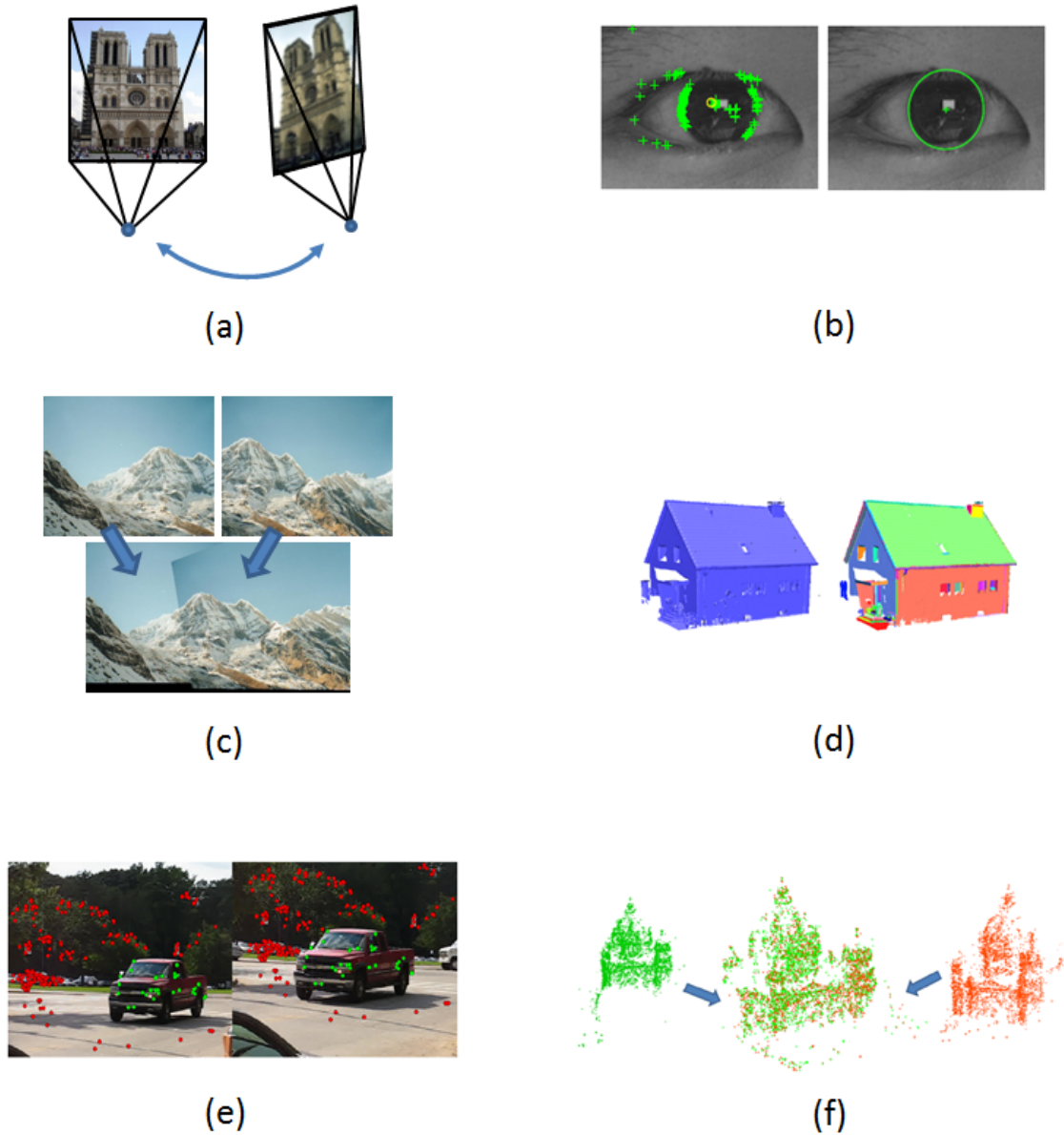


Figure 1.1: Some example problems in computer vision where models are estimated from noisy and contaminated data measurements. (a) Estimating the relative motion between a pair of camera views (Hartley and Zisserman, 2000) (b) Fitting ellipses to feature points extracted from the contours of the eye (Li et al., 2006) (c) Estimating the alignment between photographs to form panoramic images (Brown and Lowe, 2003) (d) Fitting planes to dense point clouds (Schnabel et al., 2007) (e) Estimating multiple motion models from images (Raguram et al., 2012a) (f) Estimating the 3D transformation that merges point clouds obtained via structure-from-motion (Raguram and Frahm, 2011).

the model complexity increase; it is susceptible to degenerate data configurations; and the model parameter estimates can be unstable and are typically not the globally optimal solution. Recent years

have seen exciting advances in dealing with many of these problems (Chum et al., 2003; Nistér, 2003; Matas and Chum, 2002; Capel, 2005; Matas and Chum, 2005; Chum and Matas, 2005; Chum et al., 2005; Frahm and Pollefeys, 2006; Ni et al., 2009; Sattler et al., 2009). However, a number of important issues still remain open. In this dissertation, we aim to address some of these challenges.

The goal of this dissertation is to advance the state of the art in robust estimation techniques by developing algorithms capable of *efficiently* and *accurately* delivering model parameter estimates in the face of noise and outliers. The algorithms developed in this thesis aim to improve upon the robustness, flexibility and efficiency of current techniques. In turn, we hope that these algorithmic improvements will help drive forward the state of the art in computer vision. Indeed, as the computer vision and robotics communities push towards more challenging problems on massive real-world datasets (Tanaka and Kondo, 2006; Snavely et al., 2008; Agarwal et al., 2009; Torii et al., 2009; Frahm et al., 2010) and seek real-time performance (Nister et al., 2004; Malis and Marchand, 2006; Pollefeys et al., 2008; Clipp et al., 2010), it becomes increasingly more important to develop algorithms that can handle the many challenges that are associated with this progression.

One of the motivating applications for the algorithms developed in this dissertation is that of large-scale structure from motion. Recent years have seen an explosion in consumer digital photography and a phenomenal growth of community photo-sharing websites. More than a billion photos are uploaded to the web every day, and this number shows no signs of slowing down. In addition, more and more of the Earth's cities and sights are photographed each day from a variety of viewing positions and angles, illumination conditions, and imaging devices (see, for instance, Google Street View¹ and Microsoft Bing StreetSide²). In turn, this has created a high demand for computer vision techniques that can provide intuitive and compelling visual representations of landmarks and geographic locations. This problem of estimating 3D structure from a video sequence or an unordered collection of images has received renewed attention in recent years (Schaffalitzky and Zisserman, 2002; Nister et al., 2004; Pollefeys et al., 2008; Snavely et al., 2008; Agarwal et al., 2009; Torii et al., 2009; Frahm et al., 2010). Estimating 3D structure and camera pose information from images requires the extraction of salient feature correspondences across images, followed by

¹<http://www.google.com/streetview>

²<http://www.microsoft.com/maps/streetside.aspx>

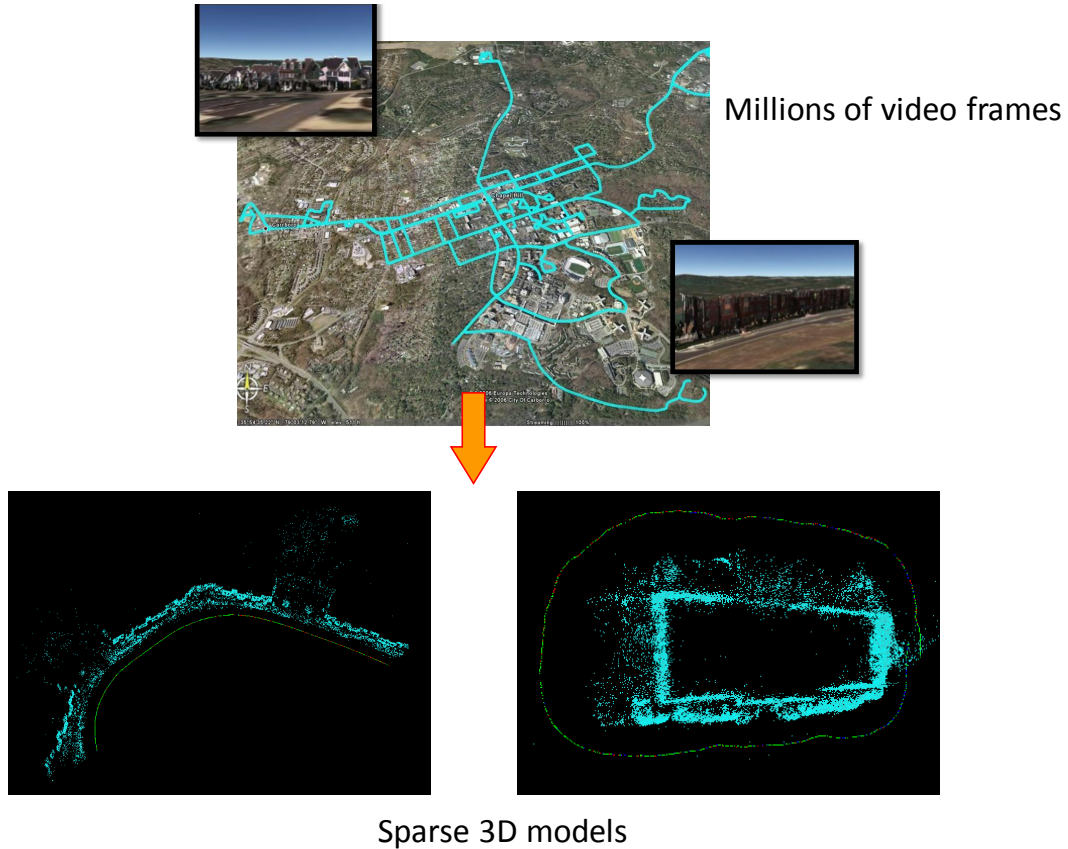


Figure 1.2: 3D reconstruction from video (see, for e.g., (Pollefeys et al., 2008)). In this dissertation, we propose a real-time algorithm that performs robust camera pose estimation at 100+ frames per second.

the robust estimation of geometric constraints that link the different views. It is worth noting that this process must be performed literally millions of times in any large-scale reconstruction system. Thus, it is essential that this module be capable of both efficient as well as robust operation; failing this, processing these large-scale datasets is simply an impossible task. The algorithms developed in this dissertation have been deployed in real-world systems that handle massive large-scale datasets, achieving *real-time* performance (Figure 1.2), and enabling the processing of *orders of magnitude* more data than the state of the art (Figure 1.3).

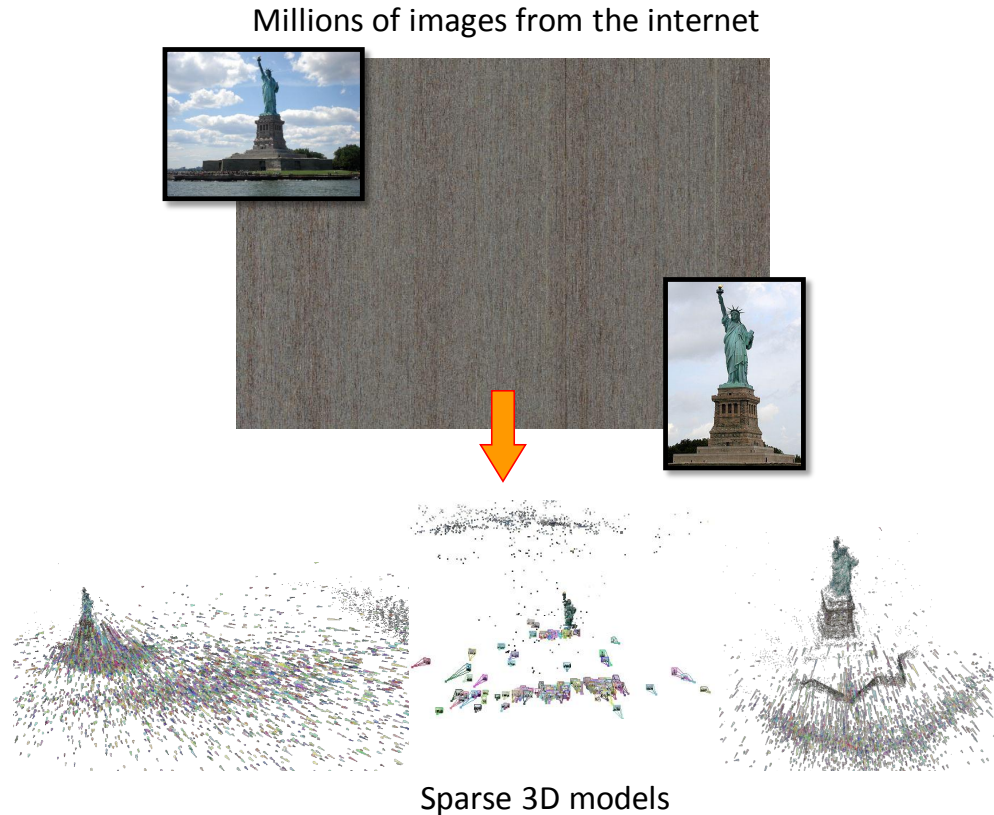


Figure 1.3: 3D reconstruction from internet image collections containing millions of images (Frahm et al., 2010). In this dissertation, we propose a high-performance robust estimator that is capable of performing geometric verification at 450+ image pairs per second, threaded across 8 CPU cores.

1.1 Thesis Statement

Given noisy and highly contaminated data measurements, it is possible to efficiently and accurately recover model parameter estimates, thus enabling real-time and large-scale applications. When the scale of noise is known, it is possible to compensate explicitly for this, yielding improved performance. In addition, in the absence of noise scale, it is still possible to perform efficient robust estimation by looking for model consistency.

1.2 Contributions

This dissertation makes the following contributions to robust estimation:

A Comparative Analysis of RANSAC Techniques (Chapter 3). We present a coherent framework for RANSAC-based robust estimation techniques, which consolidates and analyses the various improvements made over the years. In particular,

- We provide a common context for the analysis of RANSAC algorithms by introducing a new framework for robust estimation, which we call Universal RANSAC (USAC). USAC extends the simple hypothesize-and-verify structure of standard RANSAC to incorporate a number of important practical and computational considerations. This framework also provides the basis for a new high performance robust estimator that addresses many of the limitations of RANSAC within a single unified system (Raguram et al., 2012a).
- To address the issue of real-time applications, we propose a real-time reformulation of RANSAC, termed Adaptive Real-Time Random Sample Consensus (ARRSAC), which achieves high performance by integrating various ideas proposed in the literature (Raguram et al., 2008).
- We discuss the application of these two algorithms in large-scale 3D reconstruction systems. In particular, we investigate the use of an alternate form of prior information to bias sampling, with a focus on internet photo collections. As opposed to ordering feature matches based on image-to-image match scores, we build match statistics on visual words, and use this to guide the sampling. In practice, this leads to an improvement in efficiency (Raguram et al., 2012b).

Exploiting Uncertainty in Random Sample Consensus (Chapter 4). We show how to model and characterize the effects of noise and uncertainty in RANSAC, and design an algorithm that can efficiently leverage this information (Raguram et al., 2009).

- Assuming knowledge of the scale of inlier noise, we discuss how to explicitly characterize the uncertainty of the models hypothesized in the estimation procedure, in terms of their covariance.

- We then show that this covariance can be leveraged to develop an inlier classification scheme that is dependent on the model uncertainty, as opposed to a fixed threshold. In other words, given a model with associated uncertainty, we can immediately identify a set of points that support this solution, if the imperfect model estimation were taken into account. We show how this can be integrated into a robust estimation algorithm (cov-RANSAC) in order to achieve a reduction in the number of samples required.

Threshold-free Robust Estimation via Residual Consensus (Chapter 5). Finally, we address the issue of the inlier threshold, which is a user-supplied parameter that can vary depending on the estimation problem and the data being processed, and propose a new framework for robust estimation that can operate in the absence of this parameter (Raguram and Frahm, 2011).

- We make use of the simple observation that a pair of “good” models is somehow consistent in the behaviour of their residuals, and that a pair of “bad” models does not exhibit this consistency. We then develop an algorithm that identifies consistent pairs of models, without resorting to the use of an inlier threshold.
- By using residual sorting information, coupled with non-parametric statistical hypothesis tests, we develop a simple algorithm, termed Residual Consensus (RECON), that is capable of threshold-free operation, while still being efficient enough to be employed for real-world robust estimation problems.

1.3 Organization

The remainder of the dissertation is organized as follows. Chapter 2 introduces the robust estimation problem and presents the RANSAC algorithm in its standard form. Other relevant robust estimation algorithms are also discussed, with a focus on those that have found application in computer vision systems.

Chapter 3 presents a comparative analysis of RANSAC techniques, and surveys the state of the art algorithms in this area. The Universal RANSAC framework is introduced as a basis for this comparative analysis, and its performance is benchmarked relative to other RANSAC approaches. We

then extend this analysis to design a real-time RANSAC algorithm that combines various optimizations in order to achieve high performance. Finally, we also present a new scheme for weighting feature matches, based on visual word statistics, and show that this can improve the efficiency of USAC as applied to geometric verification in large scale photo collections.

Chapter 4 discusses the effect of measurement noise on the model parameters, as well as its effect on RANSAC. We then present an approach that quantifies this effect, and show how it can be leveraged to improve performance.

Chapter 5 introduces a new framework for threshold-free robust estimation, that looks at consistency between pairs of models, without requiring data or model specific threshold parameters to be provided by the user. A simple measure of consistency is proposed, and we describe an algorithm that uses this measure to achieve threshold-free robust estimation.

Finally, Chapter 6 concludes with a summary and discussion of ideas for future work.

Chapter 2

Background and Related Work

2.1 Problem Definition

We begin by defining the notion of *robustness* that we will use in this dissertation. One may argue that since one goal of computer vision is to emulate human visual perception, that the robustness of a computer vision algorithm should be evaluated against the performance of a human observer, performing a similar task. However, this is not always an appropriate benchmark, due to a number of reasons. Most notably, the notion of robustness in human vision works well for higher-level tasks and abstract concepts (e.g., “is this an picture of a chair?”). However, when presented with lower-level computational tasks (e.g., “what is the relative camera motion between this pair of images”), the human visual system is much less precise. Thus, we will not consider higher level processes when evaluating the robustness of the algorithms in this dissertation. Rather, we will use a definition of robustness that is drawn from the statistical literature (Huber, 1981):

The word “robust” is loaded with many – sometimes inconsistent – connotations. We use it in a relatively narrow sense: for our purposes, robustness signifies insensitivity to small deviations from the assumptions.

More specifically, we will define the robustness of an algorithm in terms of its ability to tolerate data contamination in the form of outliers, which represent deviations from model assumptions. That is, a robust estimator is one that is capable of providing reliable estimates of model parameters

even when some (possibly significantly large) fraction of the data is contaminated by points that do not conform to the model being estimated. Note that while the focus of this dissertation is on applications of robust estimation in computer vision, this definition of the robust estimation problem is much more general, and many of the algorithms presented here are suitable for use in other application areas as well.

Given a set of measurements \mathcal{U} containing N data points $\mathbf{u}_1, \dots, \mathbf{u}_N$, assume that a subset of size I points is consistent with some underlying model of interest. These points, or inliers, may be subject to small-scale random variations, or noise, which arise from imperfections in the measurement process. The remaining $N - I$ data points may be thought of as gross errors that do not conform to the model of interest, and are termed outliers. Let θ be a p -dimensional parameter vector that is to be estimated from the set of data measurements. The goal of the estimation procedure is to compute the model parameters corresponding to the inlier population.

Traditional maximum likelihood estimation (MLE) techniques such as least squares produce unpredictable results when given data that contains both inlier and outlier populations. Since then, much work in robust statistics and computer vision, has been targeted at automatically separating these two distinct populations in the data – inliers and outliers – prior to or during the parameter estimation process. In the remainder of this section, we will discuss some of the prominent approaches that have been proposed over the years.

2.2 Random Sample Consensus

The RANSAC algorithm was originally proposed in (Fischler and Bolles, 1981) as a general framework for model fitting in the presence of outliers. Below, we present a description of the algorithm in its most basic form.

The goal of the standard RANSAC algorithm is to efficiently explore the space of model parameters in order to maximize some objective function C . RANSAC performs this maximization in a randomized, data-driven manner. Points in the parameter space are defined by repeatedly selecting random subsets of the data points, and generating model hypotheses from the points in each subset. Each hypothesized model is then scored using the remaining data points and the hypothesis that obtains the best score is returned as the solution.

2.2.1 Objective function

In the standard RANSAC formulation, the objective function C to be maximized is the size of the *support* for a given model. More specifically, given a model with parameters θ , the support is the number of data points from \mathcal{U} that have residual errors smaller than some predefined threshold t . We assume that an error (or residual) function $r(\mathbf{u}_i, \theta)$ is provided, along with the value of the threshold t . Defining $e_i = r(\mathbf{u}_i, \theta)$, an inlier is defined as a point with $e_i^2 < t^2$, and the set of such points is called the *consensus set*. RANSAC attempts to maximize the cardinality of this set. In other words, the objective function in RANSAC can be written as

$$C = \sum_i \rho(e_i^2) \quad (2.1)$$

where the cost function $\rho(\cdot)$ is defined as

$$\rho(e_i^2) = \begin{cases} 1 & \text{for } e_i^2 \leq t^2, \\ 0 & \text{for } e_i^2 > t^2. \end{cases} \quad (2.2)$$

2.2.2 Subset size

In order to maximize the objective function C , RANSAC operates in a *hypothesize-and-verify loop*, repeatedly sampling subsets of the data to hypothesize model parameters, and then verifying the support of the models against all data points. Each subset is a *minimal sample* - in other words, the size m of each subset is defined by the minimum number of points required to uniquely compute the model parameters. This marks a departure from traditional regression techniques, which typically use all the available data to estimate model parameters; in contrast, RANSAC uses as *little* data as possible. This reason is motivated by the fact that the goal in RANSAC is to draw an uncontaminated (or all-inlier) sample from which the correct model parameters can be computed. Such a model should then, in principle, be supported by a majority of the other inliers, and thus provides us with the solution.

In this context, define $p_{success}^k$ as representing the probability of drawing an uncontaminated

subset of size $k \geq m$. We then have

$$P_{success}^k = \frac{\binom{I}{k}}{\binom{N}{k}} = \prod_{i=0}^{k-1} \frac{I-i}{N-i} \quad (2.3)$$

It can be seen that this probability of drawing an uncontaminated sample is maximized when $k = m$. Consequently, the size of the sample is chosen to be as small as is feasible.

2.2.3 Stopping criterion

Since the total number of elemental subsets, $\binom{N}{m}$, can be very large, it is computationally infeasible to exhaustively sample the space of model parameters. We now address the issue of how many samples need to be drawn before the algorithm can terminate. It can be shown (Fischler and Bolles, 1981), that in order to ensure with some confidence η_0 that at least *one* outlier-free set of m points is sampled, we must draw at least k samples, such that

$$k \geq \frac{\log(1 - \eta_0)}{\log(1 - \varepsilon^m)}, \quad (2.4)$$

where $\varepsilon = I/N$ is the fraction of inliers in the dataset. The value of η_0 is typically set to 0.95 or 0.99.

In practice, since ε is typically unknown *a priori*, it is possible to use a worst-case estimate of ε to compute the number of RANSAC trials. However, it is much more efficient to use an *adaptive* stopping criterion where ε is initialized with the worst-case assumption, and is updated as the sampling progresses, based on the size of the maximum consensus set found so far. In this context, Equation (2.4) may be viewed as the number of samples after which the likelihood of finding a better model (i.e., one with better support than the largest consensus set found so far) falls below a $1 - \eta_0$ threshold (Torr et al., 1998).

An additional useful way of thinking about the number of trials in RANSAC is the following: each sample drawn in RANSAC is either uncontaminated (a “success”) or contaminated (a “failure”). The number of uncontaminated samples drawn thus follows a binomial distribution with probability of success $p = 1/\varepsilon^m$ (the probability of drawing m inlier points). For sufficiently small values of p , this can be approximated by a Poisson distribution. In other words, the probability of

exactly n successes in k trials can be expressed as

$$p(n, \lambda) = \frac{\lambda^n e^{-\lambda}}{n!}, \quad (2.5)$$

where λ is the expected number of successes in k trials. In RANSAC, we want to ensure with some confidence that the probability of having no successes falls below some threshold. Thus, we have

$$p(0, \lambda) = e^{-\lambda} < 1 - \eta_0 \quad (2.6)$$

For $\eta_0 = 0.95$, it can be seen that on average, we expect approximately $\lambda = 3$ uncontaminated samples to be drawn before the 95% confidence in the solution is reached.

Some final observations about the stopping criterion are pertinent at this stage. Firstly, note from Equation (2.4) that the number of iterations does not depend on the number of inliers, but rather upon their proportion in the data. Secondly, note that it is this process of repeated sampling that gives the algorithm robustness to large levels of data contamination. As the inlier ratio reduces, it can be seen from equation (2.4) that the number of iterations required by RANSAC increases – however, given enough time, the algorithm will eventually find an uncontaminated sample that gathers sufficient support, and return this as the solution. Finally, one of the implicit assumptions made in the derivation of Equation 2.4 is that *any* minimal subset that contains only inliers leads to high support, and is sufficient to provide an accurate estimate of the model parameters. However, since we are relying on minimal subsets, the influence of noise is the largest possible. In practice, this can often lead to reduced accuracy, as well as an increase in the observed number of RANSAC iterations compared to the theoretically expected number.

2.2.4 Threshold selection

The most important input parameter required by RANSAC is the threshold t , which determines whether a data point supports a particular model or not. While precise knowledge of this threshold is typically unavailable, it is often possible to make a reasonable empirical choice. For instance, it is often the case when estimating multi-view geometric constraints that the acceptable reprojection error is on the order of a pixel. Furthermore, it is also possible to generalize this process of threshold

Algorithm 1 Standard RANSAC algorithm

Input: $\mathcal{U}, \eta_0, k_{max}, t$
Output: θ^*, \mathcal{I}^*
 $k = 0, I_{max} = 0$
while $k < k_{max}$ **do**
 1. Hypothesis generation
 Randomly sample minimal subset of m points
 Estimate model parameters θ_k
 2. Verification
 Calculate the support set \mathcal{I}_k
 if $|\mathcal{I}_k| > I_{max}$ **then**
 $\theta^* = \theta_k, \mathcal{I}^* = \mathcal{I}_k$
 Recompute k_{max} from eqn. (2.4) using $\varepsilon = |\mathcal{I}^*|/N$
 end if
 $k = k + 1$
end while

selection by making some assumptions about the nature of the noise affecting the data points. In particular, it is common to assume that the data points are perturbed by Gaussian noise with zero mean and standard deviation σ . In this context, the point-model error e^2 can be expressed as a sum of n squared Gaussian variables, where n is the codimension of the model (for e.g., $n = 1$ when measuring the distance of a point from a line, while $n = 2$ for point to point distance in 2D). The residuals thus follow a chi-square distribution with n degrees of freedom, and the inverse chi-square distribution can be used to determine a threshold t that captures a fraction α of the true inliers (Hartley and Zisserman, 2000):

$$t^2 = \chi_n^{-1}(\alpha)\sigma^2 \quad (2.7)$$

where χ is the cumulative chi-square distribution, and α is the fraction of inliers, typically set to 0.95. Thus, when using the threshold computed with $\alpha = 0.95$, a true inlier will be incorrectly rejected only 5% of the time.

The RANSAC algorithm, after incorporating the adaptive stopping criterion, is outlined in Algorithm 1. This is the typical form in which RANSAC is used in practice.

2.2.5 RANSAC improvements

The standard RANSAC algorithm is invariably the robust estimator of choice for computer vision applications, partly due to its simplicity (note the simple, two step hypothesize and verify structure in Algorithm 1), but also its ability to tolerate large levels of data contamination, providing reliable parameter estimates even when well over half the data consists of outliers. However, while popular, the standard algorithm does have its weaknesses. In brief, RANSAC can be computationally expensive when presented with more challenging data (e.g., low inlier ratios); it is susceptible to degenerate data configurations; and the model parameter estimates can be unstable and are typically not the globally optimal solution. Consequently, recent years have seen exciting advances in dealing with each of these problems. We now briefly touch upon some of these advances, noting that a more in-depth discussion of these algorithms will be deferred until Section 3.2.

One strategy to improve RANSAC is based on the observation that while RANSAC generates model hypotheses by sampling minimal subsets at uniformly at random from the data, it may sometimes be possible to incorporate prior information and bias the sampling with a view towards preferentially generating models that are more likely to be correct. The idea of non-uniform sampling based on prior information can have a dramatic impact on the efficiency of RANSAC, particularly for low inlier ratios. Indeed, this simple observation has been explored in a number of ways in the RANSAC literature (Myatt et al., 2002; Tordoff and Murray, 2002; Chum and Matas, 2005; Ni et al., 2009), and depending on the data and/or problem at hand, can provide orders of magnitude improvement in the number of samples required to find a solution.

While the number of samples drawn by RANSAC depends on the inlier ratio and the model complexity, an additional computational factor is the total number of correspondences N , since a hypothesized model is verified against every data point. A number of efforts in recent years have focused on making the verification stage of RANSAC more efficient (Matas and Chum, 2002; Capel, 2005; Matas and Chum, 2005; Chum and Matas, 2008). The main idea in these approaches is to conduct a statistical test on a small number of data points, and then either discard the model, or accept it and compute its full support as in the standard case. By terminating the evaluation of bad models early on, these techniques provide up to an order of magnitude improvement in runtime compared to RANSAC.

In addition to these improvements in efficiency, other approaches have focused on the question of accuracy. In (Chum et al., 2003), an approach is proposed that integrates local refinement of the model parameters into the RANSAC algorithm, improving both the accuracy of the final solution, as well as the computational efficiency of the overall algorithm. The effect of degenerate data configurations on the accuracy of RANSAC is addressed in (Chum et al., 2005; Frahm and Pollefeys, 2006), which propose techniques for detecting and recovering from degeneracy.

All the broad strategies mentioned above focus on optimizing or improving various modules of the standard RANSAC algorithm. In addition to these advances, there have been other approaches that aim to more fundamentally reformulate the algorithm in various ways. For instance, (Nistér, 2003) presents a real-time implementation called preemptive RANSAC, which switches from a depth-first approach, as in standard RANSAC, to a breadth-first approach, which is more amenable to time-bounded settings. While this enables real-time operation, the algorithm can no longer provide an η_0 -level guarantee that the correct solution has been found. Consensus Set Maximization (Li, 2009) reformulates RANSAC as a mixed integer programming problem, and solves it using a modified branch and bound algorithm. An interesting consequence of this formulation is that it provides an *optimal* solution to the robust estimation problem, unlike the randomized results provided by RANSAC. However, this comes at a significantly higher computational cost, and the algorithm is as yet impractical for real-world applications. Finally, the StaRSaC algorithm (Choi and Medioni, 2009) reformulates RANSAC to relax the requirement of a fixed threshold, by exhaustively testing all choices in some range. For each threshold, RANSAC is executed repeatedly, and threshold is chosen that shows minimum “variance of parameters”. While this approach does not require a user-specified threshold, it can also be relatively inefficient, depending on the number of thresholds tested and the number of RANSAC repetitions per threshold.

2.3 Other robust estimators

While the focus of this dissertation is primarily on RANSAC and its variants, it is worth noting that a rich variety of other robust estimation algorithms have been proposed over the years. We now present a brief discussion of some other important robust estimation algorithms which aim to optimize different cost functions, as compared to RANSAC. A more in-depth survey of these

methods may be found in (Rousseeuw and Leroy, 1987; Meer, 2004; Maronna et al., 2006).

2.3.1 M-estimators

M-estimators (Huber, 1964) are generalizations of Maximum Likelihood Estimators that seek to minimize objective functions of the form

$$C = \sum_i \rho(e_i) \quad (2.8)$$

where ρ is a robust function of the error residuals e_i , and whose growth as a function of $|e_i|$ is subquadratic (in contrast to non-robust techniques such as least squares, which use a quadratic function of the error residuals). The motivation behind M-estimators is to down-weight the penalty for large residuals, and thereby reducing the effect of outliers. The objective function in Equation 2.8 can be solved by finding θ such that

$$\sum_i \psi(e_i) \frac{\delta e_i}{\delta \theta} = 0, \quad (2.9)$$

where $\psi(e_i) = d\rho(x)/dx$ is referred to as the influence function. It is common (Holland and Welsch, 1977; Huber, 1981) to then introduce a weight function $w(x) = \psi(x)/x$, so that Equation 2.9 becomes

$$\sum_i w(e_i) e_i \frac{\delta e_i}{\delta \theta} = 0. \quad (2.10)$$

This leads to an iteratively reweighted least squares (IRLS) procedure, involving computing the weights $w(e_i)$ using the current estimate of the parameter vector θ , and then solving Equation 2.10 to estimate a new θ , holding the weights fixed.

The many M-estimators that have been proposed differ primarily in the shape of the cost function ρ , which in turn leads to different statistical properties (Beaton and Tukey, 1974; Holland and Welsch, 1977; Huber, 1981). However, while the use of modified cost functions improves robustness, it should be noted that in order to perform well in practice, and in particular to avoid local minima in the IRLS procedure, M-estimators typically require a good initial estimate of the model parameters. This is often obtained either by non-robust least squares or by the use of other robust

estimators, such as RANSAC.

2.3.2 Least Median of Squares

The Least Median of Squares technique (Rousseeuw, 1984) takes a different approach to the regression problem. Instead of minimizing the sum of the squared error residuals as in least squares (or a robust function of the residuals as in the case of M-estimators), this technique seeks to find the parameters that minimize the *median* of the squared residuals. Thus, the cost function in LMedS can be written as

$$C = \text{median}_i(e_i^2) \quad (2.11)$$

The implication of this choice is that now, up to half the points can be arbitrarily far away from the optimum model without changing the value of the objective function. Since the median is not differentiable, this cost function is typically minimized by adopting a random sampling strategy as in RANSAC (in fact, RANSAC predates LMedS by three years). While the LMedS algorithm does not require knowledge of the scale of the noise, note, however, that the method breaks down when more than half the data is contaminated by outliers. On the other hand, if the outlier fraction is known beforehand, the algorithm can be easily modified to replace the median with a different order statistic, thereby modifying the tolerance of the algorithm to a different outlier percentage.

2.3.3 M-estimator Sample Consensus (MSAC)

There have been attempts to more directly combine the advantages of M-estimator techniques with standard RANSAC. In particular, it can be observed that the process of maximizing the size of the consensus set in RANSAC is equivalent to minimizing an objective function of the form in equation (2.1), but where $\rho(\cdot)$ is now defined as

$$\rho(e_i^2) = \begin{cases} 0 & \text{for } e_i^2 \leq t^2, \\ 1 & \text{for } e_i^2 > t^2. \end{cases} \quad (2.12)$$

This cost function implies that inlying data points have no penalty, and outliers are given a constant penalty of one. Inspired by M-estimators, (Torr and Zisserman, 1998) proposed an alternate cost

function of the form

$$\rho(e_i^2) = \begin{cases} e_i^2 & \text{for } e_i^2 \leq t^2, \\ t^2 & \text{for } e_i^2 > t^2. \end{cases} \quad (2.13)$$

where outliers are still scored with a fixed penalty, but inliers are now scored based on how well they fit the model. The resulting estimator was called MSAC, or M-estimator Sample Consensus. It was shown that for the problem of epipolar geometry estimation, MSAC provides solutions that are 5-10% more accurate than standard RANSAC (Torr and Zisserman, 2000).

2.3.4 Maximum Likelihood Estimation Sample Consensus (MLE SAC)

The MSAC idea was extended a step further in the Maximum Likelihood Estimation Sample Consensus (MLE SAC) algorithm (Torr and Zisserman, 2000) where a new objective function, the likelihood of the model, is maximized.

$$C = \sum_i \log p(e_i|\theta) \quad (2.14)$$

where $p(e_i|\theta)$ characterizes the behaviour of data point residuals. To account for both noise and outliers, the error distribution for data points is represented as a mixture model of Gaussian and uniform distributions. For inliers, the residuals e_i are assumed to be Gaussian $\mathcal{N}(0, \sigma)$ with a known standard deviation, while for outliers, the distribution of residuals is assumed to be uniform $(0, v)$. In other words,

$$p(e_i|\theta) = \varepsilon \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{e_i^2}{2\sigma^2}} \right) + (1 - \varepsilon) \frac{1}{v} \quad (2.15)$$

where θ are the model parameters, σ is the (assumed known) standard deviation of the measurement noise, v is a constant (for instance, when extracting feature matches between images, this could be the diameter of the search window) and ε is the (unknown) inlier ratio, which in this case plays the role of the mixing parameter.

Like MSAC, MLE SAC offers a 5-10% improvement in model accuracy over basic RANSAC, but at the expense of additional computational cost. In practice, the main benefits offered by MSAC and MLE SAC over RANSAC are observed in cases where the likelihood function over model pa-

rameters is relatively flat, implying that there are several competing models near the vicinity of the global minimum. In such scenarios, using robust cost functions can better distinguish between multiple models that may produce identical inlier scores. On the other hand, in cases where the likelihood function has a strong peak implying the existence of a dominant model in the data, the effect of the different cost functions is minimal. In particular, as noted in Sec. 2.2.3, on average, only three uncontaminated samples are drawn before the standard termination criterion in RANSAC (as well as MSAC/MLESAC) is met (Equation 2.6). Accordingly, when there is a dominant model in the data, the cost functions in both RANSAC and MSAC/MLESAC are likely to pick the same model from this set.

2.4 Summary

In summary, it is evident from the discussion in this section that the field of robust estimation is a very active one, with a variety of challenges and open problems. In this context, our focus in this dissertation is on a category of popular algorithms – RANSAC-based robust estimators. It is worth noting that, despite their existing limitations, RANSAC-based algorithms are arguably the most widely used robust estimators in computer vision today, and form a crucial cog in many real-world vision systems. It thus becomes important to study and improve upon these algorithms, in order to help drive forward the state of the art in computer vision. In the remainder of this dissertation we present various algorithms that aim to achieve this goal.

