On the Sample Consensus Robust Estimation Paradigm: Comprehensive Survey and Novel Algorithms with Applications

by

Peter Olubunmi OLUKANMI

Student Number 215040652

Submitted in fulfillment of the requirements
for the degree of Masters in Computer Science

in the

School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
Durban, South Africa

2016

DECLARATION

UNIVERSITY OF KWAZULU-NATAL COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

The research described in this thesis was performed at the University of KwaZulu-Natal under the supervision of Professor A.O. Adewumi. I hereby declare that all materials incorporated in this thesis are my own original work except where acknowledgement is made by name or in the form of a reference. The work contained herein has not been submitted in part or whole for a degree at any other university.

Signed:

_____

Olukanmi Peter Olubunmi

Date: January, 2016

As the candidate's supervisor, I have approved/disapproved the dissertation for submission

Signed:

_____

Professor A.O. Adewumi

Date: January, 2016

DECLARATION II - PLAGIARISM

I, Olukanmi Peter Olubunmi, declare that

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This thesis has not been submitted for any degree or examination at any other university.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
    a. Their words have been re-written but, the general information attributed to them has been referenced.
    b. Where their exact words have been used, their writing has been placed in italics and in quotation marks, and referenced.
5. This thesis does not contain text, graphics or tables which are copied and pasted from the internet, unless specifically acknowledged, and the source being detailed in the thesis and in References section.

Signed:

_____

Date: February, 2016

DEDICATION

To Seun, the best wife in the world, to our lovely daughter, Grace, who landed on earth a few days ago, and ultimately to the Almighty God who blessed me with these precious jewels.

ACKNOWLEDGEMENT

My appreciation goes first to God, my Maker and the giver of all good things.

My profound and sincere gratitude goes to my darling wife, Mrs. Seun Olukanmi. Thank you for your constant encouragement, whilst at the same time, writing your own thesis. My heartfelt appreciation goes to my parents, Mr and Mrs Olukanmi, by whose many sacrifices I have come this far, as well as to my parents-in-law, Pastor (Engr.) and Mrs Fagbemi.

Special thanks goes to my supervisor, Professor Aderemi Oluyinka Adewumi for his fatherly care, guidance and counsel, throughout this programme. I also thank Dr A. Arasomwan and my colleagues at the Centre for Applied Artificial Intelligence Research, for their valuable contributions. Special thanks Peter Popoola and Ayobami Akinyelu: your questions and suggestions were very helpful. Mafunda Sowambile, your suggestions on 'proofs', were very helpful. Big thanks also to Dr Toyin Popoola, Dr T.A. Fashanu, Dr Tunde Adegbola, and Prof. Fakinlede, of the department of Systems Engineering, University of Lagos, Nigeria, all of whom have had indelible influences on my career in positive ways. A special place is reserved for the able technician, to whom I have run, for one technology solution or the other: Mr Soren Greenwood.

I appreciate Pastors Michael Olusanya and Joseph Adesina, and their respective families, as well as all the members of the Deeper Life Bible Church, KZN Province, South Africa, for being very supportive to me and my family.

All of you, and many others, have made it possible to complete this degree, in record time. May God bless and keep you all.

ABSTRACT

This study begins with a comprehensive survey of existing variants of the Random Sample Consensus (RANSAC) algorithm. Then, five new ones are contributed. RANSAC, arguably the most popular robust estimation algorithm in computer vision, has limitations in accuracy, efficiency and repeatability. Research into techniques for overcoming these drawbacks, has been active for about two decades. In the last one-and-half decade, nearly every single year had at least one variant published: more than ten, in the last two years. However, many existing variants compromise two attractive properties of the original RANSAC: simplicity and generality. Some introduce new operations, resulting in loss of simplicity, while many of those that do not introduce new operations, require problem-specific priors. In this way, they trade off generality and introduce some complexity, as well as dependence on other steps of the workflow of applications. Noting that these observations may explain the persisting trend, of finding only the older, simpler variants in 'mainstream' computer vision software libraries, this work adopts an approach that preserves the two mentioned properties. Modification of the original algorithm, is restricted to only search strategy replacement, since many drawbacks of RANSAC are consequences of the search strategy it adopts. A second constraint, serving the purpose of preserving generality, is that this 'ideal' strategy, must require no problem-specific priors. Such a strategy is developed, and reported in this dissertation. Another limitation, yet to be overcome in literature, but is successfully addressed in this study, is the inherent variability, in RANSAC.

A few theoretical discoveries are presented, providing insights on the generic robust estimation problem. Notably, a theorem proposed as an original contribution of this research, reveals insights, that are foundational to newly proposed algorithms. Experiments on both generic and computer-vision-specific data, show that all proposed algorithms, are generally more accurate and more consistent, than RANSAC. Moreover, they are simpler in the sense that, they do not require some of the input parameters of RANSAC. Interestingly, although non-exhaustive in search like the typical RANSAC-like algorithms, three of these new algorithms, exhibit absolute non-randomness, a property that is not claimed by any existing variant. One of the proposed algorithms, is fully automatic, eliminating all requirements of user-supplied input parameters. Two of the proposed algorithms, are implemented as contributed alternatives to the homography estimation function, provided in MATLAB's computer vision toolbox, after being shown to improve on the performance of M-estimator Sample Consensus (MSAC). MSAC has been the choice in all releases of the toolbox, including the latest 2015b. While this research is motivated by computer vision applications, the proposed algorithms, being generic, can be applied to any model-fitting problem from other scientific fields.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	***Full Meaning***
AutoCISAC:	Automatic Consecutive-Instance Sample Consensus
BDP:	Breakdown Point
CISAC:	Consecutive-Instance Sample Consensus
CV:	Computer Vision
MCISAC:	M-estimator Consecutive-Instance Sample Consensus
MSAC:	M-estimator Sample Consensus
MSASSAC:	M-estimator Shuffle-and-Sweep Sample Consensus
RANSAC:	Random Sample Consensus
SASSAC:	Shuffle-and-Sweep Sample Consensus

CHAPTER ONE

INTRODUCTION

1.0 Chapter Introduction

This chapter presents an overview of this research work. It begins by explaining the underlying motivation, after which the research problem is described. This is followed by an enumeration of the research objectives. Some theoretical as well as practical significance of this work are then highlighted, followed by clarification of scope of the study and an overview of adopted methodolgy. Finally, a few important terms are defined, as used in this work, before wrapping up the chapter with description of the organization of the rest of this thesis.

1.1 Background and Motivation

Model-fitting has been studied for years. It is a problem commonly encountered in nearly every scientific field. Model-fitting techniques are integral components of toolboxes in domains like machine learning, finance, econometrics, physics, statistics, engineering, and computer vision. The most popular technique used for model-fitting, standard (or ordinary) least squares regression, is however not robust to outlier contamination [1]. An outlier is a data instance that is inconsistent with the model that fits the bulk of a dataset. A single outlier can significantly affect estimates made using the standard least squares method, since each data instance error contributes equally to the error function being minimized to achieve a model of best fit. This led research in statistics to the development of robust estimation techniques.

The robust estimation problem involves fitting to data, an appropriate model which is not misled by presence of outliers. Given data points to which a model perfectly fits, if some of the points are given arbitrary bias, so that they are no longer consistent with the model, a robust estimator is still able to produce the original model. Detailed discussion of concepts, techniques, justification and technical issues surrounding robust estimation are provided in [2] and [3]. A common measure of robustness is the breakdown point (BDP) [1] [4], defined as the threshold of outlier rate beyond which the technique in question is no longer robust to outliers. It is measured as a percentage which holds for all data sizes. RANSAC is one of those relatively few robust estimators with higher BDP than fifty percent. Fifty percent is the limit of the so called high-BDP technique known as Least Median of Squares (LMedS) [5], another robust estimator that has enjoyed high popularity. Others like the M-estimator family [2] [6] have much less BDP. Conventional statistics applications typically require BDP much less than fifty percent, since outliers in such applications are ‘anomalies’ or ‘exceptions’ in the

data. However, the case is often different in computer vision applications, where outliers are only defined with respect to the best among competing models that define geometric transformations between common features in a pair of images.

A common robust estimation problem in computer vision is the image correspondence or registration problem [7] [8], which involves estimating the geometric transformation required to transform one of a pair of images such that common features in both images align. Such a problem is encountered in diverse fields including medical imagery [9], remote sensing [7], Location Detection in environmental planning (LDP) [10] omnidirectional camera model fitting [11], motion estimation [12, 13], motion segmentation [14], autonomous vehicle navigation [15], automated cartography [10]. RANSAC has become a popular and essential component of computer vision software owing to its effectiveness in handling such problems. Besides the fact that RANSAC is one of the most outlier-robust algorithms [1], there are other properties that have made it highly favoured in computer vision. It is simple both in structure and principle, easy to implement and due to its non-exhausitve search approach, it is relatively more efficient compared to many other available techniques [16]. RANSAC has no domain-specific component to it, and can therefore be applied to any model-fitting problem.

Nevertheless, RANSAC has some limitations [16], [17]. It is not guaranteed to find the optimal model. Being a stochastic algorithm, it is not repeatable, that is, multiple runs of the algorithm on the same problem typically yield varying results. There is also no upper bound on its solution time, only a theoretical lower bound for the number of hypothetical models required to be generated for a certain probability of finding a good solution. The more the time allowed, the higher the probability of finding a good solution; and too short a time may result in really bad estimates. There is also the issue of dependence on user-supplied parameters. This list of drawbacks is not exhaustive. The desire to overcome some of these drawbacks has led to a research area that has been very active for about two decades till date. Many variants have therefore been developed. As the second chapter of this thesis reveals, nearly every year in the last one-and-half decade witnessed the introduction of one or more variants. The last two years (2014 and 2015) alone witnessed the introduction more than ten variants.

However, many existing variants compromise two attractive properties of the original RANSAC: simplicity and generality. Some introduce new operations like local optimization and preprocessing, to mention a few, resulting in loss of simplicity. Many of those that do not introduce new operations modify RANSAC's search strategy by leveraging on problem-specific priors, thereby trading off generality and introducing some complexity as well as dependence on other steps of the workflow of applications. These observations may explain

the persisting trend of finding only the older, simpler variants in popular computer vision software libraries. Specifically, the OpenCV library and all releases of MATLAB's computer vision toolbox (1994 - 2015), arguably the two most popular software tools used in the computer vision field, are being referred to. For instance, although MATLAB gets updated twice every year, functions provided for estimating geometric transformations between image pairs (a robust estimation problem) have not changed: the homography estimation function, named *estimateGeometricTransform*, is still based on the M-estimator Sample Consensus (MSAC) till date, while the fundamental matrix estimation function, *estimateFundamentalMatrix*, offers both RANSAC and MSAC as options. Even if the makers of these popular software, have other reasons for sticking with RANSAC and MSAC for so many years till date despite being well aware of recent advancements, the properties that these two 'favoured' algorithms have in common, which the 'unfavoured' ones compromise, are worth paying attention to.

While this research shares with other research efforts the common goal of mitigating RANSAC's drawbacks, the foregoing observations on the 'pitfalls' of many existing approaches and the discrepancy between progress in research and state of the art in popular software, provide the motivation to pursue improvements adopting the peculiar approach described in subsequent sections of this chapter.

1.2 Problem Statement

RANSAC has undergone much development through active research efforts over the last two decades. In fact, for the past fifteen years, nearly every year, a new variant of the original algorithm is published. Yet such progress is not reflected in popular software used in education, research and practice: the persisting trend is to find only the older, simpler variants in software implementations. It is observed that many existing variants compromise two attractive properties of the original RANSAC: simplicity and generality. This is mainly because they seek performance enhancements either by introduction of new operations or by biasing sampling using problem-specific priors. This research focuses on exploring the possibility of avoiding these 'pitfalls', while still mitigating as much of RANSAC's drawbacks as possible. Since many of RANSAC's drawbacks are consequences of its random-sampling search strategy, the approach adopted to preserve simplicity is to restrict modification of RANSAC to only a replacement of its search strategy. This constraint then dictates that an effective search strategy be developed that is inherently capable of producing performance improvements over RANSAC without any additional refinement. A second constraint, serving the purpose of preserving generality is that such a strategy must require no problem-specific

priors. Developing such a search strategy is the main problem addressed in this thesis. Additionally, since the overall concern of this research is to drive forward the state of the art in computer vision software, the need for a comprehensive and up-to-date survey of such a fast paced literature as RANSAC's, is also addressed in this research.

1.3 Research Questions

This research seeks to provide answers to the following questions. The first two questions are addressed in the contributed survey while the rest of the work is devoted to addressing the last question.

i. Taking a holistic look at RANSAC literature, what are the dominant themes; what is/are the most fundamental question(s) in RANSAC research and what is/are the most pressing concern(s) of research efforts?
ii. What directions should future research efforts pursue to achieve algorithms with higher odds of being considered as good candidates for implementation in popular software?
iii. How can the drawbacks of RANSAC be overcome purely by a change in search strategy, without introducing new steps nor dependence on problem-specific priors?

1.4 Aim and Objectives

The primary aim of this study is to drive forward the state of the art in robust estimation, particularly in computer vision software through provision of a comprehensive survey of existing RANSAC variants and development of at least one new algorithm that offers performance advantages over RANSAC, while avoiding the common ''pitfalls'' of introducing new operations or dependence on problem-specific priors. The objectives highlighted below provide guidance in achieving this aim.

i. To review relevant publications and provide organized discussion of variants with analysis of literature to provide answers to the stated research questions.
ii. To develop and implement a search strategy that overcomes limitations of RANSAC's strategy, resulting in at least one variant whose advantages over RANSAC are achieved purely by search strategy replacement.
iii. To carry out extensive empirical testing to evaluate the performance of the resulting algorithm(s) and compare with RANSAC's.

1.5 Contributions and Significance of Study

This study contributes to theory, research and practice of robust estimation in computer vision in a number of ways, highlighted as follows:

The survey of RANSAC literature presented in this thesis, is the most comprehensive and up-to-date review published on this subject. Through such a survey, practitioners will be better armed to make choices for their applications; software makers will benefit from awareness of a more rounded range of options than is currently found in popular software libraries; and comparative studies, a common phenomenon in RANSAC literature, will be better guided.

A central contribution of this thesis is the presentation of theoretical insights on the generic robust estimation problem. A theorem is proposed, referred to in this work as 'the consecutive inliers theorem'. A few other propositions are included. One of the empirical studies reported, reveals interesting results on the automatic threshold estimation problem. These theoretical contributions apply to the general robust estimation problem and therefore create a new world of possibilities in RANSAC research and in robust estimation in general. As exemplified in this work, insights revealed in the consecutive inliers theorem, make possible a number of innovations, which we believe to only be the beginning of its exploration. Five novel algorithms, presented in this thesis, are borne out of these theoretical insights. The algorithms offer performance improvements over RANSAC while avoiding the earlier described 'pitfalls' of many existing algorithms. Three of the five algorithms are completely non-random. This property has never been claimed by any RANSAC variant and is indeed not common among non-exhaustive search algorithms in general.

Some significance also lie in taking these contributions beyond 'theory, algorithm and experimental results' to actual ready-to-use software implementations that can benefit practice, research and teaching in computer vision. Two of the algorithms are implemented as contributed alternatives to the homography estimation function provided in MATLAB's computer vision toolbox, a widely used software in this field, after being shown to improve on the performance of M-estimator Sample Consensus (MSAC), the variant used in the official implementation, including the 2015b release.

1.6 Scope and Limitations

For the reasons described in the background section, this thesis considers improving RANSAC only by replacing its search strategy. Otherwise, the proposed algorithms could also benefit

from many other enhancements from which RANSAC has benefited, such as preprocessing, local optimization, partial evaluation, and many other enhancement methods found in literature. But the focus is to address this 'most fundamental' problem first, having noted it to be the root of many of RANSAC's drawbacks, before considering other enhancements in future works, if at all necessary. Therefore exploration of these other directions is outside the scope of this research. This delineation reflects in the various experiments reported, in the sense that the proposed algorithms are compared to only pure-random-sampling variants like RANSAC and MSAC.

1.7 Methodolgy

The validity of the 'consecutive inliers theorem' proposed in this thesis is established through a concise proof. Properties of the proposed algorithms are studied and the claimed advantages are verified empirically. Empirical studies reported involve line-fitting, affine transformation and projective homography estimation problems. Simulated datasets are used to create a wide range of data conditions, and in situations where the results from algorithms need to be benchmarked with known ground truth. Homography estimation problems are also studied using real-life image pairs consisting of aerial maps of the Westville campus of the University of Kwazulu-Natal (UKZN) and its neighbourhood. These maps were sourced from Google maps.

Comparative analyses are performed using descriptive statistics as well as relevant visualizations. Statistical hypothesis tests are also employed. The Friedman test is used to test for significant difference in relative ranking of algorithms run on the same set of problems after which Nemenyi tests are conducted for pairwise comparisons. The purpose of these tests is to be able to infer reliable generalizations about performances of algorithms.

All software implementations are done in the MATLAB language, while statistical hypothesis tests are performed using facilities in the R software.

1.8 Definition of Terms

A few terms are defined here, as used throughout this thesis, which are consistent with popular language in RANSAC literature.

Simplicity: an algorithm is described as simple in this thesis, if it does not include any extra steps or operations such as local optimization, preprocessing and preliminary hypothesis tests.

Such an algorithm, like RANSAC or MSAC, simply involves using a search mechanism to optimize an objective function used to measure model quality. The more extra steps an algorithm contains, the more it is said to violate simplicity. While this may not necessarily result in performance disdvantages, inclusion of extra steps generally increase difficulty of implementation.

Generality: this refers to the absence of problem-dependent component. Generic algorithms, like RANSAC, do not depend on any prior information that is problem-specific. The value of generality is that such an algorithm can be used for model-fitting on any kind of data, rather than being limited, for example, to image data. Also such an algorithm is independent of the outcomes of other steps in the workflow of applications.

Repeatability: this is a measure of the ability of an algorithm to return the same (or close) results upon multiple runs on the same problem. As far as RANSAC literature is concerned, synonyms include consistence, non-randomness, stability, and precision.

Instance: a row in a data table; a point in data space.

Outliers: instances that are inconsistent with a model that fits the bulk of a dataset; instances that are remarkably different from the bulk; instances with error greater than a given threshold. All three definitions mean the same thing.

Outlier rate: also referred to as contamination level, is the fraction or percentage of data instances that are outliers.

Inliers: the opposite of outliers.

Hypothesis: a model constructed at some point in time during the run of an algorithm. It is also referred to as a solution or hypothetical model. This should not be confused with the use of the same word in the context of inferential statistics in chapter 4, where the null and alternate hypotheses represent complementary inferences about a population.

Consecutive instances: instances that follow each other (contiguous or adjacent) *in the data table.*

Minimal sample size *n*: the minimum number of instances required to fit a model of interest. For example, for a line-fitting problem, $n = 2$, for an affine transformation $n=3$, for a projective homography $n = 4$.

Data size m: number of instances in a dataset. In the computer vision sense, this is the number of matches.

1.9 Organization of Thesis

The rest of this thesis is organized as follows:

Chapter 2 presents an introduction to the working principles and limitations of the RANSAC algorithm followed by a review of robust estimation techniques and a comprehensive survey of RANSAC variants.

Chapter 3 presents the 'consecutive inliers' theorem and a proof to establish its validity. The five novel algorithms are described in this chapter, being applications of insight provided by the theorem. Also presented is a study on the problem of automatic estimation of distance threshold, an important parameter in RANSAC-like algorithms and how the theorem leads to innovation in addressing this problem.

Chapter 4 presents comparative studies involving six algorithms: four of the new ones and two random-sampling algorithms found in popular software, RANSAC and MSAC. Performance criteria include accuracy, efficiency, and repeatability (non-randomness).

Chapter 5 takes the contributions of this work a step further. Two of the algorithms are implemented as contributed alternatives to the homography estimation function provided in MATLAB's computer vision toolbox, having being shown in chapter 4 to offer performance advantages over M-estimator Sample Consensus (MSAC), the variant used in the official implementation. The chapter includes a user guide written in the official MATLAB documentation style which includes description of syntax, input arguments and output arguments for both functions.

Finally, Chapter 6 presents summary, conclusion, and directions for future works.

CHAPTER TWO

LITERATURE REVIEW

2.0 Chapter Introduction

This chapter presents an introduction to the working principles and drawbacks of the RANSAC algorithm followed by a brief overview of robust estimation techniques in general. These form a prelude to a survey of RANSAC variants which is the main contribution of this chapter. To the best of the author's knowledge, currently, this is the most comprehensive survey published on this subject. The underlying motivation is that besides development of new variants, which is the concern of subsequent chapters, collection and organized discussion of existing variants may be equally as valuable in driving forward the state of the art. The survey includes some analysis of literature which aim to answer a few interesting questions in order to provide researchers with holistic understanding of this fast-growing field. Research trends are observed and an attempt is made to cast in a single sentence, what the survey reveals to be the most fundamental research question. Also, to aid software production process which would typically involve studying variants in much more detail, quantitative and qualitative approaches are proposed to guide prioritization of original works to be studied. The survey concludes with identification of gaps and recommended directions for future research efforts.

2.1 Robust Estimation: Model Estimation in the Presence of Outliers

Perhaps the most popular model-fitting technique in most scientific fields is the least squares technique. It achieves model-fitting by minimizing the sum of squared residuals. A single outlier can significantly affect its estimates, since each data instance error contributes equally to the error function being minimized to achieve a model of best fit.

As already defined in chapter 1, an outlier is a data instance that is inconsistent with the model that fits the bulk of a dataset. Inliers are those that are consistent with the model. Therefore an instance must fall into one of the two categories. The robust estimation problem involves fitting to data, an appropriate model which is not misled by presence of outliers. Given data points to which a model perfectly fits, if some of the points are given arbitrary bias, so that they are no longer consistent with the model, a robust estimator is still able to accurately produce the original model.

2.2 Measuring Robustness: Breadkdown Point

A common measure of robustness of an estimator is the breakdown point (BDP). It is defined as:

$$BP_M(D_I) \quad \stackrel{def}{=} \min\{\frac{m}{|D_I|} : bias_M(m; D_I) = \infty\}$$

where $bias_M(m; D_I)$ is the maximum perturbation associated with a model *M* and set D_I, that leaves estimates unchanged [4].

Intuitively, the BDP of an estimator is the fraction of data that can be given arbitrary values without making its estimates arbitrarily bad. It is typically measured as a percentage of the data size. The percentage holds for all data sizes. RANSAC is one of those relatively few robust estimators with BDP beyond fifty percent. Fifty percent is the limit for so called high BDP technique known as Least Median of Squares (LMedS), another popular technique. Others like the M-estimator family have much lower BDP. While fifty-percent BDP is sufficient for most statistics applications, since the outliers in such applications are 'anomalies' or 'exceptions' in the data, the case may be different in computer vision applications, where outliers are only defined with respect to the best among competing models.