Chapter 3

A Comparative Analysis of RANSAC Techniques

3.1 Overview

Over the past decade, a number of improvements to RANSAC have been proposed, each addressing a specific weakness of the original algorithm. However, while recent work has focused on addressing issues with RANSAC, relatively less attention has been paid to a unified review of these developments. One effort in this direction is that of (Chum and Matas, 2008), which analyses the performance of a few recent RANSAC variants on a selection of geometric estimation problems. In addition, while there exist a few surveys that focus on statistical robust estimators (for instance, Meer, 2004), the coverage of algorithms in these surveys is limited, and does not include many recent RANSAC-based techniques that are in widespread use, particularly in computer vision applications. From an evaluation perspective, to the best of our knowledge, there are no publicly available “reference” implementations, or benchmark datasets, for the evaluation of newly developed robust estimation algorithms. One of the goals of this dissertation is to fill this gap, by providing a common context within which to study these disparate techniques. In turn, this analysis will lead naturally to the development of new algorithms that combine the strengths of existing methods. In a sense, this step will consolidate years of research in RANSAC-based robust estimation techniques, to develop new frameworks for high-performance robust estimation. We now describe two frameworks that

result from this analysis.

3.2 A Universal Framework for Random Sample Consensus (USAC)

In Algorithm 1, we summarized the operation of RANSAC in the form in which it is often applied. However, note that there are some important limitations of the technique when used in this form.

1. **Efficiency:** From Section 2.2, note that time complexity in RANSAC depends on the subset size m , the inlier ratio ε , and the number of data points N . Thus, for more challenging problems (e.g., higher model complexity, more contaminated data, more data points) the computational cost of RANSAC can be prohibitively high. In addition, note that there is an implicit assumption in equation (2.4), that a model generated from an all-inlier sample will result in all inliers being found. In practice, this is seldom the case, since the use of minimal samples implies that the model hypotheses themselves are “noisy”. Consequently, the number of iterations required by RANSAC is typically more than that predicted by equation (2.4) (Tordoff and Murray, 2002; Chum et al., 2003).
2. **Accuracy:** As noted in Section 2.2.2, the reliance of RANSAC on minimal samples is primarily due to efficiency considerations. However, model parameters that are computed from minimal subsets may be affected significantly by noise in the minimal sample. It is thus worth stressing that the model parameters returned by standard RANSAC can often be sub-optimal and must be refined prior to being used for additional tasks.
3. **Degeneracy.** The objective function in RANSAC relies on the assumption that a model generated from a contaminated minimal sample will have low support, thus making it possible to distinguish between correct and incorrect models. This assumption may be violated when degenerate configurations are encountered. For instance, in the case of epipolar geometry estimation with the 7-point algorithm (Hartley and Zisserman, 2000), when the minimal sample comprises of five points that lie on a plane and two *arbitrary* points, the resulting model is consistent with all points on the plane, though the resulting epipolar geometry is incorrect.

A number of techniques have been proposed in recent years to deal with these, as well as other, limitations of RANSAC. To put these disparate techniques into context, we extend the simple

hypothesize-and-verify structure of RANSAC and propose a *Universal RANSAC* framework, illustrated in Figure 3.1. This framework generalizes RANSAC by incorporating a number of important practical and computational considerations. In the remainder of this section, we describe the various components in Figure 3.1, and discuss the many variations and extensions proposed in the literature within this unified context. In particular, note that all the techniques discussed below can be viewed as special cases of the USAC framework.

3.2.1 Prefiltering (Stage 0)

The input to RANSAC is a set \mathcal{U} , consisting of N data points. As seen in Section 2.2.3, the number of iterations required by RANSAC is a function of the contamination in this set. Given that the runtime of RANSAC is closely tied to the inlier ratio of the data, a few efforts have been targeted towards improving the quality of the data points that form the input to the algorithm. Strictly speaking, this step is not a part of the core framework itself, since it involves “cleaning up” the input data before the algorithm commences. Furthermore, this step typically involves problem specific constraints and heuristics, and is more closely tied to the step of data generation. It is worth noting, however, that performing this step when applicable can significantly benefit RANSAC.

Considering the specific problem of estimating multi-view geometric relations, the input to RANSAC is typically a set of feature matches. Considerable effort has been directed towards imposing constraints on the output of a feature detector in order to provide a more reliable set of feature matches (Schmid and Mohr, 1997; Tuytelaars and Gool, 2000; Jung and Lacroix, 2001; Cech et al., 2008). In the RANSAC literature, one recent effort that investigates the computational benefit of a spatial consistency filter applied to an initial set of feature matches is SCRAMSAC (Sattler et al., 2009). This consistency filter results in a reduced set of matches that are more likely to be correct, in turn speeding up RANSAC by up to an order of magnitude.

3.2.2 Sample minimal subset (Stage 1)

Stage 1a: Sampling

In standard RANSAC, minimal subsets are generated by sampling uniformly at random from the set of data points. In the most general setting, this implies that no assumptions are being made about

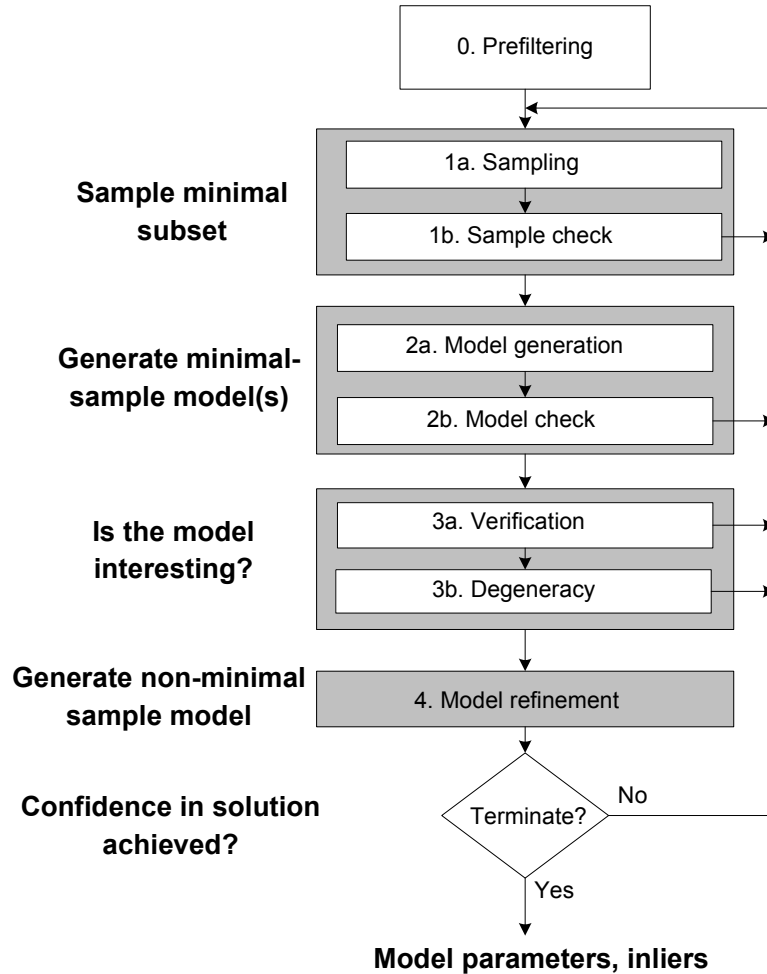


Figure 3.1: The Universal RANSAC framework. The figure represents a generalized RANSAC framework that takes into account a number of important practical and computational considerations. USAC is a synthesis of various RANSAC techniques, and provides a unified view of recent advances in this area. In addition to providing a common context for the different RANSAC variants, the combination of these techniques into a single robust estimation framework allows them to interact with, and benefit from, each other. Thus, an implementation of the USAC framework as sketched above, with state of the art algorithms for each individual module, allows us to address the various limitations of RANSAC in a unified manner.

the data. In many practical situations, however, it may be possible to incorporate prior information and bias the sampling with a view towards preferentially generating models that are more likely to be correct. This can have a dramatic effect on the efficiency of RANSAC, particularly for low inlier ratios. This simple observation has been explored in a number of ways in the RANSAC literature. A review of some prominent work in this area follows.

3.2.2.1 NAPSAC

The N-Adjacent Points Sample Consensus (NAPSAC) algorithm (Myatt et al., 2002) uses the observation that often, inliers in the data tend to be “closer” to one another than to outliers. Considering an n -dimensional space, assume that outliers are distributed uniformly within a bounded region of the space, and that inliers are distributed uniformly on a d -dimensional manifold within the same region. Now consider a hyper-sphere of radius r , centered at a point on the manifold. The number of inliers within the hypersphere is proportional to r^d , whereas the number of outliers is proportional to r^n . Since $n > d$, this implies that as the radius decreases, the probability of finding an outlier in the hyper-sphere decreases faster than the probability of finding an inlier. Thus, in NAPSAC, an initial data point \mathbf{x}_0 is selected uniformly at random. Next, the algorithm finds a set of points $S_{\mathbf{x}_0}$ such that all points in the set lie within a hyper-sphere of radius r , centered around \mathbf{x}_0 . If the number of points in $S_{\mathbf{x}_0}$ is less than the size of the minimal sample, then this set is discarded, and the sampling process repeats afresh. Otherwise, points are selected uniformly from $S_{\mathbf{x}_0}$ until a minimal sample of size m has been selected.

One of the notable advantages of NAPSAC arises in the case of very high dimensional models, where the probability of drawing an uncontaminated sample is often low, even for relatively uncontaminated datasets. On the other hand, one of the disadvantages of a NAPSAC style strategy is that enforcing the spatial proximity requirement can make the algorithm prone to degeneracies. In fact, specific techniques have been proposed to enforce exactly the *opposite* requirement – that samples are prevented from incorporating data points that are too close to each other (Zhang et al., 1995; Zhang and Kanade, 1998).

3.2.2.2 Guided sampling for MLESAC:

One of the deficiencies of the MLESAC approach (Section 2.3.4) lies in the computation of the mixing parameter ε . As noted earlier, this parameter is estimated by EM for every hypothesized model. However, note that this parameter is really a prior property of the data, and not dependent on the individual models. In (Tordoff and Murray, 2002), an observation is made that better hypothesis have better likelihood scores for *any* mixing parameter. Thus, it is not necessary to solve for it using EM at every iteration. To select the best hypothesis, it is actually sufficient to use a fixed mixing

parameter for all hypotheses.

Going further, it is argued in (Tordoff and Murray, 2002) that instead of using a global mixing parameter, it is preferable to use individual priors on a data point being correct, if available. In other words, equations 2.15 can be rewritten as

$$p(e_i) = p(v_i) \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{e_i^2}{2\sigma^2}} \right) + (1 - p(v_i)) \frac{1}{v} \quad (3.1)$$

where v_i is an indicator variable denoting whether the i -th data point is an inlier or not, and $p(v_i)$ are the prior probabilities. For two-view geometric estimation problems, one way to obtain these priors is to use scores from the feature matching stage. More specifically, assuming the i -th feature has n_i possible matches, denote the correctness of each of these matches as v_{ij} , and the quality score as s_{ij} with $j = 1 \dots n_i$. For a number of narrow-baseline image pairs, the distribution of the quality scores s_{ij} for mismatches (\bar{v}_{ij}) was empirically shown to follow the distribution

$$p(s_{ij}|\bar{v}_{ij}) = \frac{3(1 - s_{ij})^2}{4} \quad (3.2)$$

and for correct matches,

$$p(s_{ij}|v_{ij}) = a \frac{(1 - s_{ij})}{\alpha^2} e^{-\left[\frac{1-s_{ij}}{\alpha}\right]^2} \quad (3.3)$$

where a is a normalisation constant and α is a ‘‘compactness’’ parameter. These probabilities can then be used to compute a value for $p(v_{ij}|s_{i,1 \dots n_i})$, which is the probability that a tentative match is correct, given the scores of all tentative matches. This probability can be used in equation (3.1) as an estimate of the prior probability $p(v_i)$. In addition, hypothesis generation can be performed through the use of Monte-Carlo sampling based on the computed priors, which was shown to reduce the number of iterations in MLESAC by close to an order of magnitude.

3.2.2.3 PROSAC

The Progressive Sample Consensus (PROSAC) algorithm (Chum and Matas, 2005) uses a measure of the quality of the data points in order to preferentially generate hypotheses that are more likely to be valid. As opposed to Guided MLESAC (Section 3.2.2.2), PROSAC is less reliant on having a very accurate prior distribution; rather, the prior is used to simply sort the points, and a new

sampling strategy is used to select samples in a more meaningful order compared to RANSAC.

PROSAC can be viewed as a process that starts by deterministically testing the most promising hypotheses (generated from the most promising data points), gradually shifting to the sampling strategy of RANSAC as the confidence in the *a priori* sorting based on quality scores decreases. Consider a sequence of T_N samples of size m drawn by RANSAC from the set of all N correspondences. This sequence is denoted by $\{\mathcal{M}_i\}_{i=1}^{T_N}$. Let $\{\overline{\mathcal{M}}_{(i)}\}_{i=1}^{T_N}$ denote a sequence of the same samples, but now sorted in descending order of sample quality. To find a sampling strategy, let \mathcal{U}_n denote a subset of \mathcal{U}_N containing n points with the highest quality. The sequence $\{\mathcal{M}_i\}_{i=1}^{T_N}$ contains, on average, T_n samples containing points only from \mathcal{U}_n , where

$$T_n = T_N \frac{\binom{n}{m}}{\binom{N}{m}} \quad (3.4)$$

Since we are interested in a sampling strategy that uses a progressively larger sampling pool, a recurrence relation for T_{n+1} may be obtained as

$$T_{n+1} = \frac{n+1}{n+1-m} T_n \quad (3.5)$$

Observe that there are T_n samples containing points only from \mathcal{U}_n and T_{n+1} samples containing points only from \mathcal{U}_{n+1} . Since the subsets \mathcal{U}_{n+1} and \mathcal{U}_n are related as $\mathcal{U}_{n+1} = \mathcal{U}_n \cup \mathbf{u}_{n+1}$, there are thus $T_{n+1} - T_n$ samples that contain a data point \mathbf{u}_{n+1} and $m - 1$ data points drawn from \mathcal{U}_n . Therefore, to generate samples in the order $\mathcal{M}_{(i)}$, one can draw $T_{n+1} - T_n$ samples consisting of a point \mathbf{u}_{n+1} and $m - 1$ data points drawn at random from \mathcal{U}_n , for $n = m \dots N$.

In practice, PROSAC has been shown to achieve very significant computational savings, since good hypotheses are generated early on in the sampling process. However, while the ordering based on quality scores is less “local” than the NAPSAC strategy, it has been observed that invariably, PROSAC too needs safeguards against degeneracies (Raguram et al., 2008). In addition, when the quality scores are less helpful, for instance in the case of scenes with significant repetitive structure, the performance gains obtained with PROSAC are less significant (Ni et al., 2009; Sattler et al., 2009).

3.2.2.4 GroupSAC

While the NAPSAC strategy aimed to exploit the observation that inliers tend to be “closer” to each other than outliers, the recent GroupSAC algorithm (Ni et al., 2009) uses a more general version of this observation – that inliers are often more “similar” to each other. Assuming that we can separate data points into a number of groups that are similar according to some criterion, GroupSAC is a sampling strategy that aims to exploit this grouping.

In RANSAC, the number of inliers in a minimal sample \mathcal{S} can be expressed as $I_{\mathcal{S}} \sim B(m, \varepsilon)$, where $B(m, \varepsilon)$ is a binomial distribution with parameter ε , the inlier ratio. Thus, the probability of an all inlier subset may be expressed as

$$p(I_{\mathcal{S}} = m) = \prod_{x_i \in \mathcal{S}} p(v_i = 1) \quad (3.6)$$

where v_i is an indicator variable denoting x_i being an inlier. In the absence of prior information, this probability reduces to ε^m . While the techniques from (Chum and Matas, 2005; Tordoff and Murray, 2002) leverage the observation that $p(v_i = 1)$ is not necessarily the same for all points, the inlier probabilities are still considered to be independent.

In GroupSAC, the data points are partitioned into a set of groups $\{G_i\}$, $i = 1 \dots K$, with the intuition that these groups tend to have either a high or a low fraction of inliers. Considering the feature matching problem for an image pair, groups may be obtained by optical flow based clustering. The assumption here is that the largest, more consistent clusters tend to have a higher fraction of inliers. As in PROSAC, sampling scheme is devised that starts by testing a small subset of the most promising groups, and gradually expands this to include all points.

While GroupSAC was shown to improve sampling efficiency, its applicability hinges on finding a grouping that makes sense in the context of the problem being considered. Furthermore, since the grouping stage is a part of the robust estimation module, the particular grouping strategy used (e.g., optical flow, image segmentation, etc.) should itself be very efficient, so as to not appreciably increase the overall runtime.

Stage 1b: Sample check

Once a minimal sample has been generated, the next step in standard RANSAC is to compute model parameters from this minimal sample. In some cases, however, it may be possible to insert a simple test to first check whether the sample is suitable for computing model parameters. For instance, when computing a homography from a sample of four points, a degenerate case occurs when three of the four points are collinear. Including a simple test for collinearity would help immediately discard such samples. While this step can eliminate spurious cases, care must be taken to ensure that the test does not significantly impact the overall runtime.

3.2.3 Generate minimal sample model(s) (Stage 2)

Stage 2a: Model generation

In this step, model parameters are computed from the minimal subset. Note that in the minimal case, a single sample could result in the generation of multiple solutions, each of which needs to be evaluated by the algorithm. It is also worth noting that since the model parameters are generated from minimal samples, the influence of noise is the largest possible. Consequently, it is advisable that these minimal sample models should be used only after some form of refinement, either within the algorithm, or through a *post hoc* procedure.

Stage 2b: Model check

In standard RANSAC, once a model has been generated, its support is determined by verifying the model against all data points. It may be possible, however, to first introduce a preliminary test that checks the model based on application-specific requirements or constraints, and then performs the verification only if required. As a simple example, consider the case of fitting a circle to a set of points, where only circles with a certain range of radii are desired. By first testing the validity of the model parameters, it may be possible to completely skip the more expensive verification stage.

In (Chum et al., 2004), the computational benefit of a model check stage is demonstrated for the case of epipolar geometry estimation. By noting that for real cameras, only points in front of the camera are visible, it is possible to enforce *oriented* epipolar constraints via the model check stage.

Each of the fundamental matrices is first tested to see whether it satisfies the oriented constraint. If this test is passed, the support of the fundamental matrix is computed as usual. If not, then the model may be discarded without further inspection. Since the oriented test is very efficient relative to verifying the model against all data points, this leads to a significant computational benefit. In (Chum et al., 2004), it was shown that, depending on the scene, anywhere between 10%-92% of the models could be rejected just based on this simple oriented constraint test.

3.2.4 Is the model interesting? (Stage 3)

In standard RANSAC, the goal of the model verification stage is to compute the support of the model by evaluating all data points, and to return the model that gathers the largest consensus set. We take a more general view, where the goal of this stage can be posed as providing an answer to the question: *is the model interesting?*

Models generated from contaminated samples have arbitrary model parameters, and thus reveal no useful information about data points that are not contained in the minimal sample itself. These models can be viewed as “non-interesting”, and the goal of the verification process is to filter out these models. On the other hand, “interesting” models, or those that are likely to lead to the largest consensus set, are worth investigating further. In the USAC formulation, the notion of a model being “interesting” has two components:

- The model is likely to be uncontaminated and gather large support.
- The model is non-degenerate.

While RANSAC seeks the answer to this question, the standard formulation is limited by the assumptions made, as well as efficiency concerns. In the USAC framework, the objective is to answer this question while improving upon both the efficiency and the robustness of the standard algorithm.

Stage 3a: Model verification

While the number of samples drawn by RANSAC depends on the inlier ratio and the model complexity (equation (2.4)), an additional computational factor is the total number of correspondences N , since a hypothesized model is verified against every data point. In particular, the runtime of

RANSAC can be expressed as

$$t = k(t_M + \bar{m}_S t_V) \quad (3.7)$$

where k is the number of samples, m_S is the number of models per minimal sample, and t_M and t_V denote the time required to compute model parameters, and to verify the model, respectively. If for convenience, the time taken to verify a single point is chosen as the unit of time, then for RANSAC, $t_V = N$, where N is the number of data points. When N is large, the verification stage can thus consume a large fraction of the total runtime. However, as noted in Section 2.2.3, the number of uncontaminated samples generated in RANSAC is typically small. In turn, this implies that almost almost all the hypothesized models are likely to be contaminated. If we assume that these contaminated models will be consistent with only a few data points, then it may be possible to discard “bad” models early on in the verification stage. As noted earlier, this assumption may be violated when degenerate data is encountered; this issue will be discussed shortly.

A number of efforts in recent years have focused on making the verification stage of RANSAC more efficient. In this section, we review some of the important approaches in this regard. The techniques in this section propose variations of the same basic idea: conduct a statistical test on a small number of data points, and discard or accept the model based on the results of the test. The basic statistical test can be generalized as follows: given a model from the hypothesis generation stage, determine whether the model is “good” – i.e., it leads to the solution with maximal support, or it is “bad” – i.e., one of the data points in the sample is an outlier. The property “good” is a hidden state that is not directly observable, but is statistically linked to observable events. In the case of RANSAC, the observable events are the results of the individual point evaluations.

The aim of the statistical test is to reduce the number of verified data points, and thereby the time complexity of the verification step. On the other hand, this test has the side-effect that a “good” model may be incorrectly rejected, which in turn causes an increase in the total number of samples drawn to achieve the η_0 confidence in the solution. Typically, as more data points are processed by the test, the probability that a “good” model will be rejected reduces. The task is thus to find a balance between the number of data points evaluated by the test and the increase in the number of samples, such that the total runtime t from equation (3.7) is minimised.

3.2.4.1 The $T_{d,d}$ Test

In (Matas and Chum, 2002), model verification is first performed using a subset of d randomly selected points (where $d \ll N$). The remaining $N - d$ points are evaluated only if the first d points are all inliers to the model. An expression for the number of verified correspondences per test, t_V , can be derived as a function of d as

$$t_V(d) = P_g((1 - \alpha)N + \alpha\bar{t}_\alpha) + (1 - P_g)(\beta N + (1 - \beta)\bar{t}_\beta) \quad (3.8)$$

where the first term represents the result of the test on a good sample, and the second is contributed by a bad sample. P_g is the probability of drawing a good sample, α is the probability that a good sample does not pass the pre-verification test, and β is the probability that a bad sample passes the test. \bar{t}_α and \bar{t}_β represent the average number of points tested in the two cases. Note that the efficiency of the test hinges on the relation $\beta \ll (1 - \alpha)$, in order for a bad sample to be consistent with far fewer points than a good sample. The value α may be approximated as $1 - \varepsilon^d$ and β as δ^d , where δ is the probability that a data point is consistent with an incorrect model. In (Matas and Chum, 2002), an optimal setting of $d = 1$ was derived by minimizing the average time spent in the pre-verification step. In other words, the $T_{1,1}$ test checks a model against a randomly drawn data point. If the model is consistent with the point, verification continues. If not, the model is discarded and a new sample is generated. Note that a valid hypothesis may be mistakenly rejected by the $T_{d,d}$ test. Thus, one of the consequences of this approach is that it requires many more hypotheses than standard RANSAC. However, providing the hypothesis generation step is not too expensive, the overall runtime is typically reduced due to the preverification procedure (Matas and Chum, 2002).

3.2.4.2 Bail-Out Test

The idea of early termination of bad hypotheses was further extended in (Capel, 2005). Given a model to be scored, a randomly selected subset of n points is evaluated against the model. If the observed inlier ratio within this subset, ε_n , is significantly less than the best inlier ratio observed so far, ε^* , then it is unlikely that the current model will yield a larger consensus set than the current maximum and can be discarded. More formally, given that I_n inliers have been seen within a subset

of size n , the probability that the total number of inliers \bar{I} for the current hypothesis will be greater than the current best inlier count, I^* is given by

$$p_{conf} = p(\bar{I} > I^*) = \sum_{\bar{I}=I^*}^N p(\bar{I}|I_n, n, N) \quad (3.9)$$

When this probability drops below a certain threshold (e.g., 1%), the model can be discarded. Since equation (3.9) is difficult to directly compute, an alternative is presented. The distribution of the number of inliers I_n in a subset of n points follows a hyper-geometric distribution. The idea behind the bail-out test is to compare the number of inliers seen within a subset of n points against a lower bound I_n^{min} . If $I_n < I_n^{min}$, then bail-out occurs. In order to compute I_n^{min} , note that the hypergeometric lower bounds may be approximated either using a binomial distribution for small values of n , or as a normal distribution for large values of n .

It was shown in (Capel, 2005) that the bail-out test typically results in a factor of 2-7 improvement in performance compared to standard RANSAC. Note, however, that the strategy outlined in (Capel, 2005) does not account for the case where a good hypothesis might be incorrectly rejected by the bail-out test. Thus, while the test is effective in practice, it is not, strictly speaking, optimal.

3.2.4.3 SPRT test

Most recently, an optimal randomized model verification strategy was described in (Matas and Chum, 2005; Chum and Matas, 2008). The test is based on Wald's theory of sequential testing (Wald, 1947). The theory was first applied for quality control in industrial inspection, with the goal of deciding whether a batch of products was "good" or "bad", while making the smallest number of observations possible. The verification stage of RANSAC has a similar goal: to decide whether a model is "good" (H_g) or "bad" (H_b). Wald's SPRT test is a solution of a constrained optimization problem, where the user supplies acceptable probabilities for errors of the first type (rejecting a good model) and the second type (accepting a bad model) and the resulting optimal test is a trade-off between the time to decision, and the errors committed. The SPRT test is based on the likelihood ratio

$$\lambda_j = \prod_{r=1}^j \frac{p(x_r|H_b)}{p(x_r|H_g)} \quad (3.10)$$

where x_r is equal to 1 if the r -th data point is consistent with a given model, and 0 otherwise. $p(1|H_g)$ denotes the probability that a randomly chosen data point is consistent with a good model, and this can be approximated by the inlier ratio ε . Similarly, $p(1|H_b)$ is the probability that a randomly chosen data point is consistent with a bad model, and this can be modeled using a Bernoulli distribution with parameter δ . If, after evaluating j data points, the likelihood ratio (3.10) becomes greater than some threshold A , the model is rejected.

The decision threshold A is the main parameter of the SPRT test and can be set to achieve optimal runtime, assuming that the two parameters ε and δ are known *a priori*. In practice, however, these parameters are typically unknown, and have to be estimated during the evaluation process, adjusting the value of the threshold A , based on current estimates. For the case of multi-view geometry problems, for instance, an initial estimate of the parameter δ can be obtained through geometric constraints. An initial value of ε can be estimated from the maximum number of RANSAC iterations that the user is willing to perform, and this may be updated by using the size of the largest support found so far.

In the multi-view geometry estimation experiments in (Chum and Matas, 2008), it was shown that the randomized verification procedure based on the SPRT test results a factor 2-9 runtime improvement over standard RANSAC. While the performance of the bail-out and SPRT tests are comparable for the case of higher inlier ratios (where the time to solution is low), the SPRT test was found to perform approximately 20% faster than the bail-out test for more challenging problems with lower inlier ratios.

3.2.4.4 Preemptive verification

The verification schemes discussed above are all *depth-first* in nature, meaning that a particular model is completely evaluated before moving on to the next model. An alternate strategy is that of *breadth-first* evaluation, introduced in (Nistér, 2003). A fixed number of model hypotheses are generated beforehand and scored only on a subset of the data points. Subsequently, the models are reordered based on the results of this scoring procedure, and only a fraction of these are evaluated on the next subset of data. The process continues until only one hypothesis remains, or all data points have been used. This type of evaluation procedure lends itself naturally to real-time applications,

with a bound on the acceptable runtime.

A non-increasing preemption function $f(i), i = 1, \dots, N$ defines the number of model hypotheses retained before evaluating the i -th data point. The preemption function used in (Nistér, 2003) is

$$f(i) = \lfloor M2^{(-\frac{i}{B})} \rfloor \quad (3.11)$$

where M is the number of models in the initial set, and B defines the block size, which is the number of data points that a model is evaluated against, before the reordering and preemption step takes place. In effect, this preemption function halves the number of surviving models after each block of data points has been evaluated.

It is important to note that the objectives of standard RANSAC and preemptive RANSAC are significantly different. The goal in RANSAC is to find, with predefined confidence, the best model according to some cost function; the goal in preemptive RANSAC is to find the best model within a fixed time budget. Since preemptive RANSAC starts with a set of hypotheses of fixed size, it is not meant to adapt to the level of data contamination. In particular, if the inlier ratio in the data is overestimated (i.e., too few hypotheses are generated), then the algorithm can return an arbitrarily bad solution. On the other hand, for low-contamination problems, where only a few samples are needed to find a solution, preemptive RANSAC can be much slower than standard RANSAC.

To experimentally evaluate the effectiveness of the three depth-first verification methods (Sections 3.2.4.1–3.2.4.3) presented here, we compare their performance on synthetically generated test data (these results were presented in (Raguram et al., 2008), with similar results being independently presented in (Chum and Matas, 2008)). For this experiment, we estimate the epipolar geometry from synthetically generated feature matches, over a wide range of variations in inlier ratio (ε) and number of matched features (N). Gaussian noise ($\mu = 0, \sigma = 0.2$) was added to the pixel locations. The performance of the $T_{d,d}$, bail-out, and SPRT tests are compared against baseline RANSAC. A representative subset of the results are shown in Figures 3.2 and 3.3. Figure 3.2 shows the number of verifications per model as a function of the inlier ratio ε , for four different values of N . Figure 3.3 shows the variation in the number of models generated, again as a function of ε and N . The results are averaged over 200 runs for each tested case.

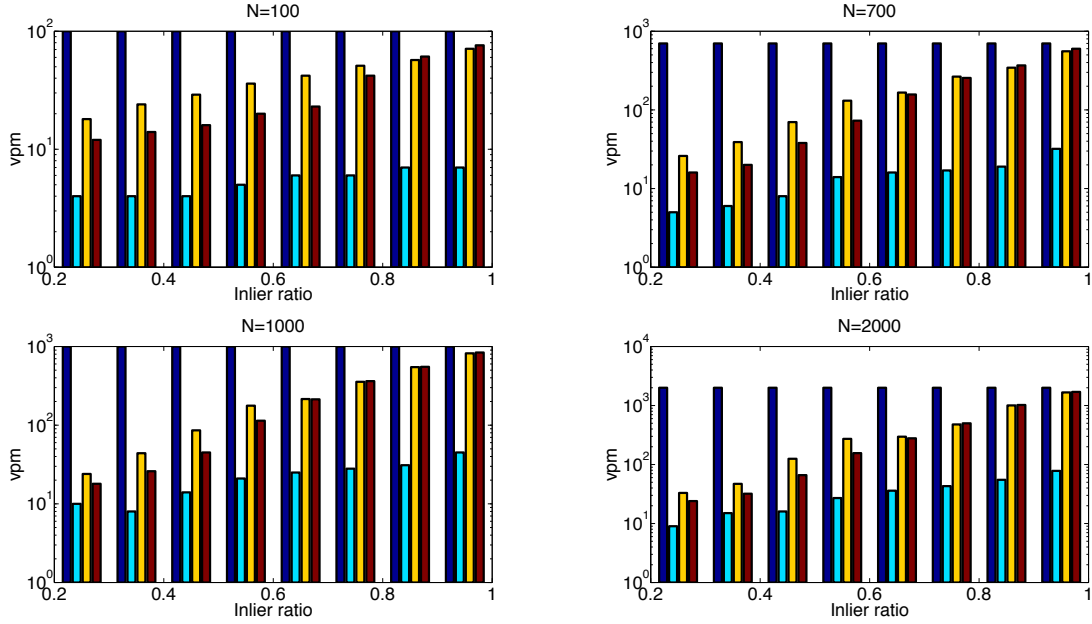


Figure 3.2: Results of synthetic data evaluation. Number of verifications per model (vpm) vs. inlier ratio for four different values of N . From left-right in each group: RANSAC, $T_{d,d}$, bail-out, SPRT. Note that the y-axis uses a log scale

Figure 3.2 shows that the $T_{d,d}$ test succeeds in greatly reducing the number of verifications per model. On the other hand, from Figure 3.3, it can be seen that the corresponding number of hypotheses required is much larger. It can be seen that the performance of the bail-out and SPRT tests are comparable over the range of parameter variations, with the SPRT producing greater speed-ups on average. However, for high inlier ratios, when the total number of models generated is small, the bail-out test can perform marginally better than SPRT, since the parameters in the SPRT are initialized conservatively.

Stage 3b: Degeneracy

In general, data is said to be degenerate if it does not provide sufficient constraints to compute a unique solution. Since RANSAC, in its standard form, does not have a safeguard against degeneracy, this can lead to incorrect solutions being returned by the algorithm. We now discuss some approaches that address this problem.

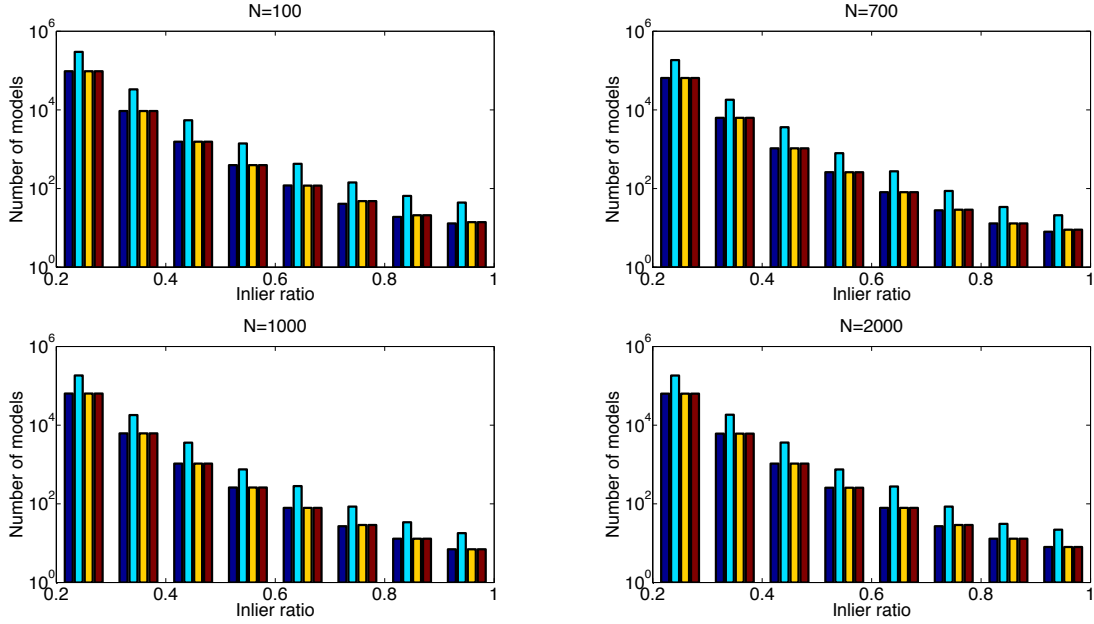


Figure 3.3: Results of synthetic data evaluation. Number of models evaluated vs. inlier ratio for four different values of N . From left-right in each group: RANSAC, $T_{d,d}$, bail-out, SPRT. Note that the y-axis uses a log scale

3.2.4.5 Model Selection

One approach that has been employed to deal with the issue of degeneracy is that of model selection. In this context, a degenerate solution can be thought of as a model with fewer parameters than the correct model. This was the approach taken in (Torr et al., 1995), where a RANSAC technique incorporating model selection, named PLUNDER, was used for epipolar geometry estimation. The algorithm estimates multiple models, accounting for the various cases of degenerate estimation; for instance, when the goal is to estimate a fundamental matrix, but the data only provides enough constraints to estimate a homography or affine fundamental matrix. By using a suitable cost function that penalizes models with more parameters, it becomes possible to find the simplest model that still provides a good fit to the data. One such cost function is the Geometric Information Criterion (GIC) (Akaike, 1974; Kanatani, 1996). For the problem of fitting a variety of dimension d to r dimensional points, the GIC can be represented as

$$\text{GIC} = -2L + 2(dn + p) \quad (3.12)$$

where L is the log-likelihood of the model, n is the number of data points and p is the number of parameters in the chosen model. It can be seen that the GIC equation has two terms: the first corresponds to the badness of fit and the second to a penalty on the complexity of the model. In the case of several competing models the objective is to find the model with the minimum value of GIC, which represents the model with highest information content but least complexity. More specific to the case of estimating geometric entities, (Torr, 1999) describes the Geometric Robust Information Criterion (GRIC), which generalizes the GIC as

$$\text{GRIC} = -2L + \lambda_1 dn + \lambda_2 p \quad (3.13)$$

where λ_1 and λ_2 are now chosen to reduce overfitting. For the estimation of two view geometry from feature matches, it is noted that $\lambda_1 = 2$ and $\lambda_2 = 4$ have provided good results over a wide range of conditions. Note that while the model selection approach is general, one of its main disadvantages is that it requires the separate estimation of multiple competing models. In practice, this can be a significant computational overhead.

3.2.4.6 DEGENSAC

A recent example that shows the utility of a test for degeneracy is that of DEGENSAC (Chum et al., 2005), where a detailed derivation is given for a nonlinear problem of epipolar geometry estimation. In this case, a notable example of degeneracy is that of points lying on a scene plane. In particular, when five of the seven image correspondences comprising the minimal sample for fundamental matrix estimation lie on the same plane in 3D, then the resulting model is consistent with all points that lie on the same plane. Consequently, in the case of images containing a dominant scene plane, an incorrect solution is often returned by the algorithm, since the support for this incorrect model may still be significantly high. To deal with this situation, a specific test was devised that aimed to identify samples where five or more correspondences in a minimal sample are related by a homography. The corresponding models are then subject to a model completion step, where the goal is to find, if possible, non-degenerate inliers that could be used to compute the correct model, given knowledge of the degeneracy.

3.2.4.7 QDEGSAC

A more general technique for dealing with degeneracy was proposed in (Frahm and Pollefeys, 2006), which describes a framework for RANSAC with (quasi-) degenerate data (QDEGSAC). The quasi-degenerate problem occurs when a majority of the data is degenerate, and thus does not provide enough constraints to compute the model uniquely, and only a few non-degenerate data points provide the remaining constraints. However, in the quasi-degenerate case, the probability of these few points being sampled in RANSAC is considerably low, and thus RANSAC will most often return the degenerate solution. QDEGSAC provides a solution to this problem for the case of estimating linear relations. Given a data matrix A , whose rows contain the constraints provided by the data points used to estimate the relation, the QDEGSAC framework can be viewed as a process that robustly measures the rank r_A of the data matrix, by using a sequence of calls to the RANSAC algorithm. The first execution, $\text{RANSAC}(p)$, estimates the most general model, which is appropriate for non-degenerate data. Following this, a series of RANSAC runs are performed successively on the inliers of the previous run, each run adding a linear constraint on the data. Thus, at the i -th stage, the best rank $p - i$ approximation to the data is computed in a robust sense (i.e., maximizing the number of rows in the data matrix that can be considered inliers). Finally, the most constraining model that successfully explains at least 50% of the original inliers to the first $\text{RANSAC}(p)$ is returned as the solution. For the case of general linear estimation problems, QDEGSAC thus provides a solution to the degeneracy issue, while requiring a modest increase in computational cost. Note that QDEGSAC is not a component of RANSAC per-se; rather, it can be viewed as a “meta-framework” that uses a number of RANSAC runs to effectively perform robust model selection. In the context of Figure 3.1, for instance, QDEGSAC can be thought of as a wrapper around the USAC framework.

3.2.5 Model refinement (Stage 4)

The use of minimal samples in RANSAC, coupled with the fact that data points are corrupted with noise, has two important consequences:

1. A model generated from an all-inlier sample may not be consistent with all inliers. In effect, this leads to an increase in the number of RANSAC iterations.

2. The final model returned by RANSAC may still be far from the global optimum, since noise is not being compensated for in any way.

In practice, 1) is usually ignored, since a correct solution is still returned by the algorithm, only at the expense of more computation. 2) is usually compensated for by a *post hoc* refinement procedure which uses the solution returned by RANSAC as a starting point. However, it is also possible to simultaneously account for both 1) and 2) within the framework of the algorithm itself.

3.2.5.1 Local Optimization

In (Chum et al., 2003), an algorithm for Locally Optimized RANSAC (LO-RANSAC) was proposed, which uses the observation that a model that is computed from an uncontaminated minimal sample invariably contains a large fraction of inliers within its support. Thus, an optimization process is inserted into the algorithm, using the current best solution as the starting point.

A number of optimization strategies can be adopted at this stage, trading off accuracy for efficiency. For instance, an *inner-RANSAC* procedure can be performed, wherein *non-minimal* samples are selected from the set of inliers, and then used to compute refined models. This process is carried out for a fixed number of iterations (typically 10 – 20), and the refined model with best support is stored. Alternatively, an iterative approach may be employed, by first selecting all data points with an error smaller than Kt with respect to the best model, where K is a predefined scale factor and t is the error threshold. A new model can then be estimated using all the selected points. The threshold scale factor is then reduced and this process iterated, until the threshold reaches a value of t . These strategies may be combined: each (non-minimal) sample in the inner-RANSAC can be further subject to an iterative refinement procedure. This combined procedure was found to reduce the number of iterations required in RANSAC by a factor of 2-3, bringing it to agreement with the theoretically expected number. It can be shown that the number of times the local optimization is invoked is approximately equal to the logarithm of the number of samples drawn (Chum et al., 2003). However, care must still be taken so that it does not significantly impact the overall runtime.

3.2.6 USAC Implementation

In Section 3.2, we introduced a Universal RANSAC framework to provide a common context within which to analyse various variants of RANSAC. However, it becomes immediately clear that this framework can also provide the basis for a high performance robust estimator that addresses many of the limitations of RANSAC within a single unified system. Thus, one of the goals of this work is to provide a stand-alone C++ implementation that implements the Universal RANSAC framework presented in the previous section. More specifically, while Section 3.2 discussed a number of alternatives for each stage of the framework, we now describe a specific implementation that uses an appropriate choice of algorithm for each module, keeping in mind concerns about robustness, performance and generality. These algorithmic choices are also illustrated in Figure 3.4. We call this implementation USAC-1.0 to emphasize that it is based on current state of the art techniques, and can be extended as new advances emerge.

3.2.6.1 Sample minimal subset

Given a (possibly prefiltered) set of data points, the goal of this step is to generate a sequence of minimal samples. From the discussion in Section 3.2.2, it is clear that the various sampling strategies proposed in the literature all come with trade-offs. In particular, all these sampling strategies invariably involve the preferential selection of samples that are in sensitive spatial configurations. Consequently, it is advisable to couple this the non-uniform sampling stage with modules that handle degeneracy detection and model refinement.

In our implementation, we choose the PROSAC (Section 3.2.2.3) sampling strategy for the sample generation stage. This choice was made to reach a compromise between performance, generality and susceptibility to degenerate configurations. PROSAC is more easily applicable in the general case than GroupSAC, and less susceptible to degenerate configurations than NAPSAC. PROSAC requires an additional input in the form of ordering information for the data points in terms of their quality scores, which is often available. For the two-view geometry experiments presented in the next section, we use SIFT ratio scores (Lowe, 2004) to order putative feature correspondences.

In addition, once a minimal sample has been selected, problem-specific checks can be enforced to eliminate undesirable minimal samples, as outlined in Section 3.2.2 (Stage 1b).

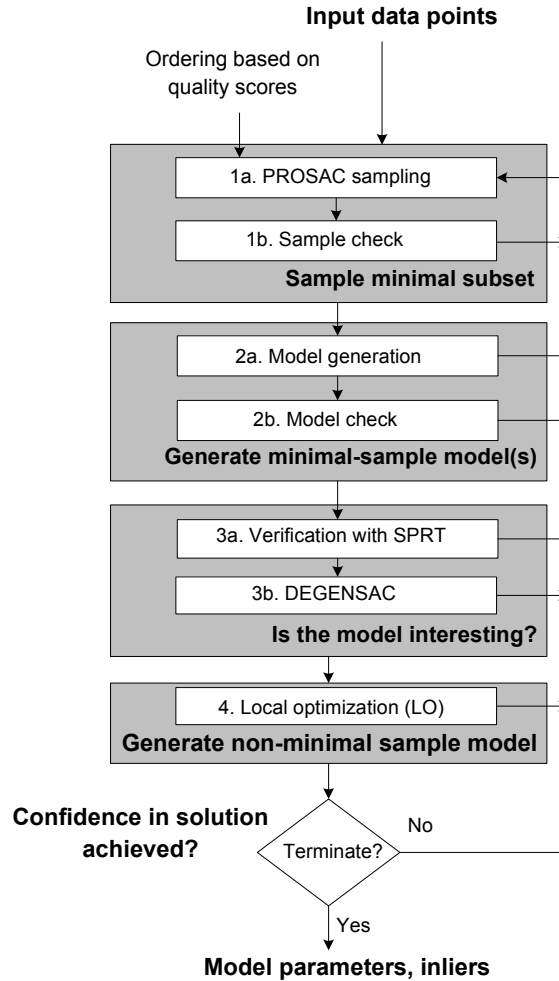


Figure 3.4: USAC-1.0 implementation. We present a high-performance implementation of the general framework of Figure 3.1, by use of state of the art algorithms for the individual modules. The combination of these techniques results in a robust estimator that address many of the limitations of standard RANSAC within a unified software package. In addition, by optionally invoking the individual modules in isolation (e.g., just PROSAC sampling), the implementation can be modified to behave like the corresponding variant of RANSAC.

3.2.6.2 Generate minimal sample model(s)

The process of model generation and checking are both dependent on the problem under consideration. While the complexity of model generation lies outside the scope of USAC, it should be noted that incorporating the model checking stage can significantly reduce the number of models that propagate to subsequent stages of the algorithm. For the two-view geometry estimation case, we use the oriented constraints check as outlined in Section 3.2.3.

3.2.6.3 Is the model interesting?

The goal of this step is to efficiently verify the model, and to check for degeneracy. We perform verification using SPRT, as outlined in Section 3.2.4.3, which is the optimal strategy for randomized model verification. This test requires two additional input parameters, ε and δ . As noted in Section 3.2.4.3, these parameters are initialized conservatively, and are updated as the algorithm progresses.

We choose to employ a specific degeneracy detection module that calls a user provided function to check for model degeneracy once it has been verified to be the current best model. As seen in the case of DEGENSAC (Section 3.2.4.4), this is with a view towards increasing robustness while maintaining computational efficiency. Note, however, that integrating the USAC algorithm with QDEGSAC is simple: the sequence of calls to RANSAC that occur within QDEGSAC need to be simply replaced by calls to USAC.

3.2.6.4 Generate non-minimal sample model

For refining the current best model, we use the Lo-RANSAC approach (Section 3.2.5.1), due to its generality. We have found that the inner-RANSAC approach, coupled with an iterative reweighted least squares procedure gives good results in practice. In our implementation, we make an additional optimization in the interests of efficiency: before performing the local optimization step, a check is performed to determine the extent to which the current inlier set overlaps with the best inlier set found so far. If this overlap is substantial (e.g., $\geq 95\%$), then it is unlikely that the local optimization will significantly improve upon the best result, and this step can be skipped. The effect of this enhancement is to prevent excess time from being spent in the computation of a locally refined solution. In addition, when the number of inliers being input to the local optimization step is large, this can lead to a computational bottleneck as well. In this event, performing the local optimization step on a subset of the inliers helps mitigate this effect and reduces the relative overhead, particularly for the case of high inlier ratios.

3.2.6.5 Confidence in solution achieved?

As discussed in Sec. 2.2.3, the standard RANSAC algorithm terminates when the probability of finding a set of inliers that is larger than the best support found so far falls under a predefined

threshold η_0 . For RANSAC, this probability can be computed as

$$\eta_R = (1 - 1/\varepsilon^m)^k, \quad (3.14)$$

where k is the number of samples drawn so far. This gives rise to the standard stopping criterion in equation (2.4). Two important modifications to this stopping criterion are discussed below.

Non-uniform sampling:

The use of non-uniform sampling results in good samples being drawn earlier on in the estimation process. However, while good samples are drawn earlier, if the standard stopping criterion from equation (3.14) is used with ε being computed over the whole dataset, then the total number of samples drawn will be the same as in standard RANSAC. Thus, a new method of computing the stopping criterion is required. Since we use PROSAC for generating minimal samples we consider two factors that determine when the algorithm should stop: non-randomness and maximality. The intuition is to look for a *stopping length* n^* that satisfies both these constraints.

The probability of having i random inliers *by chance* within the subset \mathcal{U}_n containing the n best data points is given by a binomial distribution

$$p_n(i) = \delta^{i-m}(1 - \delta)^{n-i} \binom{n-m}{i-m}, \quad (3.15)$$

where δ is the probability of a random point being consistent with a contaminated model. For each n , the minimum size of “non-random” support is calculated as

$$I_n^{min} = \min\{j : \sum_{i=j}^n p_n(i) < \psi\}, \quad (3.16)$$

where the inequality represents a simple p-value test to determine, for each n , the smallest support such that the probability of randomness falls below ψ (set to 0.05 for a 5% significance test). The stopping length n^* must have $I_{n^*} \geq I_n^{min}$.

The maximality constraint is again given by equation (3.14), and is used to ensure, with some confidence, that no better solution exists within the set \mathcal{U}_n . Thus $\eta = (1 - 1/\varepsilon_n^m)^{k_n}$, where ε_n is given

by I_n/n . The stopping length n^* is chosen to minimize the number of samples k_{n^*} , subject to the non-randomness constraint.

Randomized verification:

When a randomized verification strategy is adopted, there is a chance that good models may be erroneously rejected by the verification step. Thus, the stopping criterion must now account for this by drawing additional samples to achieve the same confidence. Since we use the SPRT test, the probability of finding a better model than the current best one is given by

$$\eta = (1 - (1 - \alpha)/\varepsilon^m)^k, \tag{3.17}$$

where α is the probability that a “good” model is rejected by the SPRT. In practice, α is not constant, being reestimated based on current estimates of the test parameters, ε and δ . Given an SPRT threshold A_i computed using the current estimates of ε_i and δ_i , the probability of rejecting a good model, with inlier fraction ε , is $\alpha_i = A_i^{-h_i}$, where h_i is computed from the relation

$$\varepsilon \left(\frac{\delta_i}{\varepsilon_i}\right)^{h_i} + (1 - \varepsilon) \left(\frac{1 - \delta_i}{1 - \varepsilon_i}\right)^{h_i} = 1. \tag{3.18}$$

This implies that whenever the value of the SPRT threshold A_i is updated, this results in a new α_i . Given the l^{th} SPRT test, the probability η from equation (3.17) is now given by the product over all individual tests, as

$$\eta = \prod_{i=0}^l (1 - (1 - A_i^{-h_i})/\varepsilon^m)^{k_i}. \tag{3.19}$$

3.2.7 Evaluation

In this section, we evaluate the performance of the USAC-1.0 implementation against current state of the art techniques, on a variety of estimation problems. Note that while robust estimators are perhaps most frequently applied in geometric vision problems, USAC is a general robust estimator, meaning that it can be used in virtually any scenario where models are being estimated from contaminated data measurements.

One interesting feature of the software library we provide is that it has a modular structure,

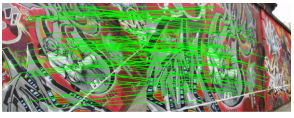
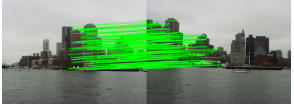
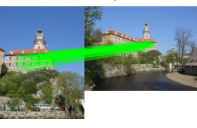
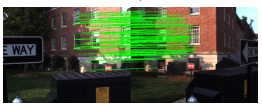
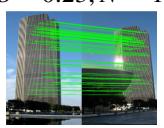
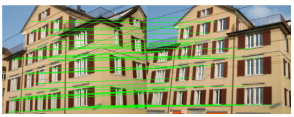
		RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
A : $\varepsilon = 0.46, N = 2540$ 	<i>I</i>	994±68	991±72	911±160	1137±27	1148±2
	error	1.71±0.21	1.62±0.41	2.14±0.93	1.17±0.12	1.04±0.00
	<i>k</i>	220	241	11	115	7/0
	models	220	241	11	115	7/0
	vpm	2540.0	80.3	2540.0	2540.0	426.7
	time	40.62	7.16	2.46	34.72	5.71
B : $\varepsilon = 0.57, N = 1760$ 	<i>I</i>	983±38	987±32	681±108	1003±3	1003±0
	error	1.80±0.44	1.54±0.44	5.69±1.81	0.88±0.14	0.88±0.02
	<i>k</i>	50	51	4	42	3/0
	models	50	51	4	42	3/0
	vpm	1760.0	106.3	1760.0	1760.0	640.6
	time	6.83	1.60	0.71	4.75	1.28
C : $\varepsilon = 0.15, N = 514$ 	<i>I</i>	70±4	71±4	56±11	74±0	74±3
	error	1.88±0.68	1.87±0.52	4.63±3.44	1.13±0.07	1.19±0.33
	<i>k</i>	16766	18800	11	11548	9/1
	models	16766	18800	11	11548	8/0
	vpm	514.0	25.4	514.0	514.0	110.4
	time	940.73	470.07	0.66	652.88	1.81
D : $\varepsilon = 0.43, N = 1967$ 	<i>I</i>	732±48	740±47	509±154	823±12	826±5
	error	2.81±0.66	2.97±0.66	6.01±3.88	1.77±0.10	1.76±0.03
	<i>k</i>	268	276	13	150	11/2
	models	268	276	13	150	10/0
	vpm	1967.0	41.6	1967.0	1967.0	494.7
	time	39.54	7.17	2.13	28.44	3.46
E : $\varepsilon = 0.23, N = 1317$ 	<i>I</i>	286±17	287±17	203±30	302±1	302±6
	error	1.63±0.44	1.63±0.54	8.49±9.12	0.89±0.06	0.89±0.13
	<i>k</i>	2433	2534	4	1717	4/0
	models	2433	2534	4	1717	4/0
	vpm	1317.0	18.4	1317.0	1317.0	374.1
	time	254.62	57.50	0.50	196.90	1.26
F : $\varepsilon = 0.24, N = 454$ 	<i>I</i>	96±6	96±6	92±11	104±0	104±0
	error	2.99±0.93	2.98±0.86	4.12±1.74	1.94±0.04	1.94±0.08
	<i>k</i>	2887	2994	769	1753	520/396
	models	2887	2994	769	1753	125/0
	vpm	454.0	18.3	454.0	454.0	26.5
	time	148.44	71.65	39.64	98.48	7.84

Table 3.1: Results for homography estimation. The table lists, for each algorithm, the number of inliers found (*I*), the error of the solution with respect to the “true” result (error), the number of samples (*k*) and models (models), the number of verifications per model (vpm), and the total runtime (time in milliseconds). For USAC-1.0, the table additionally lists, the number of samples and models that are rejected by the sample/model check tests. The results are averaged over 500 runs of each algorithm (Table contd. on next page)



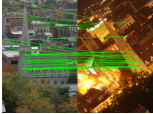
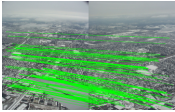
		RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
G : $\varepsilon = 0.10, N = 1114$ 	<i>I</i>	95±7	96±6	94±10	106±0	106±0
	error	2.43±0.83	2.45±0.60	2.78±1.07	1.35±0.03	1.35±0.02
	<i>k</i>	111956	130890	10727	59736	6158/5347
	models	111956	130890	10727	59736	811/0
	vpm	1114.0	33.1	1114.0	1114.0	37.8
	time	10447.26	3301.01	994.00	5637.83	26.34
H : $\varepsilon = 0.34, N = 495$ 	<i>I</i>	151±11	152±10	99±24	168±0	168±0
	error	2.22±0.45	2.23±0.51	6.81±3.12	1.44±0.00	1.43±0.00
	<i>k</i>	663	669	9	354	8/2
	models	663	669	9	354	5/0
	vpm	495.0	15.1	495.0	495.0	124.0
	time	36.00	16.05	0.53	27.71	6.13
I : $\varepsilon = 0.13, N = 979$ 	<i>I</i>	115±8	116±7	99±16	122±2	122±0
	error	2.67±2.19	2.63±2.01	11.57±18.51	1.15±0.77	1.04±0.09
	<i>k</i>	29622	32247	1059	20327	791/688
	models	29622	32247	1059	20327	103/0
	vpm	979.0	26.6	979.0	979.0	47.6
	time	2508.60	790.90	89.65	1732.38	8.23
J : $\varepsilon = 0.10, N = 994$ 	<i>I</i>	93±6	93±5	90±7	99±0	99±0
	error	3.43±1.42	3.13±1.01	11.43±4.10	2.52±0.36	2.59±0.27
	<i>k</i>	75950	84329	12059	49533	7266/6511
	models	75950	84329	12059	49533	755/0
	vpm	994.0	30.8	994.0	994.0	38.0
	time	6532.22	2134.44	1034.98	4266.51	25.74

Table 3.2: Results for homography estimation (contd.)

and can be easily configured to behave in a number of different ways. In particular, as noted in Section 3.2, many of the important RANSAC variants can be interpreted as special cases of this implementation. In its most reduced form, the implementation behaves exactly like the standard RANSAC algorithm outlined in Algorithm 1. By “switching on” the individual modules of Figure 3.4, it becomes possible to independently evaluate the effect that an specific module (e.g., PROSAC based non-uniform sampling) has on the estimation results. The choice of modules and parameters can be easily specified using a common configuration file. In other words, the implementation in Figure 3.4 not only represents USAC-1.0, but, it also in effect provides an implementation of a number of RANSAC variants, all within a single, easily configurable software package. For the experimental comparisons in this section, we thus configured our implementation to represent state of the art robust estimators. In particular, we compare the performance of USAC-1.0 against the

following:

1. RANSAC: the standard RANSAC algorithm, as laid out in Alg. 1. This denotes the baseline for comparison.
2. SPRT: RANSAC with optimal randomized verification, which represents the state of the art for efficient model verification.
3. PROSAC: RANSAC with non-uniform sampling based on ordering data points by quality scores, leading to efficient hypothesis generation.
4. LO: RANSAC with local optimization, resulting in more accurate solutions.

In all experiments, since the ground truth is unknown, the true inlier ratios (along with the ground truth inliers) were estimated by performing 10^7 evaluations of random models, followed by a refinement step. Following this, the inliers were manually inspected to ensure validity of the data points and to verify that no outliers were erroneously included. In all experiments, the results we report are average values obtained over 500 executions of each algorithm.

3.2.7.1 Homography

We evaluate the performance of USAC-1.0 for the problem of 2D homography estimation. The results are tabulated in Tables 3.1 and 3.2. The inlier ratios in the data range from 10%-57%, making this a challenging dataset for robust estimation algorithms. The table lists, for each algorithm, the number of inliers found, the number of hypotheses and models, the number of verifications per model, and the total runtime (in milliseconds). For USAC-1.0, the table also lists the number of hypotheses/models that are rejected by the sample/model check steps from Figure 3.4 (steps 1b and 2b). To determine the accuracy of the solution, we compute the Sampson error (Hartley and Zisserman, 2000) and report the root mean square (RMS) value over 500 executions of each algorithm.

It can be seen from the results that USAC-1.0 consistently delivers solutions that capture the most inliers, and which are the most accurate in terms of mean error. This is due to the local optimization step incorporated in the algorithm. In addition, the non-uniform sampling, optimized model verification and sample/model check modules all result in significant computational savings.

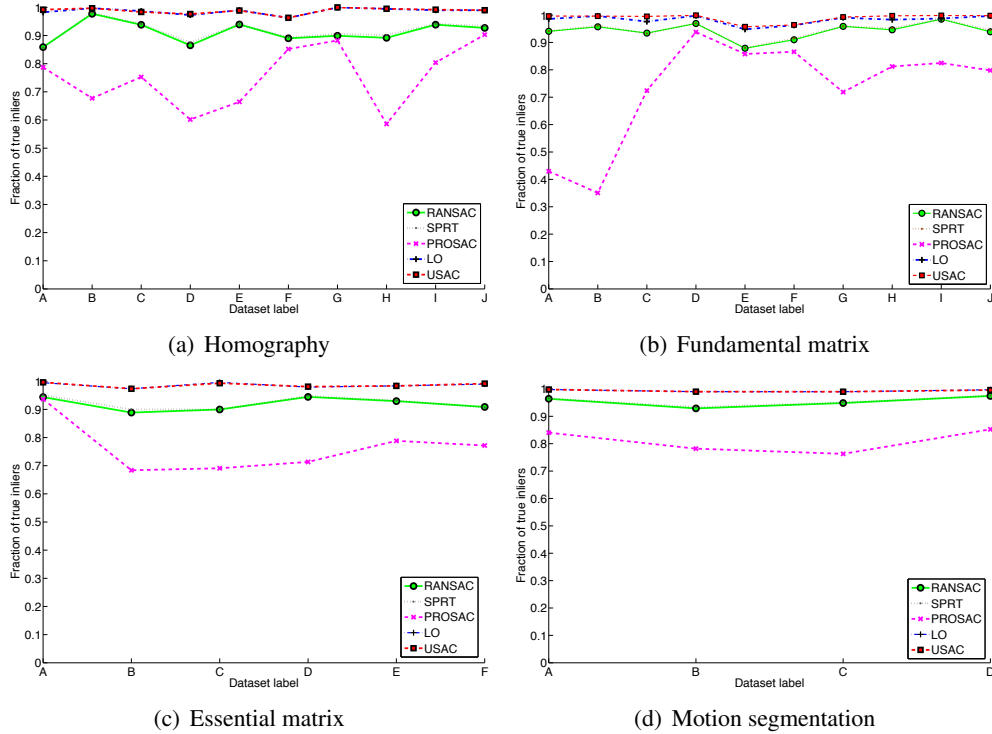


Figure 3.5: Fraction of true inliers returned by each algorithm for four estimation problems. The labels on the X-axis correspond to the datasets in Tables 3.1, 3.2, 3.3, 3.4, 3.5 and 3.6. The plots represent average values over 500 runs, for each algorithm and dataset. Note that for all four estimation problems, USAC-1.0 consistently returns close to 100% of the true inliers. The USAC-1.0 and LO curves are virtually identical, as are the RANSAC and SPRT curves. PROSAC, while fast, often returns solutions that have fewer inliers.

In particular, USAC-1.0 generates orders of magnitude fewer samples than standard techniques, and evaluates fewer points per model. The overall runtime is thus on the order of a few milliseconds, even for very low inlier ratios, representing effective speedups ranging between 5x-520x compared to RANSAC. It is worth noting that the combination of techniques in USAC results in consistently good performance, even when individual techniques are relatively ineffective – for e.g., in examples **G**, **I** and **J** in Table 3.2, where PROSAC-based sampling results in a significant runtime, while USAC-1.0 remains very efficient (10x-40x faster than PROSAC) due to the other optimizations present.

USAC-1.0 in general has much lower runtimes compared to the other techniques tested, save for PROSAC, which can sometimes be slightly faster than USAC-1.0. However, note that the solutions delivered by PROSAC can be unstable, due to the fact that the points selected are often poorly distributed in the image; we observed that the highest ranking points are typically spatially very

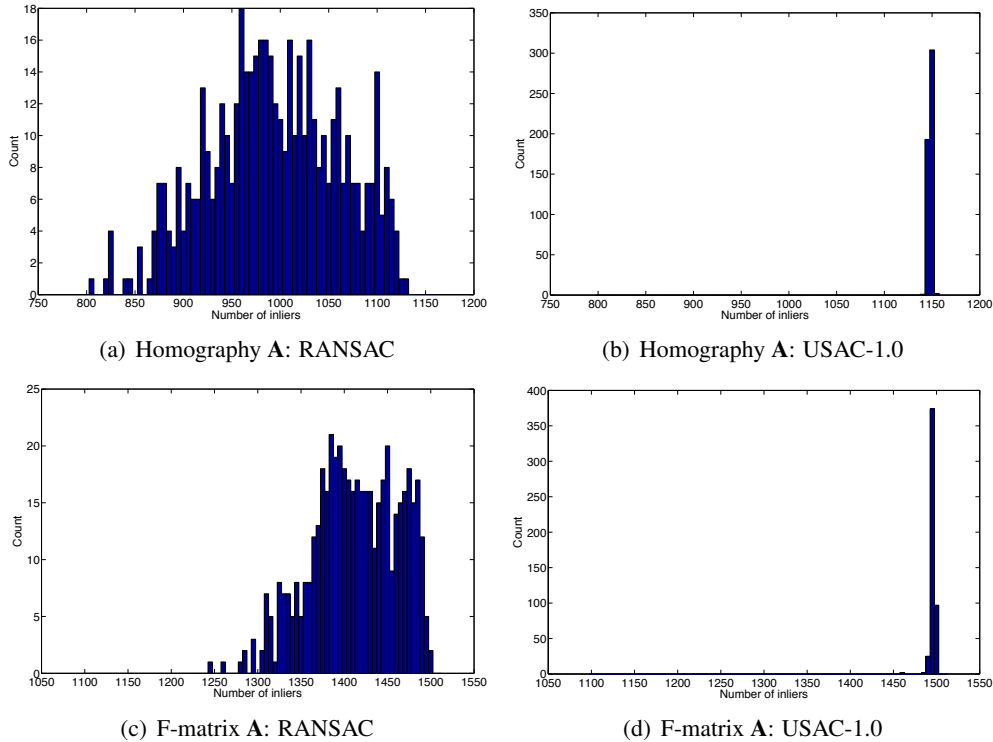


Figure 3.6: Histogram of inlier counts (y-axis) over 500 runs, for RANSAC vs. USAC-1.0. The graphs show results for homography estimation (a-b) for dataset A from Table 3.1 and fundamental matrix estimation (c-d) for dataset A from Table 3.3. Note that the histogram of inlier counts is more spread out for RANSAC, indicating that different inlier maxima are reached in different runs, owing to the randomized nature of the algorithm. USAC-1.0, on the other hand, has a much more peaked distribution, implying that the correct maxima is being reached on most executions.

close to each other. In USAC-1.0, this effect is mitigated by the local optimization step, which incurs a small additional computational cost but provides much more accurate and stable results. This effect is illustrated in Figure 3.5(a), which shows the fraction of true inliers that are returned by each algorithm. Note from the graphs that USAC-1.0 typically returns a significant fraction of the inliers, while the corresponding values for PROSAC are often much lower. This is due to the fact that the local optimization step uses non-minimal samples to improve the accuracy of the model parameter estimates, thus resulting in a larger consensus set. By reestimating model parameters using the set of all inliers, the local optimization is able to “break out” of the spatially localized fits provided by PROSAC-based sampling.

Another visualization of the stability of the results may be obtained by running each robust estimator multiple times on the same dataset, and calculating the fraction of runs in which a partic-

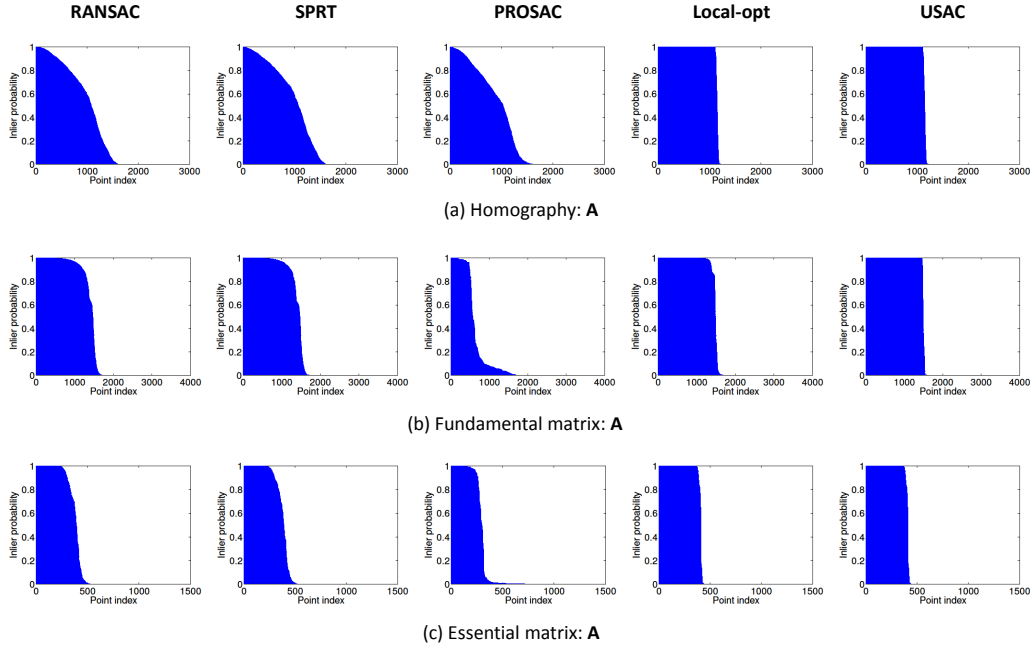


Figure 3.7: Visualization of the stability of the estimation results for each of the five algorithms evaluated. The results are shown for (a) homography estimation (dataset **A** from Table 3.1) (b) fundamental matrix estimation (dataset **A** from Table 3.3) (c) essential matrix estimation (dataset **A** from Table 3.5). In each graph, for every feature correspondence in the dataset (x-axis), we determine the fraction of runs, out of 500, where that correspondence is classified as an inlier (y-axis). The points on the x-axis are sorted in decreasing order of these probabilities. In the ideal case, each true inlier would have a score of 100%, and each true outlier would have a score of 0%. It can be seen that RANSAC and SPRT show comparable trends, while PROSAC is often significantly worse, missing true inliers on many of the runs. The results for LO and USAC-1.0 are very stable, and correspond closely to the ideal situation.

ular correspondence is classified as being an inlier. For homography estimation, this visualization is presented in Figure 3.7(a), for dataset **A** from Table 3.1. In each graph, for each input feature correspondence (x-axis), we plot the fraction of runs in which that correspondence is classified as an inlier (y-axis). This provides an empirical estimate of the probability that a true inlier is correctly classified by each robust estimation algorithm. The points on the x-axis are sorted in decreasing order of these probabilities. In the ideal case, each true inlier would have a score of 100%, and each true outlier would have a score of 0%. It can be seen that RANSAC, SPRT and PROSAC show comparable trends; each of these algorithms misses some inliers on each run, thus further underscoring the randomized nature of these methods. On the other hand, the results for LO and USAC-1.0 are very similar, and correspond closely to the ideal situation. This is an indication that

these methods successfully classify a majority of the inliers on every run of the algorithm and are thus very stable.

Finally, note that the standard deviation of the number of inliers returned by USAC-1.0 is significantly lower than that of RANSAC. Figures 3.6(a) and 3.6(b) compare the histograms of retrieved inlier counts over 500 runs of RANSAC and USAC-1.0, for image pair **A** from Table 3.1. It can be seen that the number of inliers returned by RANSAC can vary quite widely, while the histogram for USAC-1.0 is much more “peaked”, implying that a much more stable solution is returned in the vicinity of the global maximum. This again, is due to the application of local optimization, which by using non-minimal samples to compensate for inlier noise, ensures that the model parameters are more stable. This also indicates that while computationally very expensive techniques such as (Li, 2009) can be used to maximize the consensus set, USAC-1.0 achieves approximately similar results, at a small fraction of the computational cost. USAC-1.0 is thus able to deliver more accurate and stable solutions than current techniques, while doing so extremely efficiently.

3.2.7.2 Fundamental matrix

We evaluate the performance of USAC-1.0 for the problem of fundamental matrix estimation using the 7-point algorithm. These results are tabulated in Tables 3.3 and 3.4. The inlier ratios in the data range from 22%-96%, and examples were chosen to cover a wide range of challenging cases: narrow and wide baselines, scale change, repetitive textures, and dominant scene planes (a degenerate configuration for the fundamental matrix). The table lists the same statistics as for homography estimation.

As before, it can be seen from the results that USAC-1.0 consistently delivers stable and accurate solutions with low overall runtimes, with up to 4x-7000x speedups compared to RANSAC. The case of fundamental matrix estimation is particularly challenging since care must be taken to avoid degenerate solutions. PROSAC, in particular, can be prone to degeneracies, since even when there is no dominant plane in the scene, the top ranked points often lie on the same surface. In urban scenes, this will often lead to a degenerate solution (or one that is poorly conditioned). Note, for instance, examples **A** and **B** in Table 3.3, where PROSAC returns solutions very quickly, but with high errors. The DEGENSAC module in USAC-1.0 (step 3b in Figure 3.4) is able to effectively detect and handle


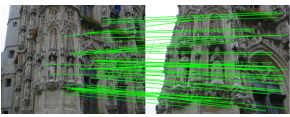
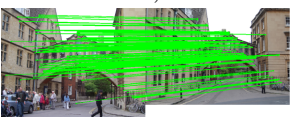



		RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
A : $\varepsilon = 0.48, N = 3154$ 	<i>I</i>	1412±50	1413±48	645±120	1480±33	1495±8
	error	1.28±0.58	1.26±0.55	21.31±11.14	0.85±0.50	0.63±0.24
	<i>k</i>	1420	1455	4	957	2/0
	models	3499	3587	10	2359	4/0
	vpm	3154.0	20.1	3154.0	3154.0	940.9
	time	255.90	18.19	2.01	191.16	11.84
B : $\varepsilon = 0.57, N = 575$ 	<i>I</i>	315±12	316±11	115±13	328±1	328±3
	error	0.71±0.18	0.69±0.17	60.99±13.05	0.45±0.04	0.45±0.06
	<i>k</i>	385	371	7	240	3/0
	models	941	906	17	588	8/1
	vpm	575.0	11.5	575.0	575.0	423.8
	time	14.98	4.35	1.35	11.22	3.61
C : $\varepsilon = 0.38, N = 1088$ 	<i>I</i>	381±13	381±12	295±16	398±11	406±4
	error	1.79±1.27	1.81±1.22	30.49±2.68	1.07±0.83	0.58±0.28
	<i>k</i>	7935	8403	4	5762	2/0
	models	19578	20735	9	14221	3/0
	vpm	1088.0	22.6	1088.0	1088.0	472.2
	time	546.53	108.77	5.41	408.26	13.74
D : $\varepsilon = 0.22, N = 1516$ 	<i>I</i>	324±10	324±9	313±19	333±3	334±3
	error	0.93±0.47	0.94±0.46	1.30±0.63	0.54±0.27	0.49±0.22
	<i>k</i>	267465	280961	6	198554	4/0
	models	661141	694471	16	490798	12/4
	vpm	1516.0	17.3	1516.0	1516.0	268.6
	time	23892.92	3434.69	1.77	18008.08	3.32
E : $\varepsilon = 0.38, N = 422$ 	<i>I</i>	141±6	141±5	137±5	152±6	153±5
	error	2.96±2.46	2.75±2.35	2.06±1.37	1.89±2.10	1.34±1.25
	<i>k</i>	12218	12776	798	7560	161/0
	models	30628	32028	1984	18948	402/318
	vpm	422.0	15.0	422.0	422.0	8.2
	time	391.62	160.47	25.58	247.36	16.79
F : $\varepsilon = 0.38, N = 346$ 	<i>I</i>	118±6	119±6	113±9	125±2	125±1
	error	2.15±1.47	2.07±1.44	3.18±2.30	1.58±0.65	1.58±0.59
	<i>k</i>	11427	11580	158	6490	64/0
	models	28260	28632	386	16055	156/121
	vpm	346.0	13.4	346.0	346.0	8.3
	time	319.29	147.61	4.61	195.69	4.87

Table 3.3: Results for fundamental matrix estimation. The table lists, for each algorithm, the number of inliers found (*I*), the error of the solution with respect to the “true” result (error), the number of samples (*k*) and models (models), the number of verifications per model (vpm), and the total runtime (time in milliseconds). For USAC-1.0, the table also lists the number of samples and models that are rejected by the sample/model check tests. Results are averaged over 500 runs of each algorithm. (Table contd. on next page)



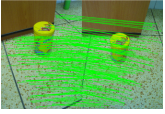

		RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
G : $\varepsilon = 0.45, N = 904$ 	I	388±10	388±11	291±53	401±5	402±7
	error	1.92±1.14	1.86±1.13	8.74±5.02	1.04±0.74	0.89±0.77
	k	1885	1956	23	1413	19/0
	models	4659	4835	56	3491	46/15
	vpm	904.0	15.7	904.0	904.0	224.5
	time	111.37	24.55	2.57	94.32	5.52
H : $\varepsilon = 0.92, N = 786$ 	I	685±37	690±36	588±0	712±31	722±0
	error	2.77±6.43	2.24±5.78	23.65±0.00	1.85±5.57	0.29±0.00
	k	14	15	1	11	1/0
	models	34	36	3	25	3/0
	vpm	786.0	299.8	786.0	786.0	675.7
	time	0.85	0.59	0.12	9.31	15.61
I : $\varepsilon = 0.96, N = 2776$ 	I	2621±6	2622±7	2195±0	2628±9	2657±1
	error	6.15±3.41	6.51±3.53	7.92±0.00	4.81±3.63	0.47±0.06
	k	5	5	1	5	1/0
	models	12	12	1	11	1/0
	vpm	2776.0	1845.4	2776.0	2776.0	2770.5
	time	0.91	1.09	0.22	5.76	15.05
J : $\varepsilon = 0.93, N = 2456$ 	I	2142±80	2157±71	1821±0	2264±11	2277±4
	error	4.26±2.64	3.98±2.35	21.84±0.00	2.52±0.61	0.32±0.30
	k	12	14	1	6	1/0
	models	27	32	3	14	3/0
	vpm	2456.0	647.8	2456.0	2456.0	2454.4
	time	2.07	1.09	0.36	11.57	12.46

Table 3.4: Results for fundamental matrix estimation (contd.)

these cases. When the scene contains a dominant plane (examples **H** – **J**), all techniques that do not account for degeneracy return incorrect solutions. This is illustrated in Figure 3.8, which depicts sample epipolar geometries for degenerate and non-degenerate solutions provided by RANSAC and USAC-1.0, respectively. Note that for degenerate solutions, the epipolar lines are consistent with points on the dominant plane, but not with off-plane inliers. Figure 3.5(b) shows the fraction of true inliers that are returned by each algorithm. Note that USAC-1.0, due to the local optimization, typically returns all true inliers. In addition, the histograms of inlier counts (Figures 3.6(c) and 3.6(d)) indicate that, as for the homography, USAC-1.0 returns very stable solutions due to the local optimization.