2.3 Robust Estimation as a Combinatorial Optimization Problem

Consider taking a sample of *m* data points, where *n* is the minimum number of points required to define a model of interest. If there are *m* instances in the data, then the possibilities are $\binom{m}{n}$ in number. If an appropriate objective function is used to evaluate each resulting model, then the model that best fits the data can be selected. Such an approach to model fitting is robust to outliers without first detecting and deleting them. This is because each model is constructed from a minimal sample set. It is easy to show that the best model will be constructed from a minimal sample set that are all inliers. Clearly, this is a combinatorial optimization problem.

However, the immediate problem with this approach is that which plagues most practical combinatorial optimization problems. The space of all possible models, also known as the solution space, easily gets large, making exhaustive enumeration infeasible for even relatively small sized problems. Fischler and Bolles [10] introduced RANSAC to handle this problem. RANSAC and its variants, have since become very popular in computer vision, a field rife with extreme robustness requirements.

2.4 The RANSAC Algorithm

RANSAC is simple in structure, yet is is very powerful and effective. The structure of the algorithm is summarized as follows:

2.4.1 Outline of the RANSAC Algorithm

1. Repeat a and b until sufficient number of trials
    a. Randomly select minimal sample set and fit hypothetical model. The minimal sample set consists of the minimum number of data instances required to define a given model. For example, when fitting a straight line, the minimal sample size is two; three for an affine transformation; four for projective homography; and so on.
    b. Evaluate hypothesis –criterion: number of data instances consistent with the hypothetical model (a number of other functions have been proposed)
2. Return the best model

The cost function which RANSAC seeks to minimize is stated as follows:

$$\rho_1(e^2) = \begin{cases} 0, & e^2 < T \\ T^2, & e^2 \geq T \end{cases}$$

where e is the point error and *T* is the error threshold used to distinguish inliers from outliers.

The optimization problem that RANSAC seeks to solve is:

$$\hat{\theta} = \underset{\theta}{\arg min} \sum_{i=1}^{m} \rho_1(e^2, \theta)$$

where θ is the estimate of model parameters and *m* is the data size.

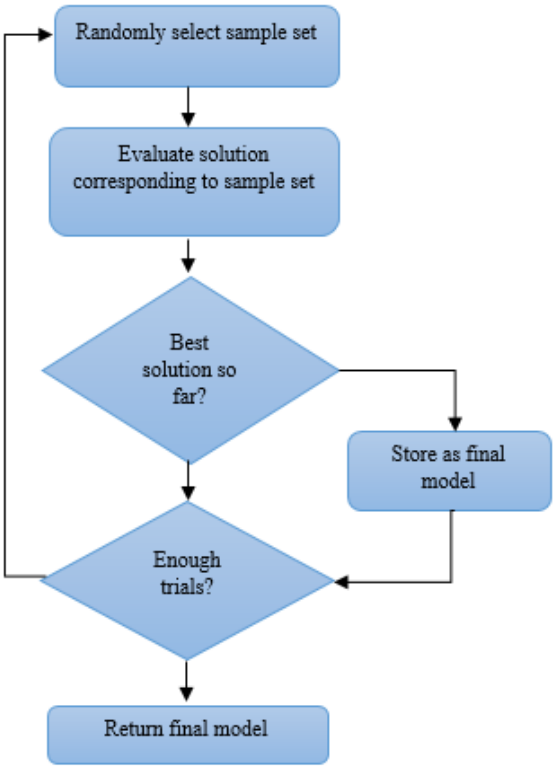


Figure 2.1: Flowchart of RANSAC

2.4.2 Drawbacks of RANSAC

Listed below are some of the drawbacks of the RANSAC algorithm. Although this list of drawbacks is not exhaustive, it contains most of the drawbacks that have been widely identified and studied.

1. Efficiency: There is no upper bound on its solution time, only a theoretical lower bound for the number of hypothetical models required to be generated for a certain probability of finding a good solution. The more the time allowed, the higher the probability of finding a good solution; and too short a time may result in bad estimates.
2. Convergence: RANSAC's solutions do not improve progressively as more iterations are performed. An iteration is simply a random trial which may compare arbitrarily with previous trials. The final solution returned is simply the best found so far. This has implications on efficiency too, as the best solution may have been found much earlier than the termination condition dictates, resulting in much waste of time, but RANSAC has no way of knowing this in order to terminate early enough.

3. Accuracy: As is typical of stochastics algorithms, RANSAC is an approximation algorithm. There is therefore no guarantee of finding the optimal solution.
4. Repeatability: Being a stochastic algorithm, it is not repeatable. That is, multiple runs of the algorithm on the same problem typically yield varying results. Although given an appropriately computed number of trials, the probability of a good solution is high. However, it is possible for RANSAC to return a bad solution.
5. Lack of robustness to degenerate configurations.
6. There is also the issue of dependence on user-supplied parameters.

2.5 Homography Estimation

Any pair of images sharing the same surface can be related by a homography if we assume a pin-hole camera model. In other words, homography is a special case of the fundamental matrix. Such models are applied in problems like image registration, image rectification, camera-motion estimation, image mosaicking, video stabilization, image in-painting, and so on.

Given point correspondences, x' and x, a homography transformation H satistfies:

$$x' = Hx$$

H is 3 by 3.

In the most general homography case, at least four point pairs are required to define a model.

The various kinds of homography include similarity, affine, and projective homographies. Under homographic transformations, quadrilaterals are transformed into quadrilaterals.

2.5.1 Non-Refective Similarity

This may include one or more of rotation, scaling and translation. This implies that shapes and angles are preserved, when such a transformation is applied. Such a model requires two points pairs to define it.

2.5.2 Affine Transformation

A similarity transformation is a special case of affine transformation. In addition to the possibility of rotation, translation and scaling, affine transformation may include shearing. Though straight lines remain straight as in similarity transformation, shapes and angles are not preserved. Affine transformations require a minimum of three point pairs to define the model.

2.5.3 Projective Transformation

This is the most general case of homography. Affine transformations are special cases of projective homography. The only constraint in projective homography is that quadrilaterals map to quadrilaterals. Projective transformations require a minimum of four poin pairs to define the model.

Given point correspondences (x, y) and (u, v), affine and similarity transformations are given as:

$$[u\ v] = T[x\ y\ 1]^T$$

Projective transformation T is given by:

$$[up\ vp\ wp] = T[x\ y\ w]$$

where $u = \frac{up}{wp}$, and $v = \frac{vp}{wp}$.

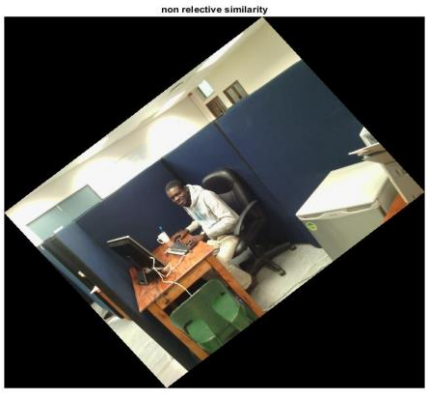
Equivalently,

$$u = \frac{Ax + By + C}{Gx + Hy + I}$$

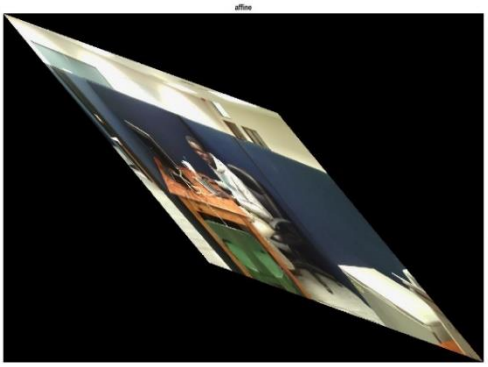
$$v = \frac{Dx + Ey + F}{Gx + Hy + I}$$

$$T = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix}$$

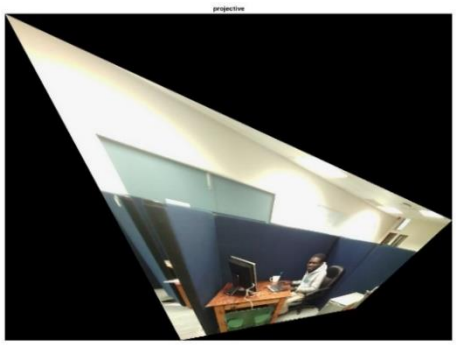
(a) Original (untransformed)



(b) Similarity



(c) Affine



(d) Projective

Figure 2.2: Homography transformations illustrated

2.6 On Existing Reviews of RANSAC Variants

A few attempts have been made in the past by some authors to provide organized discussions of RANSAC variants. Choi et al [16] presented one such discussion under three themes according to improvement focus – accuracy, speed and robustness. The discussion by Raguram et al [18] highlights three other improvement themes which are all related to improvement of speed – optimizing model verification, improving hypothesis generation, preemptive RANSAC strategy, and local optimization. Their discussion culminated in the discussion of an algorithm representing a fourth subject – adaptive real-time RANSAC. A more recent review [17], contributed by this same Raguram and four other authors, each of whom had in the past

developed some of the most successful variants. Their goal is to present the idea that each variant that exists can be treated as a special case of RANSAC under specific practical and computational considerations. They therefore present an integrated framework named Universal RANSAC (USAC) that aggregates the strengths of a number of variants each constituting a module in the framework. Their review is a discussion of the functional requirements of each module, and the various options that exist in literature for meeting the requirements, leading to their choices in the final implementation of USAC. This framework is discussed in further detail in section 2.9.6.

The survey presented in this chapter is substantially more comprehensive and up-to-date, than any of the above, in terms of coverage of variants as well as discussion of themes. It treats in more depth, the subjects of holistic literature understanding, gap identification and provision of guidance for future research. Chronological analysis of literature is also presented. Approaches to identifying 'classics' to aid prioritization of works for very detailed variant studies, such as may be required in software development settings, are also suggested. Indeed the goal is to provide a roadmap for navigating this vast literature.

2.7 Brief Overview of the General Field of Robust Estimation

This overview begins with robust estimation techniques used in the broader statistics community, before narrowing focus to those used in computer vision. Though the list of algorithms represents a good coverage of techniques in these fields, it is not exhaustive.

As mentioned earlier, the field of robust estimation is concerned with tackling the problem of avoiding outlier-misled model-fitting. Many robust estimators have been developed. One measure of the level of robustness of these techniques is known as BDP, defined as the maximum outlier contamination level that an estimator can resist without arbitrarily large bias in the estimates. One popular category of techniques is the M-estimate-type techiques such as Huber, Tukey's bisquare, Andrew's, Fair's, Welsch's, Cauchy's, and Talwar's weighted M-estimators. Techniques with higher BDP than M-estimators include LmedS [5] technique which performs estimation by minimizing the median of squared residuals. Least Trimmed Squares (LTS) [19] which minimizes the sum of squared residuals computed using fixed cardinality subsets, and S-estimates, which minimize the variance of residuals. In the field of computer vision, a field rife with robust estimation problems with high BDP requirements, popular techniques used include LmedS, Minimize the Probability of Randomness (MINPRAN) [20], and RANSAC. RANSAC seems to be the most popular.

The High-BDP estimators mentioned – LMS, LTS, MINPRAN, RANSAC – are all capable of accurate model estimation even in the presence of up to fifty percent outliers. Roussew argues that the highest breakdown limit is 50% [20], on the basis that higher contamination may result in having outliers that 'conspire' to fit well to a model which becomes the best-fitting model due to the number of outliers rather than being the 'right' model. However, if it is assumed that this 'conspiracy' does not arise, then the robustness of the high BDP algorithms can still be compared on contamination beyond fifty percent. While an algorithm like LmedS would breakdown beyond 50% contamination, RANSAC can exhibit robustness to contamination beyond 50%. Other robust estimators used in the field of computer vision are discussed in the next subsection.