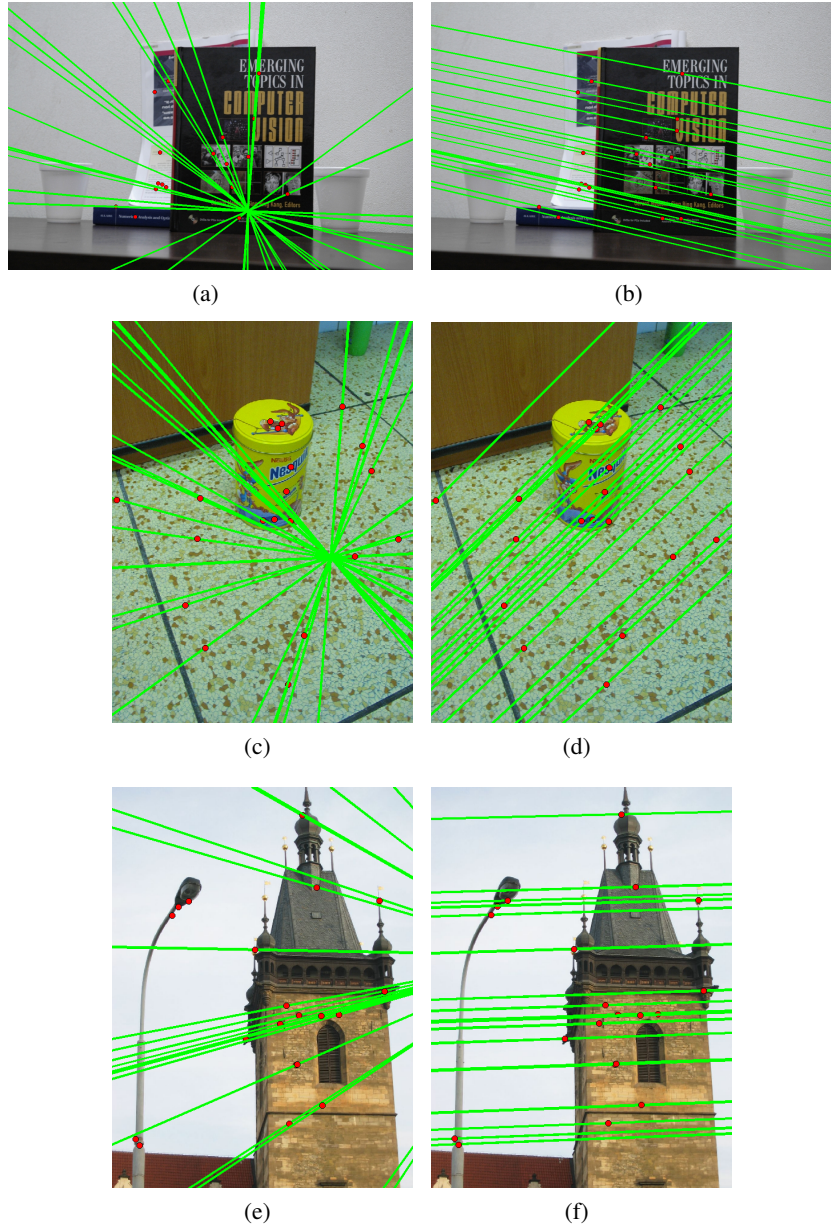


Figure 3.8: Example degenerate (left) and non-degenerate (right) epipolar geometries for three scenes containing dominant scene planes. Figures (a), (c) and (e) are results produced by RANSAC, while (b), (d) and (f) are computed with USAC-1.0. The figures show some selected epipolar lines corresponding to inlier correspondences. Note that for the degenerate cases, the epipolar geometry is consistent with points on the dominant plane, but not with the (fewer) off-plane inliers.

3.2.7.3 Essential matrix

We evaluate the performance of USAC-1.0 for the problem of essential matrix estimation using the 5-point algorithm (Nistér, 2004). These results are tabulated in Table 3.5, with inlier ratios in the


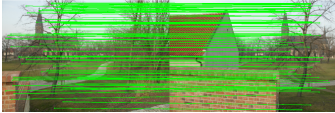
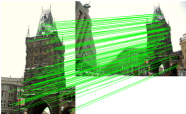
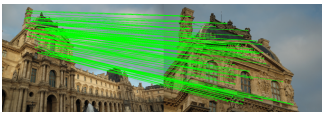
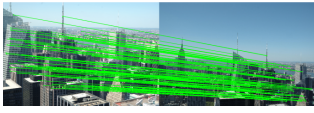

		RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
A : $\varepsilon = 0.85, N = 2471$ 	<i>I</i>	1975±22	1990±21	1960±20	2082±5	2085±5
	error	1.49±1.01	1.44±1.07	1.91±1.83	0.83±0.41	0.81±0.18
	<i>k</i>	14	16	1	10	1/0
	models	58	66	6	40	6/1
	vpm	2471.0	199.9	2471.0	2471.0	410.2
	time	9.12	3.03	1.05	8.34	1.18
B : $\varepsilon = 0.77, N = 582$ 	<i>I</i>	399±14	404±14	307±19	437±9	437±8
	error	2.18±1.53	2.14±1.44	2.46±2.11	1.09±0.43	1.07±0.43
	<i>k</i>	32	31	7	25	7/0
	models	140	137	30	88	31/13
	vpm	582.0	37.9	582.0	582.0	38.0
	time	8.42	4.65	1.93	7.91	2.12
C : $\varepsilon = 0.65, N = 1110$ 	<i>I</i>	646±18	646±17	496±14	715±3	713±4
	error	2.94±2.20	2.99±2.25	3.99±3.06	1.01±0.25	1.06±0.31
	<i>k</i>	73	77	12	48	12/0
	models	281	299	45	214	44/11
	vpm	1110.0	37.2	1110.0	1110.0	94.5
	time	24.41	10.60	4.10	17.02	5.21
D : $\varepsilon = 0.60, N = 962$ 	<i>I</i>	544±12	546±14	411±12	564±7	565±6
	error	4.56±2.03	4.51±2.25	8.51±4.40	1.57±0.80	1.54±0.76
	<i>k</i>	80	82	6	65	6/0
	models	354	366	20	290	21/2
	vpm	962.0	60.1	962.0	962.0	148.1
	time	27.56	12.58	1.79	17.48	2.03
E : $\varepsilon = 0.35, N = 1207$ 	<i>I</i>	395±16	396±15	335±32	418±8	418±8
	error	2.21±1.48	2.47±1.74	9.27±9.55	0.99±0.59	0.98±0.54
	<i>k</i>	1321	1363	17	887	19/0
	models	5948	6139	73	3989	83/25
	vpm	1207.0	18.3	1207.0	1207.0	72.2
	time	522.77	193.18	6.71	348.49	6.12
F : $\varepsilon = 0.26, N = 2273$ 	<i>I</i>	537±23	536±22	456±28	585±5	586±5
	error	1.82±0.57	1.76±0.62	5.39±2.03	1.10±0.25	1.08±0.28
	<i>k</i>	6826	7830	41	4117	35/0
	models	29281	33580	198	17631	155/64
	vpm	2273.0	20.7	2273.0	2273.0	25.6
	time	4100.09	1101.12	27.41	3746.64	17.81

Table 3.5: Results for essential matrix estimation. The table lists, for each algorithm, the number of inliers found (*I*), the error of the solution with respect to the “true” result (error), the number of samples (*k*) and models (models), the number of verifications per model (vpm), and the total runtime (time in milliseconds). For USAC-1.0, the table additionally lists the number of samples and models that are rejected by the sample/model check tests. The results are averaged over 500 runs of each algorithm.

data ranging from 26%-86%. We tabulate the same statistics as for the fundamental matrix, with the error again being given by the Sampson distance. The dominant plane degeneracy is not an issue for the 5-point method. However, the time to compute a minimal solution is more than that for the 7-point. In this context, note that even for very low inlier ratios (such as in example **F** in Table 3.5), USAC-1.0 is still able to deliver correct solutions with low runtimes – in fact, well within real-time, which is useful for real-time 3D reconstruction systems.

3.2.7.4 Motion segmentation

Finally, as an example of multi-model fitting, we evaluate the performance of USAC-1.0 for the problem of segmenting multiple motions. While USAC, like RANSAC, is not specifically designed to handle this case, it can still provide good results, particularly when the number of motions is known beforehand. When multiple models are to be estimated, a common strategy is to use a sequence of calls to the algorithm:

1. Apply the algorithm to the original dataset in order to recover the most dominant model, along with its set of inliers. All other points (including inliers to the other models) are considered to be outliers to this first model.
2. Remove all inliers for the current model, and repeat step 1 until all models have been recovered.

This form of iterative robust estimation has been investigated for many applications, including motion segmentation (Tron and Vidal, 2007). For the experiments in this section, we consider a pair of images and compute tentative feature correspondences. Then, we estimate a sequence of epipolar geometries using the 7-point algorithm as before. These results are tabulated in Table 3.6. For each dataset, the table lists summary statistics for each model present. The number of models ranges from 2-3, and is assumed to be known beforehand. Note that the effective inlier ratio for each individual model can be significantly low, since inliers to one model are outliers to the others. Note that even for these low inlier ratio problems, USAC-1.0 is still able to return accurate solutions in a computationally efficient manner.

			RANSAC	SPRT	PROSAC	Local-opt	USAC-1.0
A : $\varepsilon_1 = 0.72, N_1 = 1643$ $\varepsilon_2 = 0.33, N_2 = 465$ 	model						
	M1	<i>I</i>	1152±21	1153±20	1015±10	1176±5	1176±4
		error	1.02±0.51	1.06±0.49	8.51±5.14	0.35±0.09	0.33±0.04
		<i>k</i>	81	85	3	53	3/0
		models	170	175	7	107	7/0
		vpm	1643.0	19.1	1643.0	1643.0	640.2
	time	15.90	5.19	1.51	9.16	4.84	
	M2	<i>I</i>	132±15	133±14	104±21	153±1	153±1
		error	1.23±0.64	1.26±0.63	4.21±2.04	0.25±0.08	0.22±0.07
		<i>k</i>	39335	43133	79	13404	66/0
models		82604	88423	175	29491	141/34	
vpm		491.0	14.1	628.0	467.0	20.5	
time	1082.30	198.19	13.01	442.12	6.70		
B : $\varepsilon_1 = 0.43, N_1 = 619$ $\varepsilon_2 = 0.36, N_2 = 353$ 	model						
	M1	<i>I</i>	245±18	246±17	215±13	263±3	263±4
		error	1.12±0.78	1.16±0.75	3.21±1.14	0.83±0.13	0.84±0.14
		<i>k</i>	3233	3693	7	2014	4/0
		models	6942	7034	15	4269	8/0
		vpm	619.0	18.4	619.0	619.0	240.9
	time	115.31	42.52	1.31	81.67	3.34	
	M2	<i>I</i>	121±10	122±11	93±8	127±4	127±4
		error	1.07±0.41	1.11±0.53	5.34±2.39	0.76±0.12	0.76±0.14
		<i>k</i>	14495	19645	18	6865	16/0
models		33341	43808	41	14794	36/14	
vpm		369.0	20.1	404.0	356.0	57.2	
time	365.13	78.91	4.19	162.15	9.31		
C : $\varepsilon_1 = 0.31, N_1 = 858$ $\varepsilon_2 = 0.37, N_2 = 592$ 	model						
	M1	<i>I</i>	259±5	259±3	215±13	263±1	263±1
		error	0.84±0.22	0.82±0.21	2.45±1.17	0.34±0.10	0.33±0.11
		<i>k</i>	18834	21671	16	12789	12/0
		models	41540	47676	41	28136	28/6
		vpm	858.0	18.8	858.0	858.0	332.6
	time	1006.90	188.98	3.01	714.60	9.84	
	M2	<i>I</i>	201±12	202±11	155±16	217±3	217±2
		error	1.78±0.62	1.76±0.65	12.32±5.81	0.57±0.11	0.53±0.14
		<i>k</i>	7377	7831	217	3657	189/0
models		15492	16545	477	8045	440/351	
vpm		599.0	14.3	643.0	595.0	39.9	
time	264.90	78.32	7.01	151.16	9.14		
D : $\varepsilon_1 = 0.31, N_1 = 1277$ $\varepsilon_2 = 0.36, N_2 = 879$ $\varepsilon_3 = 0.39, N_3 = 559$ 	model						
	M1	<i>I</i>	393±5	393±4	344±11	397±0	397±0
		error	1.43±0.52	1.46±0.55	2.27±0.64	1.05±0.01	1.06±0.01
		<i>k</i>	12877	13907	14	10921	8/0
		models	28137	31013	29	24026	20/2
		vpm	1277.0	27.5	1277.0	1277.0	307.3
	time	886.91	184.24	3.55	567.42	8.38	
	M2	<i>I</i>	314±5	314±3	287±4	320±0	319±1
		error	1.08±0.33	1.12±0.31	2.44±1.08	0.21±0.01	0.23±0.03
		<i>k</i>	5289	5817	281	3921	204/0
		models	11435	13205	646	8626	500/312
		vpm	884.0	16.6	933.0	880.0	120.0
	time	319.63	48.17	33.39	191.31	17.37	
	M3	<i>I</i>	203±9	204±8	165±2	214±1	214±1
		error	2.53±0.66	2.51±0.65	6.40±3.82	0.67±0.13	0.63±0.14
<i>k</i>		5042	5445	572	2818	369/0	
models		10878	12415	1186	6481	913/692	
vpm		570.0	24.9	646.0	560.0	28.5	
time	216.21	47.08	117.36	137.76	29.72		

Table 3.6: Results for motion segmentation. The images show inliers to each individual motion, denoted with different colours. The table lists the same summary statistics as before, reported for each individual motion. The results are averaged over 500 runs.

3.2.8 Summary

In this section, we have presented a comprehensive overview of the state of the art in RANSAC-based robust estimation. To provide a unified context for this analysis, we have proposed a Universal RANSAC framework, which is a synthesis of the various optimizations and improvements that have been proposed to RANSAC over the years. This framework provides a common context within which to analyse current state of the art robust estimation algorithms, and to investigate the interplay between these varied algorithmic components. As a second contribution, we have developed a stand-alone C++ library that implements the Universal RANSAC framework using state of the art algorithms for each stage (USAC-1.0). The implementation is modular, and the effect of individual optimizations can be studied in isolation, as well as in combination with each other. We have evaluated its effectiveness on a challenging collection of estimation problems, thereby demonstrating the significant advantages of unifying the various RANSAC techniques.

3.3 Adaptive Real-Time Random Sample Consensus (ARRSAC)

Section 3.2 discussed a number of RANSAC techniques that aim to improve upon the efficiency and robustness of the standard RANSAC algorithm. While these efforts have shown considerable promise, it is also the case that none of them are directly applicable in situations where real-time performance is essential. This is primarily due to the fact that in order to achieve an η_0 confidence in the solution, all these methods draw an arbitrary number of samples, which could potentially be quite large for low inlier ratios. In the case of real-time systems the goal is to achieve the threshold when possible; however, when this is not the case, it is desirable to obtain the best solution given the time constraints. More specifically, the fixed time constraint implies that the maximum number of hypotheses generated must be bounded by some (predefined) number. The main challenge is thus to enable robust estimation even under this hard constraint.

Relatively fewer efforts have been directed towards the goal of formulating RANSAC in a manner that is suitable for this scenario. In Section 3.2.4.4, we discussed one such effort – preemptive RANSAC – where a *fixed* number of hypotheses are evaluated in a *parallel, multi-stage* setting.

In this case the goal is to find, within a fixed time budget, the best solution from a restricted set of hypotheses. As opposed to depth-first preemptive schemes (Sections 3.2.4.1–3.2.4.3), where a particular hypothesis is completely evaluated before moving on to the next hypothesis, preemptive RANSAC uses a *breadth-first* approach, where all the hypotheses generated are evaluated on a subset of the data points. Subsequently, the hypotheses are reordered based on the results of this scoring procedure, and only a fraction of the hypotheses are evaluated on the next subset of data. A non-increasing preemption function defines the number of hypotheses retained after each stage of preemption. The preemption function used in (Nistér, 2003) is of the form

$$f(i) = \lfloor M2^{(-\frac{i}{B})} \rfloor, \quad (3.20)$$

where M is the number of models in the initial set, and B defines the block size, which is the number of data points that a model is evaluated against, before the reordering and preemption step takes place. In effect, this preemption function halves the number of surviving models after each block of data points has been evaluated. This process continues until only one hypothesis remains, or all data points have been used.

While preemptive RANSAC facilitates real-time implementation, there exist a few limitations in the scheme. One of the primary limitations of preemptive RANSAC is its inherent non-adaptiveness to the data. The selection of a fixed number of hypotheses implicitly implies that a good prior estimate of the inlier ratio is available; in practice, this is often not the case. For low contamination problems, preemptive RANSAC is often slower than standard RANSAC, since it evaluates many more hypotheses than necessary. On the other hand, when the inlier ratio is too low, preemptive RANSAC is unlikely to find a good solution, since it does not test enough hypotheses.

In this section we propose a new framework for real-time robust estimation, termed Adaptive Real-Time Random Sample Consensus (ARRSAC), that aims to overcome these limitations of preemptive RANSAC. One of the main challenges in real-time robust estimation lies in being able to adapt to the strict limitations presented by the fixed time budget; namely, the generation of a pre-defined number of hypotheses. The framework we develop is able to cope with variations in the contamination level of the data, while at the same time providing accurate estimation at real-time speeds. As in Section 3.2, our approach builds upon the strengths of state of the art RANSAC

methods, and combines them in a way that is suitable for real-time robust estimation. In this sense, ARRSAC may be viewed as a variation of USAC, with modifications made to enable real-time operation.

3.3.1 The approach

In this section, we describe the main ideas we adopt in order to design a real-time robust estimation framework. From the discussion on preemptive RANSAC (Section 3.2.4.4), it can be seen that the number of hypotheses M in the initial candidate set is fixed *a priori*. In a real-time setting, this ensures that the runtime of the algorithm is bounded. However, fixing this number beforehand effectively places a restriction on the inlier ratio. When the true inlier ratio is high, preemptive RANSAC evaluates far too many samples. On the other hand, when the true inlier ratio is low, the likelihood that preemptive RANSAC finds a correct solution decreases drastically.

The breadth-first scan advocated by preemptive RANSAC is indeed a natural formulation for real-time applications, since the primary constraint is the fixed time-budget. On the other hand, adapting the number of hypotheses to the contamination level of the data requires an estimate of the inlier ratio, which necessitates the adoption of a depth-first scan. The ARRSAC framework retains the benefits of both approaches (i.e, bounded run-time as well as adaptivity) by operating in a *partially depth-first* manner. Hypotheses for the initial candidate set are generated one-by-one, and are evaluated on the first data block. Bad hypotheses are discarded based on a depth-first preemptive strategy. Hypotheses that pass this verification procedure provide an estimate of the inlier ratio, which is used to determine the required number of hypotheses, keeping the upper limit fixed at M . Note that since the evaluation is performed on only a subset of the data (thus, partially depth-first), the estimated inlier ratio may be either above or below the true inlier ratio. In view of this fact, the ARRSAC algorithm allows the generation of additional hypotheses at a later stage in the evaluation process.

While the above discussion describes a method for adaptively determining the number of hypotheses, note that there is still an upper limit on the size of the hypothesis set. In such a scenario, it is then important to choose a strong set of candidate hypotheses, since this facilitates good performance at low inlier ratios. ARRSAC accomplishes this by adopting a non-uniform sampling

approach, generating hypotheses from the highest quality matches. Note, however, that even in the absence of information to guide non-uniform sampling (or when this does not help), ARRSAC still has a bounded runtime. This is in contrast with strategies such as PROSAC, which in the worse case operates like RANSAC (and hence has a potentially large runtime).

From the point of view of model accuracy, once a hypothesis passes the verification test, it is also possible to generate new hypotheses by sampling from data points that are consistent with this good hypothesis – in other words, to perform local optimization (see Section 3.2.5). In the interest of computational efficiency, we adopt a simpler local optimization scheme than in USAC, simply generating a fixed number of non-minimal model hypotheses (the “inner RANSAC” approach from (Chum et al., 2003)) from the set of inliers to the current best model. An important advantage of the competitive evaluation framework is that ARRSAC does not spend excessive time in local optimization, since the hypotheses generated in the inner RANSAC are forced to compete among themselves. This prevents the algorithm from spending excessive time in the refinement, which is advantageous when the inlier ratio is high.

Finally, while the partially depth-first evaluation strategy described above serves to adaptively determine the number of hypotheses, it also provides an additional computational advantage. In the original preemptive RANSAC algorithm, hypotheses are evaluated on all B data points in a block before the reordering and preemption step takes place. As we have noted earlier, this constitutes an unnecessary overhead since most of the generated hypotheses are likely to be contaminated. Thus, in ARRSAC, bad hypotheses are discarded after partial evaluation and only the hypotheses that survive are selected for propagation to subsequent stages of evaluation. The preemption function in Equation 3.20 thus effectively defines an upper bound on the number of hypotheses required for the next stage of evaluation. The number of good hypotheses that need to be reordered and propagated in ARRSAC is typically much less than this bound.

To summarize the main contributions of the ARRSAC algorithm: the adoption of a partially depth-first strategy for hypothesis generation provides a larger degree of control over the initial set. Adaptively determining the number of hypotheses ensures computational efficiency for higher inlier ratios. Furthermore, this strategy also allows more control over the quality of hypotheses that are included in the initial set, thereby increasing the likelihood of obtaining a good solution even for low

Algorithm 2 The ARRSAC algorithm

Initialization

Set values for M (max. number of candidate hypotheses) and B (block size)

1. Generate initial hypothesis set (Algorithm 3)

Set n = total number of hypotheses generated in the initial stage

2. Preemptive evaluation

for $i = B + 1$ to N **do**

for all valid models in the current set **do**

 Score model hypotheses using data point i

 Evaluate SPRT test (Section 3.2.4.3)

if model rejected **then**

$n = n - 1$

 Mark current model as invalid and continue to next

else

 Update score for current model

end if

end for

if $(i \bmod B) = 0$ **then**

$n = \min(f(i), n/2)$

 Reorder based on hypothesis scores and select hypotheses $h(1), \dots, h(n)$

if $n = 1$ **then**

 Break with the remaining hypothesis as the top one

end if

end if

end for

Return best model in $h(1), \dots, h(n)$

inlier ratios. The use of optimized verification techniques is geared towards improving the overall efficiency of the algorithm. Finally, it is also possible to integrate a simple local optimization scheme into the algorithm, thereby improving the accuracy of the estimated model parameters. It is worth noting that unlike preemptive RANSAC, the proposed method does not make a strong assumptions about the inlier ratio in the data being processed. This ability to efficiently adapt to the data and provide accurate real-time robust estimation makes ARRSAC easily deployable in real-time vision systems.

3.3.2 Algorithm Description

The ARRSAC algorithm is presented in Algorithms 2 and 3. Below, we discuss some of the aspects of the algorithm.

Algorithm 3 ARRSAC: Generating the initial hypothesis set

$k = 1$, $M' = M$, $count_{in} = 0$, Inner RANSAC flag: $flag_{in} = 0$
Set max number of inner RANSAC iterations: M_{in}
repeat
 if $flag_{in} = 0$ **then**
 Generate hypothesis $h(k)$ by selecting the k -th PROSAC sample
 else
 Generate hypothesis $h(k)$ by sampling non-minimal subset from \mathcal{U}_{in}
 $count_{in} = count_{in} + 1$
 if $count_{in} = M_{in}$, reset $count_{in} = 0$, $flag_{in} = 0$, $\mathcal{U}_{in} = \{\}$
 end if
 Evaluate hypothesis $h(k)$ using SPRT test (Section 3.2.4.3)
 if hypothesis $h(k)$ is accepted **then**
 if largest support so far **and** $flag_{in} = 0$ **then**
 $flag_{in} = 1$, $count_{in} = 0$
 Set $\mathcal{U}_{in} =$ support of hypothesis $h(k)$
 end if
 Calculate termination length and number of hypotheses M' (Equation 3.14, 3.16)
 end if
 $k = k + 1$
 until ($k > M'$)
Return k , set containing accepted hypotheses

3.3.2.1 Generating the initial hypothesis set:

Algorithm 3 sketches out the selection of the initial candidate set of hypotheses in ARRSAC. The main differences between this step and preemptive RANSAC are that (a) we perform PROSAC-style sampling in order to preferentially generate better hypotheses in the candidate set (b) we use a depth-first preemption strategy when generating this set, in order to discard bad hypotheses early on (c) use of simple local refinement to improve the quality of the hypotheses and (d) we determine the number of hypotheses adaptively, using the PROSAC stopping criterion (see Section 3.2.6.5).

One way to think about this stage of the algorithm is as a batch-mode PROSAC, where the evaluation is restricted to a subset (i.e., block of size B) of data points with the highest quality scores. In addition, note that we cap the maximum number of hypotheses in the initial set at M , ensuring that the total number of generated hypotheses is bounded. In the event that the ordering based on quality scores does not help, it is indeed the case that ARRSAC too might fail to return a good solution, much like preemptive RANSAC. There is thus a trade-off between efficiency (i.e., real-time performance) and accuracy. However, as our experiments have indicated, this situation does

not arise frequently in practice, and ARRSAC is able to operate effectively in real-world application scenarios.

3.3.2.2 Preemptive evaluation:

The initial candidate set in ARRSAC is generated using a partially depth-first approach. For the evaluation of the surviving hypotheses, we adopt a strategy similar to preemptive RANSAC, but with some key differences. In particular, while preemptive RANSAC uses the fixed preemption function in Equation 3.20, ARRSAC is more flexible in this regard. At any given point, there are typically much less than $f(i)$ hypotheses in the set, due to the fact that contaminated hypotheses are discarded using the SPRT test. In practice, this turns out to be a significant computational saving (see Section 3.3.3). In Algorithm 2, after every evaluation of B data points, the surviving models are re-ordered based on the results of the evaluation on the current block, and the best half are propagated for the next round of evaluation.

3.3.3 Experimental evaluation

We now present an evaluation of the proposed algorithm on synthetic and real data. Recall from the discussion in Section 3.3, that the main challenge for real-time robust estimation is the ability to deal with a predefined limit on the maximum number of hypotheses. In other words, the algorithms should ideally be able to cope with variation in the level of data contamination, etc., while still retaining robustness in the face of the hypothesis constraint. To this end, we fix the maximum number of hypotheses for both preemptive RANSAC and ARRSAC, and test the ability of these algorithms to handle challenging datasets.

We first evaluate the effectiveness of preemptive RANSAC on synthetic data, over a range of parameter variations. The goal in this case is not to examine the number of verifications or the execution time (which are constant). Rather, we seek to examine the ability of preemptive RANSAC to cope with varying, and *a priori* unknown, inlier ratios. To do so, we generate synthetic 2D feature matches corresponding to a hypothetical fundamental matrix, and perturb the feature locations in each image by zero-mean Gaussian noise with $\sigma = 0.50$. A varying number of outlier matches are then added to the set of inliers to obtain a set of putative correspondences with a range of inlier

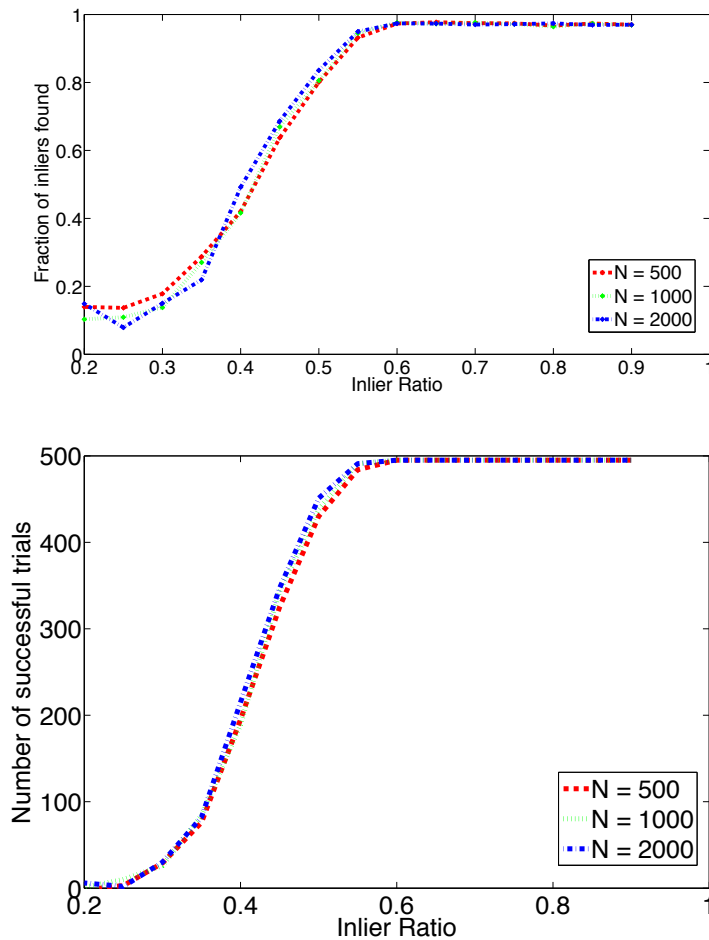


Figure 3.9: (a) Variation in fraction of inliers found by preemptive RANSAC as a function of inlier ratio. (b) Number of successful runs (out of 500), defined as those where preemptive RANSAC returned $\geq 50\%$ of the inliers in the data, plotted as a function of inlier ratio.

ratios. We then run preemptive RANSAC on these sets of putative feature matches, fixing M (the number of hypotheses in preemptive RANSAC) to 400. In the context of the standard stopping criterion (Equation 2.4), using the the 7-point algorithm for fundamental matrix estimation, this number of iterations corresponds approximately to a setting of $\eta_0 = 0.95$ and $\varepsilon = 0.50$.

Figure 3.9(a) shows the fraction of true inliers returned by preemptive RANSAC, as a function of ε , for three different values of N (total number of correspondences). The plots represent average values over 500 runs of the algorithm. The main trend to be observed from the graph is that for higher inlier ratios (e.g., $\varepsilon \geq 0.55$), preemptive RANSAC returns at least 95% of the true inliers on average. Note that this does not correspond perfectly to the true parameter setting of $\varepsilon = 0.50$, which

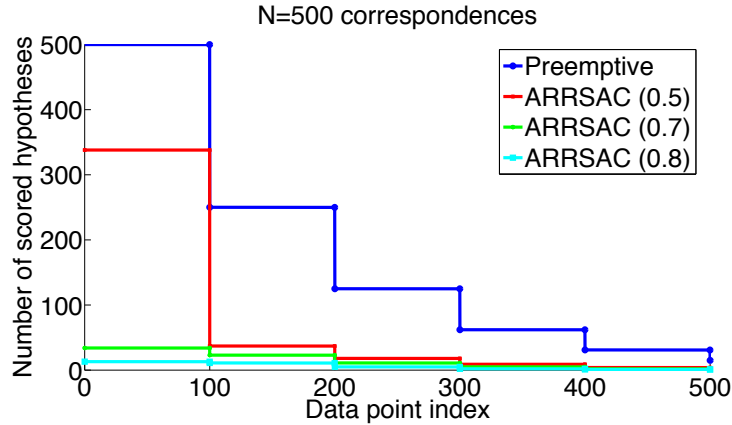


Figure 3.10: Number of hypotheses scored at each stage of evaluation in preemptive RANSAC and ARRSAC, for three values of ε . For this experiment, the number of data points $N = 500$, and the block size B was set to 100.

was used to obtain the number of hypotheses $M = 400$. This is because we are not compensating for noise in the minimal sample, and the resulting inlier counts are consequently lower than the theoretically expected number. It can also be seen that as the inlier ratio drops below $\varepsilon = 0.50$, the performance of preemptive RANSAC starts to suffer, in terms of fraction of inliers returned. This is due to the fact that for these low inlier ratios, preemptive RANSAC does not evaluate a sufficient number of hypotheses, and thus an incorrect solution, with lower inlier count, is invariably returned. This is further illustrated in Figure 3.9(b), which shows the number of runs, out of 500, for which preemptive RANSAC returned at least 50% of the inliers in the input dataset. It can be seen that the accuracy of preemptive RANSAC drops off rapidly and, particularly for lower inlier ratios, often fails completely. This highlights the limitation of preemptive RANSAC discussed in Section 3.3: when the true inlier ratio in the data deviates from the strict assumption made in preemptive RANSAC, it often significantly impacts the accuracy of the system.

From the point of view of efficiency, Figure 3.10 compares the number of hypotheses retained at each stage of preemption for preemptive RANSAC, as compared to ARRSAC. These results correspond to the same synthetic fundamental matrix estimation problem as above, with $N = 500$ correspondences and a block size of $B = 100$. The graph shows the number of hypotheses retained (y-axis) following each round of evaluation (x-axis). Note that for this experiment, it was assumed that a PROSAC-based ordering of the hypotheses was unavailable, and thus the sampling in both cases is performed uniformly at random. It can be seen that even without PROSAC, the partially

depth-first evaluation strategy used in ARRSAC causes an appreciable decrease in the number of hypotheses evaluated at each stage. In particular, note that the number of hypotheses for each round in preemptive RANSAC is fixed; this corresponds to the fixed preemption function in Equation 3.20. On the other hand, due to the fact that bad hypotheses are discarded early on in ARRSAC, the number of good hypotheses that survive is considerably lower as compared to preemptive RANSAC. Thus, the preemption function in ARRSAC is able to *adapt* to the data, and always remains below that of equation 3.20, providing considerable computational savings.

Finally, we provide an evaluation of ARRSAC on real test data, comparing its performance against other robust estimators. A sample of the test images chosen for this experiment are shown in Table 3.3.3, and were selected to cover a range of inlier ratios and number of correspondences. As in Section 3.2.7, corresponding features in the image pairs were detected using SIFT matching, and the ratio scores were used to sort feature matches based on quality. The maximum number of hypotheses was held fixed to 500 for both preemptive RANSAC and ARRSAC. Table 3.3.3 lists the summary statistics for five methods: baseline RANSAC, SPRT, PROSAC, preemptive RANSAC and ARRSAC. While the first three columns are listed for comparison against traditional RANSAC approaches, note that only the last two columns are directly comparable, since preemptive RANSAC and ARRSAC are the two methods that are intended for real-time applications, which is the focus of this work.