2.8 Robust Estimation Techniques in Computer Vision

There are two broad categories of techniques, namely, stocahatic techniques and deterministic alternatives.

2.8.1 Stochastic Techniques

Stochastic techniques are essentially non-exhaustive search techniques. One group of such algorithms found in robust estimation literature, involve optimization of some function of residuals over several generated models. RANSAC falls into this category. It generates hypothetical models and maximizes the cardinality of the inlier set. MSAC [21], a RANSAC variant, maximizes an error function formulated in the M-estimate framework. MLESAC [22] implements the same approach as MSAC using a slightly different error function. MAPSAC [23] maximizes *a posterior* probability of inlier set. Many other variants, discussed later in this work, differ from these in their strategy for generating and verifying models, but generally employ one of these optimization objectives. Generally, algorithms that belong to the RANSAC family depend on the user to supply of an appropriate distance threshold used by the algorithm to distinguish between outliers and inliers.

Several other techniques exist that optimize other functions of residuals, which are not dependent on the supply of threshold. One popular technique is the earlier mentioned LmedS which minimizes the median of squared residuals. However, its BDP is lower than those of the RANSAC family: it breaks down when data contamination exceeds fifty percent. Minimum Probability of Randomness (MINPRAN), minimizes the probability that a combination of model and corresponding inliers occurred by chance. While it does not require noise threshold to be supplied, it assumes the knowledge of the dynamic range of the outlier

data. Another robust estimator is the Minimum Unbiased Estimate (MUSE) [24], which minimizes order statistics of the squared residuals. It has been noted to have limited outlier robustness [17]. Projection-based M-estimator (pbM) [25], computes threshold automatically by approaching the M-estimator objective of MSAC, as a projection pursuit problem. But it is noted to be a computationally expensive technique, especially as the model complexity increases [17].

Table 2-1: Optimization Objectives for High-BDP Estimators in Computer Vision

Technique	**Objective**
LMedS	Minimize median of squared residuals [5]
LTS	Minimize least squares over fixed-sized subsets of data [19]
RANSAC	Maximize support i.e. inlier set cardinality [10]
MSAC	Maximize inlier set likelihood by minimizing M-estimate error
MLESAC	A different version of MSAC's objective [21]
MAPSAC	Maximize a-posterior probability of inlier set [23]
MINPRAN	Minimize chance probability of inliers- model set [20]
MUSE	Minimize order statistics of squared residuals [24]
pbM	Same objective as MSAC formulated as projection pursuit [25]

Raguram et.al [17] note that the foregoing techniques that optimize functions of residuals, generally rely on one or more user-supplied parameter(s). Those which do not rely on supply of the threshold, rely on the supply of the number of hypotheses to be tested. This they argue, relies in turn, on knowledge of the inlier rate, or a worst case assumption that guarantees success for the worst inlier rate possible, for the given problem. They argue that these user inputs can be difficult to compute. One technique that avoids these limitations, is the Residual Consensus (RECON) Algorithm. It adopts a different paradigm, testing pairs of models for consistency. The heuristic that is the basis for this approach is that the residuals for good

models are likely to be consistent with each other. This is of course, computationally expensive, since hypothetical models have to be paired.

Another group of stochastic algorithms are those that detect inliers based on the distribution of residuals. The approaches in this category work based on the idea that the distribution of residuals with respect to a sufficiently large set of randomly selected models can reveal which points are outliers or not. Such techniques are also very well suited for multi-model problems. While this approach has been shown to be an effective approach, it depends on the generation of a sufficient number of hypothetical models, which can be a limitation in terms of computational complexity. Examples of techniques that belong in this category include J-Linkage [26], Ensemble Method [27], and Kernel-Fitting [28].

2.8.2 Deterministic Techniques

A few robust estimation approaches exist that are deterministic, in that they do not explore the solution space in a randomized way. Examples are Joint Compatibility Branch and Bound [29], Active Matching and Consensus Set Maximization algorithm of [30]. The first two leverage on prior information on location of image features, while the last reformulates the consensus set maximization objective of RANSAC as a mixed-integer programming problem, solving it using a branch and bound technique. Olsson et.al [31] presented an approach based on computational geometry theory. They propose an $O(k^{n+1})$ polynomial time algorithm, where k is the number of matches and transformation space is of n degrees of freedom.

Litman et. al [32] point out that the foregoing method could not be used in practice for spaces of more than a few degrees of freedom. A three-fold contribution is made by these authors, in a work which is an interesting entry at the Computer Vision and Pattern Recognition (CPVR) 2015 conference. First is the introduction of a scheme for efficient sampling of the space of transformations. The second contribution is an algorithm that finds the best transformation given the inlier rate. The last is an algorithm that estimates the inlier rate without explicitly detecting them. The authors consider the last as their main contribution, noting that without it the rest of the framework has no practical applicability. In the approach of the framework, the best transformation is found using a branch-and-bound technique. The authors introduce a quantity v(p) that depends on the sample density E, and is a function of the inlier rate p. The main insight of their paper, they note, is that this quantity, which is easy to compute, attains a minimum at the 'true' inlier rate p*. They further established the existence of this minimum, theoretically. Using the branch-and-bound approach, the technique minimizes the error of data points, given this estimate inlier rate.

Broadly speaking, one advantage of these deterministic techniques is that they avoid dependence on inlier error threshold. However, a major drawback of these class of algorithms is that they are generally computationally costly, which is a major problem in practical applications. This may be a reason why they are not as popular or accepted as the RANSAC family, in computer vision software and practice.

2.8.3 On the Popularity of Stochastic Algorithms in Practice

A survey of popular computer vision software reveals that deterministic techniques are not very popular compared to stochastic techniques. Due to the nature of solution spaces in real-life computer vision applications, it is found that existing deterministic techniques are generally limited in the kinds of problems they can handle. Generally speaking, in the light of practical applications they are too costly computationally, though some of these algorithms give guarantees of global optimum in cases they can handle. It is no wonder therefore, that state-of-the-art software opt for their stochastic counterparts which adopt non-exhaustive search, among which the RANSAC paradigm is favoured. Good results are still achievable, and a wide range of practical problems can be solved by such an approach.

2.9 Survey of RANSAC Variants

Since its introduction in 1981, RANSAC has been through various developments resulting in quite a number of variants. Each variant is designed to improve the performance of the basic algorithm, in some way. As mentioned in the introductory section of this paper, Choi et.al [16] identify the main research directions as improvement of accuracy, speed and robustness. Their discussion reveals that by robustness, they refer specifically to low sensitivity to poor choice of threshold. Similarly, Raguram et.al [17] identify the performance concerns efficiency, accuracy, and lack of robustness to degeneracy. The functional themes with which their work is concerned include prefiltering matches for better efficiency, (guided) sampling, model checking, partial evaluation, preemptive verification, handling degeneracy, and local optimization. In [18], a discussion is presented, of speed-related themes: optimized model verification, improving hypothesis generation, preemptive RANSAC strategy, and local optimization. The survey that is presented in the sections that follow builds upon these existing works and is more comprehensive and up-to-date than any of them.

2.9.1 Pursuit of Improved Accuracy

Accuracy, in RANSAC literature, is used to convey either of two related concepts. The first refers to effectiveness of the technique adopted for evaluated hypothetical models. The second use of the term accuracy refers to the effectiveness of the search strategy used by an algorithm to explore the space of possible hypotheses given any of the optimization objectives. Both uses of the term are related since both the search strategy and the optimization objective interact to produce the final estimates returned by the algorithm. Therefore accuracy generally bores down to mean correctness of the estimates returned by an algorithm. The foregoing definitions are simply different directions authors looked in order to improve RANSAC's accuracy. Other strategies is found in literature, for improving accuracy, include local optimization and adopting of appropriate stopping criteria. Local optimization involves refining the initial estimates of RANSAC using a local optimization algorithm. Since RANSAC performs time-constrained optimization, the point of using a stopping criterion is to compute the lower bound on the number of trials required for a certain probability of selecting good solutions.

Figure 2.1 is a visual representation of the various approaches that have been adopted in pursuit of improved accuracy.

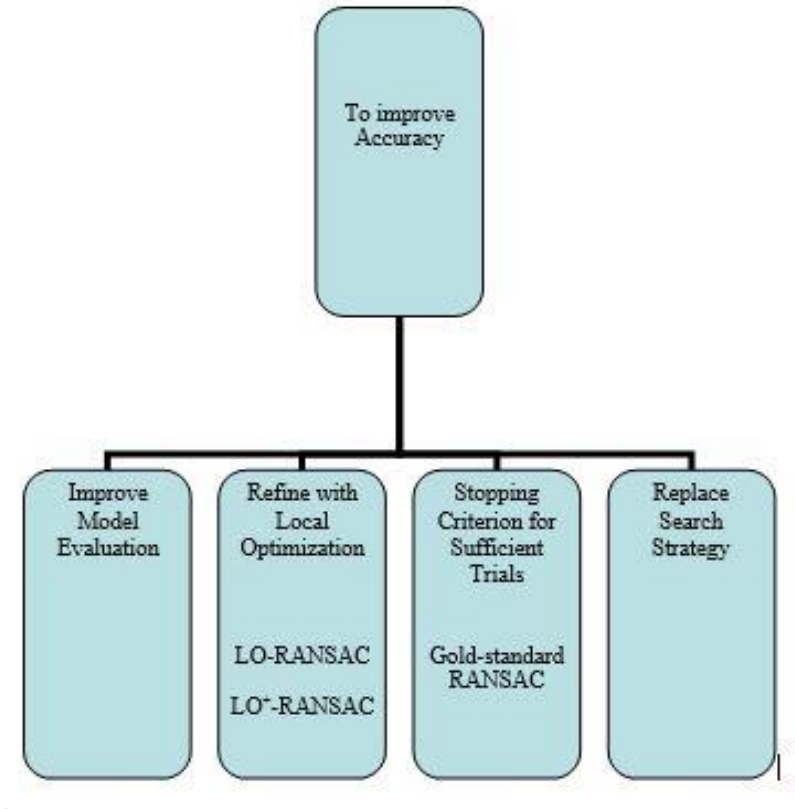


Figure 2.3: Strategies for Improving Accuracy

Figure 2.4 extends the diagram in Figure 2.3 to show a few subthemes that have been studied in the pursuit of effective model evaluation.

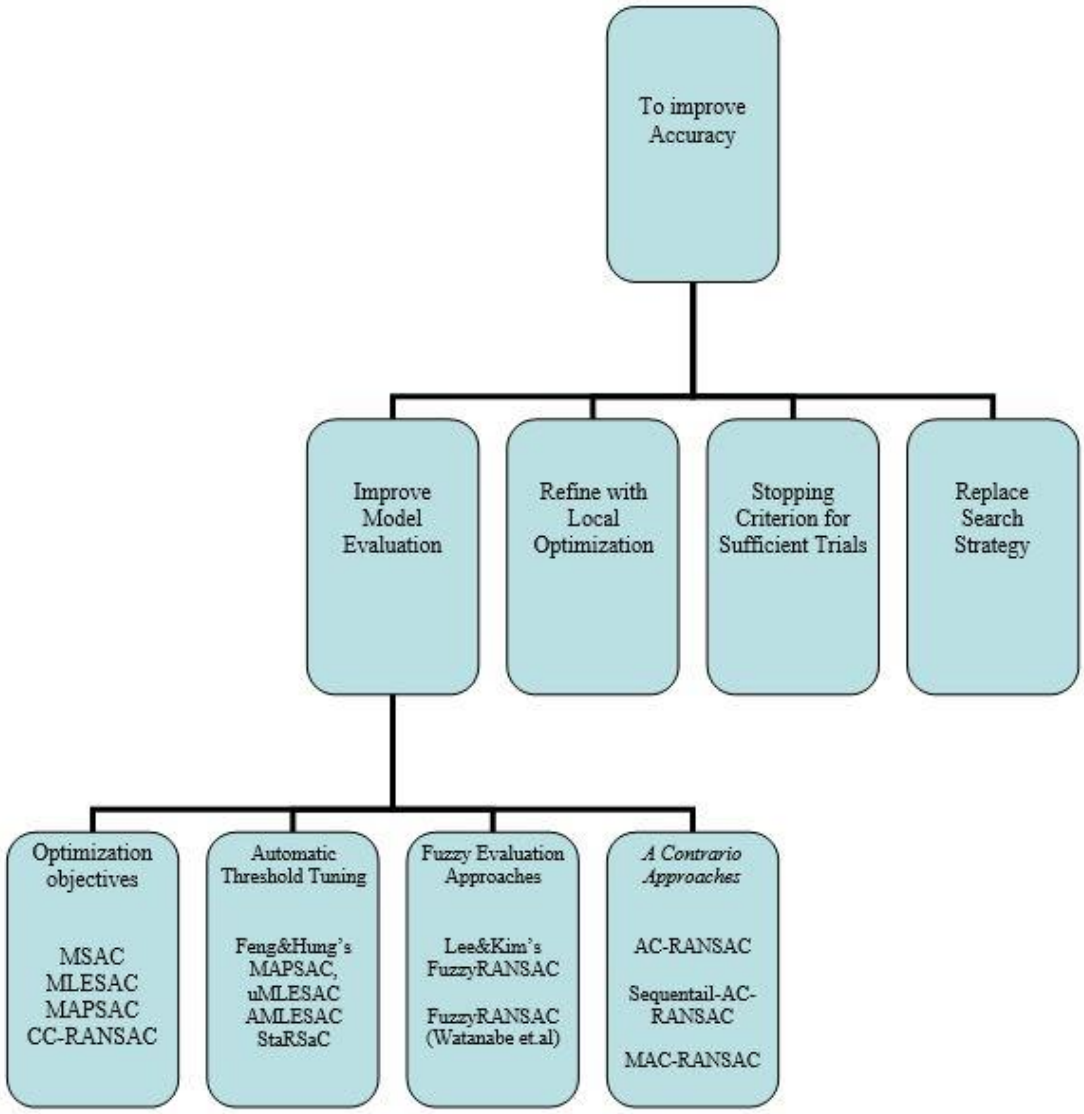


Figure 2.4: Approaches to Model Evaluation

2.9.1.1 Model Quality Measures and Optimization Objectives

As described in chapter 2, the original RANSAC seeks to maximize the cardinality of the consensus set. The assumption inherent in this objective is that the best model that fits the data is the one that records the highest number of inliers. This assumption does not always hold. MSAC [21] and MLESAC [22] replace this objective with slightly different objective functions

in the M-estimate framework, both proposed by Torr and Zisserman. Intuitively, and loosely speaking, these functions seek to maximize the tightness of inliers, round the model. MAPSAC [23] employs a Bayesian approach, maximizing *a posterior* probability of inlier set.

In a paper published in 2010, Gallio et.al [33], noted the unreliability of RANSAC in situations with clustered patches of limited extent. In such cases, a single plane crossing through two such patches may contain more inliers than the correct model. This happens with images containing structures like steps, curbs ramps, and so on, in range sensor applications. The focus of the authors is to mitigate the effect of such unreliability for safe parking of cars and robot navigation. A modification of RANSAC, named CC-RANSAC is therefore proposed. The difference between CC-RANSAC and RANSAC is that, instead of evaluating hypothetical models based on the total cardinality of consensus set as RANSAC does, CC-RANSAC's objective is to maximize the connected components of inliers. An assumption of the algorithm is that inliers cluster together into one large connected component. While the authors argue that this holds for the concerned applications, they admit the necessity for further investigation.

2.9.1.2 Threshold Selection Techniques

As mentioned earlier, the distance threshold used for distinguishing outliers from inliers, is an important parameter required by RANSAC and many variants. When the threshold is set to a value that is too large, the algorithm becomes highly susceptible to noise, and outliers may be regarded as inliers. This is because of the low discrimination of the algorithm between points. On the other hand, when the threshold is set to a value that is too small, many inliers will be falsely rejected as outliers. Both cases result in bad estimates. Therefore selecting an optimal threshold value is an important step towards achieving accurate model estimates. Under this heading, existing approaches to selecting this crucial parameter are discussed.

Empirical Approach

In many situations it is possible to make a reasonable empirical choice of the distance threshold. This is probably still the most common approach in practice. A proabale reason may be the fact that in many situations, it is fairly easy to determine the threshold by experiments. In modern computer vision software like Matlab 2015b and OpenCV, implementations of the RANSAC algorithm for estimating homographies and fundamental matrices require the user to supply a value for this parameter.

Theoretical Approach

While it is possible in many applications to choose the value of the distance threshold by experimentation, a more formal process can be adopted [17]. The approach assumes Gaussian noise with zero mean and standard deviation σ. The point-model error d^2 can therefore be expressed as a sum of *n* squared Gaussian variables, *n* being the co-dimension of the model. The residuals follow a chi-square distribution with *n* degrees of freedom. The inverse chi-square distribution can be used to determine a threshold *t* that captures a fraction α of the true inliers:

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

Where χ is the cumulative chi-square distribution, and α is the confidence level, that is, 1-α is the probability of incorrectly rejecting a true inlier.

Automatic Tuning

Some variants have been developed to avoid dependence on user-supplied distance threshold. These variants incorporate techniques for estimating the threshold. One such variant published in 2003 - Feng and Hung's MAPSAC [25] - performs simultaneous estimation of inlier rate and threshold, using a mixture model of Gaussian and uniform distributions and computation of the transformation by minimizing the 2D projection error. Another variant uMLESAC [25] is a user-independent version of MLESAC that uses expected maximization (EM) for automatic estimation as well as adaptive termination using failure rate and error tolerance. AMLESAC [34], uses uniform search and gradient descent to estimate the threshold, and EM to estimate the inlier rate, whilst including local optimization in its framework. StaRSaC [35] achieves automatic estimation using a measure known as 'variance of parameters' (VoP) to compute a stable range of solutions over a pool of transformations. The underlying principle of StaRSaC is that a threshold value that is too small, produces a tight fitting, unstable solution. The degree of instability increases with both the variance of the uncertainty and with the number of outliers. Similarly, a threshold that is too large produces fits that are also unstable due to the influence of outliers erroneously treated as inliers. The observation of the authors is that there exists a region or range of distance threshold values, typically wide, that produce stable solutions. Once this region is found, then the model that maximizes the RANSAC objective is chosen. So, basically, StaRSaC runs multiple RANSAC's using various thresholds and choses the one that minimizes the VoP.

While the advantage of automatic tuning may be obvious, in that it makes the algorithm independent of the user, it generally comes with significant increase in computational cost and runtime.

2.9.1.3 Threshold-Independent Model Evaluation

Besides automatic threshold estimation, other approaches have been proposed for achieving user-independence. Two such approaches are discussed under this heading, namely, *a contrario* approaches and fuzzy approaches.

A Contrario Approaches

Moisan and Stival [36] propose a computational definition of rigidity along with a probabilistic criterion to rate the meaningfulness of a rigid set as a function of the number of matched pairs, as well as the accuracy of the matches. This criterion, they argue, yields an objective way to compare precise matches of a few points, and make inference about a larger set. It guarantees that the expected number of meaningful rigid sets found by chance in a random distribution of points is as small as desired. The basic idea of the *a contrario* approach is to combine RANSAC with a hypothesis testing framework, as a way of avoiding dependence on threshold selection.

According to Rabin et.al [37], who refer to the variant as *a contrario* RANSAC (AC-RANSAC), this technique has the advantage of allowing the automatic tuning of parameters without any *a priori* on the distribution of inliers. They further extend AC-RANSAC to develop Sequential AC-RANSAC and MAC-RANSAC. Sequential AC-RANSAC extends AC-RANSAC to the case of multi-model estimation while MAC-RANSAC combines Sequential AC-RANSAC with spatial filtering and transformation fusion detection with a fusion splitting criterion.

Fuzzy Techniques

Some authors have proposed the use of fuzzy techniques for avoiding drawbacks of conventional threshold-dependent RANSAC. Variants that belong in this category evaluate models based on membership functions of a fuzzy set. One such variant which incoporates fuzzy theory into RANSAC, is the fuzzy RANSAC algorithm proposed by Lee and Kim in 2007 [38]. It classifies samples as good, bad and vague. Good sample sets are those whose degree of inlier membership is high and the rate of membership change is small. Bad sample sets are those whose degree of inlier membership is high and the rate of membership change is small. Vague sample sets are those whose rate of membership change is large without relation to any degree of membership. The algorithm then improves classification accuracy, omitting outliers by iteratively sampling only from good sets.

Watanabe proposed another fuzzy RANSAC algorithm [39] which combines a fuzzy model evaluation approach with extended sampling method based on reinforcement learning. They argue that RANSAC's model estimation precision can be improved by increased variation in

the size of samples from which hypothetical models are constructed. This however, increases the size of the solution space. They propose a Monte-Carlo sampling, performed in proportion to evaluation values, which is learned using reinforcement learning. The claim of their work, substantiated by homography estimation experiments, is that the technique is more accurate and efficient than RANSAC for the cases tested.

2.9.1.4 Local Optimization

The basic idea of local optimization is to use an optimization algorithm to refine the solution from the basic RANSAC in a depth-first manner. This can be any suitable optimization algorithm. The goal is to resort to locally optimize the best solution returned by RANSAC within a reasonable number of iterations. This strategy was proposed by Chum et al [40] and the resulting variant was named accordingly: Locally optimized RANSAC (LO-RANSAC). They proposed this in response to the problem found in the fact that the number of iterations required for RANSAC to produce near-optimal results, in practice, is usually much higher than the theoretically computed lower bound. Their work points out the fact that while the optimal solution must be constructed from an all-inlier sample, an all-inlier sample does not necessarily produce an optimal solution. Four different approaches to local optimization were proposed by the authors, tagged simple, iterative, inner-RANSAC, and inner-RANSAC with iteration. The simple local optimization strategy applies a linear optimization algorithm to all data points judged by basic RANSAC as inliers. Iterative local optimization, as the name suggests applies a linear algorithm iteratively, while the threshold is being reduced per iteration. Inner-RANSAC applies RANSAC successively to initially detected inliers, without requiring samples to be minimal. In the iterative inner-RANSAC, each run of inner RANSAC is processed using the iterative local optimization procedure.

Nine years after LO-RANSAC was published, another work was contributed, named LO$^+$-RANSAC [41]. It has two (out of three) authors in common with LO-RANSAC. It is an improved version of LO-RANSAC. Two key contributions of this work are the use of a truncated quadratic cost function and an introduction of a limit on the number of inliers used for the least squares computation. They show through experiments that the algorithm is quite stable. That is, less random; precise under a broad range of conditions; less sensitive to the choice of inlier-outlier threshold; and is better for initializing bundle adjustment than the gold-standard RANSAC.

Though the 'LO' variants do not offer guarantees of global optimum and absolutely repeatable results, they achieve very significant improvements, which are still competitive in literature to date. These improvements come with additional computational burden.

2.9.1.5 Search Strategy Change for Improved Accuracy

RANSAC's search strategy is stochastic. There is therefore no guarantee of finding the global optimum or even a good solution for that matter. However, the probability of finding a good solution increases, as the number of trials is increased. The problem is that there is no upper bound on the time it takes to find a good solution. This problem becomes more pronounced in applications where a large number of trials cannot be allowed, such as in real-time applications.