A few important trends are noticeable from the table. Firstly, for images with higher inlier ratios (**A–C**), it can be observed that both preemptive RANSAC and ARRSAC deliver solutions that are comparable in terms of the number of inliers returned. In particular, preemptive RANSAC returns more inliers on average than baseline RANSAC, since it evaluates many more hypotheses than necessary. However, this comes at a higher computational cost; for these scenarios, preemptive RANSAC in fact operates slower than standard RANSAC (refer ‘spd-up’ in Table 3.3.3). On the other hand, ARRSAC is able to efficiently estimate the inlier ratio and accordingly limit the number of hypotheses, producing a considerable speed-up while retaining quality. Secondly, for images with low inlier ratios (**D–F**), preemptive RANSAC is often unable to find the correct solution since it evaluates too few hypotheses. In contrast, the non-uniform hypothesis generation step in ARRSAC enables accurate estimation of the epipolar geometry even for very low inlier ratios. In addition,

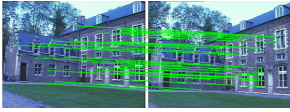
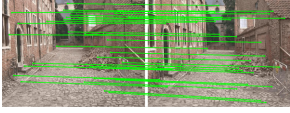
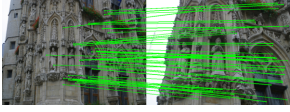
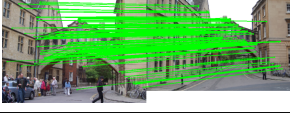

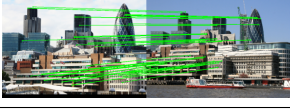
		RANSAC	SPRT	PROSAC	Preemptive	ARRSAC
A : $\varepsilon = 0.72, N = 713$ 	I	485±21	486±21	364±73	510±15	512±8
	k	76	78	8	500	22
	vpm	713.0	67.2	713.0	198.9	210.6
	time (ms)	6.9	2.2	0.8	17.3	0.6
	spd-up	1.0	3.2	8.8	0.4	11.2
B : $\varepsilon = 0.67, N = 735$ 	I	498±22	492±21	349±73	492±21	495±11
	k	86	99	9	500	28
	vpm	735.0	77.8	735.0	199.0	204.6
	time (ms)	10.4	3.7	1.4	17.3	0.9
	spd-up	1.0	2.8	7.3	0.6	10.8
C : $\varepsilon = 0.57, N = 575$ 	I	315±12	316±11	115±13	324±7	326±5
	k	371	385	7	500	27
	vpm	575.0	11.5	575.0	196.0	204.8
	time (ms)	15.5	4.4	1.4	17.2	0.8
	spd-up	1.0	3.5	11.1	0.9	20.4
D : $\varepsilon = 0.38, N = 1088$ 	I	381±13	381±12	325±16	217±61	403±10
	k	7935	8403	14	500	48
	vpm	1088.0	22.6	1088.0	200.0	208.7
	time (ms)	546.53	108.77	5.41	17.3	3.74
	spd-up	1.0	3.2	8.8	31.6	147.7
E : $\varepsilon = 0.38, N = 422$ 	I	141±6	141±5	137±5	112±26	152±8
	k	12218	12776	498	500	161
	vpm	422.0	15.0	422.0	189.0	204.2
	time (ms)	391.62	160.47	25.58	17.3	9.79
	spd-up	1.0	2.8	7.3	22.6	39.9
F : $\varepsilon = 0.45, N = 904$ 	I	388±10	388±11	311±33	301±55	400±8
	k	1885	1956	23	500	56
	vpm	904.0	15.7	904.0	199.8	187.6
	time	111.37	24.55	2.57	17.3	5.12
	spd-up	1.0	3.5	11.1	6.4	21.8

Table 3.7: Evaluation results for real image pairs. The real-time approaches, preemptive RANSAC and ARRSAC, are listed in the last two columns. The table lists the number of inliers returned (I), number of models evaluated (k), the number of verifications per model (vpm), along with the runtime (in milliseconds), and the speed-up (spd-up) compared to standard RANSAC. It can be observed from the above results that the ARRSAC approach produces significant computational speed-ups, while simultaneously providing accurate robust estimation in real-time. Note that the number of hypotheses evaluated by ARRSAC is always less than preemptive RANSAC. In addition, the correct epipolar geometry is always recovered.

the efficient preemption strategy employed ensures that even for these cases of low inlier ratios, the overall runtime is still well below the real-time budget. The results in Table 3.3.3 highlight the fact that ARRSAC is able to elegantly adapt to the data and ensure accurate robust estimation over a wide range of inlier ratios, and is thus suitable for challenging real-time robust estimation scenarios.

3.3.4 Summary

In this section, we presented an algorithm for real-time robust estimation. In particular, ARRSAC can be viewed as a special case of USAC (Section 3.2), with various optimizations made to adapt the algorithm for real-time applications. In particular, while traditional robust estimation approaches place no hard limit on the number of hypotheses, real-time estimators limit this number based on the desired running time. The approach we present leverages state of the art techniques: non-uniform sampling, local optimization and depth- and breadth-first preemptive approaches in order to achieve high performance. While our results are promising, note that unlike RANSAC and the other variants discussed in Section 3.2, neither preemptive RANSAC nor ARRSAC guarantee that the correct solution will be returned in the worst case. As noted in Section 3.3, in real-time systems, the goal is to obtain the best solution given the time constraints. While preemptive RANSAC achieves this goal under certain conditions, our aim in ARRSAC is, in some sense, to extend the range of operating conditions across which this goal is achieved. Thus, while preemptive RANSAC makes several strict assumptions about the data, ARRSAC relaxes these assumptions, and is consequently applicable in a wider range of more challenging scenarios.

3.4 Large Scale Applications

As mentioned in Chapter 1, one of the motivating applications for the algorithms developed in this dissertation is that of large-scale structure from motion. To this end, both the algorithms introduced in this chapter – USAC and ARRSAC – have been implemented and used in large-scale 3D reconstruction systems. In these systems, robust estimators are employed most commonly to perform the task of geometric verification: in other words, given a set of tentative feature matches between a pair of images, the goal is to compute the corresponding epipolar geometry (if any) between the two images. This is a fundamental component of any real-world system that seeks to model large-scale imagery either in the form of video sequences (Nister et al., 2004; Malis and Marchand, 2006; Pollefeys et al., 2008; Clipp et al., 2010) or photographs gathered from the internet (Snavely et al., 2008; Agarwal et al., 2009; Frahm et al., 2010; Raguram et al., 2011).

3.4.1 Real-Time Robust Estimation with ARRISAC

In the context of video, ARRISAC has been integrated into the UrbanScape real-time structure from motion system described in (Pollefeys et al., 2008). In this system, ARRISAC is used for 5-point (Nistér, 2004) and 3-point (Haralick et al., 1994) pose estimation, and operates at 100+ Hz on average. In the context of internet photo collections, ARRISAC has also been employed in (Frahm et al., 2010; Raguram et al., 2011), whose focus is in the modeling and organization of landmark image collections. In particular, the goal in (Frahm et al., 2010) is to “build Rome on a cloudless day” – i.e., to reconstruct 3D models of Rome on a single PC (as opposed to a cluster of computers), within the span of a day. Thus, this is again a scenario with a fixed time budget, making ARRISAC a good candidate for robust estimation. In this system, ARRISAC is used primarily for 7-point epipolar geometry estimation and achieves speeds of 450+ Hz, when implemented in a multi-threaded application across 8 CPU cores.

3.4.2 Improved Geometric Verification with USAC

Most recently, we have investigated the use of USAC for processing large-scale internet photo collections, with a focus on finding a better strategy to weight feature matches based on their quality (Raguram et al., 2012b). As the results in Sections 3.2.7 and 3.3.3 indicate, taking prior information into account can often significantly improve the overall computational efficiency of the robust estimator. It is worth noting, however, that thus far, virtually the only kind of ordering information that has been used in PROSAC has been purely image-to-image (Chum and Matas, 2005; Raguram et al., 2008; Sattler et al., 2009; Ni et al., 2009). Note that this kind of ordering does not leverage any information that might have been obtained from prior matching rounds – in other words, each pair of images is verified completely independently of the others. Particularly for the case of photo collections, where images of the same 3D scenes are repeatedly encountered, there is much to be gained by altering the ordering scheme to take prior matching results into account.

3.4.2.1 Related Work

Thus far, the typical way to perform geometric verification in image collections (Snavely et al., 2008; Agarwal et al., 2009; Chum and Matas, 2010a; Frahm et al., 2010; Philbin et al., 2011) has

been to estimate the geometric relationship between image pairs *independently*. Our main idea in this work is simple: as the geometric verification progresses, we learn information about the image collection, and subsequently use this learned information to improve efficiency. More specifically, since images of the same geometric structures are being repeatedly verified against each other, this process of repeated matching reveals useful information about the stability and validity of low-level image features. While current techniques either ignore this information, or leverage it for other tasks via an offline processing stage, we feed this information directly back into the verification pipeline.

Relevant to our approach are techniques for the related problems of location recognition and image retrieval, where the goal is to efficiently identify and return images that are geometrically related to a given query image. Given that efficiency and accuracy are important in this setting, a number of recent approaches have addressed the problem of learning how to select informative image features – or, alternatively, how to suppress uninformative features (Li et al., 2010; Turcot and Lowe, 2009; Naikal et al., 2011; Sattler et al., 2011; Knopp et al., 2010). Also closely related is the work of (Mikulík et al., 2010), which uses the output of geometric verification to learn a probabilistic similarity measure between visual words, thus linking together words that are likely to be related. While similar in spirit to our idea, our goal is quite different – our aim is to fully utilize this information in an *online* way. This is a distinguishing characteristic from current techniques that have thus far obtained this information via an offline, preprocessing step, or through a *post-hoc* phase that uses the output of structure-from-motion. In addition, in contrast to techniques such as (Li et al., 2010; Turcot and Lowe, 2009; Knopp et al., 2010), which operate at the *feature* level (i.e., using the results of exhaustive geometric verification to either prioritize or prune image features for every image in the dataset), our approach explores the weighting of *visual words*. In this respect, our approach is similar to that of (Naikal et al., 2011), which uses an offline training stage to identify a subset of the visual vocabulary that contains information that is most useful for landmark identification. However, in contrast, we do not require a dedicated learning stage or labeled training data; our system incrementally learns as it processes new image pairs.

3.4.2.2 Approach

Identifying useful visual words

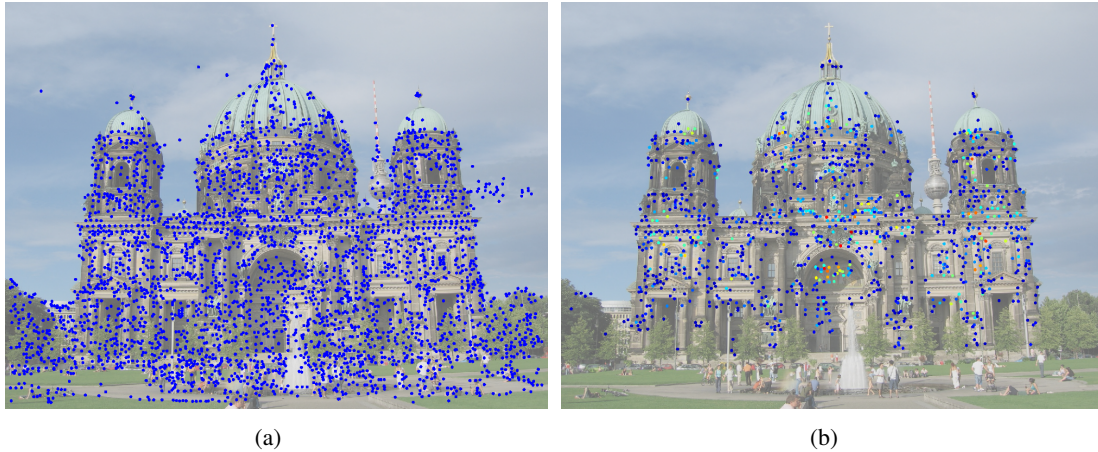


Figure 3.11: (a) All detected features for a single image (b) Features filtered based on the results of geometric verification – only visual words that were inliers in at least 10 previous image pairs are shown. The features in (b) are heatmap colour coded based on the inlier counts. (Figure best viewed in colour).

As a motivating example, consider Figure 3.11(a), which shows all detected SIFT features for a single image. Note that a large number of features lie in areas of the image that are very unlikely to pass any geometric consistency check (for e.g., features on vegetation, people, and in the sky). Now, if we have previously verified *other* images of the same scene, we can weight each visual word in the current image by the number of times that the word has previously passed the geometric consistency check in other image pairs (see Figure 3.11(b)). Note, in particular, that this weighting emphasizes visual words that are stable, reliable, and more likely to be geometrically consistent (for instance, those in the central portion of the structure), while also suppressing spurious visual words. Similar ideas have been explored to some extent in recent work (Knopp et al., 2010; Turcot and Lowe, 2009), but at the feature-level. Our approach extends this idea in two ways: (1) similar to (Mikulík et al., 2010) we work at the visual word level, which in turn allows us to predict, for a never before seen image, which feature matches are likely to be geometrically consistent, and (2) since our goal is geometric verification, we incorporate this visual word prioritization strategy into the verification step itself (i.e., no preprocessing or labeling of images is required).

Computing visual word priorities

In the interests of computational efficiency, we adopt a very simple strategy to identify potentially useful visual words. Consider a visual vocabulary, $W = \{w_1, w_2, \dots, w_N\}$, consisting of N visual

words. Typically, this vocabulary is generated by (approximate) k-means clustering, using a diverse set of image descriptors (Philbin et al., 2007). In addition, consider a set of visual word *priorities*, $C = \{c_1, c_2, \dots, c_N\}$, where each c_i represents a score that is proportional to the validity of the visual word. In the absence of any prior information, we start by assigning each of these visual words the same priority (i.e., $c_i = 0, \forall i$). We then carry out geometric verification on the image collection, selecting image pairs using either a retrieval-based method as in (Agarwal et al., 2009; Chum and Matas, 2010a), or a clustering based-method as in (Frahm et al., 2010). For each pair of selected images, we match SIFT features, and then run a robust estimator to identify a set of inliers.

Each pair of matching features is associated with a visual word from the set W (for simplicity, we ignore for now the case where a pair of matched features gets assigned to different visual words in each image; we will return to this point later). For every pair of images that we *successfully* verify (where a “success” is considered to be a pair of images with $> I_{min}$ inliers; commonly chosen values of I_{min} range between 15-20), we then update the priority of the inlier visual words based on the results of this process. The simplest possible scheme would consider the set C to be a set of inlier counts – in other words, for each feature match that was found to be an inlier, we update a count c_i for the corresponding visual word. In the case where the matched points are assigned to different words, we simply update the counts for both visual words. Intuitively, over time, we expect that these counts will help identify visual words that are frequently matched as inliers, as well as words that repeatedly fail the geometric consistency check. Note that the set C is maintained globally and is used across all image pairs.

Improving USAC sampling

Assume we are given a set of counts $C = \{c_1, c_2, \dots, c_N\}$, obtained by matching a set of image pairs. This weighting of visual words can then easily be incorporated into USAC in order to bias the sampling in favour of the more reliable words. In particular, given a pair of images, consider a set \mathcal{S} containing N matched features. For each matched feature in \mathcal{S} , we have a corresponding visual word w_i , with associated count c_i . We simply order the matches in \mathcal{S} based on the counts c_i , and then run USAC as before.

3.4.2.3 Results

We first evaluate the effect of the modified ordering scheme based on visual words counts on the robust estimation stage. We report results on two experiments, representing different usage scenarios:

Experiment 1: We consider a dataset of 10000 images representing a single landmark (downloaded by doing a keyword search for “Berlin Dome” on Flickr). This dataset is relatively clean, though a small fraction ($\approx 5\%$) of unrelated images are present in the dataset. We process this dataset using the approach of (Agarwal et al., 2009), by retrieving 20 match candidates for each image in the dataset, and performing geometric verification. We compare the performance of three estimators: (a) baseline RANSAC (denoted as **R1**) (b) USAC with ordering based on SIFT matching scores (**R2**) and (c) USAC with the proposed ordering based on visual words (**R3**). The only difference between **R2** and **R3** is in the ordering used to prioritize matches. In all cases, we estimate the epipolar geometry using the 7-point algorithm. For this experiment, we use a visual vocabulary with 20,000 words, computed by k-means clustering using a set of randomly chosen descriptors from the dataset.

Experiment 2: In this experiment, we process a much larger dataset, consisting of 2.77 million images of Berlin, also used in (Frahm et al., 2010). To handle datasets of this magnitude, we use the clustering based approach described in (Frahm et al., 2010), which is capable of scaling well to these larger datasets. We extract binarized gist features for each image, and then cluster using k-medoids with $k = 100000$ centers. We then perform geometric verification independently on each cluster, by first trying to identify a set of $m (=3, \text{ in our experiments})$ consistent images in each cluster, denoting the image with the most inliers as the “iconic”. We then register all remaining images in the cluster to this iconic. Compared to the approach used for Experiment 1, this strategy dramatically reduces the total number of pairwise image verifications that need to be performed (Frahm et al., 2010). For this experiment, we compare the performance of **R2** and **R3**, since RANSAC (**R1**) is impractical for datasets of this size. In this case, we use a larger vocabulary with 10^6 visual words, computed using approximate k-means clustering (Philbin et al., 2007).

In all cases, note that there is no dedicated “training” phase for **R3**. Initially, we start with all visual word priorities set to zero, and then progressively accumulate inlier counts. As the matching progresses, the efficiency of **R3** increases rapidly, and after verifying about 500 pairs on average, it becomes more efficient than the matching score based ordering. This is an empirical observation

Table 3.8: Experiment 1 (BerlinDome-10k)

	R1	R2	R3
Mean time per image pair (ms)	389.6	41.2	28.9
Mean # of hypotheses per image pair	14218.4	604.1	399.7
Total runtime (hours:minutes)	23:48	04:08	03:16

Table 3.9: Experiment 2 (Berlin-2.77M)

	R2	R3
Mean time per image pair (ms)	26.7	18.8
Total runtime (hours:minutes)	02:50	02:28
# iconics	9841	9912
# registered images	132,719	133,830

at this stage, and investigating this progression in more detail is an interesting direction for future work. In the results we report, for method **R3**, we start by using the matching score based ordering, and then switch over to the visual word based ordering after 500 image pairs. The results for the two experiments are shown in Tables 1 and 2. For the smaller BerlinDome-10k dataset, it can be seen that the new ordering scheme improves on the runtime of the matching score based ordering by about 30%. This provides strong evidence that using information accumulated during the matching process helps improve efficiency. For Expt. 1, we do not distribute our processing across multiple CPU/GPU cores; thus, the reported total runtime is for a single thread. For the large-scale Berlin-2.77M dataset, much the same trend holds for the per-image pair estimation time, which indicates that the weighting based on visual words is still reliable even with the presence of multiple landmarks in the dataset. It can be seen that the mean time to verify a single image pair is reduced compared to the results in Table 1; this is a consequence of the viewpoint-based clustering which increases the mean inlier ratio by grouping similar images in the same cluster. Finally, the overall runtime numbers reported in Table 2 are for a parallel implementation that distributes image pairs across 16 CPU threads for geometric verification. Note that the overall runtimes are not directly comparable between Tables 1 and 2, since these use two different strategies (Agarwal et al., 2009; Frahm et al., 2010) to perform the verification.

A specific example is shown in Figure 3.12(a). The inlier ratio for this image pair is significantly low ($\approx 20\%$), due to large changes in viewpoint and scale, coupled with repetitive patterns and symmetries. For this low inlier ratio, standard RANSAC requires close to 1.07×10^6 samples, which is computationally prohibitive. USAC with feature matching scores requires about 22000 samples, while USAC with visual word ordering takes 595 samples, which represents a 36x improvement compared to ordering using matching scores.

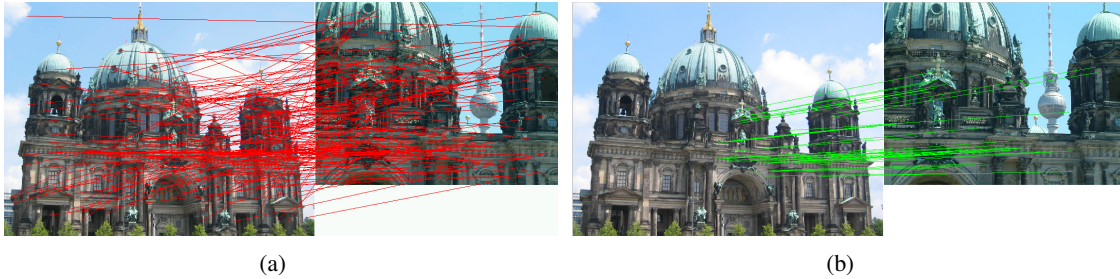


Figure 3.12: (a) Putative features matches and (b) inliers. In this case, the inlier ratio is $\approx 20\%$.

3.4.3 Summary

In this section, we have presented a simple technique for taking advantage of the information generated during geometric verification, to improve the efficiency of the robust estimation stage. We show that reliable statistics can be computed on the visual-word level and used to guide the sampling process. Our approach thus integrates online knowledge extraction seamlessly into structure-from-motion systems, and is particularly relevant for large-scale image collections. While this is a promising first step, there are interesting directions for future work. In particular, while we currently use the visual word weights only to guide sampling, it might be possible to integrate this kind of learned information into the matching stage as well. For instance, the work of (Mikulík et al., 2010) describes a way to use the output of geometric verification to link together words that are likely to be related. Incorporating this type of learned information into the feature matching stage could further improve the efficiency of the overall geometric verification process.

3.5 Discussion

In this chapter, we introduced two algorithms for robust estimation, USAC and ARRSAC. The common thread linking these two algorithms is that they represent a synthesis of state of the art RANSAC techniques. By combining various ideas from the literature, these methods achieve both high performance as well as robustness, and represent the state of the art in robust estimation. Both these algorithms have been integrated into 3D reconstruction systems, and have enabled real-time operation as well as the processing of very large scale datasets.

As a secondary contribution, we have developed a stand-alone C++ library that implements the USAC framework using state of the art algorithms for each stage (USAC-1.0). This library provides

state of the art performance and we plan to make it freely available to the community. The library can be used as a stand-alone robust estimator that can be plugged-in for use in specific applications; as a benchmark against which to compare various robust estimators; and as an easily extendible base for developing new algorithms, for others to use and to build upon.

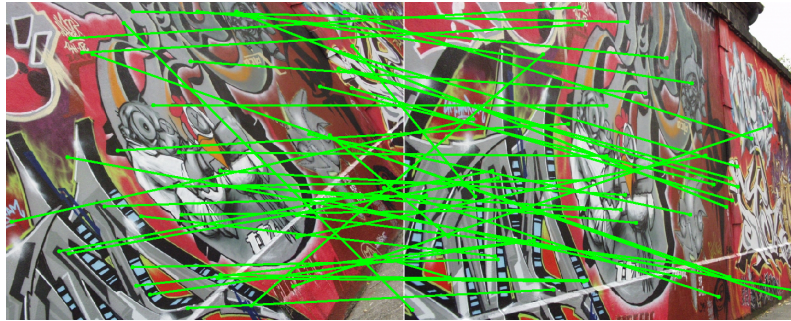
Chapter 4

Exploiting Uncertainty in Random Sample Consensus

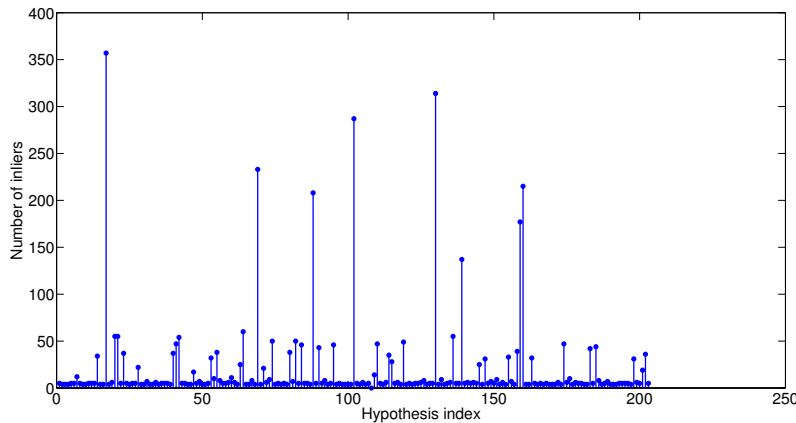
4.1 Introduction

In this chapter, we turn our attention to the standard RANSAC stopping criterion (Section 2.2.3), and look more closely at the assumptions therein. As we have seen in the previous chapter, in its standard formulation, RANSAC operates in a simple hypothesize and verify loop, relying on the fact that with enough iterations, a “good” sample (i.e., a minimal subset that is free from outliers) will eventually be drawn, resulting in a model that has largest support. Since it is computationally infeasible to try every possible minimal subset, the standard termination criterion in RANSAC (see Equation 2.4) determines the number of trials required to ensure with some confidence η_0 , that at least one minimal subset is drawn which is free from outliers.

The implicit assumption in the above formulation is that a model produced from an all-inlier minimal subset will be consistent with all other inliers in the data. In practice, this assumption is often violated, due to the effects of noise. To illustrate this with an example, consider Figure 4.1(a), which shows a subset of the putative feature matches for a pair of images of a planar scene. We can run RANSAC to estimate a 2D homography from this set of putative matches. The results for a complete execution of the RANSAC algorithm are shown in Figure 4.1(b), which plots the number of inliers (y-axis) corresponding to each model hypothesis generated during the RANSAC execution



(a)



(b)

Figure 4.1: (a) Subset of matched SIFT features for a pair of images (b) Results of a single execution of RANSAC using the putative features from (a). The graph plots the number of inliers (y-axis) corresponding to each model hypothesis generated during the RANSAC execution (x-axis).

(x-axis). It can be seen from the graph that there are a few samples that obtain high inlier counts, while the majority of samples result in only a few inliers. We can focus our attention on these high-inlier hypotheses, and look more closely at the minimal samples that correspond to these. This is illustrated in Figure 4.2, which highlights the eight hypotheses whose support is significantly larger than the rest (labels ❶–❸). Figure 4.2 also shows the minimal samples corresponding to these model hypotheses. The main observation to be made from inspecting these minimal samples is that each of these is an all-inlier minimal sample – in other words, each is a sample that should have gathered all other inliers, but failed to do so.

This effect has been observed previously in the robust estimation literature (Tordoff and Murray, 2002; Chum et al., 2003). Indeed, as discussed in Section 3.2.5, the presence of noise in the data, coupled with the use of minimal samples, implies that the estimated model parameters are also affected by noise. In addition to a reduction in accuracy, another consequence of the decreased

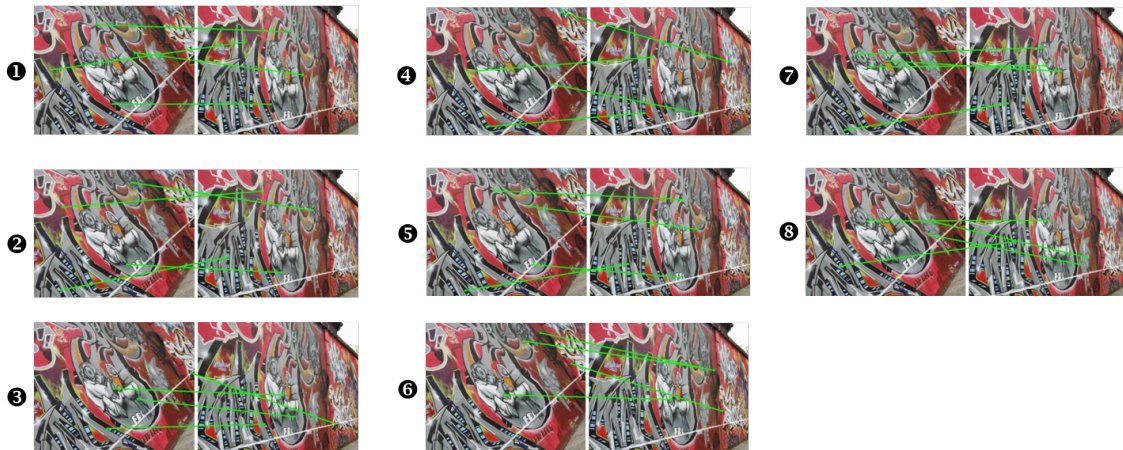
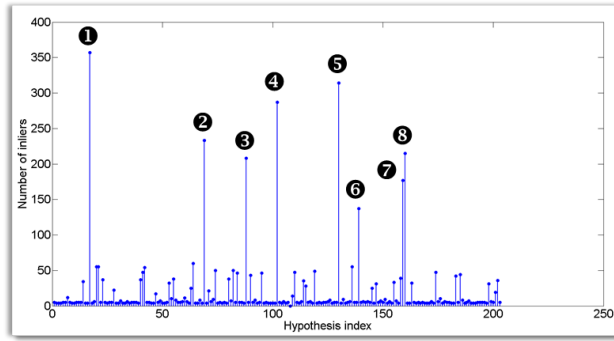


Figure 4.2: **Top:** Graph of inlier counts corresponding to the various model hypotheses generated during a RANSAC execution. **Bottom:** Minimal samples corresponding to the hypotheses labeled 1–8 in the graph.

support of these models is that the number of samples drawn in RANSAC typically increases by a factor of two to three compared to the theoretically expected number. Going further, we have observed previously (see Section 2.2.3) that all-inlier minimal subsets are sampled multiple times in RANSAC before the algorithm terminates. In particular, to achieve a 95% confidence in the solution, approximately three all-inlier samples are drawn by RANSAC. These two observations explain the presence of the multiple peaks in Figures 4.1 and 4.2 – simply put, RANSAC generates multiple all-inlier samples, each of which has reduced support due to the effects of noise.

Our goal in this chapter is to address these issues, by quantifying the effects of noise on the model parameters. More specifically, we introduce a new framework for robust estimation, which addresses some of the issues mentioned above, by *explicitly* characterizing the uncertainty of the models hypothesized in the estimation procedure in terms of their covariance. This uncertainty arises from imprecise localization of the data points from which the model is computed, as well as

from their spatial distribution. We show that by explicitly accounting for these uncertainties, it is possible to account for the effects of noise in the data. In other words, we show that knowledge of the uncertainties can be leveraged to develop an inlier classification scheme that is dependent on the model uncertainty, as opposed to a fixed threshold. This strategy has the advantage that, given a model with associated uncertainty, we can immediately identify a set of points that support this solution, if the imperfect model estimation were taken into account. We then also demonstrate how the standard RANSAC stopping criterion can be altered to take this into account, thus achieving a reduction in the total number of samples required.

4.2 Related Work

The Locally Optimized RANSAC algorithm (Chum et al., 2003), discussed in Section 3.2.5, was the first approach that addressed the effects of noise in RANSAC. To recap: given a hypothesis with the largest support so far, LO-RANSAC performs an “inner RANSAC” loop, where non-minimal subsets are sampled from within the support of the current best solution, thus resulting in more accurate parameter estimates. The RANSAC algorithm then resumes sampling on all data points, carrying out the local optimization step every time a hypothesis with better support is found. The effect of the local optimization step is to mitigate the effects of noisy data points by using non-minimal subsets to generate hypotheses. Since this typically results in an increase in the support, the net effect is that the overall number of iterations required by the algorithm moves closer to the theoretically predicted number. Our approach in this work extends the LO-RANSAC idea by explicitly modeling the effect of noise on the model parameters. Thus, while LO-RANSAC focuses on the effect of noise (reduced support), we focus on the cause (uncertainty in model parameters). The two approaches are also, in some sense, complementary; in particular, the uncertainty analysis we develop helps in determining a set of inliers, and we can perform local optimization in order to refine this solution. In addition, our method improves upon LO-RANSAC by using a different criterion to determine when to stop sampling. In practice, this results in a reduction in the number of samples required.

There exists a wide body of work in photogrammetry and computer vision that deals with error analysis and propagation with respect to image measurements. A survey of this literature can be

found in (Hartley and Zisserman, 2000; McGlone, 2004), with more specific applications in (Hartley, 1994; Meidow et al., 2009a). We build on this foundation, but restrict our attention to some common geometric entities that are often encountered in computer vision – namely, the homography and the fundamental matrix. Uncertainty analysis for these two entities has been developed in the literature (Csurka et al., 1997; Criminisi et al., 1999; Sur et al., 2008), and our main contribution in this work is to leverage these techniques within the context of robust estimation. Similar ideas have been used (Ochoa and Belongie, 2006) to perform guided matching using point covariances. While similar in spirit to our approach, we aim to integrate this step into a RANSAC algorithm in order to improve accuracy and efficiency.

4.3 Uncertainty Analysis

As noted in Section 4.2, there exists a wide body of work in photogrammetry and computer vision that deals with error analysis and uncertainty estimation. In this work, we restrict our attention to some common geometric entities in order to specifically demonstrate how knowledge of uncertainties may be integrated into a random sampling framework. Before describing the proposed robust estimation algorithm, we first provide a discussion of uncertainty analysis as applied to the estimation of two common geometric relations, the homography and fundamental matrix, and show how point uncertainties are transformed under the covariance of an estimated relation. The discussion in this section refers mainly to (Criminisi et al., 1999) and (Sur et al., 2008); the former develops the theory behind characterizing the uncertainty for homography estimation in terms of its covariance, while the latter deals with fundamental matrix estimation.

4.3.1 Uncertainty for homography estimation

Given a set of corresponding points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, where \mathbf{x}_i and \mathbf{x}'_i are in \mathbb{P}^2 , the 2D homography \mathbf{H} is the 3×3 projective transformation that takes each \mathbf{x}_i to \mathbf{x}'_i . Since each point correspondence provides two independent equations in the elements of the matrix \mathbf{H} , a total of four corresponding points are required to compute \mathbf{H} up to scale.

Expressing \mathbf{x}_i and \mathbf{x}'_i in terms of homogeneous 3-vectors, the pair of equations for correspondence i is given by:

$$\begin{bmatrix} w'_i \mathbf{x}_i^T & \mathbf{0}^T & -x'_i \mathbf{x}_i^T \\ \mathbf{0}^T & -w'_i \mathbf{x}_i^T & y'_i \mathbf{x}_i^T \end{bmatrix} \mathbf{h} = \mathbf{0}$$

where $\mathbf{x}'_i = (x'_i, y'_i, w'_i)^T$ and \mathbf{h} is the 9-vector made up of the entries of the matrix \mathbf{H} . By stacking the equations corresponding to four point correspondences, we then obtain a set of equations $\mathbf{A}\mathbf{h} = \mathbf{0}$, where \mathbf{A} is an 8×9 matrix. The Direct Linear Transformation (DLT) algorithm (Hartley and Zisserman, 2000) for determining the solution then involves computing the nullspace of \mathbf{A} using the SVD. The unit singular vector corresponding to the smallest singular value is the solution \mathbf{h} (up to scale). In practice, the correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ used in the computation of the homography are imperfectly localized in each image (for simplicity, we ignore for now the issue of outliers). It is common to consider the data points as having errors modeled by a homogeneous, isotropic Gaussian noise process with known variance σ . Note that the standard RANSAC algorithm also uses an estimate of this quantity to set the threshold for inlier classification (see Equation 2.7). Note that while we do make the simplifying assumption of a common, isotropic covariance matrix for every point, as discussed in (Kanatani, 2004), this is a reasonable assumption to make in practice.

Assuming that the points in each image are perturbed by Gaussian noise, these errors in position cause uncertainty in the estimated transformation. This uncertainty can be characterized in terms of a 9×9 matrix representing the *covariance* of the transformation. In general, this covariance matrix depends on the error in point locations, the spatial distribution of the points used to compute the transformation, and the method of estimation. In (Criminisi et al., 1999), it was shown that the covariance matrix $\Lambda_{\mathbf{h}}$ can be computed as

$$\Lambda_{\mathbf{h}} = \mathbf{J}\mathbf{S}\mathbf{J}, \quad (4.1)$$

where $\mathbf{J} = -\sum_{k=2}^9 \mathbf{u}_k \mathbf{u}_k^T / \lambda_k$, with \mathbf{u}_k representing the k^{th} eigenvector of the matrix $\mathbf{A}^T \mathbf{A}$ and λ_k the corresponding eigenvalue. \mathbf{S} is a 9×9 matrix with a closed-form expression. We refer the reader to (Weng et al., 1989; Criminisi et al., 1999) for details on the derivation and the coefficients of \mathbf{S} .

In standard RANSAC, which ignores this uncertainty in the estimated transformation, the computed homography is used to determine the error for each data point. One common measure when

computing the homography is the symmetric transfer error, given by

$$d_{\perp} = d(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i)^2 + d(\mathbf{x}_i, \mathbf{H}^{-1}\mathbf{x}'_i)^2, \quad (4.2)$$

where \mathbf{H} and \mathbf{H}^{-1} represent forward and backward transformations, respectively. For a pair of matched points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, if the computed error from Equation 4.2 is below a predefined threshold t , then that correspondence is classified as an inlier. As noted in Section 2.2.4, if we make the assumption that the point-model errors correspond to a sum of n squared Gaussian variables (where n is the codimension of the model), then the value of the threshold t can be computed as

$$t^2 = \chi_n^{-1}(\alpha)\sigma^2, \quad (4.3)$$

where χ is the cumulative chi-square distribution. In this setting, t represents the value of the threshold that captures a fraction α of the set of inliers; this is typically set to 0.95. However, note that this derivation ignores the fact that the estimated model parameters are also affected by noise. Consequently, if Equation 4.3 is used to compute a fixed threshold t , for use in RANSAC, this threshold will not strictly capture a fraction α of the inliers. This is the reason for the reduced inlier counts seen in Figures 4.1 and 4.2.