One way to address this problem, is to bias sampling towards hypotheses that are more likely to be good, so that they are selected earlier, and with higher priority, than less promising ones. The earliest, and probably the most popular category of variants that sought to achieve better speed and efficiency are known as guided-sampling algorithms. But many other search paradigms have been developed. Some are refinements of the guided sampling concept, while others represent significant departures from this concept. Each category is discussed in section 2.10.

Change or modification of search strategy is not connected to accuracy alone. It usually has impact on speed as well. Therefore detailed discussion of search strategies is reserved for section 2.9.3, to avoid repetition.

2.9.1.6 Stopping Criterion for Sufficient Trials

Like search strategy change, the use of appropriate stopping criterion may have an impact on accuracy as well as speed. From the viewpoint of achieving better accuracy, it is necessary to compute a lower bound for the number of trials that should be allowed for RANSAC to find a good solution, with a given probability or confidence level. The absence of such criterion may impact accuracy a great deal, since an arbitrarily number of allowed trials may be insufficient. The gold-standard RANSAC [42], therefore, offers this advantage by incorporating a stopping criterion, which has become quite popular in literature. The number of trails *k*, for samples that have to be drawn for a given probability P_I of drawing an uncontaminated sample is given by:

$$k = \frac{\log(\eta)}{\log(1 - P_I)}$$

$$P_I = \frac{\binom{I}{n}}{\binom{m}{n}}$$

$$\prod_{j=0}^{n-1} \frac{I-j}{m-j}$$

$$\approx \left(\frac{I}{m}\right) n$$

where I is the number of inliers, m is the number of points in the full data, and n is the minimal sample size.

2.9.2 Pursuit of Improved Efficiency

Efficiency is often related to speed, which is a priority in low-time-budget applications. Speed has to do with fast achievement of results, in terms of run time. Efficiency can be in terms of time or computational cost. An algorithm is more efficient than another, if it finds good solutions early or with less computation. Where an algorithm finds good solutions early, it is safe to terminate it early enough to fit the speed requirements of the application.

Various strategies are found in RANSAC literature for improving efficiency. Some variants employ guided-sampling, leveraging problem-dependent information to bias sampling towards more promising candidates, instead of the uniform sampling strategy of RANSAC. Other approaches adopt search strategies that are significant departures from RANSAC's strategy (see section 2.9.2). Some other variants save time and computational cost by running preliminary tests on each hypothetical model to decide whether or not to proceed to full evaluation. The stopping criterion discussed earlier as a means for achieving better accuracy, may also impact efficiency significantly. The logic is that the algorithm may be confidently terminated if the sufficient number of trials computed, has been reached. Another approach to reducing run time and computational cost is the inclusion of preprocessing step, to derive a reduced, more reliable set of matches, with higher inlier rate from the original data. This is effective because RANSAC's efficiency has been shown to deteriorate as data size and contamination level increases [43],[44]. Each strategy is discussed in greater detail in the rest of this subsection. Figure 2.3 summarizes existing approaches adopted in literature in pursuit of better efficiency than the original RANSAC.

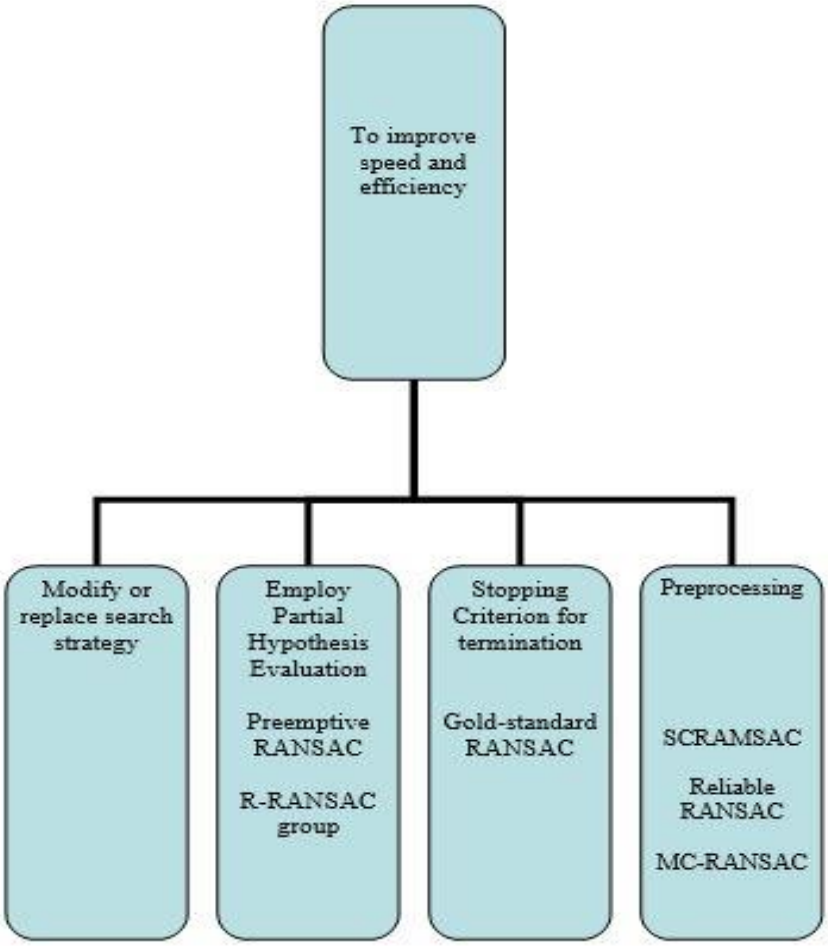


Figure 2.5: Strategies for improving efficiency

2.9.2.1 Search Strategy Change for Improved Efficiency

As stated under a similarly named heading, in section 2.9.2, to avoid repetition, search strategies are discussed in a separate subsection. In context, it suffices to say that biasing sampling, or employing completely new search strategies, could have impact not only on accuracy, but also on efficiency.

2.9.2.2 Partial Hypothesis Evaluation

One operation that takes a high proportion of RANSAC's run time is the evaluation of a hypothetical model generated from a minimal sample. An approach that has proven quite successful in achieving significant reduction in run time, and savings in computational cost is the use of preliminary tests to decide whether or not a model is promising. The purpose of such a test is to decide whether it is necessary to proceed to full evaluation of the model, or not.

Notable variants that adopt the strategy of partial evaluation of hypotheses include Preemptive RANSAC [13] and Randomized RANSAC (R-RANSAC) group of algorithms [45],[45],[38].

The R-RANSAC group carry out a preliminary test which when violated, implies that a model is not likely to be a good one. The implication is that it is not worth proceeding to full evaluation of such a model. A number of tests have been proposed including the $T_{d,d}$ test [46], Wald's sequential probability ratio test (SPRT) [45], and Bail-out test [47]. Preemptive RANSAC uses a breadth-first approach, generating and evaluating a fixed number of models in parallel. The evaluation is done on a subset of the data. The models are ranked according to the result of the evaluation. Only a fraction of them are evaluated on the next subset of the data. This process continues until only one model is left or all subsets of the data have been used. The number of hypothetical models retained before evaluating a given data point is given according to some predefined preemption function. While the use of preliminary tests generally reduces computation time, the tests are not guaranteed to be accurate. Therefore it is possible to reject a good model.

2.9.2.3 Stopping Criterion and its Impact on Efficiency

Like search strategy change, the use of appropriate stopping criterion may impact accuracy as well as efficiency. From the viewpoint of achieving better efficiency, it is useful to compute the number of trials required for RANSAC to find a good solution with a given probabilistic confidence level. Keeping in mind that RANSAC is non-convergent, and the solution per iteration does not improve progressively, the demand for efficiency suggests that the algorithm should be terminated once the computed number of trials has been reached. The absence of such a criterion may impact efficiency a great deal, since exceeding the required number of trials may prove to be a substantial waste of time and resources. Therefore the stopping of criterion of the gold-standard RANSAC achieves the balance between efficiency and accuracy.

2.9.2.4 Preprocessing

The number of RANSAC iterations required depends on the outlier rate in the data. Specifically, the lower the outlier rate, the lower the number of iterations required. A few authors have therefore pursued the goal of improved efficiency by including a preprocessing step to extract a reduced set with higher inlier rate from the original data. SCRAMSAC [43] proposed by Sattler et.al in 2008, uses a spatial consistency filter to derive a refined dataset with reduced size and increased inlier rate from the original set of matches. Two new works have been contributed along this direction within the last two years. In 2013, Wang et.al propose a variant named Reliable RANSAC [48] , which uses a relaxation technique to select matches that are

more likely to be correct, thereby resulting in a reduced, more reliable set. MC-RANSAC [49], proposed by Trivedi et.al in 2014, uses a Monte-Carlo approach, to achieve a preprocessed sample of hypothetical inliers.

It may be worth noting here that while the strategies discussed are the main ones by which speed and efficiency are directly addressed, some efficiency gains can also be realized through local optimization, discussed under the broad performance theme of accuracy. RANSAC may be run briefly enough and then refined by local optimization, to save time.

2.9.3 Review of Search Strategies in RANSAC Literature

Under the broad themes of accuracy and efficiency, a functional theme – search strategy modification or replacement - was mentioned as an effective strategy. This theme has been reserved for this separate heading to avoid repetition, since it lies in the overlap of multiple performance themes. It is indeed a very important theme, since many of the limitations of RANSAC are direct consequences of its serial uniformly-random sampling search strategy. Some authors have modified this strategy into biased or guided sampling while others propose entirely new paradigms such as metaheuristics, fuzzy sampling, and so on.

2.9.3.1 'Guided' Sampling

The earliest approach to modifying RANSAC's search strategy is commonly referred to as guided sampling. Guided-sampling algorithms use prior information on a problem to bias sampling. In 2002, NAPSAC [50], the oldest variant found in this category, was proposed. The concern of the authors is on the performance of RANSAC on high-dimensional problems, in which they show that biasing the sampling towards clusters is preferable. NAPSAC works based on the heuristic that an inlier is likely to be close to other inliers. Three years later, two other variants surfaced: PROSAC [45] by Chum and Matas, and guided-MLESAC [51] by Tordoff and Murray. In PROSAC's strategy, samples are drawn from progressively larger sets of top-ranked correspondences according to a similarity score that predicts correctness of matches. Guided-MLESAC modifies MLESAC, guiding sampling by leveraging *a priori* information on probability of validities of correspondences. In 2009, another work was published by Ni [4] along the guided sampling direction. As the name, GroupSAC, suggests, it biases sampling on the assumption that there exists some logical grouping in the data. The authors term this concept group sampling. Such grouping, they suggest, might be clustering based on optical flow, or grouping based on image segmentation. Ni also put the earlier discussed LO-RANSAC in the category of modified sampling strategy variants [4]. This

applies specifically where local optimization is achieved using sampling based techniques like the inner-RANSAC proposed by LO-RANSAC's authors.

Zhao et.al published a variant FRANSAC [52], which works similar to PROSAC. It ranks matches according to a distance measure defined as the ratio between the distance of a point's nearest neighbour and that of the next neighbour. Another variant published in the same year, FSC [53], divides the data set into two parts: the sample set and the consensus set. The sample set has high correctness rate and consensus set has a large number of correct matches. An iterative method is employed to increase the number of correct correspondences.

While the guided sampling variants have been established in literature as effective ways to improve efficiency, the drawback is that they generally depend on problem-dependent prior information. Although many authors argue that they are usually available in practice, such such priors generally limit the resulting variant in applicability, for example, to the computer vision field. This is unlike RANSAC which can be applied to general robust estimation problems. This limitation is avoided by some variants, discussed in the subsections that follow. This new category of algorithms replace RANSAC's strategy, usually with strategies from the field of metaheuristic.