In order to determine the effect that the uncertainty of the transform has on point transfer, we consider the forward transformation $\hat{\mathbf{x}}'_i = \mathbf{H}\mathbf{x}_i$ (the formulation is symmetric in the reverse direction). Given a computed homography \mathbf{H} and its covariance, as estimated from Equation 4.1, consider a pair of matched features $\mathbf{x} \leftrightarrow \mathbf{x}'$ which were not used in the computation of \mathbf{H} . Given the (assumed) 3×3 covariance matrix $\Lambda_{\mathbf{x}}$ for point \mathbf{x} , note that the covariance of the transferred point $\hat{\mathbf{x}}'$ may no longer be an isotropic Gaussian distribution, due to the uncertainty in \mathbf{H} . In particular, it can be shown (Criminisi et al., 1999) that the uncertainty in the transferred point is given by

$$\Lambda_{\hat{\mathbf{x}}'} = \mathbf{B}\Lambda_{\mathbf{h}}\mathbf{B} + \mathbf{H}\Lambda_{\mathbf{x}}\mathbf{H} \quad (4.4)$$

where \mathbf{B} is the 3×9 matrix

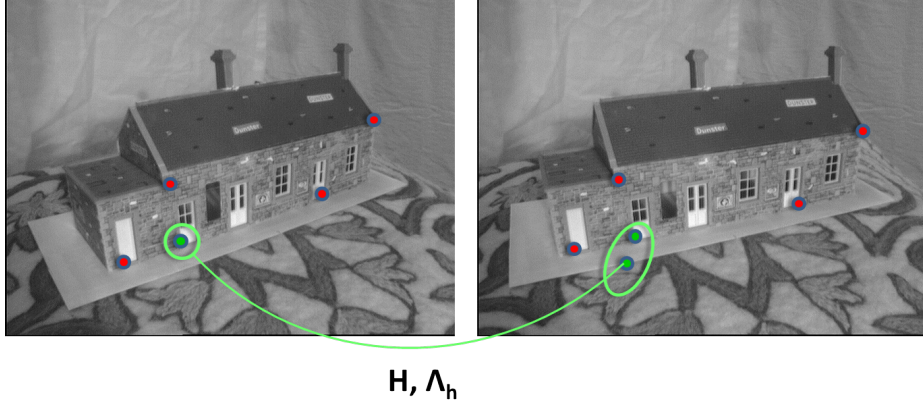


Figure 4.3: Using the covariance matrix of the estimated homography in point transfer.

$$\begin{pmatrix} \mathbf{x}^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{x}^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & \mathbf{x}^T \end{pmatrix}$$

The first term in equation (4.4) represents the uncertainty in $\hat{\mathbf{x}}'$ given an exact point \mathbf{x} and uncertain \mathbf{H} . The second term represents the uncertainty given an exact \mathbf{H} and uncertain point measurement \mathbf{x} . The 3×3 matrix $\Lambda_{\hat{\mathbf{x}}'}$ thus defines a covariance ellipse that represents the uncertainty in point transfer due to both uncertainty in point measurement as well as uncertainty in the estimated homography.

This result is illustrated in Figure 4.3, where a homography \mathbf{H} is computed from a set of four point correspondences (denoted by the red circles), along with its associated covariance matrix $\Lambda_{\mathbf{H}}$. This covariance matrix is then used to transfer the uncertainty of another point (denoted by the green circle) from the left image into the right. It can be observed that the uncertainty of the transferred point is now represented in terms of an ellipse, which represents a level set of the probability density function describing the noise spread around the ellipse centre. Since the measured point in the right image and the uncertainty region of the transformed point overlap, this implies that by accounting for the various uncertainties, this correspondence can be explained as a true match. Note that in this case, we do not use a fixed threshold t for computing the support over all points, as in RANSAC. Rather, as shown in Equation 4.4, both the uncertainty of the transformation as well as the uncertainty of the point are taken into account when making this decision.

4.3.2 Uncertainty for fundamental matrix estimation

Given a set of corresponding points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, the fundamental matrix \mathbf{F} is the 3×3 matrix that satisfies $\mathbf{x}'_i{}^T \mathbf{F} \mathbf{x}_i = 0$. Each correspondence provides one linear equation in the entries of \mathbf{F} . Since \mathbf{F} is defined up to a scale factor, a simple linear solution may be found from 8 point matches. In fact, by enforcing the rank constraint, \mathbf{F} may be computed from 7 point matches. Due to the relative simplicity of the linear 8-point algorithm in terms of the discussion of its uncertainty analysis, we focus on this algorithm in the presented work. Thus, given 8 point correspondences, an appropriately normalized (Hartley, 1997b) matrix \mathbf{M} may be constructed, where

$$\mathbf{M} = \begin{pmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ x'_2 x_2 & x'_2 y_2 & x'_2 & y'_2 x_2 & y'_2 y_2 & y'_2 & x_2 & y_2 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{pmatrix}$$

Solving $\mathbf{M}\mathbf{f} = \mathbf{0}$, where \mathbf{f} is a 9-vector made up of the entries of \mathbf{F} , followed by rank 2 constraint enforcement, then provides the solution.

The uncertainty analysis for the 8-point algorithm was developed in (Csurka et al., 1997; Sur et al., 2008). Given the measurement noise in the pixel coordinates, the goal is to characterize the uncertainty of \mathbf{F} in terms of its covariance matrix $\Lambda_{\mathbf{F}}$. If we denote the solution to the linear system of equations as $\tilde{\mathbf{f}}$, the covariance of $\tilde{\mathbf{f}}$ can be shown to be:

$$\Sigma_{\tilde{\mathbf{f}}} = \sigma^2 \mathbf{J}_X \mathbf{J}_X^T \quad (4.5)$$

where \mathbf{J}_X is an 8×32 matrix that represents the Jacobian of the transformation that generates $\tilde{\mathbf{f}}$ from the vector of point matches X , and σ is the standard deviation of the noise in point measurements.

The rank 2 constraint is then enforced by computing

$$\mathbf{F} = \mathbf{U} \mathbf{D} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V}^T$$

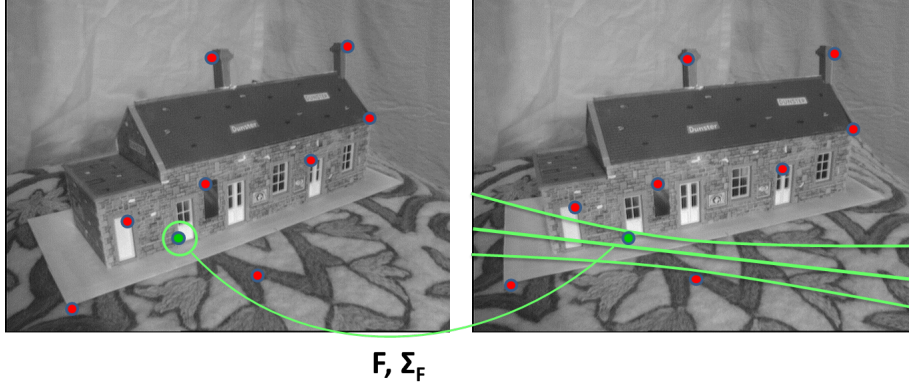


Figure 4.4: Using the covariance matrix of the estimated fundamental matrix in point transfer.

where \mathbf{U} , \mathbf{D} and \mathbf{V} are obtained from the singular value decomposition $\tilde{\mathbf{F}} = \mathbf{U}\mathbf{D}\mathbf{V}^T$. An efficient method for computing the Jacobian of the SVD is described in (Papadopoulos and Lourakis, 2000), and can be used to compute the 9×9 Jacobian matrix \mathbf{J}_{SVD} . The final covariance matrix for \mathbf{F} is then obtained as

$$\Sigma_F = \mathbf{J}_{SVD} \begin{pmatrix} \Sigma_{\tilde{\mathbf{f}}} & \mathbf{0}_{8,1} \\ \mathbf{0}_{1,8} & 0 \end{pmatrix} \mathbf{J}_{SVD}^T$$

Along the lines of the discussion in Section 4.3.1, the uncertainty of the transform may be used to map the covariance ellipse of a point in one image into the other. However, since the fundamental matrix defines a point-line mapping, the interpretation of this uncertainty is now altered. Specifically, given the fundamental matrix \mathbf{F} and point \mathbf{x} , we have the epipolar line $\mathbf{l} = \mathbf{F}\mathbf{x}$ corresponding to \mathbf{x} . The uncertainty of this line is represented in terms of a covariance matrix

$$\Sigma_{\mathbf{l}} = \mathbf{J}_F \Sigma_F \mathbf{J}_F^T + \sigma^2 \mathbf{J}_x \mathbf{J}_x^T \quad (4.6)$$

where \mathbf{J}_F and \mathbf{J}_x are the Jacobian of the point-line mapping with respect to \mathbf{F} and \mathbf{x} . This covariance matrix defines a conic C_k , which corresponds to a hyperbola in the second image, illustrated in Figure 4.4. As in the case of the homography, the validity of a match can be determined in terms of whether the point \mathbf{x}' lies within the mapped uncertainty area.

4.4 Incorporating uncertainty into RANSAC

Given the analysis described in the previous section, we now discuss how knowledge of the uncertainty of the estimation may be leveraged in RANSAC to obtain a more efficient algorithm.

4.4.1 Computing (uncertain) support:

The discussion in Sections 4.3.1 and 4.3.2 introduced the idea of characterizing the uncertainty of the estimated transformations, and indicated the advantage of using this uncertainty when computing the support, as compared to a fixed, precomputed threshold. To make this intuition more rigorous, we note that this uncertainty analysis has a simple probabilistic interpretation – in particular, in terms of the Mahalanobis distance (Mahalanobis, 1936).

As an example, consider the case of homography estimation, where we are given a set of matched points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, and are required to compute the homography \mathbf{H} . Given our noise model, a point \mathbf{x} in the first image is denoted as having mean $\bar{\mathbf{x}}$ and covariance matrix $\Lambda_{\mathbf{x}}$. The corresponding point \mathbf{x}' is similarly denoted as having mean $\bar{\mathbf{x}'}$ and covariance matrix $\Lambda_{\mathbf{x}'}$. As in Section 4.3.1, we can transfer the point from the first image to the second image via the estimated homography, in order to obtain the point $\hat{\mathbf{x}}' = \mathbf{H}\mathbf{x}$.

It can be seen that $\hat{\mathbf{x}}'$ is a point with mean $\overline{\mathbf{H}\bar{\mathbf{x}}}$ and covariance matrix $\Lambda_{\hat{\mathbf{x}'}}$, as defined in Equation 4.4. If we make the assumption that $\bar{\mathbf{x}'} = \overline{\mathbf{H}\bar{\mathbf{x}}}$, then it can be seen that the quantity $\mathbf{x}' - \mathbf{H}\mathbf{x}$, i.e., the distance between the detected point \mathbf{x}' and the transferred point $\mathbf{H}\mathbf{x}$, is a random variable with zero mean, and covariance matrix $\Lambda_{\mathbf{x}} + \Lambda_{\hat{\mathbf{x}'}}$. Defining $\mathbf{e} = \mathbf{x}' - \mathbf{H}\mathbf{x}$, this can be written in terms of the Mahalanobis distance d_M :

$$d_M(\mathbf{e}) = \left(\mathbf{e}^T (\Lambda_{\mathbf{x}} + \Lambda_{\hat{\mathbf{x}'}})^{-1} \mathbf{e} \right)^{\frac{1}{2}}, \quad (4.7)$$

which provides a measure of distance between the points, taking the various uncertainties into account. Given equation 4.7, and assuming that the random variables are Gaussian, the squared Mahalanobis distance follows a chi-square distribution. It thus follows, much like in RANSAC, that a fixed fraction α of inliers will satisfy the condition

$$\mathbf{e}^T (\Lambda_{\mathbf{x}} + \Lambda_{\hat{\mathbf{x}'}})^{-1} \mathbf{e} \leq k^2, \quad (4.8)$$

where k is the value of the inverse cumulative chi-square distribution corresponding to α . In other words, this represents a new criterion for identifying inliers, which takes point and model uncertainties into account.

While the above discussion describes a probabilistic interpretation, a simple geometric formulation may also be constructed, following (Ochoa and Belongie, 2006). Consider the transferred point $\hat{\mathbf{x}}' = \mathbf{H}\mathbf{x}$, with mean denoted by $\mu_{\hat{\mathbf{x}}'}$ and covariance $\Lambda_{\hat{\mathbf{x}}'}$. Given some probability α , and the corresponding value of k , the goal is to determine whether the measured point \mathbf{x}' lies within this bound. A simple geometric interpretation can be formed by noting that the set of equal likelihood points that satisfy

$$(\mathbf{x}' - \mu_{\hat{\mathbf{x}}'})^T \Lambda_{\hat{\mathbf{x}}'}^{-1} (\mathbf{x}' - \mu_{\hat{\mathbf{x}}'}) = k^2, \quad (4.9)$$

corresponds to a conic

$$\mathbf{C} = \left[\mu_{\hat{\mathbf{x}}'} \mu_{\hat{\mathbf{x}}'}^T - k^2 \Lambda_{\hat{\mathbf{x}}'} \right] \quad (4.10)$$

which for the case of the homography, represents an ellipse. The point \mathbf{x}' lies within this conic if $\mathbf{x}'^T \mathbf{C} \mathbf{x}'$ has the same sign as $\mu_{\hat{\mathbf{x}}'}^T \mathbf{C} \mu_{\hat{\mathbf{x}}'}$ (Ochoa and Belongie, 2006). Note that while this discussion has centered around the homography, an analogous argument holds for the case of the fundamental matrix, invoking the duality between lines and points. In other words, given an epipolar line $\hat{\mathbf{l}}'$, with mean $\mu_{\hat{\mathbf{l}}'}$ and covariance $\Sigma_{\hat{\mathbf{l}}'}$ (as computed from Equation 4.6), the conic that forms the envelope of lines is given by

$$\mathbf{C} = \left[\mu_{\hat{\mathbf{l}}'} \mu_{\hat{\mathbf{l}}'}^T - k^2 \Sigma_{\hat{\mathbf{l}}'} \right], \quad (4.11)$$

which in this case is a hyperbola, with symmetric branches around $\mu_{\hat{\mathbf{l}}'}$. A similar test holds to check whether the point \mathbf{x}' lies within the branches of the hyperbola.

The above discussion illustrates how knowledge of the model and point uncertainties can be used to compute the support for a given model. In particular, given a model hypothesis with associated covariance, the above tests can be applied to each correspondence to determine which points can be classified as inliers, if the model uncertainty is *explicitly* taken into account. In analogy with RANSAC, this process should result in a fraction α of the true inliers being recovered. However, note that it does not make sense to apply the uncertainty analysis to a sample that is contaminated, since the formulation described in Section 4.3 makes the assumption that the only source of error is

noise in the point locations, and does not account for outlying matches. We now address the issue of determining when to apply the covariance test within RANSAC.

4.4.2 Applying the test:

Given that we can compute a set of inliers for a particular hypothesized transform, we then address the issue of when this covariance check should be applied. As mentioned above, applying the test to every sample, contaminated as well as uncontaminated, is not meaningful, and would incur a significant computational overhead. Ideally, if we could identify a sample that had a good chance of being uncontaminated, we could then apply the covariance test to gather all potential inliers to this sample. For example, considering Figure 4.2, if we could identify first uncontaminated sample (i.e., Sample ❶), we could then apply the covariance test to identify all the inliers that support this model, taking the estimation uncertainty into account.

There have been approaches that address the issue of identifying these “non-random” model hypotheses. For instance, one way to address this problem is to estimate how likely it is that a configuration of points that form the support for a particular solution, could have occurred by chance. This is the idea behind the Minimum Probability of Randomness (MINPRAN) algorithm (Stewart, 1995) which estimates this probability by assuming that outlier residuals follow a uniform distribution. However, as noted in (Torr et al., 1998), applying this technique to the estimation of geometric entities such as the fundamental matrix is not straightforward. An alternate approach is to formulate the problem as a p-value test, as in (Chum and Matas, 2005). The probability of having i random inliers *by chance* for a dataset with N points is given by a binomial distribution

$$p_N(i) = \delta^{i-m}(1 - \delta)^{N-i} \binom{N - m}{i - m}, \quad (4.12)$$

where δ is the probability of a random point being consistent with a contaminated model. Consequently, the minimum size of “non-random” support can then be calculated as

$$I_N^{min} = \min\{j : \sum_{i=j}^N p_N(i) < \psi\}, \quad (4.13)$$

which represents a simple p-value test to determine the smallest support that has a probability of

randomness below ψ (set to 0.05 for a 5% significance test).

A Bayesian approach to this problem was provided in (Brown and Lowe, 2003). In particular, given a model hypothesis with I inliers, let h denote an indicator variable taking on values of 0 and 1, depending on whether the model hypothesis is bad (i.e., contaminated) or good (i.e., uncontaminated). We thus have

$$\begin{aligned}
 p(h = 1|I) &= \frac{p(I|h = 1)p(h = 1)}{p(I)} \\
 &= \frac{p(I|h = 1)p(h = 1)}{p(I|h = 1)p(h = 1) + p(I|h = 0)p(h = 0)} \\
 &= \frac{1}{1 + \frac{p(I|h=0)p(h=0)}{p(I|h=1)p(h=1)}}.
 \end{aligned} \tag{4.14}$$

A hypothesis can be considered to be non-random if $p(h = 1|I) > \psi$. From the denominator of Equation 4.14, we see that this quantity depends on the ratio of two values: the probability of having I inliers to a bad model, and the probability of having I inliers to a good model. If p_0 and p_1 denote the probability of a point being consistent with a bad and a good model respectively, then we assume the following two binomial distributions:

$$p(I|h = 0) = B(I; N, p_0) \tag{4.15}$$

$$p(I|h = 1) = B(I; N, p_1). \tag{4.16}$$

As an example, choosing $p_0 = 0.1$, $p_1 = 0.6$, $p(h = 1) = 10^{-6}$ and $\psi = 0.99$ gives the condition $I > 0.31N + 7.0$, which represents a simple check to identify consensus sets with non-random support.

4.4.3 Algorithm

With the above discussion in mind, we now describe a modified RANSAC algorithm that incorporates knowledge of the uncertainties into the estimation process. Algorithm 4 outlines the operation of the cov-RANSAC algorithm. Steps 1 and 2 proceed as in standard RANSAC – minimal subsets are randomly sampled from the data in order to compute model hypotheses, whose support is then verified on all data points. Note that the optimizations discussed in Section 3.2 can

Algorithm 4 RANSAC with uncertainty estimation: cov-RANSAC

Input: Set of N data points \mathcal{U} , maximum number of iterations k_{max} , noise variance σ^2

Set $k = 0$, repeat steps 1–4 while $k < k_{max}$

1. Hypothesis generation:

Sample a minimal subset of size m at random from \mathcal{U}

Set $k = k + 1$

Estimate model parameters from this minimal subset

2. Verification:

Evaluate the model and compute the support set \mathcal{I}_k

Check if support is non-random: $p(\text{model} = \text{good} | |\mathcal{I}_k|) > \psi$ (see Equations 4.13 and 4.14)

if model with non-random support found **then**

3. Covariance test and refinement

3a. Compute support accounting for model uncertainty

Compute least squares fit using inlier set \mathcal{I}_k

Compute the covariance Λ_T of the transform (Section 4.3)

Find set of inliers \mathcal{I}_{cov} using covariance tests (Section 4.4.1)

3b. Local refinement

Run K iterations of inner (non-minimal) RANSAC

Store the support set \mathcal{I}_{cov}^* corresponding to the best model found

4. Confidence in the solution:

Compute the reduced set $\mathcal{U} = \mathcal{U} \setminus \mathcal{I}_{cov}^*$

Compute number of samples k_{max} from Equation 2.4, using $\varepsilon = \frac{|\mathcal{I}_{cov}^*|}{(N - |\mathcal{I}_{cov}^*|)}$

Go to hypothesis generation step

end if

be applied here as well; for instance, prior information can be easily integrated into the sampling stage to achieve non-uniform sampling (Section 3.2.2), and hypothesis verification can be performed using a randomized verification strategy (Section 3.2.4). Following verification, we then check if the current model hypothesis has sufficiently large support, as outlined in Section 4.4.1. The goal of this step is to identify promising fits to the data, which can then be further refined.

There are two possible outcomes at stage: either the model is rejected as being random fit, or is accepted. In the first case, we move on to the next hypothesis. In the second case, we have a set of inliers \mathcal{I}_k , obtained using the standard RANSAC threshold. As we have seen (refer Figure 4.2), the use of this threshold results in a subset of the inliers being found, since we have not compensated for model uncertainty. To do so, we first compute a least squares fit to the set of inliers, and then compute the covariance of the resulting transformation (Step 3a). There are two main reasons for this: first, as noted in (Hartley and Zisserman, 2000: Sec. 5.2.6), if the covariance of the transform is estimated from a the minimal sample and used to transfer additional points (that were not used

in its computation), this may lead to errors. The reason for this is that extrapolation far beyond the set of points used to compute the transform may be unreliable. Secondly, as observed in (Criminisi et al., 1999; Hartley and Zisserman, 2000), the uncertainty of the model reduces with the number of points used in its computation. To handle both these issues, we use all available inliers at this stage, in order to compute an accurate estimate with reduced uncertainty. Following this, we can then carry out the covariance-based inlier tests as discussed in Section 4.4.1 to compute the support, denoted by \mathcal{I}_{cov} . This set contains all points that support the current (refined) model, *explicitly* taking the estimation uncertainty into account. Further refinement is possible as well – for instance, a local optimization may be carried out, similar to LO-RANSAC (Chum et al., 2003). A simple strategy (Step 3b) is to generate K non-minimal model hypotheses within an inner-RANSAC loop. The consensus set \mathcal{I}_{cov}^* corresponding to the best model in this local optimization loop is then stored.

In the standard LO-RANSAC algorithm, following the local optimization step, sampling then resumes as before, on the entire dataset. The reason is this done is twofold: (a) repeated sampling might provide a better starting point for the local optimization, thus improving the support slightly and (b) there might exist a different model in the data, with potentially larger support. It is due to these two reasons that sampling then continues on the entire dataset \mathcal{U} . However, an additional optimization is possible. More specifically, since we are explicitly compensating for estimation uncertainty, the set of inliers \mathcal{I}_{cov} contains, on average, a fraction α of the true inliers (recall that α is typically set to 0.95 or 0.99, and is used to compute the chi-square statistic in Equation 4.9). In other words, at this point, we have identified (almost) the entire set of inliers corresponding to some model in the data. Thus, if we are confident that there is only a single model in the data, note that we could potentially stop sampling at this stage, and simply return the best set of inliers found so far. However, we can make a small modification to the standard stopping criterion, in order to ensure that we have not missed a different model with potentially larger support. This is shown in Step 4 of Algorithm 4. Given the support set \mathcal{I}_{cov}^* , this provides a lower bound on number of inliers in the data. We can remove these points from the original dataset, yielding a reduced set $\mathcal{U} = \mathcal{U} \setminus \mathcal{I}_{cov}^*$. We then need to carry out a sufficient number of iterations to ensure, with confidence η_0 , that no model with support greater than $|\mathcal{I}_{cov}^*|$ exists within the remaining $(N - |\mathcal{I}_{cov}^*|)$ data points. The number of iterations can then be computed from the standard stopping criterion (Equation

2.4), with $\varepsilon = \frac{|\mathcal{I}_{cov}^*|}{(N-|\mathcal{I}_{cov}^*|)}$. This small modification in the stopping criterion allows cov-RANSAC to terminate in fewer iterations compared to LO-RANSAC.

4.5 Results

In this section, we evaluate the performance of cov-RANSAC on homography and fundamental matrix estimation. The performance of the approach was evaluated against the standard RANSAC algorithm and LO-RANSAC (Section 3.2.5.1). Note that for these experiments, we do not assume the existence of prior information regarding correctness of the correspondences, so we do not compare against techniques that specifically exploit this information. The assumed standard deviation for the measurement error was set to $\sigma = 0.60$, and the number of inner RANSAC repetitions was set to $K = 5$. For the standard LO-RANSAC, which uses iterative least squares with threshold tightening, we set the threshold multiplier to 2.0, and use 4 iterations.

4.5.1 Homography estimation

The results for homography estimation are tabulated in Table 4.1, which shows average numbers over 500 runs for inliers found (I), samples evaluated (k) and estimation error (error). The main observation to be made is that using the covariance information in RANSAC leads to a 4–5 fold reduction in the number of samples evaluated compared to standard RANSAC, and a 1.5–2 fold decrease compared to LO-RANSAC. In addition, the accuracy of the solution returned by cov-RANSAC, along with the size of the final support, is virtually identical to that of LO-RANSAC.

To see the effect of applying the covariance test within RANSAC, consider Figure 4.5. Figures 4.5(a) and 4.5(c) show two test images pairs. The green points overlaid in these figures represent inliers to some uncontaminated model hypothesis generated during the RANSAC sampling process. Due to the effects of noise, only a subset of inliers are recovered. We can now compute the covariance of the homography and use this to find the support set \mathcal{I}_{cov} , as detailed in Section 4.4.1 and Algorithm 4. This is illustrated in Figures 4.5(b) and 4.5(d), with inlier points overlaid on the right image of the corresponding image pair. The images also show the uncertainty ellipses for the case where points are transferred from the left image to the right, with the ellipses computed from Equation 4.10 with $\alpha = 0.95$. Note that by explicitly compensating for model uncertainty, a number


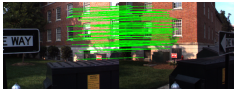


		RANSAC	LO-RANSAC	cov-RANSAC
A : $\varepsilon = 0.46, N = 2540$ 	<i>I</i> <i>k</i> error	994 ± 68 160 1.66 ± 0.29	1147 ± 7 74 1.12 ± 0.14	1148 ± 2 35 1.08 ± 0.07
B : $\varepsilon = 0.43, N = 1967$ 	<i>I</i> <i>k</i> error	732 ± 48 247 2.63 ± 0.46	823 ± 4 96 1.81 ± 0.10	822 ± 5 54 1.78 ± 0.08
C : $\varepsilon = 0.24, N = 454$ 	<i>I</i> <i>k</i> error	96 ± 6 2687 3.04 ± 0.80	104 ± 0 1001 1.94 ± 0.09	104 ± 0 618 1.99 ± 0.08
D : $\varepsilon = 0.34, N = 495$ 	<i>I</i> <i>k</i> error	151 ± 11 663 2.32 ± 0.47	168 ± 0 254 1.47 ± 0.01	168 ± 0 141 1.47 ± 0.01

Table 4.1: Comparison of the different RANSAC techniques for homography estimation on real data. The table shows, for each image pair and algorithm, the average number of inliers (I), samples evaluated (k) and estimation error (error). Note that cov-RANSAC achieves a reduction in the number of samples, while maintaining the same accuracy as LO-RANSAC.

of additional inliers are recovered, thus significantly increasing the support.

4.5.2 Fundamental matrix estimation

A similar performance comparison was performed for the task of estimating the epipolar geometry for image pairs. The results of the evaluation are shown in Table 4.5.2, which tabulates average numbers over 500 runs, for all techniques. The results again demonstrate the advantage of using the uncertainty information. It can be seen that the proposed technique leads to a 6–8 fold reduction in the number of samples evaluated compared to RANSAC, while again maintaining the same accuracy as LO-RANSAC. In addition, due to the modified termination criterion, cov-RANSAC requires approximately 1.5–3.0 times fewer samples than LO-RANSAC. These results again underscore the benefit of incorporating knowledge of the uncertainty of the estimation process into a RANSAC framework.

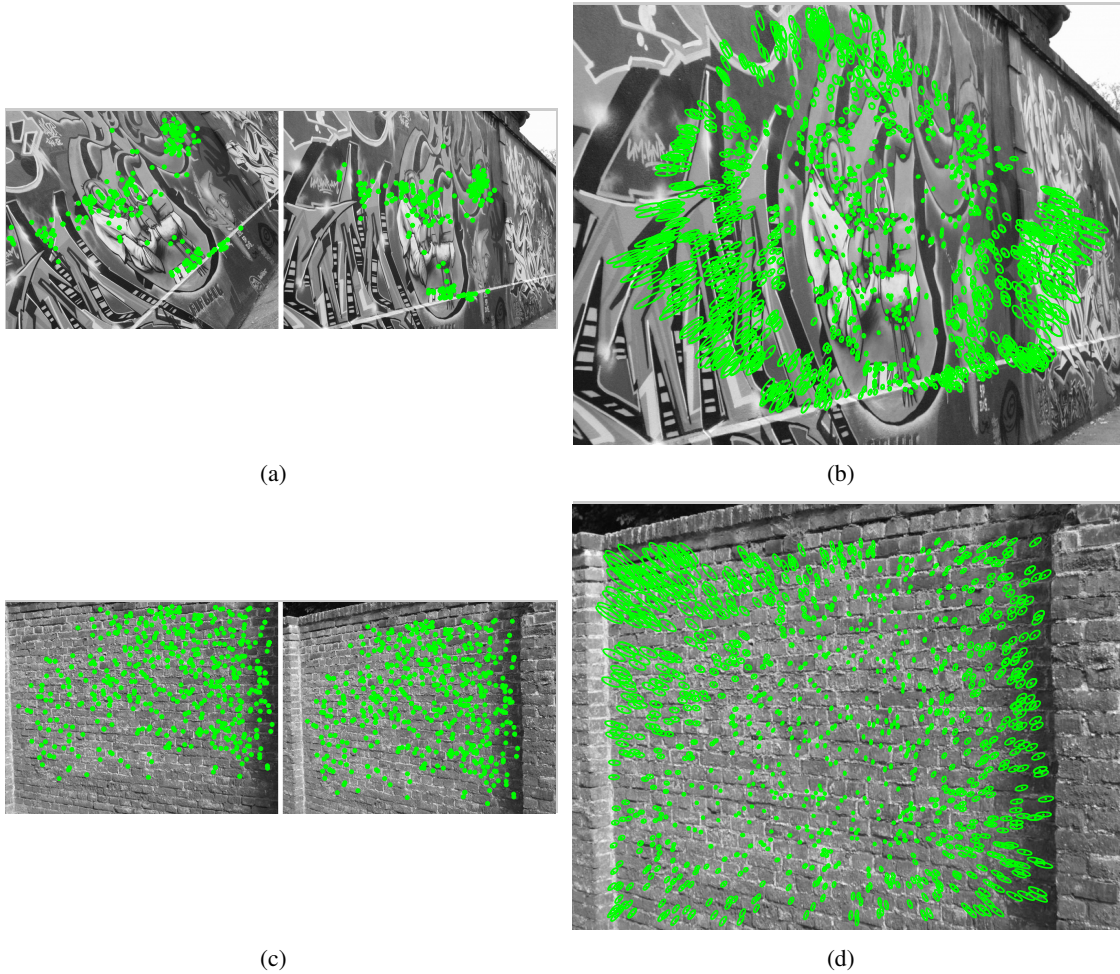


Figure 4.5: Example results showing the use of the covariance test in computing the support for homography estimation. Figures (a), (c) show the images pairs used. The green points overlaid in Figures (a), (c) represent inliers to an uncontaminated model hypothesis sampled in RANSAC. Due to the effects of noise, only a subset of inliers are recovered. Figures (b), (d) show the results of applying the covariance-based test to find inliers, overlaid on the right image of the corresponding image pair. The images also show the uncertainty ellipses corresponding to each point. Note that by explicitly compensating for model uncertainty, a number of additional inliers are recovered.

4.6 Discussion

In this chapter, we demonstrated how uncertainty information may be incorporated into a random sampling framework, in order to robustly estimate the support for a model generated from noisy data points. The resulting algorithm, cov-RANSAC, achieves a significant reduction in the number of samples compared to standard RANSAC, and provides a modest improvement over LO-RANSAC.

While the framework we describe is flexible, and can be combined with other techniques (for





		RANSAC	LO-RANSAC	cov-RANSAC
A : $\varepsilon = 0.57, N = 575$ 	<i>I</i> <i>k</i> error	315±12 440 0.71±0.18	328±1 175 0.45±0.04	328±1 58 0.43±0.05
B : $\varepsilon = 0.38, N = 1088$ 	<i>I</i> <i>k</i> error	381±13 7806 1.79±1.27	402±4 2951 1.07±0.53	404±2 1112 0.88±0.28
C : $\varepsilon = 0.45, N = 904$ 	<i>I</i> <i>k</i> error	388±10 2315 1.92±1.14	403±5 818 1.04±0.74	402±5 299 0.99±0.83
D : $\varepsilon = 0.38, N = 346$ 	<i>I</i> <i>k</i> error	118±6 9745 1.75±0.77	128±2 3538 1.58±0.35	128±1 1278 1.58±0.29

Table 4.2: Comparison of the different RANSAC techniques for fundamental matrix estimation on real data. The table shows, for each image pair and algorithm, the average number of inliers (I), samples evaluated (k) and estimation error (error).

instance, randomized model verification), there are some limitations. For instance, note from Algorithm 4, and the discussion in Section 4.4.3, that we use non-minimal samples while computing the model covariance, in order to reduce uncertainty in the parameter estimates. While this proves to be effective in practice, there is still the issue of degenerate data configurations. For instance, considering the simple case of a line, if the data points used to estimate the line are spatially very close together, the corresponding uncertainty in the parameter estimates may be high, which in turn will lead to a large support in cov-RANSAC. It is possible to mitigate this effect by placing some constraints on the covariance matrix – for instance, a threshold on the largest eigenvalue of the covariance matrix Σ_T . However, this may prove to be difficult to set in general, and thus, cov-RANSAC requires other safeguards against these degenerate configurations.

In this work, we assumed a common, isotropic covariance matrix for each feature point, in order to simplify the discussion. We note that there exist methods to more precisely model the location

uncertainty for common feature detectors. For instance, (Zeisl et al., 2009) describes a method to compute anisotropic covariance matrices for SIFT and SURF feature points. Integrating this into cov-RANSAC, and analysing the performance, both in terms of accuracy, as well as computational overhead, would be an interesting direction for future work.

Chapter 5

Scale Adaptive Robust Estimation via Residual Consensus

5.1 Introduction

At this stage, we turn our attention to the issue of the inlier threshold parameter in RANSAC. As we have seen, since models in RANSAC are scored based on their support, it is precisely this parameter that allows RANSAC to distinguish between good and bad model fits. Indeed, the cost function (Section 2.2.1) in RANSAC may be viewed as being equivalent to finding the location of the densest band of data points, with width specified by the user. While it may indeed be possible to often “guess” a reasonable threshold value, there are applications, such as the alignment of 3D reconstructions with arbitrary scale (see Figure 5.1), where this choice is less obvious. In addition, it has been shown that if the chosen threshold deviates significantly from the true value, RANSAC can provide biased, or simply incorrect results (Choi and Medioni, 2009).

In this chapter, we present a new robust estimation framework that is agnostic to the threshold parameter. Instead, the algorithm we present is based on a simple observation: that the residual errors for “good” models are in some way consistent with each other. We show that it is possible to efficiently identify this stable behaviour by exploiting residual ordering information coupled with simple non-parametric statistical tests. This leads to an very simple, yet effective, algorithm whose performance over a range of noise levels and inlier ratios is comparable to that of RANSAC, whilst

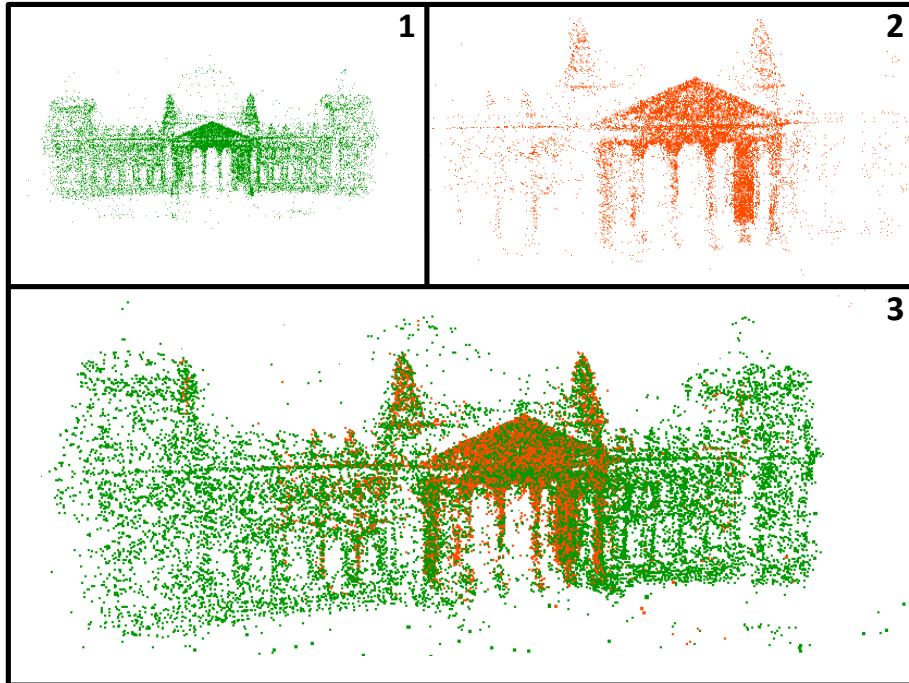


Figure 5.1: (1-2) Two sparse 3D submodels obtained from a structure-from-motion system. The two reconstructions are to be merged into a common reference frame via a 3D similarity transformation. Since the scale of each reconstruction is arbitrary, a good choice of threshold is not obvious. (3) Merged reconstruction (best viewed in colour) obtained using our method, requiring no prior knowledge of noise scale or inlier ratio.

completely eliminating the inlier threshold parameter.

To summarize the main contributions of this chapter: (1) we develop a new strategy for robust estimation, based on the behaviour of data point residuals. (2) the method does not require an inlier threshold, or a worst-case assumption about the level of data contamination; this is a strongly distinguishing characteristic compared to current techniques, which require at least one of these assumptions to be made. (3) the effectiveness of the algorithm is demonstrated on a variety of challenging estimation problems covering a range of inlier ratios, noise levels, and model complexities.

5.2 Related Work

Since the introduction of the original RANSAC algorithm, a number of popular variants have emerged, many of which were discussed in Section 3.2. Orthogonal to these advances, there have also been attempts to develop robust estimators capable of operating without a threshold parameter.

Below, we discuss some of the important work in this area.

Since scoring models based on their support, as in RANSAC, requires a threshold, one possible approach is to optimize a different cost function. Least Median of Squares (LMedS) (Rousseeuw, 1984), scores models based on their median error residual, returning the model with lowest score. However, this breaks down when the data consists of more than 50% outliers, since the median residual is no longer informative. While it is, in principle, possible to replace the median by a different order statistic, this requires *a priori* knowledge of the level of data contamination. MINPRAN (Stewart, 1995) builds on the assumption that outliers are uniformly distributed within some range, and then searches for a combination of model parameters and inliers that are least likely to have occurred by chance; however, this assumes knowledge of the dynamic range of the outlier data.

A number of closely linked techniques have been proposed to address the problem of multi-model fitting without prior scale, primarily for segmenting multiple structures from range data (Miller and Stewart, 1996; Lee et al., 1998; Wang and Suter, 2004). MUSE (Miller and Stewart, 1996) and ALKS (Lee et al., 1998) are related techniques that minimize the k^{th} order statistics of the squared residuals. It has been noted (Wang and Suter, 2004) that their ability to handle large outlier ratios is limited. ASSC (Wang and Suter, 2004) uses a mean-shift algorithm to detect modes in the residual distribution, using this to estimate the noise scale. However, this can be computationally expensive, particularly as the model complexity increases (Zhang and Kosecka, 2006). It is worth noting that while many of the above techniques have been evaluated for the range data problem, it has been observed (Torr and Murray, 1997) that extending them to other estimation problems, such as multi-view geometric relations, can be challenging. The recently proposed StaRSaC algorithm (Choi and Medioni, 2009) relaxes the requirement of a fixed threshold, by exhaustively testing all choices in some range. For each of T thresholds, RANSAC is executed K times, and a value is chosen that shows minimum “variance of parameters”. Even for reasonable values of T and K , this quickly becomes computationally infeasible (e.g., testing $T = 30$ thresholds with $K = 30$ requires almost three orders of magnitude more computation than RANSAC).

Two recent approaches that are similar in spirit to ours are those of (Zhang and Kosecka, 2006) and Kernel-Fitting (Chin et al., 2009b). Both these works use the observation that for a given point, the distribution of residuals with respect to a large set of randomly generated models can be

used to reveal whether the point is an outlier or an inlier. In particular, the kernel-based measure introduced in (Chin et al., 2009b) has been shown to yield good results for the problem of fitting multiple structures to noisy data (Chin et al., 2009a; Chin et al., 2010). However, one limitation of these strategies is their implicit reliance on sampling a “sufficient” number of uncontaminated models. It is worth noting that models that are contaminated carry no reliable information about the validity of a data point not contained in the minimal sample itself, since the error of any inlier data point to a bad model is arbitrary. Thus, it becomes imperative to sample enough models for the distribution of residuals to be sufficiently “informative”. We take a different view: instead of the residual distributions of *points*, we observe the residual distributions of *models*. The intuition here is that two uncontaminated models will be “alike” in some way, while contaminated models will disagree in an unstructured way. This results in a simpler, yet much more efficient algorithm. A more recent extension of the ideas in (Chin et al., 2009b; Chin et al., 2009a; Chin et al., 2010) is presented in (Wong et al., 2011), which combines two ideas: sorting models based on their distances to data points (as in (Chin et al., 2009b)), as well as sorting data points based on their distances to models (which is similar to the approach we take). While the method we propose shares a very similar idea, the goal is different: in our case, we focus on the case of single model robust estimation, with the goal being to efficiently compute the model parameters without requiring any data or model specific thresholds. In contrast, the goal in (Wong et al., 2011) is multi-model robust estimation, and the sorting is used to speed up the stage of hypothesis generation. However, there is still an implicit reliance on a user defined parameter; namely, the inlier ratio in the data.

Finally, we note that one fundamental advantage of our technique compared to the methods discussed in this section (Rousseeuw, 1984; Stewart, 1995; Miller and Stewart, 1996; Lee et al., 1998; Wang and Suter, 2004; Zhang and Kosecka, 2006; Chin et al., 2009b; Chin et al., 2009a; Chin et al., 2010; Wong et al., 2011) is the ability to adaptively stop evaluation. In standard RANSAC, the current best inlier count can be used to adaptively set the maximum number of trials. However, in the absence of a threshold, current techniques either require knowledge of the inlier ratio, or must run the algorithm under a worst-case assumption—in other words, drawing enough samples to guarantee success for the *worst* inlier ratio that could potentially occur in the data. RECON does not have this limitation; we are able to adapt “on-the-fly” to the level of data contamination, without

requiring any data-specific parameter tuning. We note that this advantage is a consequence of the fact that, much like RANSAC, we address the specific problem of single model robust estimation, and develop an efficient threshold-free framework for this problem. While the single-model case indeed arises in a number of practical scenarios of interest, it is worth noting that there are certain applications (e.g., motion segmentation), for which the proposed method is not an ideal fit.

5.3 Approach

5.3.1 Residual Consensus

The key idea behind RANSAC is that a model generated from an uncontaminated minimal sample (i.e., containing only inliers) will have larger support than a contaminated model (generated from samples containing at least one outlier). Provided enough random samples have been drawn to guarantee, with some confidence, that at least one of the minimal samples is free from outliers, RANSAC will terminate successfully, returning the corresponding model with highest support. Observe that RANSAC uses the threshold to essentially distinguish between good and bad model estimates. In the absence of a threshold, however, it becomes harder to make this distinction. Our technique is based on the intuition that, near the true threshold value, uncontaminated models capture a stable set of points (i.e., the set of inliers). On the other hand, contaminated models do not demonstrate this consistent behaviour. In other words, looking at the relationship between *subsets* of models can reveal useful information about the validity of the models themselves. To formalize this intuition, we now consider more closely the behaviour of uncontaminated vs. contaminated models.

5.3.1.1 Uncontaminated models

Given an uncontaminated model and access to the true noise variance, it is possible to compute a region containing a fixed fraction α of the total number of inliers (Hartley and Zisserman, 2000; Raguram et al., 2009). To see why this is the case, recall the discussion in Section 2.2.4; under the typical assumption that the measurement error for inliers is Gaussian, with zero mean and variance σ^2 , the squared point-model errors between an inlier and the “true” model can be represented as a sum of n squared Gaussian variables, where n is the codimension of the model – i.e., a chi-

square distribution with n degrees of freedom. Thus, in order to recover a fraction α of inliers, an appropriate threshold t can be computed as

$$t^2 = \chi_n^{-1}(\alpha)\sigma^2, \quad (5.1)$$

where $\chi_n^{-1}(\cdot)$ is the inverse cumulative chi-square distribution. Let our estimate of the (*a priori* unknown) variance be $\hat{\sigma}^2$. If our estimate $\hat{\sigma}^2$ is close to the true σ^2 , then using the threshold t for an uncontaminated model will result in a fraction α (usually set to 0.95 or 0.99) of the inliers being found. For the case of line fitting, this is shown in Fig. 5.2(a).

Note, however, that to be strictly correct, one must take the uncertainty of the estimated models into account, as discussed in Chapter 4. To recap: since models are generated from noisy data points, this error propagates to the model itself. As a consequence, when using the “ideal” threshold, uncontaminated models do not always capture a fixed fraction α of inliers. However, if we characterize the uncertainty of a model \mathbf{M} via its covariance matrix $\Sigma_{\mathbf{M}}$ and explicitly perform error propagation, this can be remedied (Raguram et al., 2009). For instance, when fitting a line, we can derive an expression for the covariance matrix Σ_L of the line (Meidow et al., 2009b). To capture a fraction α of true inliers, we now need to consider the α -percentile error envelope defined by Σ_L , which is in the form of a hyperbola (Fig. 5.2(b)).

From the discussion above, given that a fraction α of the inliers can always be recovered for an uncontaminated model, now consider *a pair* of uncontaminated models. It can be seen that the expected fraction of total inliers in the intersection of the two error envelopes is given by α^2 . For typical values of α close to 1.0, note that this expected fraction is close to α , implying that uncontaminated models capture a stable set of points with respect to each other (Fig. 5.2(c)). In other words, if $\hat{\sigma}^2 \approx \sigma^2$, uncontaminated models will have a fraction $\approx \alpha$ of their inliers in common.

5.3.1.2 Contaminated models

Assuming unstructured outliers, observe that models generated from contaminated subsets have arbitrary parameters. Given our estimate of the noise variance $\hat{\sigma}^2$, we can break down the behaviour of contaminated models into two scenarios, based on the relationship between the estimated $\hat{\sigma}^2$, and the true σ^2 .

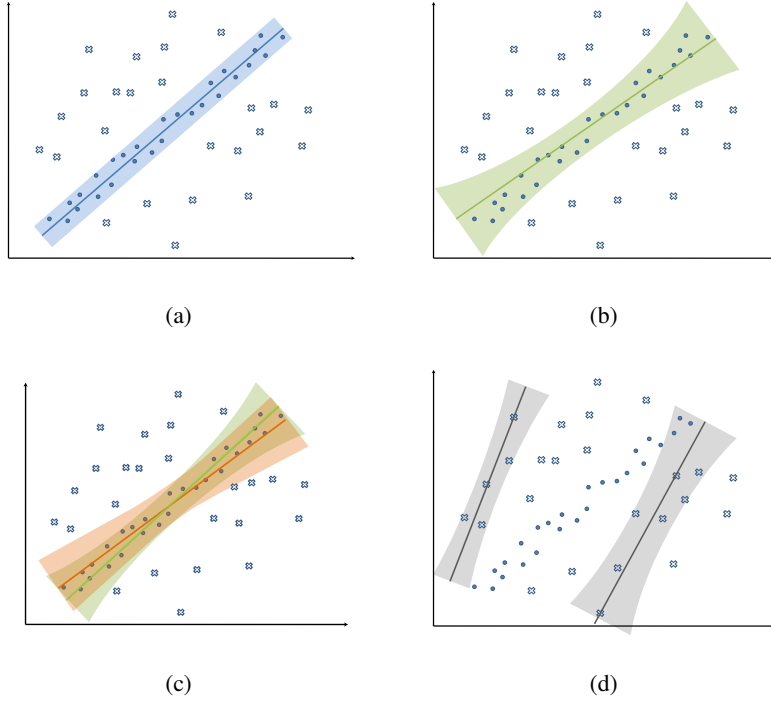


Figure 5.2: (a) Error bound that captures $\alpha = 99\%$ of true inliers with respect to a noise-free, uncontaminated model. Using the inverse chi-square cdf, this gives $t = 2.57\sigma$. (b) Error envelope, following model covariance estimation, capturing $\alpha = 99\%$ of the inliers for a model generated from noisy inliers. (c) Overlap between two uncontaminated models, with a fraction $\approx \alpha$ of their inliers in common (d) Error envelopes for contaminated models, capturing random data points. (Figure best viewed in colour)

(1) $\hat{\sigma}^2 \leq \sigma^2$: First, assume $\hat{\sigma}^2 = \sigma^2$. As before, we can use the true noise variance to compute α -percentile envelopes, but they now capture essentially arbitrary data points (Fig. 5.2(d)). In particular, the probability of a pair of contaminated models having significant overlap in their inlier sets tends to zero, since this would amount to picking essentially the same minimal sample twice. To see this, assume that we have N data points, with I inliers. The probability of picking an uncontaminated minimal sample of size m is given by

$$P_{good} = \prod_{i=0}^{m-1} \frac{I-i}{N-i} \approx (I/N)^m \quad (5.2)$$

Now, assume we have generated a contaminated model. Picking a second model that happens by

chance to be consistent with the first is governed by the probability

$$P_{bad} = \prod_{i=0}^{m-1} \frac{I' - i}{N - i} \quad (5.3)$$

where I' is the number of points supporting the first “bad” model. Assuming unstructured outliers, we have $I' \ll I$, implying that $P_{bad} \ll P_{good}$. Note that exactly this same reasoning holds for the case of $\hat{\sigma}^2 < \sigma^2$.

(2) $\hat{\sigma}^2 > \sigma^2$: The extent of the α -percentile envelopes is governed by $\hat{\sigma}^2$. In particular, for $\hat{\sigma}^2 \gg \sigma^2$, the computed error envelopes might become so large that they include all data points. This leads trivially to a large overlap for any pair of models, uncontaminated or otherwise, since all points fall in the intersection. If no prior is available on the potential values of $\hat{\sigma}$, we must use additional constraints to determine whether we have an overestimate of σ . We do so by leveraging knowledge of the distribution of residuals.

Given two models \mathbf{M}_1 and \mathbf{M}_2 , assume that there is significant overlap in their inlier sets. Assuming that the size of this overlap is n , we have two residual vectors:

$$\begin{aligned} R^1 &= \{r_1^1, r_2^1, \dots, r_n^1\} \\ R^2 &= \{r_1^2, r_2^2, \dots, r_n^2\}, \end{aligned} \quad (5.4)$$

measured from each model to the n points. Under typical assumptions, we know that the distribution of inlier residuals from an uncontaminated model approximates a chi-square distribution. In other words, if \mathbf{M}_1 and \mathbf{M}_2 are uncontaminated and $\hat{\sigma} \approx \sigma$, the vectors R^1 and R^2 can be thought of as two independent random samples drawn from a *common* underlying distribution. On the other hand, when one or more of the models is contaminated and $\hat{\sigma} \gg \sigma$, the intersection will contain an *arbitrary combination of inliers and outliers*, and it is unlikely that a single underlying distribution could have generated both sets of residuals. Thus, a statistical test that evaluates the similarity between residual distributions can reveal useful information about the nature of the models. The Kolmogorov-Smirnov (KS) (Chakravarti et al., 1967) and Anderson-Darling (AD) (Stephens, 1974) tests are statistical tools that may be used to compare a sample with a reference distribution or to

compare two empirical samples. The KS test computes the statistic

$$D_{n,n} = \sqrt{\frac{n}{2}} \sup_x |F_{1,n}(x) - F_{2,n}(x)|, \quad (5.5)$$

where n is the number of points, $F_{1,n}(x)$ is an empirical distribution function and $F_{2,n}(x)$ is either a reference distribution (one-sample KS test) or a second empirical distribution function (two-sample KS test). We wish to test the null hypothesis H_0 that the two samples R^1 and R^2 are from the same underlying distribution (note that this is a non-parametric test, so the exact form of this common distribution does not have to be specified). H_0 is rejected at a level β (typically 1% or 5%) if $D_{n,n} > C_\beta$, where C_β is the critical value of the Kolmogorov distribution (for details, refer (Chakravarti et al., 1967)). Simply put, if the residuals offer strong evidence against the null hypothesis, the models can be discarded as invalid.

Summarizing the discussion thus far, we now have two constraints.

1. At the true σ , a pair of uncontaminated models will have a fraction $\approx \alpha^2$ of their inliers in common. Conversely, the probability of a pair of contaminated models having this level of overlap is very small.
2. The inlier residuals for a pair of uncontaminated models should correspond to the same underlying distribution

We refer to these two conditions together as α -consistency, and we can now develop a simple algorithm that exploits this behaviour. In brief: iterate over possible values of σ , and pick the lowest σ that gives rise to a set of α -consistent models (i.e., models with significant overlap, and whose residuals are statistically likely to correspond to the same distribution). The algorithm is outlined in Algorithm 5.

While our initial experiments have verified the correctness of this approach, one significant limitation is that it is computationally expensive. The algorithm requires us to examine all possible σ values and to compute the overlap in support for each. In the context of this limitation, we now present a practical algorithm that efficiently exploits *residual ordering* information to achieve the same results.

Algorithm 5 Iterative robust estimation via α -consistency

Repeat 1-3 until K mutually α -consistent models found

1. Hypothesis generation:

Hypothesize model \mathbf{M}_i from random minimal sample

2. Model verification:

Compute and store the residuals of \mathbf{M}_i to all data points

3. Test α -consistency

for all $\hat{\sigma}$ in range $(\sigma_{min} \dots \sigma_{max})$ **do**

for all prior models \mathbf{M}_j in $\{\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_{i-1}\}$ **do**

 Use $\hat{\sigma}$ to compute overlap between the inlier sets for \mathbf{M}_i and \mathbf{M}_j

end for

if set of K pairwise consistent models found **then**

if inlier residuals pass the KS statistical test **then**

 Return $\hat{\sigma}$, models and set of inliers

end if

end if

end for

5.3.2 Efficient Residual Consensus

Given a model \mathbf{M}_i , consider its residuals to all N data points, $R^i = \{r_1^i, r_2^i, \dots, r_N^i\}$. Assume that for each model, we sort the residuals to be in increasing order, and retain the indices of the sorted points. In other words, we have a set $S^i = \{\lambda_1^i, \lambda_2^i, \dots, \lambda_N^i\}$, representing a permutation of the indices $1, 2, \dots, N$ such that $r_{\lambda_1^i}^i \leq r_{\lambda_2^i}^i \leq \dots \leq r_{\lambda_N^i}^i$. We define the partial overlap set $\Theta_n^{i,j}$ as

$$\Theta_n^{i,j} = S_{1:n}^i \cap S_{1:n}^j \quad (5.6)$$

where i and j denote two models \mathbf{M}_i and \mathbf{M}_j and $S_{1:n}^i$ is the subset containing the first n points of S^i . We also define the *normalized partial overlap*, $\theta_n^{i,j}$, to be the scalar quantity

$$\theta_n^{i,j} = \frac{1}{n} |\Theta_n^{i,j}| \quad (5.7)$$

This represents the number of common elements when considering the first n points, normalized by the subset size. We finally define the expected fraction of common points when considering a subset of size n , E_{θ_n} ,

$$E_{\theta_n} = E[\theta_n^{i,j}] = \frac{1}{n} \sum_{k=0}^n k p(|\Theta_n^{i,j}| = k) \quad (5.8)$$

One way to think about the index sets $S_{1:n}^i$ is the following. By choosing a subset of size n , we are implicitly defining a threshold t_n^i (and thus, some $\hat{\sigma}$), and then selecting from the vector R^i , those points with $r^i < t_n^i$. The approach we take is based on the intuition that for uncontaminated models, inlier indices should loosely group together at the front of set S^i . On the other hand, contaminated models should represent random permutations of the point indices. We shall now look more closely at this behaviour.

1. Uncontaminated models: Given sorted index sets S^i and S^j for a pair of uncontaminated models, it is evident that inlier indices should appear towards the front of these sets (by definition, inliers are closer to uncontaminated models than outliers). Note that due to model noise, the *relative ordering* of the inlier indices is arbitrary. However, as we consider progressively larger subsets $S_{1:n}^i$ and $S_{1:n}^j$, this defines implicit thresholds t_n^i and t_n^j . From Section 3.1, we know that when both t_n^i and t_n^j are near the true threshold, the subsets $S_{1:n}^i$ and $S_{1:n}^j$ each contain a significant fraction α of the inliers. In other words, the normalized overlap $\theta_n^{i,j}$ should approach α^2 near the true threshold.

2. Contaminated models: Given a pair of contaminated models with arbitrary parameters, we do not expect to see consistent structure when inspecting their residuals. In other words, after sorting by residuals, the sets S^i and S^j for a pair of bad models are likely to be random permutations of the point indices $1, 2, \dots, N$. The probability that the subset $S_{1:n}^i$ for model \mathbf{M}_i , has k elements *by chance* in common with the subset $S_{1:n}^j$ can be derived as:

$$p(|\Theta_n^{i,j}| = k) = \binom{n}{k} \binom{N-n}{n-k} / \binom{N}{n} \quad (5.9)$$

Substituting into Eqn (5.8), this results in

$$E_{\theta_n} = n/N \quad (5.10)$$

In other words, the normalized overlap between two contaminated models approaches α^2 only as $n \approx N$. This poses a question: how do we distinguish this case from that of two good models where the inlier ratio is close to 100%? Some consideration will reveal that this ambiguity corresponds

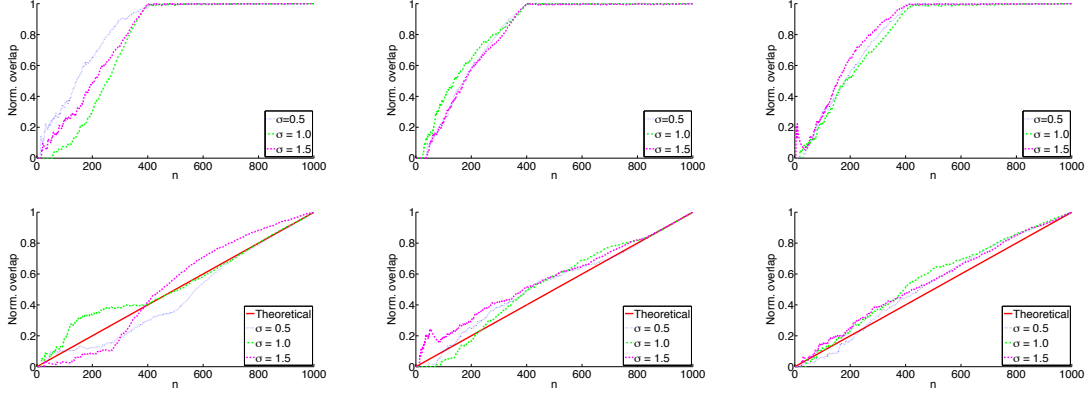


Figure 5.3: (Left-Middle-Right): 2D-line, homography, 3D similarity transform estimation. (Top row): Normalized overlap $\theta_n^{i,j}$, vs. subset size n for pairs of uncontaminated models. Note that close to the true inlier ratio (40%), $\theta_n^{i,j} \approx \alpha^2$. (Bottom Row): Normalized overlap $\theta_n^{i,j}$ vs. n for random pairs of contaminated models. Also overlaid is the predicted value from Eqn. (5.10). Note that $\theta_n^{i,j} \approx \alpha^2$ as $n \approx N$.

exactly to the discussion in Sec. 3.1.2, of the case $\hat{\sigma} \gg \sigma$ where the error envelopes include essentially all data points. Note that given a set of inlier residuals $\{r_1^i, r_2^i, \dots, r_n^i\}$, a robust estimate of the noise variance may be easily computed (Miller and Stewart, 1996; Torr and Murray, 1997) as

$$\hat{\sigma} = C \sqrt{\text{median}\{r_m^i{}^2\}}, \text{ where } m = 1, \dots, n \quad (5.11)$$

and C is a constant that makes $\hat{\sigma}$ unbiased for a particular target distribution of residuals. For instance, $C = 1.4286(1 + 5/(n - m))$ for Gaussian distributed residuals, where m is the minimal sample size. When a pair of bad models is found to have large overlap, the corresponding scale estimate from Eqn. (5.11) is typically dramatically larger than the true value. Thus, if a loose estimate (even up to an order of magnitude) of the maximum possible scale is available, it can be used to quickly discard bad model pairs. Alternatively, the same KS statistical test from Section 3.1 can be used to determine the validity of the residuals. Note that all the above arguments hold when one of the two models is bad; the probability of a random model producing the same sorted indices as a good model is also given by Eqn. (5.9).

To empirically verify our analysis of the behaviour of contaminated vs. uncontaminated models, we conducted the following experiments. Synthetic data was generated for three model-fitting examples: line, homography and 3D similarity estimation. The inlier ratio in all cases was fixed

at 40% and $N = 1000$ points were generated. We then plotted the normalized partial overlap $\theta_n^{i,j}$, between random pairs of good models and bad models (Fig. 5.3). The following observations are pertinent:

1. For pairs of uncontaminated models (top row in Fig. 5.3), we observe that when $n \approx I$ (close to the true inlier ratio of 40% in these experiments), the value of $\theta_n^{i,j}$ approaches α^2 . This supports our observation regarding α -consistency for uncontaminated models.
2. For pairs of contaminated models (bottom row in Fig. 5.3), it can be seen that there is close agreement between the theoretical and obtained plots. In particular, $\theta_n^{i,j} \approx \alpha^2$ only for $n \approx N$, as predicted in Eqn (5.10). This validates the argument that the residuals for contaminated models are unstructured.

5.3.3 RECON: Algorithm Description

The details of the Residual Consensus (RECON) algorithm are presented in Algorithm 6.

Algorithm 6 RECON: REsidual CONsensus

Repeat 1-3 until K consistent models found

1. Hypothesis generation:

Hypothesize model \mathbf{M}_i from random minimal sample

2. Model verification:

Sort residuals, store sorted index set S^i and residual set

3. Test α -consistency

for all prior models \mathbf{M}_j in $\{\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_{i-1}\}$ **do**

if $\mathbf{M}_i, \mathbf{M}_j$ have $\theta_n^{i,j} \geq \alpha^2$ for some n **then**

if $\hat{\sigma} < \sigma_{max}$ (Eqn. 5.11) or KS test passed (Eqn. 5.5) **then**

 Add to set of α -consistent models

if K mutually consistent models found **then**

 Return set of K models and inliers

end if

end if

end for

4. Refine to obtain final solution

Sort inliers by mean of their residuals to the K models

Generate M non-minimal models using PROSAC-based sampling of the sorted inlier set

Recompute pairwise $\theta_n^{i,j}$, select minimum n that satisfies $\theta_n^{i,j} \geq \alpha^2$ for $i, j \in 1, \dots, M$.

Re-estimate final model using the best inlier set.

As in Algorithm 5, note that in practice, we look for K pairwise α -consistent models to provide

robustness to structure in the outliers. In all our experiments, setting $K = 3$ was sufficient in practice. From a computational viewpoint, once enough samples have been drawn to rule out high inlier ratios (say, $\varepsilon > 90\%$), the residual test can be bypassed. This is because a bad model pair has $\theta_n^{i,j} \approx \alpha^2$ only for $n \approx N$ (Eqn. 5.10), which implies that the resulting inlier ratio is close to 100%. If enough models are drawn to rule out this case, α -consistent models with large n can be directly excluded. An additional speedup can be obtained by discarding models that violate certain basic checks (such as oriented constraints, for geometry estimation). As noted in (Chum et al., 2004), this can result in upwards of 70-80% of the models being discarded. This yields a much smaller core set of models that needs to be checked (Step 3 in Algorithm 6).

For clarity, we have thus far deferred the issue that a small number of outliers may satisfy our (implicit) threshold constraints, and be included in the overlap sets. In addition, the index sets $S_{1:n}$ can be viewed as an approximation to the covariance envelopes in Section 5.3.1. The practical implication of these two points is that we sometimes overestimate the inlier set. In other words, when $\theta_n^{i,j} \approx \alpha^2$, we sometimes have $n > I$, implying that we have included additional outliers. Observe that the source of this problem is noise in the minimal sample. To mitigate this, we generate M non-minimal samples from the returned inlier set, using PROSAC-based non-uniform sampling (Step 4 in Algorithm 6). Reestimating the overlap should now result in $n \approx I$, since we are explicitly compensating for model noise. As will be shown in Sec. 5.4, this results in the final set of inliers being very close to the ground truth.

5.4 Experiments

In this section, we evaluate RECON’s performance on a number of estimation problems. We stress that RECON does not require an inlier threshold, or any model-specific parameter tuning. The only required parameters are α and K , which are set to 0.99 and 3 respectively, for all experiments. Finally, note that RECON is able to adapt to the level of data contamination, and does not require the generation of an *a priori* fixed number of hypotheses.

		Line		Plane		H		F	
		k	err	k	err	k	err	k	err
$\varepsilon = 0.6$ $\sigma = 3.0$	RANSAC _F	106.8	1.38	137.1	1.55	218.3	3.33	266.2	3.58
	LMedS _F	11	1.24	23	1.38	47	2.9	382	2.98
	RANSAC _T	23.1	1.23	31.5	1.41	66.4	2.97	271.1	3.1
	LMedS _T	11	1.24	23	1.38	47	2.9	382	2.98
	RECON	29.6	1.2	38.2	1.35	49.4	2.91	133.1	3.04
$\varepsilon = 0.4$ $\sigma = 2.0$	RANSAC _F	176.2	0.91	389.5	1.2	489.2	3.17	3544.4	2.98
	LMedS _F	11	0.87	23	1.41	47	9.31	382	12.21
	RANSAC _T	56.4	0.7	99.7	0.88	244.2	2.05	2431.5	2.28
	LMedS _T	47	0.72	191	0.8	766	2	49082	1.88
	RECON	46.9	0.73	72.5	0.89	166.9	2.04	1879.3	2.34
$\varepsilon = 0.3$ $\sigma = 2.0$	RANSAC _F	289.8	1.08	629.2	1.36	1101.8	3.18	23450.5	3.26
	LMedS _F	11	2.01	23	5.72	47	17.2	382	25.33
	RANSAC _T	79.7	0.7	212	0.82	891.2	2.55	30153.5	2.42
	LMedS _T	47	0.77	191	0.74	766	2.51	49082	2.13
	RECON	58.3	0.75	149.5	0.76	490	2.56	13751.4	2.44
$\varepsilon = 0.3$ $\sigma = 4.0$	RANSAC _F	344.6	1.62	702.3	2.49	1798.3	6.03	94303.3	5.39
	LMedS _F	11	2.85	23	6.12	47	24.4	382	46.45
	RANSAC _T	115.1	0.96	291.8	1.67	1373.4	4.61	35983.9	4.06
	LMedS _T	47	1.03	191	1.65	766	4.48	49082	3.61
	RECON	77.2	0.97	172.4	1.72	612.3	4.47	13894.4	3.98

Table 5.1: Results on synthetic data. The table lists the number of samples (k) and mean error (err) for each dataset and algorithm. err is the mean error of ground truth inliers to the recovered model (perpendicular dist. for line and plane, symmetric transfer error for **H**, Sampson error for **F**). RANSAC_T/LMedS_T refer to running the algorithms with ground truth parameters, while RANSAC_F/LMedS_F refer to fixed parameters. Note that RECON provides the same accuracy as RANSAC_T/LMedS_T, without any *a priori* knowledge of the parameters.

5.4.1 Synthetic data:

We exhaustively evaluated the performance of RECON on four estimation problems: 2D line, 3D plane, homography (**H**), and fundamental matrix (**F**). The inlier ratio ε was varied over a wide range [0.25-1.0], in steps of 0.05. For the line and plane experiments, we generated points within a square (resp., cube) of side 500, and added zero-mean Gaussian noise to the inliers, with σ varying from 0.5-6.0. For the **H** and **F** tests, we generated synthetic 2D correspondences, varying σ from 0.5-4.0 (roughly the accuracy of current feature detectors (Mikolajczyk and Schmid, 2004)). For each (ε, σ) pair, the total number of points was randomly chosen in the range [500, 2000].

Given the large number of parameter combinations tested, we present a representative selection of the results, noting briefly that RECON provided accurate results over the entire test dataset. Table 5.1 provides a baseline comparison of RECON with RANSAC and LMedS, listing the number of samples drawn (k) and the mean error (err), averaged over 500 runs. RANSAC_F and LMedS_F

denote “realistic” cases, where the algorithms are run using commonly chosen parameter values. We use $\sigma = 1.0$ to set the RANSAC_F threshold, and simply use the median score in LMedS_F , with the number of trials computed using a 50% inlier ratio. To denote the “ideal” case, we also run the algorithms with *true* parameter values which, of course, are typically unknown. For these cases, denoted by RANSAC_T and LMedS_T , the RANSAC threshold is set using the true σ , and LMedS is adapted to switch between the median or the 25% lower quartile, depending on the true ε .

Perhaps the most important observation to be made from the results is that RECON is able to provide results with virtually the same accuracy as RANSAC_T and LMedS_T , while requiring *no* parameter tuning. In addition, when RANSAC and LMedS are run with incorrect parameter values, their performance deteriorates significantly. An interesting statistic to note from Table 5.1 is that the number of trials for RECON is appreciably lower than for RANSAC, even when RANSAC is run using the true threshold value. Indeed, as shown in Figure 5.4(a), the number of samples drawn by RECON is significantly close to that predicted by the standard RANSAC stopping criterion using a 95% confidence level. This is not a coincidence; as noted in Section 2.2.3, it can be shown that for a 95% confidence in the solution, three uncontaminated samples are drawn on average in RANSAC before the confidence in the solution is achieved. Since these experiments were run looking for $K = 3$ consistent models, it is these samples that RECON finds. RANSAC has the opposite problem: since noisy models have lower support, this means more samples must be drawn to meet the stopping criterion. Figure 5.4(b) plots the fraction of true (i.e., ground truth) inliers found by RECON. Note that in all cases, RECON returns close to 100% of the true inliers, while RANSAC suffers, particularly for higher noise levels. This relates back to the discussion in Chapter 4; using a fixed threshold with noisy models implies that not all inliers will be found. On the other hand, since we look for *consistent* sets of inliers, and do not impose a fixed threshold, a higher fraction of true inliers is returned.

To further test the sensitivity of RANSAC-based methods to incorrect estimates of the scale, we generate synthetic putative matches for homography estimation, with $\varepsilon = 0.4$ and $\sigma = 0.5$. Subsequently, we run RANSAC and LO-RANSAC, with the threshold computed using a range of noise scales, $\sigma_{est} \in [0.01, 10.0]$. These scale estimates are substituted into Equation 5.1, with $\alpha = 0.95$, to obtain the corresponding threshold values. We also run MSAC and LO-MSAC, which

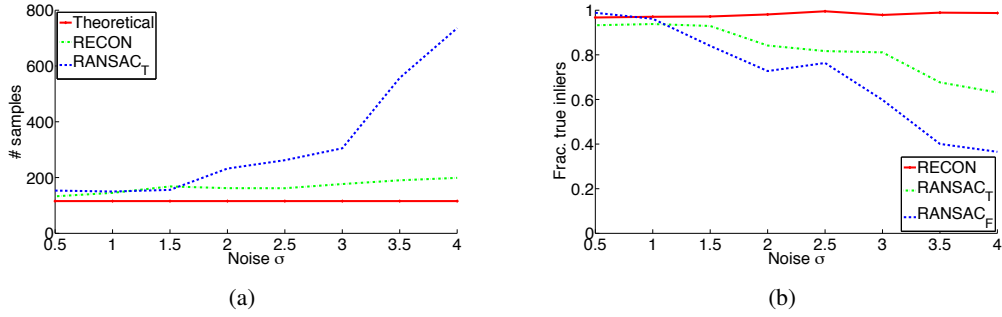


Figure 5.4: Plots for homography estimation with $\varepsilon = 0.4$. (a) Number of samples vs. σ . Note that RECON is significantly close to the theoretical estimate. (b) Fraction of true inliers recovered vs. σ . RECON returns close to 100% of the inliers.

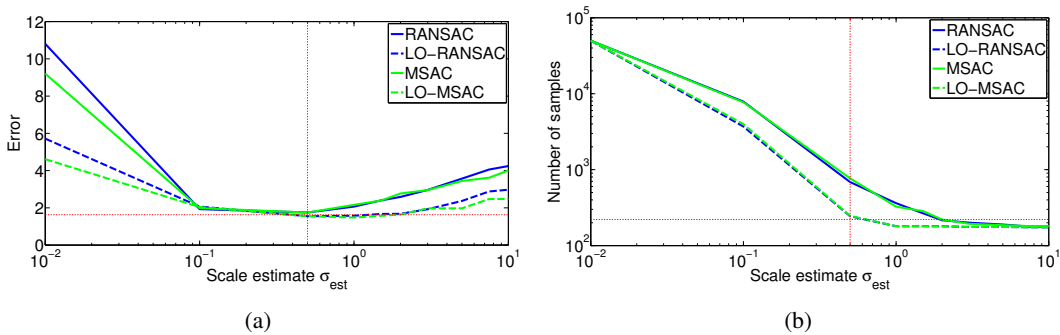


Figure 5.5: Plots for homography estimation with $\varepsilon = 0.4$ and $\sigma = 0.5$. (a) Estimation error vs. prior estimate of noise scale σ_{est} . The x-axis is on a log-scale (b) Number of samples vs. prior estimate of noise scale σ_{est} . Both axes use a log-scale. In both plots, the dashed vertical line corresponds to the true value of $\sigma = 0.50$. The dashed horizontal lines represent the values for RECON (i.e., RECON estimation error in (a), and number of samples in (b)).

are similar to their RANSAC counterparts, but where the standard RANSAC cost function (Equation 2.2) is replaced by the M-estimator cost function (2.12); i.e., outliers are assigned a constant penalty, but inliers are now scored based on how well they fit the model. The results for this experiment are shown in Figure 5.5. Figure 5.5(a) plots the estimation error as a function of the assumed noise scale (and thus the computed inlier threshold). The horizontal red line represents the estimation error for RECON, which does not require the scale parameter to be specified. The main point worth noting from the graph is the robustness of the baseline methods to inaccuracy in the scale estimates: the estimation error for RANSAC and MSAC, particularly when using local optimization, is relatively low over a wide range of scale estimates. This indicates that even if our estimate of the scale of inlier noise is off by up to an order of magnitude, it may still be possible to obtain relatively accurate results using just the baseline methods. However, it is also worth noting that, particularly when the

assumed scale is a severe underestimate, this can lead to a significant increase in the number of samples drawn. An underestimate of the scale corresponds to a lower computed threshold value and thus effectively, a lower inlier ratio – fewer “true” inliers will satisfy the threshold constraint, thus reducing the support. This is illustrated in Figure 5.5(b), which plots the number of samples required as a function of the scale estimate. It can be seen that at low scale values, all these approaches require many more iterations than RECON. In summary, the main points to take away from these results is that RANSAC and MSAC are, in reality, fairly resilient to mistakes in the estimate of noise scale. Using these methods is likely to provide a computational advantage, particularly when using the various optimizations described in Section 3. However, there are caveats – for large underestimates of the scale, RECON can prove to be faster, drawing far fewer samples.