2.9.3.2 Metaheuristics

Some variants adopt problem-independent search strategies. Many of the works that fall under this category use search strategies from the field of metaheuristics. The earliest found in the course of conducting this survey, is GASAC [54], published in 2006, which uses an evolutionary algorithm for its search. SwarmSAC [55], another variant published in 2008, uses a discrete particle swarm optimization (PSO) algorithm for its search. A much more recent work ANTSAC, published by Otte et.al in 2014, adopts concepts from the ant colony optimization (ACO) algorithm such as volatile memory, in its search. These techniques were shown by their authors to be generally more accurate than RANSAC. They also offer efficiency advantages in high contamination or large-search-space situations. Unlike the 'guided sampling' variants, the metaheuristic-based variants are generic since they do not require any problem-dependent information. The field of metaheuristics, a field concerned with developing search strategies for optimization problems, is noted by this survey to hold much promise for the RANSAC community, in terms of developing search strategies that are problem-independent.

2.9.3.3 Conditional Sampling

A sampling approach proposed by Mela et.al [56], involves incremental building of sampling sets. Data points are selected conditional on previously selected data. They argue that such an approach provides more suitable samples in terms of inlier ratio, and have better potential for accuracy. Again, like many biased-sampling variants, it depends on prior cues. BetaSAC as the resulting algorithm is named, is presented as a general guided sampling framework in which any kind of available prior information, can be easily used. The method classifies general inlier samples into four types: inlier samples, consistent samples, samples that are consistent with additional information, and suitable samples. Inlier samples are defined as those containing purely inliers. Most of the guided-sampling methods including PROSAC, seek this kinds of samples. However, there is still the possibility of constructing poor models from such a sample. Consistent samples are inlier samples that satisfy some consistency constraints. As the authors point out, different consistency constraints have been studied such as those originating from oriented projective geometry used in epipolar geometry, and 4-dimensional linear subspace constriants which holds for all relative homographies of a pair of planes. Some heuristics can also serve this purpose. A good example is NAPSAC's heuristic which is based on the observation that an inlier tends to be closer to other inliers than outliers. The third class, samples that pass a consistency test with additional information, are even higher potential samples than the foregoing. Such information include those from the image signal itself such as information derived from segmentation as used in GroupSAC. Finally, the highest potential samples, are the ones classified as suitable samples. Generating such a sample is the desired goal of guided sampling. According to the authors, such samples do not only have high potential to lead to correct models but are also not affected by degeneracy and measurement noise.

Botterill et.al [57], proposed two variants in 2009, both of which adopt conditional sampling strategies for early selection of sets that are most likely to lead to good hypotheses. The authors argue that existing guided-sampling variants fail to take into account information gained by testing hypothesis sets and finding them to be contaminated by outliers. Two algorithms, BaySAC and SimSAC, are proposed to take advantage of such information gain. Both algorithms take into account the observation that a model with low inlier rate likely results from samples that are contaminated by one or more outliers. Therefore it is a waste of time to try the same sample again or to try sample sets with one or more data points in common with these samples already taken to be contaminated. This holds for any prior probability distribution, be it the uniform distribution of RANSAC or the non-uniform distributions of several guided-

sampling variants. The goal therefore, is to choose samples that are most likely to contain no outliers based on the prior probabilities as well as the described sampling history.

Due to the intractability, and probably non-existence of a closed-form solution for this posterior probability, two approximation approaches were proposed by the authors, leading to the two algorithms. BaySAC adopts a Naïve Bayes method which involves choosing *n* data points which are most likely to be inliers based on initial prior probabilities being used, and then updating this inlier probabilities based on history. This hypothesize-verify-update process is repeated until sufficient trials have been made. The second variant, SimSAC follows an alternative approach to computing inlier probabilities, using simulation. Inlier/outlier statuses are initially assigned to points at random. It samples from this prior distribution of inlier/outlier status vectors, and updates this sample conditional on observation of samples that contain outliers, by finding peaks in accumulated histograms of inlier counts for each of the data points. SimSAC is however found to be the slower and more computationally complex of the two. BaySAC, according to the authors, works well when there are few large intersections between inputs and output sets, but works poorly in some cases when the data size is small since the points are still largely equiprobable even after the updates in such cases. According to the authors, both algorithms improve on the computational efficiency and speed of RANSAC significantly, while decreasing the failure rate in real-time applications.

Five years after the original BaySAC was published, Kang et.al published an optimized BaySAC [81]. Instead of using specific characteristic information about a primitive, the authors of the optimized BaySAC propose a technique for statistical testing of candidate model parameters to compute the prior probability of each data point, which is predictably model-free. The probability update is implemented by means of a simplified Bayes formula.

2.9.3.4 Fuzzy Sampling

Earlier discussed under threshold selection, the fuzzy RANSAC algorithm [38] proposed by Lee and Kim in 2007, introduces another sampling strategy. It classifies samples as good, bad and vague. Good sample sets are those whose degree of inlier membership is high and the rate of membership change is small. Bad sample sets are those whose degree of inlier membership is high and the rate of membership change is small. Vague sample sets are those whose rate of membership change is large without relation to any degree of membership. The algorithm then improves classification accuracy omitting outliers by iteratively sampling only from good sets.

2.9.3.5 Sampling Based on Reinforcement Learning

The fuzzy RANSAC [39] of Watanabe et.al, mentioned earlier under the discussion of fuzzy model evaluation techniques, incorporates in its framework a sampling method based on reinforcement learning. The authors argue that the precision of RANSAC's model estimation can be improved by increased variation in the size of samples from which hypothetical models are constructed. This however, increases the size of the solution space. They propose a Monte-Carlo sampling, performed in proportion to evaluation values, which is learned using reinforcement learning. The authors discuss a number of expected advantages of the method. One is balance of search exploration and exploitation. Other advantages discussed are better efficiency than RANSAC, robustness to random noise by the learning mechanism, reduced computational cost and accuracy, simplicity and wide applicability.

2.9.3.6 Importance Sampling

One other sampling strategy found in literature involves the use of importance sampling function for outlier-contaminated data. This was proposed in a framework named Importance Sampling Consensus (IMPSAC) by Torr and Davidson in 2003 [58]. Their work presents synthesis of very useful statistical techniques and posterior distribution of a two-view relation at a coarse level to obtain that of a finer level. The technique works by using a Monte Carlo Markov Chain which is seeded using RANSAC, and used to generate the importance sampling function.

2.9.3.7 Purposive Sampling

Another sampling paradigm was proposed by Wang and Luo [59], named Purposive Sampling Consensus (PURSAC). Instead of following RANSAC's assumption of uniform probability distribution of data points, PURSAC seeks the points' differences and 'purposively' selects sample sets. Using sampling noise information which the author argue, always exists, sampling is performed according to the sensitivity analysis of a model against the noise. In addition, the algorithm includes local optimization.

The authors discuss two examples: a line-fitting problem and visual odometry. For line-fitting, through analysis of geometry of the data points, confirmed by a Monte Carlo test, they show that the smaller the distance between the two points that make up the minimal sample, the more this sample is affected by sampling noise. Therefore, the conclusion is drawn that points sampled should be far enough from each other for better likelihood of finding a good model. The next step in the algorithm is to limit sampling to inliers though verification is still done

using the whole data set. In the final step, similar to LO-RANSAC, local optimization is performed using an inner iteration.

For visual odometry, the concept is implemented thus: first all the points are ranked by their matching scores, and the one with the highest rank is selected. Features close to this highest ranking point according to a given threshold are excluded from subsequent sampling attempts. This continues until all matches have been included in either the selection or exclusion list. Sample sets are then picked only from the selected group, according to their ranking, though the resulting hypothetical models are verified using the entire dataset.

By experiments, the authors show that PURSAC can achieve higher accuracy, precision, and efficiency, than RANSAC, the number of iterations being close to the theoretical expected lower bound. However, PURSAC's implementation requires some quantitative analysis to design the rules for purposive sampling, which depend on the model of interest.

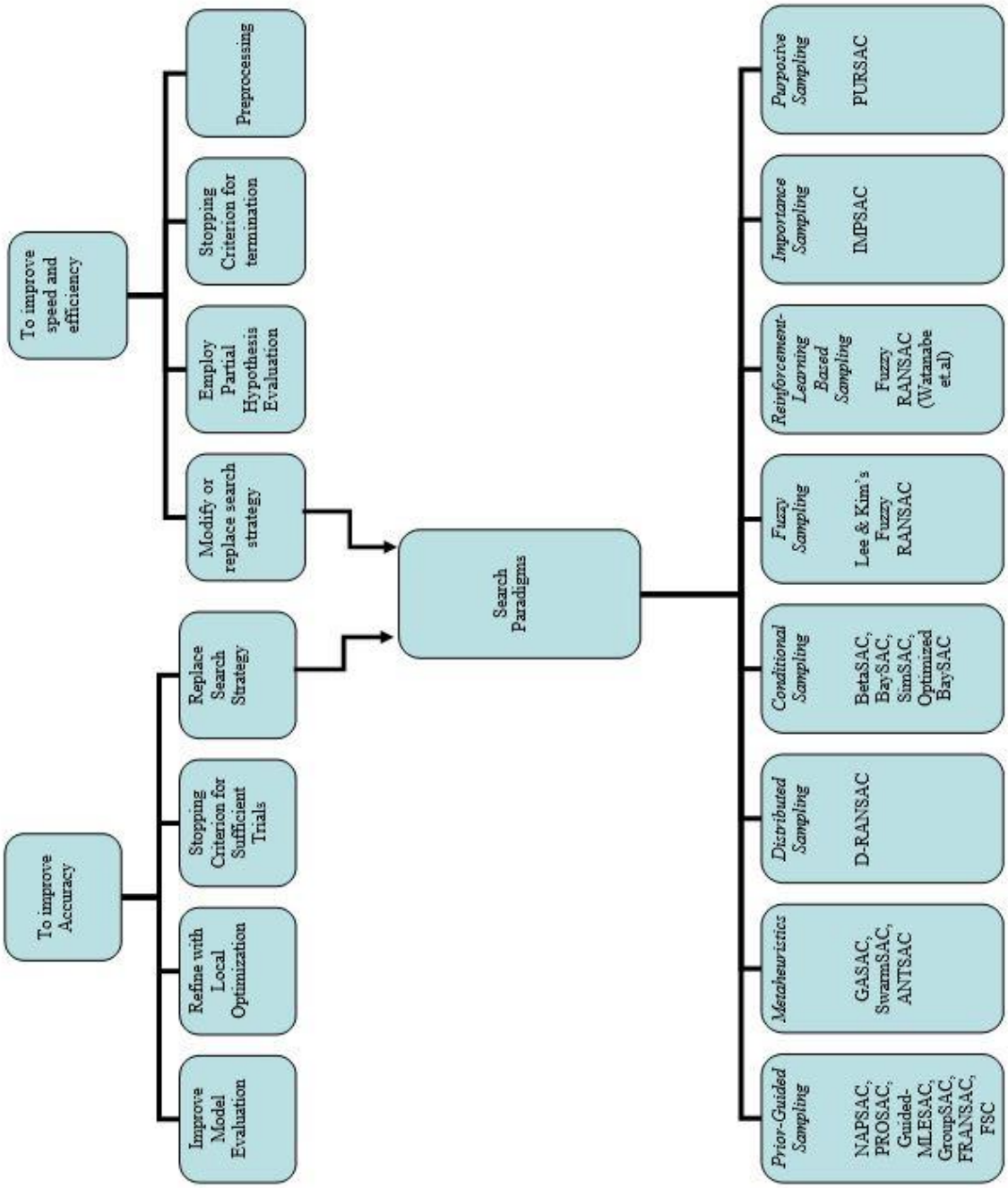


Figure 2.6: Search Paradigms in RANSAC Literature

2.9.4 Robustness Concerns in RANSAC Literature

In addition to the general concept of robustness to outlier-contamination which is the possessed (to different degrees) by every robust estimation algorithm there exists other concerns within

the RANSAC community to achieve robustness to certain peculiar image or parameter conditions. Some of this concerns are discussed in this section.