5.4.2 Real data

We also tested RECON’s performance on real data, for four model fitting problems: homography, essential matrix, fundamental matrix and 3D similarity. Since the ground truth noise variance is unknown, we use the commonly used setting of $\sigma = 1.0$ for the two-view estimation problems. For obtaining putative matches in the 3D similarity case, we compute a mean SIFT descriptor for each 3D point, and then run the standard matching process between the point sets, using the ratio test to suppress less distinctive matches. Setting the RANSAC inlier threshold for the 3D similarity estimation problem is much more difficult, since the submodels are reconstructed up to an arbitrary scale. For this case, the threshold was determined using an exhaustive trial-and-error approach. The results are presented in Table 5.2. Note that the performance of RECON is typically better than RANSAC, while requiring no threshold. In addition, when using local optimization in RANSAC, both LO and RECON produce results of similar quality, drawing a comparable number of samples in both cases. Our implementation of RECON, though as yet unoptimized for speed, has runtimes between 30 milliseconds and 2.5 seconds for the experiments in Table 5.2, while the corresponding range for RANSAC is 22ms–1.7s. It is worth noting that while RECON has a computational overhead compared to RANSAC and LO-RANSAC, it is still much faster than current threshold-free methods. For instance, the recent approach in (Chin et al., 2010) takes on the order of a minute to compute the epipolar geometry for a pair of images – irrespective of the inlier ratio. RECON’s




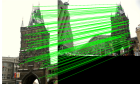

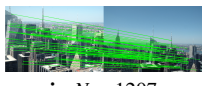


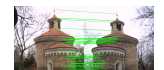


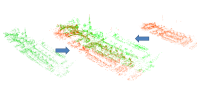
Homography (H)		RANSAC	LO	RECON	Essential (E)		RANSAC	LO	RECON
 a : $N = 2450$	k I err	66.5 ± 21 1344 ± 96 2.63 ± 0.53	36.2 ± 2 1670 ± 29 1.98 ± 0.25	37.9 ± 7 1663 ± 38 2.02 ± 0.27	 g : $N = 962$	k I err	72.3 ± 25 556 ± 21 3.52 ± 1.01	58.7 ± 15 600 ± 8 1.54 ± 0.69	58.9 ± 15 598 ± 12 1.55 ± 0.63
 b : $N = 495$	k I err	588.3 ± 154 153 ± 9 2.59 ± 0.49	238.6 ± 44 172 ± 12 2.03 ± 0.11	224.6 ± 43 174 ± 3 2.09 ± 0.16	 h : $N = 1110$	k I err	108.2 ± 31 602 ± 33 0.69 ± 0.21	74.3 ± 17 632 ± 24 0.49 ± 0.15	78.5 ± 19 632 ± 25 0.48 ± 0.13
 c : $N = 1967$	k I err	256.6 ± 121 739 ± 52 1.47 ± 0.45	77.9 ± 25 990 ± 15 1.07 ± 0.22	73.3 ± 22 995 ± 23 1.07 ± 0.28	 i : $N = 1207$	k I err	1092.3 ± 312 417 ± 41 1.74 ± 0.77	416.2 ± 66 461 ± 18 1.12 ± 0.34	407.4 ± 57 465 ± 20 1.17 ± 0.38
Fundamental (F)					3D similarity				
 d : $N = 2454$	k I err	52.3 ± 26 1854 ± 33 1.80 ± 0.85	45.7 ± 18 1890 ± 30 0.97 ± 0.26	39.1 ± 16 1897 ± 28 1.01 ± 0.26	 j : $N = 4529$	k I err	78.1 ± 25 1640 ± 27 $0.13 \pm 3 \times 10^{-3}$	57.4 ± 7 1888 ± 12 $0.09 \pm 2 \times 10^{-3}$	59.9 ± 8 1887 ± 14 $0.08 \pm 1 \times 10^{-3}$
 e : $N = 930$	k I err	11514 ± 2213 311 ± 8 1.91 ± 0.72	4190 ± 184 338 ± 3 1.41 ± 0.42	4134 ± 165 341 ± 4 1.48 ± 0.59	 k : $N = 1955$	k I err	43.3 ± 12 967 ± 36 $8.2 \times 10^{-3} \pm 2 \times 10^{-4}$	35.4 ± 12 982 ± 16 $6.2 \times 10^{-3} \pm 2 \times 10^{-4}$	31.1 ± 8 982 ± 20 $6.7 \times 10^{-3} \pm 5 \times 10^{-4}$
 f : $N = 838$	k I err	148.8 ± 48 520 ± 30 1.78 ± 0.83	73.8 ± 18 569 ± 13 1.58 ± 0.36	72.5 ± 23 568 ± 18 1.68 ± 0.56	 l : $N = 2549$	k I err	140.8 ± 28 825 ± 51 $0.05 \pm 3.1 \times 10^{-3}$	100.8 ± 28 851 ± 17 $0.02 \pm 3.1 \times 10^{-3}$	99.0 ± 12 844 ± 20 $0.02 \pm 1.0 \times 10^{-3}$

Table 5.2: Results on real data, for four estimation problems: homography (**a-c**), fundamental matrix (**d-f**), essential matrix (**g-i**) and 3D similarity (**j-l**). The table lists the number of hypotheses generated (k), the number of inliers returned (I), the total number of points (N), and the mean error (err) for each of the three methods used: RANSAC, RANSAC with local optimization (LO), and RECON. err is the mean over inliers with respect to the returned model (symmetric transfer error for **H** and 3D similarity, Sampson distance for **F** and **E**). Units are in pixels for **H**, **F** and **E**. Note that the error for 3D similarity is arbitrarily scaled. By using real world measurements, the error in **I** was found to correspond to ≈ 8 cm.

ability to adaptively stop evaluation is thus especially favourable when contrasted with current techniques that use either a pessimistic estimate of the number of hypotheses (e.g., 10000 in (Wang and Suter, 2004)) or a fixed time window (e.g., 60s in (Chin et al., 2010)).

From an application perspective, we have integrated RECON into our 3D reconstruction pipeline, where it is applied to the output of sparse structure from motion. Often, these systems output multiple submodels that then need to be automatically aligned to obtain a complete reconstruction. Figure 5.6 shows some sample results, where the colours (red/green) represent the individual sub-

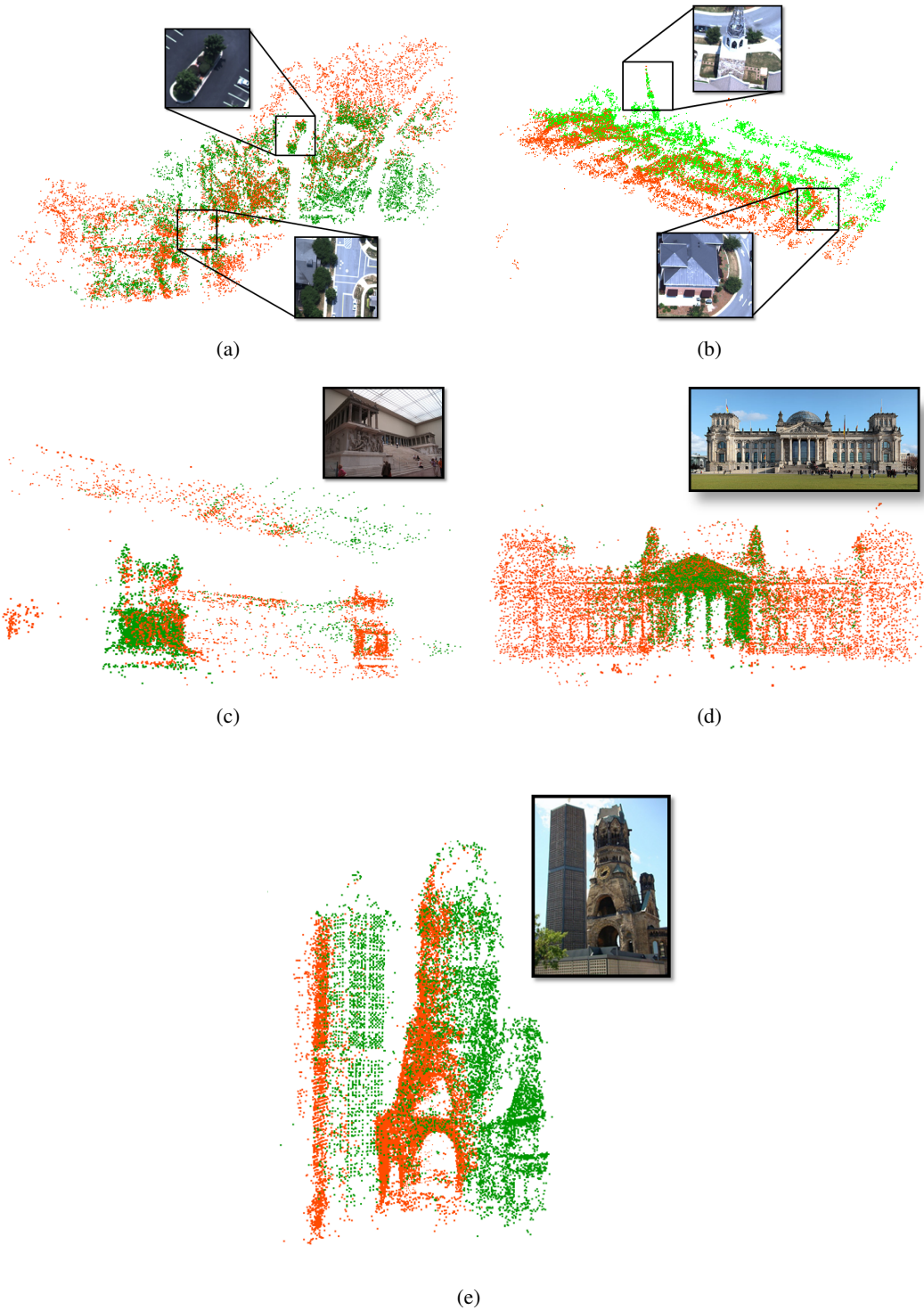


Figure 5.6: Alignment of 3D submodels using RECON. The colours (green/red) indicate different 3D submodels obtained via structure from motion. The alignment is computed automatically with RECON. (a) and (b) are from an aerial video sequence, while (c)-(e) are reconstructions from an internet photo collection representing the city of Berlin.

models. Figures 5.6(a)-(b) show 3D models obtained from aerial video recordings. In this case, video sequences were acquired by flying an airplane along parallel “stripes”, with small overlap between neighbouring paths. Figure 5.6(c)-(e) show selected reconstructions from an internet photo collection; in this case, 2.7 million images of the city of Berlin. The advantage of using RECON in these large-scale systems is that it does not require any data specific parameter tweaking, and is still efficient enough to be deployed on real-world robust estimation problems.

5.5 Discussion

In this section, we have presented a new, threshold-free framework for robust estimation. In contrast to prior work, we use *consistency* of models as a cue to detect model estimates that are close to the true solution. We develop the theoretical framework, and then present a practical algorithm, RECON, that can be used to efficiently perform robust estimation. The algorithm is very simple, easy to implement, and effective in practice. RECON produces results of the same quality as algorithms that specifically use fine-tuned parameter estimates. An additional benefit of the flexible consensus measure we use, is the ability to elegantly adapt the number of samples drawn to the contamination level of the data.

While these results are promising, there is room for improvement. Firstly, note that in this work, we specifically address the issue of single model robust estimation. The issue of multiple models (or, alternatively, a single model with structured outliers) is a harder problem to solve. While there exist other approaches (e.g., Chin et al., 2009b; Wong et al., 2011) that address this problem, these are still relatively inefficient and not directly applicable to large scale problems. Being able to estimate multiple models, while still retaining the adaptability of RANSAC and RECON is an interesting, and challenging, problem.

While RECON is typically much more efficient than current threshold-free methods, it still has a significant computational overhead, particularly compared to the optimized RANSAC techniques presented in Chapter 3. One reason for this is that in the absence of an inlier threshold, RECON inspects model pairs to determine consistency. For the case of very low inlier ratios, where the number of hypothesized models is large, this can lead to a significant overhead. We note that one possible way to mitigate this effect is to note that, given a set of previously generated models with

sorted inlier sets, the goal in RECON is to *retrieve* the closest matches to a newly hypothesized model, according to some similarity measure based on residual ordering. If each model in RECON is viewed as a set of sorted inlier indices, this process requires estimating how similar two sets are. Efficient methods – such as MinHash (Broder, 1997) – have been proposed for precisely this problem. Integrating these techniques into RECON could yield significant computational benefits, and potentially bring its performance on par with RANSAC, but without requiring the threshold parameter.

Chapter 6

Conclusion

6.1 Summary

Revisiting the dissertation statement from Chapter 1:

Given noisy and highly contaminated data measurements, it is possible to efficiently and accurately recover model parameter estimates, thus enabling real-time and large-scale applications. When the scale of noise is known, it is possible to compensate explicitly for this, yielding improved performance. In addition, in the absence of noise scale, it is still possible to perform efficient robust estimation by looking for model consistency.

The dissertation statement highlights some of the challenges involved in robust estimation – handling the effects of noise and outliers in a manner that is efficient, accurate, and flexible. These are crucial ingredients that are necessary in order for these algorithms to transition into challenging real-world applications, operating on diverse and large datasets. This has motivated much of the work in this dissertation. In Chapter 3, we described two algorithms: USAC and ARRSAC, which represent state of the art implementations that bring together many optimizations and ideas, in order to achieve high performance. We also demonstrated the use of an alternate source of prior information, namely, the reliability of low-level visual words, in order to further improve performance. This proves to be useful when processing large scale internet image collections.

In Chapter 4, we proposed a method to address the issue of noise in RANSAC, and in particular, its effect on the estimated model parameters. We characterize the uncertainty of the model in terms

of its covariance, and show how this can be used to develop a flexible method to compute the support for a noisy model. We then integrated this into a robust estimation framework, which we term cov-RANSAC, which achieves an improvement in efficiency compared to standard techniques.

Finally, in Chapter 5, we developed a simple framework for threshold-free robust estimation, called RECON. In lieu of a user-specified threshold parameter, RECON examines pairs of models in order to identify consistent fits to the data. We explore the use of residual ordering information in order to design an efficient algorithm for threshold-free robust estimation.

Many of the algorithms developed in this dissertation have found application in real-world systems, operating on large-scale datasets. In particular, ARRSAC has been integrated into a 3D reconstruction system operating on video sequences (Pollefeys et al., 2008), and has enabled vision-based pose estimation to operate in real-time. ARRSAC and USAC have also found application in the processing of large scale internet image collections (Frahm et al., 2010; Raguram et al., 2011; Raguram et al., 2012b), where they have enabled the processing of millions of images – this is an order of magnitude more data than other similar approaches (Agarwal et al., 2009). Recently, we have also integrated RECON into this system, where it is applied to the output of sparse structure from motion, in order to merge submodels to obtain more complete 3D reconstructions. Manually adjusting parameters is often not scalable when applied to these large scale problems, and the threshold-free nature of RECON is appealing in this context.

6.2 Future Work

As noted in Chapter 2, the field of robust estimation is a very active one, with a rich variety of challenges and open problems. This dissertation has focused on one aspect of this problem – RANSAC-based robust estimators – and we have proposed algorithms that improve upon the state of the art in this area. There are also a number of interesting directions for future work:

Multi-model robust estimation: In this dissertation, we have focused on the issue of single model robust estimation, which is one that arises frequently in practice. However, the issue of multiple models is also important, and is a harder problem to solve. In addition, if it is assumed that the scale of noise is unknown, this adds an additional layer of complexity. Many approaches have been

proposed to handle this case (Miller and Stewart, 1996; Lee et al., 1998; Wang and Suter, 2004; Toldo and Fusiello, 2008; Chin et al., 2009b; Wong et al., 2011); however, these approaches are still limited in their ability to adapt to the data, and often require other implicit assumptions to be made (such as knowledge of the inlier ratio). As a consequence, these approaches have not yet found widespread application in real-world systems. With RECON, we have explored the use of residual information as a source of identifying consistent fits to the data. There have been complementary approaches that apply similar ideas to the case of multi-model fitting (Chin et al., 2009b; Wong et al., 2011). Combining the adaptivity of RECON with the multi-model estimation capability of these approaches is an interesting, and challenging, direction for future work.

Reformulating RANSAC: In some applications, such as structure from motion, there are cases where we have multiple hypothesis generators which provide alternate ways of computing the same solution. For instance, the camera motion of a stereo rig can be computed from either line correspondences, or point correspondences, or some combination of the two (Pradeep and Lim, 2010; Chandraker et al., 2009). Standard RANSAC does not have a direct way to handle this problem, and would thus involve estimating all competing models. An interesting way to think about robust estimation in this context is to cast it as a *multi-armed bandit* problem (Gittins, 1989). Multi-armed bandit (MAB) problems form a class of sequential resource allocation problems, which address the issue of allocating one or more resources among several competing projects. In the MAB context, each hypothesis generator can be viewed as a lever of a slot machine, with some (unknown) distribution of rewards (in this case, the probability of drawing a good solution). The objective of the bandit is to maximize the sum of rewards earned through a sequence of lever pulls. Solving this problem typically involves the use of exploration (to figure out which levers, or hypothesis generators, are promising) and exploitation (to draw good samples from the promising levers). Applying the machinery of MAB problems, along with the many algorithms developed in this area, to the problem of robust estimation, is an interesting avenue for future work.

Incorporating higher-level knowledge: There have been approaches (see Section 3.2.2), that aim to incorporate some form of prior information into RANSAC. While these methods are promising,

there is significant scope for improvement. In this dissertation, we explored one possibility – namely, to use the information that arises as a byproduct of geometric verification in image collections – to bias the sampling in favour of visual words that are more likely to be reliable. This idea can be taken even further. For instance, one bottleneck in the verification process is the stage of feature matching. Currently, this is usually done by matching SIFT descriptors between a pair of images, in order to identify putative correspondences. However, since the SIFT descriptors for a large-scale image collection have a significant memory footprint, they typically have to be loaded on demand from disk. The use of visual words to directly perform feature matching has been explored to some extent (Philbin et al., 2007), but there are challenges; this often leads to low inlier ratios, as well as the problem of many-to-many feature matches. While we explored the use of visual word weights in RANSAC, this idea can potentially be taken further. For instance, (Mikulík et al., 2010) describes a way to use the output of geometric verification to link together words that are likely to be related. Incorporating this type of learned information into the feature matching stage as well, could potentially yield a significant benefit. In addition, exploiting the co-occurrence relationships between features is likely to help as well. For instance, if certain sets of visual words often tend to occur together as inliers, then it should be possible to integrate this into the sampling scheme. More specifically, given that a subset of size $s < m$ has been picked, the selection of the $s + 1^{\text{th}}$ point can be guided by this co-occurrence information. While this has been explored for the case of image retrieval (Chum and Matas, 2010b), it should be possible to more directly integrate it into the matching and verification stages as well.

Another source of prior information, again in regard to internet image collections, is via the use of EXIF tags. These tags often provide an estimate of the focal length of the camera, in addition to other metadata about the photo. It has been shown that this type of information can be leveraged to seed structure from motion (Crandall et al., 2011). However, the use of this information within RANSAC has not yet been fully leveraged. For instance, when computing the relative pose using the 5-point algorithm, the focal length from the EXIF tags can be used to generate a prior on this parameter. It may then be possible to sample essential matrices that are consistent with this prior information, which could potentially yield a performance improvement.




Appendix A

A.1 List of image pairs used

Below, we list the various image pairs used in the evaluation of the algorithms presented in this dissertation. Links to the original sources are provided where available.

A.1.1 Homography

Images	Notes
	Graffiti: Table 3.1, 4.1, 5.2. Available at http://www.robots.ox.ac.uk/~vgg/data/data-aff.html
	Boston: Table 3.1. Available at http://www.vision.cs.rpi.edu/gdbicp/dataset/
	Castle: Table 3.1. Available at http://cmp.felk.cvut.cz/~cechj/SCV/
	UNC-Wall: Table 3.1, 4.1, 5.2. Available at http://cs.unc.edu/~rraguram/usac
	Albany-Empire-State: Table 3.1. Available at http://www.flickr.com/photos/wallyg/3812586429/ and http://www.flickrriver.com/photos/tags/harrisonabramovitz/interesting/
	Facade-Windows: Table 3.1, 4.1. Available at http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html
	OptiFree: Table 3.2. Available at http://cs.unc.edu/~rraguram/usac

Images	Notes
	Extreme Zoom: Table 3.2, 4.1, 5.2. Available at http://www.vision.cs.rpi.edu/gdbicp/dataset/
	Day-Night (Summer): Table 3.2. Available at http://www.vision.cs.rpi.edu/gdbicp/dataset/
	Cap. Reg.: Table 3.2. Available at http://www.vision.cs.rpi.edu/gdbicp/dataset/


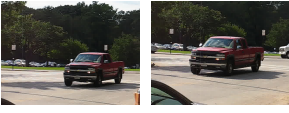


A.1.2 Epipolar geometry

Images	Notes
	Merton College: Table 3.3. Available at http://www.robots.ox.ac.uk/~vgg/data/data-mview.html
	City-Hall-Leuven: Table 3.3, 3.7, 4.2, 5.2. Available at http://cvlab.epfl.ch/data/strechamvs/
	Oxford: Table 3.3, 3.7, 4.2. Available at http://cs.unc.edu/~rraguram/usac
	Trees: Table 3.3. Available at http://cmp.felk.cvut.cz/~cechj/SCV/
	London-Victoria: Table 3.3, 3.7. Available at http://cmp.felk.cvut.cz/~cechj/SCV/

Images	Notes
	<p>Great-Wall: Table 3.3, 4.2. From (Chum and Matas, 2005), available at http://cs.unc.edu/~rraguram/usac</p>
	<p>London-Gherkin: Table 3.4, 3.7, 4.2. Available at http://www.flickr.com/photos/andreicutza_ok/4140443295/sizes/l/in/photostream/ http://www.flickr.com/photos/davidgardener/4536034140/sizes/l/in/photostream/</p>
	<p>Vision-Textbook: Table 3.4. Available at http://cs.unc.edu/~rraguram/usac</p>
	<p>Box: Table 3.4. From (Chum et al., 2005), available at http://cs.unc.edu/~rraguram/usac</p>
	<p>Lamppost: Table 3.4. From (Chum et al., 2005), available at http://cs.unc.edu/~rraguram/usac</p>
	<p>Aerial: Table 3.5. Available at http://cs.unc.edu/~rraguram/usac</p>
	<p>Vysehrad: Table 3.5. Available at http://cmp.felk.cvut.cz/~cechj/SCV/</p>
	<p>Powder-Gate: Table 3.5, 5.2. Available at http://cs.unc.edu/~rraguram/usac</p>
	<p>Louvre: Table 3.5, 5.2. Available at http://cs.unc.edu/~rraguram/usac</p>

Images	Notes
	Manhattan: Table 3.5, 5.2. Available at http://cs.unc.edu/~rraguram/usac
	UNC-South-Building: Table 3.5. Available at http://cs.unc.edu/~rraguram/usac
	Leuven-Castle: Table 3.7. Available at http://www.cs.unc.edu/~marc/data/castlejpg.zip
	Begijnhof: Table 3.7. Available at http://cs.unc.edu/~rraguram/usac
	Rotunda: Table 5.2. Available at http://cmp.felk.cvut.cz/projects/is3d/Data.html
	Valbonne: Table 5.2. Available at http://www.robots.ox.ac.uk/~vgg/data/data-mview.html

A.1.3 Motion Segmentation

Images	Notes
	Traffic-1: Table 3.6. Available at http://www.vision.jhu.edu/data/hopkins155/
	Traffic-2: Table 3.6. Available at http://www.vision.jhu.edu/data/hopkins155/
	Desk-1: Table 3.6. Available at http://cs.unc.edu/~rraguram/usac
	Desk-2: Table 3.6. Available at http://cs.unc.edu/~rraguram/usac

Bibliography

- Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building Rome in a day. In *International Conference on Computer Vision (ICCV)*.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.
- Beaton, A. E. and Tukey, J. W. (1974). The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185.
- Broder, A. (1997). On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences, SEQUENCES '97*, pages 21–.
- Brown, M. and Lowe, D. G. (2003). Recognising panoramas. In *International Conference on Computer Vision (ICCV)*, pages 1218–.
- Capel, D. (2005). An effective bail-out test for RANSAC consensus scoring. In *British Machine Vision Conference (BMVC)*.
- Cech, J., Matas, J., and Perd'och, M. (2008). Efficient sequential correspondence selection by cosegmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.
- Chakravarti, I. M., Laha, R. G., and Roy, J. (1967). *Handbook of Methods of Applied Statistics*, volume I.
- Chandraker, M. K., Lim, J., and Kriegman, D. J. (2009). Moving in stereo: Efficient structure and motion using lines. In *International Conference on Computer Vision (ICCV)*, pages 1741–1748.
- Chin, T. J. et al. (2009a). The Ordered Residual Kernel for Robust Motion Subspace Clustering. In *Neural Information Processing Systems*.
- Chin, T.-J., Wang, H., and Suter, D. (2009b). Robust fitting of multiple structures: The statistical learning approach. In *International Conference on Computer Vision (ICCV)*.
- Chin, T.-J., Yu, J., and Suter, D. (2010). Accelerated hypothesis generation for multi-structure robust fitting. In *European Conference on Computer Vision (ECCV)*.
- Choi, J. and Medioni, G. (2009). StaRSaC: Stable random sample consensus for parameter estimation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Chum, O. and Matas, J. (2005). Matching with PROSAC - progressive sample consensus. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Chum, O. and Matas, J. (2008). Optimal randomized RANSAC. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(8):1472–1482.
- Chum, O. and Matas, J. (2010a). Large-scale discovery of spatially related images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(2):371–377.

- Chum, O. and Matas, J. (2010b). Unsupervised discovery of co-occurrence in sparse high dimensional data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3416–3423.
- Chum, O., Matas, J., and Kittler, J. (2003). Locally optimized RANSAC. In *DAGM-Symposium*, pages 236–243.
- Chum, O., Werner, T., and Matas, J. (2004). Epipolar geometry estimation via RANSAC benefits from the oriented epipolar constraint. In *International Conference on Pattern Recognition (ICPR)*, pages 112–115.
- Chum, O., Werner, T., and Matas, J. (2005). Two-view geometry estimation unaffected by a dominant plane. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 772–779.
- Clipp, B., Lim, J., Frahm, J.-M., and Pollefeys, M. (2010). Parallel, real-time visual slam. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- Crandall, D. J., Owens, A., Snavely, N., and Huttenlocher, D. (2011). Discrete-continuous optimization for large-scale structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3001–3008.
- Criminisi, A., Reid, I., and Zisserman, A. (1999). A plane measuring device. *Image and Vision Computing*, 17(8):625 – 634.
- Csurka, G., Zeller, C., Zhang, Z., and Faugeras, O. D. (1997). Characterizing the uncertainty of the fundamental matrix. *Computer Vision and Image Understanding*, 68(1):18–36.
- Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- Frahm, J.-M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., and Pollefeys, M. (2010). Building rome on a cloudless day. In *European Conference on Computer Vision (ECCV)*, volume 6314, pages 368–381.
- Frahm, J.-M. and Pollefeys, M. (2006). RANSAC for (quasi-)degenerate data (QDEGSAC). In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 453–460.
- Gittins, J. C. (1989). *Multi-armed Bandit Allocation Indices*. Wiley, Chichester, NY.
- Haralick, R. M. (1994). Propagating covariance in computer vision. In *In Proc. Workshop on Performance Characteristics of Vision Algorithms*.
- Haralick, R. M., Lee, C.-N., Ottenberg, K., and Nölle, M. (1994). Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):331–356.
- Hartley, R. (1997a). In defense of the eight-point algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(6):580 –593.
- Hartley, R. I. (1997b). In defense of the eight-point algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(6):580–593.

- Hartley, R. I. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge University Press.
- Holland, P. W. and Welsch, R. E. (1977). Robust regression using iteratively reweighted least squares. *Communications in Statistics: Theory and Methods*, pages 813–828.
- Hough, P. (1962). Method and means for recognizing complex patterns. U.S. Patent 3.069.654.
- Huber, P. J. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, (35):73–101.
- Huber, P. J. (1981). *Robust Statistics*. John Wiley & Sons.
- Illingworth, J. and Kittler, J. (1988). A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116.
- Jung, I.-K. and Lacroix, S. (2001). A robust interest points matching algorithm. In *Computer Vision, Eighth IEEE International Conference on*, volume 2, pages 538–543.
- Kanatani, K. (1996). *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science Inc., New York, NY, USA.
- Kanatani, K. (2004). Uncertainty modeling and model selection for geometric inference. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(10):1307–1319.
- Knopp, J., Sivic, J., and Pajdla, T. (2010). Avoiding confusing features in place recognition. In *European Conference on Computer Vision (ECCV)*, pages 748–761.
- Lee, K.-M., Meer, P., and Park, R.-H. (1998). Robust adaptive segmentation of range images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20:200–205.
- Legendre, A.-M. (1805). *Nouvelles méthodes pour la détermination des orbites des comètes*. Paris.
- Li, D., , and Parkhurst, D. J. (2006). Open-source software for real-time visible-spectrum eye tracking. In *Proceedings of the COGAIN Conference*, pages 18–20.
- Li, H. (2009). Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *International Conference on Computer Vision (ICCV)*, pages 1074–1080.
- Li, Y., Snavely, N., and Huttenlocher, D. P. (2010). Location recognition using prioritized feature matching. In *European Conference on Computer Vision (ECCV)*, pages 791–804.
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.
- Mahalanobis, P. C. (1936). On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55.
- Malis, E. and Marchand, E. (2006). Experiments with robust estimation techniques in real-time robot vision. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- Maronna, R. A., Martin, D. R., and Yohai, V. J. (2006). *Robust Statistics: Theory and Methods (Wiley Series in Probability and Statistics)*. John Wiley & Sons, 1 edition.

- Matas, J. and Chum, O. (2002). Randomized RANSAC with $T_{d,d}$ test. *British Machine Vision Conference (BMVC)*, pages 448–457.
- Matas, J. and Chum, O. (2005). Randomized RANSAC with sequential probability ratio test. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 1727–1732 Vol. 2.
- McGlone, C., editor (2004). *The Manual of Photogrammetry, 5th Ed.* The American Society of Photogrammetry.
- Meer, P. (2004). Robust techniques for computer vision. pages 107–190.
- Meidow, J., Beder, C., and Förstner, W. (2009a). Reasoning with uncertain points, straight lines, and straight line segments in 2d. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(2):125–139.
- Meidow, J., Beder, C., and Förstner, W. (2009b). Reasoning with uncertain points, straight lines, and straight line segments in 2d. *ISPRS JPRS*, 64(2):125–139.
- Mikolajczyk, K. and Schmid, C. (2004). Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86.
- Mikulík, A., Perdoch, M., Chum, O., and Matas, J. (2010). Learning a fine vocabulary. In *European Conference on Computer Vision (ECCV)*, pages 1–14.
- Miller, J. and Stewart, C. V. (1996). MUSE: Robust surface fitting using unbiased scale estimates. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 300–306.
- Myatt, D. R., Torr, P. H. S., Nasuto, S. J., Bishop, J. M., and Craddock, R. (2002). NAPSAC: high noise, high dimensional robust estimation. In *British Machine Vision Conference (BMVC)*, pages 458–467.
- Naikal, N., Yang, A., and Sastry, S. S. (2011). Informative feature selection for object recognition via sparse pca. Technical report.
- Ni, K., Jin, H., and Dellaert, F. (2009). GroupSAC: Efficient consensus in the presence of groupings. In *International Conference on Computer Vision (ICCV)*.
- Nistér, D. (2003). Preemptive RANSAC for live structure and motion estimation. In *International Conference on Computer Vision (ICCV)*.
- Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26:756–777.
- Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 652–659.
- Ochoa, B. and Belongie, S. (2006). Covariance propagation for guided matching. In *Workshop on Statistical Methods in Multi-Image and Video Processing (SMVP)*.
- Papadopoulos, T. and Lourakis, M. I. A. (2000). Estimating the jacobian of the singular value decomposition: Theory and applications. In *European Conference on Computer Vision (ECCV)*.

- Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition*, pages 1–8.
- Philbin, J., Sivic, J., and Zisserman, A. (2011). Geometric latent dirichlet allocation on a matching graph for large-scale image datasets. *International Journal of Computer Vision*, 95(2):138–153.
- Pollefeys, M., Nistér, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S. J., Merrell, P., Salmi, C., Sinha, S. N., Talton, B., Wang, L., Yang, Q., Stewénus, H., Yang, R., Welch, G., and Towles, H. (2008). Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167.
- Pradeep, V. and Lim, J. (2010). Egomotion using assorted features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1514–1521.
- Raguram, R., Chum, O., Pollefeys, M., Matas, J., and Frahm, J.-M. (2012a). USAC: A universal framework for random sample consensus.
- Raguram, R. and Frahm, J.-M. (2011). RECON: Scale-adaptive robust estimation via residual consensus. In *International Conference on Computer Vision (ICCV)*.
- Raguram, R., Frahm, J.-M., and Pollefeys, M. (2008). A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision (ECCV)*, pages 500–513.
- Raguram, R., Frahm, J.-M., and Pollefeys, M. (2009). Exploiting uncertainty in random sample consensus. In *International Conference on Computer Vision (ICCV)*.
- Raguram, R., Tighe, J., and Frahm, J.-M. (2012b). Improved geometric verification for large-scale landmark image collections. In *British Machine Vision Conference (BMVC)*.
- Raguram, R., Wu, C., Frahm, J.-M., and Lazebnik, S. (2011). Modeling and recognition of landmark image collections using iconic scene graphs. *International Journal of Computer Vision*, 95(3):213–239.
- Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880.
- Rousseeuw, P. J. and Leroy, A. M. (1987). *Robust regression and outlier detection*. John Wiley & Sons.
- Sattler, T., Leibe, B., and Kobbelt, L. (2009). SCRAMSAC: Improving RANSAC’s efficiency with a spatial consistency filter. In *International Conference on Computer Vision (ICCV)*.
- Sattler, T., Leibe, B., and Kobbelt, L. (2011). Fast image-based localization using direct 2d-to-3d matching. In *International Conference on Computer Vision (ICCV)*, pages 667–674.
- Schaffalitzky, F. and Zisserman, A. (2002). Multi-view matching for unordered image sets, or “how do i organize my holiday snaps?”. In *European Conference on Computer Vision (ECCV)*, volume 1, pages 414–431. Springer-Verlag.

- Schmid, C. and Mohr, R. (1997). Local grayvalue invariants for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(5):530–535.
- Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, pages 214–226.
- Siegel, A. F. (1982). Robust regression using repeated medians. *Biometrika*, 69(1):242–244.
- Snaveley, N., Seitz, S. M., and Szeliski, R. (2008). Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.
- Stephens, M. A. (1974). Edf statistics for goodness of fit and some comparisons. *Journal of the ASA*, 69(347):pp. 730–737.
- Stewart, C. V. (1995). MINPRAN: A new robust estimator for computer vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(10):925–938.
- Strang, G. (1988). *Linear Algebra and Its Applications*. Brooks Cole.
- Sur, F., Noury, N., and Berger, M.-O. (2008). Computing the uncertainty of the 8 point algorithm for fundamental matrix estimation. In *British Machine Vision Conference (BMVC)*.
- Tanaka, K. and Kondo, E. (2006). Incremental ransac for online relocation in large dynamic environments. In *IEEE International Conference on Robotics and Automation*, pages 68–75.
- Toldo, R. and Fusiello, A. (2008). Robust multiple structures estimation with j-linkage. In *European Conference on Computer Vision (ECCV)*, pages 537–547.
- Tordoff, B. and Murray, D. W. (2002). Guided sampling and consensus for motion estimation. In *European Conference on Computer Vision (ECCV)*, pages 82–98.
- Torii, A., Havlena, M., and Pajdla, T. (2009). From google street view to 3d city models. In *International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2188 – 2195.
- Torr, P. and Zisserman, A. (1998). Robust computation and parametrization of multiple view relations. In *International Conference on Computer Vision (ICCV)*, pages 727–732.
- Torr, P. H. S. (1999). Model selection for two view geometry: A review. In *Shape, Contour and Grouping in Computer Vision*, pages 277–301, London, UK. Springer-Verlag.
- Torr, P. H. S. and Murray, D. W. (1997). The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24:271–300.
- Torr, P. H. S. and Zisserman, A. (2000). MLESAC: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156.
- Torr, P. H. S., Zisserman, A., and Maybank, S. J. (1995). Robust detection of degenerate configurations for the fundamental matrix. In *International Conference on Computer Vision (ICCV)*, page 1037, Washington, DC, USA.
- Torr, P. H. S., Zisserman, A., and Maybank, S. J. (1998). Robust detection of degenerate configurations while estimating the fundamental matrix. *Computer Vision and Image Understanding*, 71(3):312–333.

- Tron, R. and Vidal, R. (2007). A benchmark for the comparison of 3-d motion segmentation algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.
- Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.
- Turcot, P. and Lowe, D. G. (2009). Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data (WS-LAVD)*.
- Tuytelaars, T. and Gool, L. V. (2000). Wide baseline stereo matching based on local, affinely invariant regions. In *British Machine Vision Conference (BMVC)*, pages 412–425.
- von Mises, R. (1964). *Mathematical Theory of Probability and Statistics*. Academic Press, New York. H. Geiringer, ed.
- Wald, A. (1947). *Sequential analysis*. Dover, New York.
- Wang, H. and Suter, D. (2004). Robust adaptive-scale parametric model estimation for computer vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(11):1459–1474.
- Weng, J., Huang, T., and Ahuja, N. (1989). Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11:451–476.
- Wong, H. S., Chin, T.-J., Yu, J., and Suter, D. (2011). Dynamic and hierarchical multi-structure geometric model fitting. In *International Conference on Computer Vision (ICCV)*, pages 1044–1051.
- Zeisl, B., Georgel, P., Schweiger, F., Steinbach, E., and Navab, N. (2009). Estimation of location uncertainty for scale invariant feature points. In *British Machine Vision Conference (BMVC)*.
- Zhang, W. and Kosecka, J. (2006). A new inlier identification procedure for robust estimation problems. In *Robotics: Science and Systems*.
- Zhang, Z., Deriche, R., Faugeras, O., and Luong, Q. (1995). A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial intelligence*, 78(1-2):87–119.
- Zhang, Z. and Kanade, T. (1998). Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27:161–195.