2.9.4.1 Robustness to Degeneracy

RANSAC may produce a model that fits a given data but does not verify that such a model is unique. This makes it prone to failure in data with degenerate configurations. DEGENSAC [60] was proposed by Chum et.al in 2005, to include a test for degeneracy for epipolar geometry estimation. A more general degeneracy testing approach was proposed by Frahm and Pollefeys a year later, resulting in a variant dubbed QDEGSAC [61]. Their approach works by multiple sequential calls to RANSAC. The first run estimates the most general model that RANSAC would have returned ignoring the possibility of degeneracy. Constraints are then added successively to subsequent runs. The final model returned is the one that successfully explains at least fifty-percent of the inliers of the first general RANSAC run. The high computational cost of QDEGSAC should be obvious to the reader, as it is a composite of multiple RANSC runs.

2.9.4.2 Robustness to False Matching Under Drastic Occlusion and Stitching

The conventional RANSAC approach of using a size of sample equal to the minimum required to define a given model, fails in some situations with drastic occlusion and scaling caused by large viewpoint changes. This is because the conventional approach will find it difficult to find enough correct matches to compute for example, the fundamental matrix. The result is false acceptance of outliers as inliers. To tackle this problem, Chou and Wang proposed an approach that uses only two points, correspondingly dubbed 2-point RANSAC [62], to raise the success rate in planar cases. The approach was tested on loop-detection and place recognition tasks. However, the authors express some concerns about the proposed approach. The first concern is the 2-D limitation, the second being the computation speed. They hope to investigate more efficient ways for dealing with the homography matching step than exhaustive search.

2.9.4.3 Robustness to Patch Clustering

As mentioned earlier under the discussion of optimization objectives, Gallio et.al [33] noted the unreliability of RANSAC in situations with clustered patches of limited extent. In such cases, a single plane crossing two such patches may contain more inliers than the correct model, a situation that occurs with images containing steps, curbs or ramps, in range sensor applications. CC-RANSAC was therefore proposed to improve robustness to such conditions by adopting a new objective which is the maximization of connected components of inliers.

Normal Coherence RANSAC (NCC-RANSAC) [63], published in 2014 builds on the success of CC-RANSAC in overcoming the challenge. CC-RANSAC has some limitations. It succeeds when the patches are distinct but fails if they are connected. As the name implies, NCC-RANSAC performs a normal coherence test on all data points of the inlier patches in order to remove points whose normal directions contradict that of the fitted plane. The outcome is the derivation of distinct inlier patches, each of which is treated as a candidate plane. The planes are grown recursively until all planes have been completely extracted. This process of plane fitting and clustering continues until no more planes are found.

2.9.5 Other Themes

In addition to the themes already discussed. A few other themes are identified in literature which do not seem to be quite as popular as the previously discussed ones. The themes discussed under this category, generally emerged relatively recently in literature.

2.9.5.1 Multi-Model Estimation

Most RANSAC variants assume that a single model accounts for all of the data configuration. But it is found that there are cases where this assumption does not hold. A few authors have explored extension of RANSAC to handle multi-model cases. Sequential RANSAC, as it is called, involves detection of multiple models by applying RANSAC sequentially and removing the inliers from the dataset as each model is detected. Such an approach is adopted in the work of Kanazawa and Kawakami [64], and that of Vincent and Laganiere [65]. The Sequential AC-RANSAC of Rabin et.al [37] also adopts this strategy, combining it with the *a contrario* framework of AC-RANSAC. The same applies to MAC-RANSAC which extends sequential AC-RANSAC with spatial filtering and transformation fusion detection with a fusion splitting criterion.

Zuliani et.al [66] criticized the sequential approach as non-optimal and note that it is prone to inaccurate estimation. In view of this they proposed a parallel strategy that detects models simultaneously in a more principled way. Through experiments, they argue that the parallel approach seems to produce more stable estimates than the sequential approach. An important gap is also noted by the authors: the need for automatic estimation of the optimal number of models.

2.9.5.2 Robust Estimation with Non-Homogenous Correspondences

This is another relatively unexplored area of research in RANSAC literature. Most works in RANSAC literature assume homogeneous correspondences, that is, correspondences that are

of the same modality, sharing the same properties and metrics. However, one work is found that addresses the non-homogeneous case. Published in 2014 by Barclay and Kaufmann, the variant named Fault-Tolerant RANSAC (FT-RANSAC) [67], adopts PROSAC-inspired guided sampling, consensus maximization of classical RANSAC, Hough-inspired dimensionality reduction, and consistency voting mechanism. This setup helps the algorithm to compute the best model among competing multi-modal solutions.

2.9.5.3 Target Tracking and Dynamic Model Estimation

Dynamic Target tracking involves evolving-state estimates. It is a widely studied field with such applications as pedestrian tracking, vehicle tracking, bacteria tracking, air traffic control, and so on. RANSAC has been found to be useful in this field, since the problems involve robust estimation. KALMANSAC [68] is one algorithm used to track single dynamic targets using causal measurements. KALMANSAC uses RANSAC to label data points as outlier or inliers. Such labels are then used to seed the iteration of subsequent time-steps.

Recursive-RANSAC [69], unlike KALMANSAC, extends to the case of multiple target tracking. It was originally developed for estimation of multiple static signals, and later extended to the dynamic case. It achieves dynamic target estimation using a recursive RANSAC procedure. While KALMANSAC computes the estimate of the maximum *a posterior* probability estimate, Recursive-RANSAC stores multiple hypothesis tracks in memory to allow subsequent inlier measurement to refine the current estimate. Recursive-RANSAC does not require prior knowledge of the number of existing targets.

2.9.6 USAC: An Integrated 'Universal' RANSAC Framework

By the joint effort of five well-known authors in RANSAC literature, each of whom has been involved in the development of one or more of the variants already discussed, a universal RANSAC framework, USAC, was published in 2013 [17]. It is a composite of a number of existing variants that fulfil the requirement of each module. The authors present the framework designed based on the argument that each existing variant is a special case of RANSAC in terms of practical and computational considerations. Therefore each module takes care of specific functional requirements. In terms of overall performance, it is the current state of the art [32], representing excellent performance along multiple directions. The various modules are summarized below:

2.9.6.1 Prefiltering

Due to the fact that the runtime of RANSAC is determined by the inlier rate, a useful measure is to preprocess the input data to improve the inlier rate. This restricts the input data to reliable matched feature pairs, using consistency filters. This impacts the performance of RANSAC in two ways: the matches used are cleaner and more correct and the data size is reduced. This results in better efficiency. The algorithm chosen for this module is SCRAMSAC.

2.9.6.2 Sampling

To improve on the efficiency of the uniform random sampling of RANSAC, the authors of USAC provided facility for biased sampling based on some prior information on the dataset. For this purpose, they explored PROSAC, GroupSAC and NAPSAC. PROSAC is chosen among these alternatives to achieve a balance of performance, generality, and the risk of degeneracy. They argue that PROSAC is more easily applicable in the general case than GroupSAC, and less susceptible to degenerate configurations than NAPSAC. PROSAC however, requires matching scores for ordering the data points.

2.9.6.3 Preliminary Model Check

This module is aimed at carrying out preliminary tests to avoid full evaluation of a hypothetical model, if it fails such a test which is an indication that it is not a good model. This is done in USAC using the SPRT test, which the authors argue is a better choice than other alternatives like $T_{d,d}$ test.

2.9.6.4 Check for Degeneracy

The goal of this module is to build robustness to degenerate data configurations. The choice made for this module is DEGENSAC. The authors noted however, that integrating USAC with QDEGSAC, a more general alternative for degeneracy detection, is quite easy: replacing the calls to RANSAC made within QDEGSAC with calls to USAC.

2.9.6.5 Local Optimization

This module refines the initial consensus set maximization result, using LO-RANSAC. The specific local optimization strategy is inner-RANSAC couple with iterative reweighted least squares procedure. The argument for this choice is that it is found to work well in practice without substantial addition to computational cost.

The authors include a check in the implementation of USAC before the local optimization step is performed, to determine the extent of overlap between the current inlier set and the best inlier set found so far. The reason is that a substantial overlap, say 95%, indicates that significant improvement to the best result is unlikely to result from local optimization. In such a case of substantial overlap, the local optimization step is not worth performing and can therefore be skipped, to save time.

2.9.6.6 Stopping Criterion

There is the need to modify the stopping of criterion of RANSAC in the USAC framework. The authors show that accounting for the effect of biased sampling and SPRT test results in some changes in the usual probability of finding a good solution which is then used to construct a stopping criterion in the gold-standard RANSAC. Details of these modifications are provided in [17].

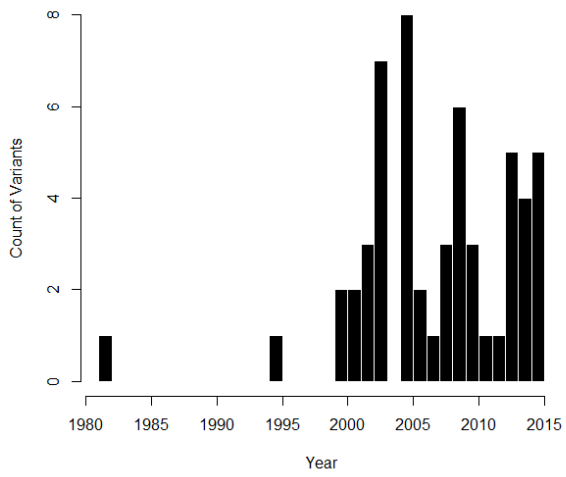
2.10 Chronological Analysis of RANSAC Literature

In this section, observations are presented on research activity from 1981 when the original RANSAC was published, to date. One outcome is the identification of trends. Some of the observations made also aid in measuring popularity of works and the rate at which each is attracting attention in literature. These metrics are put to further use in the section on classics identification. Aggregated with our theme-based discussions, the observations made in this section, naturally lead to discussion of gaps in literature.

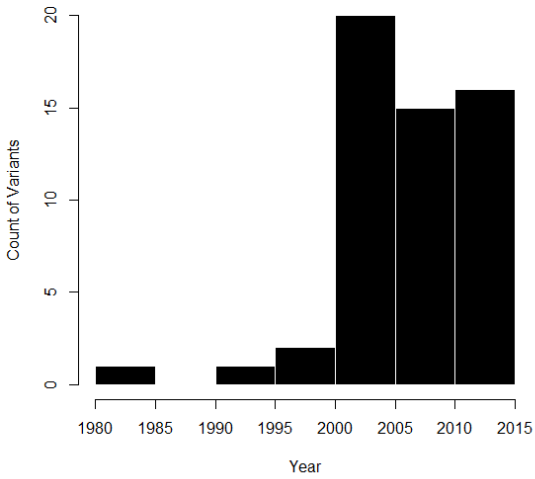
Figure 2.7 tells part of the story of the evolution of the RANSAC research. It is easily seen that RANSAC research really started to become active at the beginning of the 21st century. After the original RANSAC which was published in 1981, only two works are found by this survey to be dated earlier than the year 2000. Since 2000, nearly every year has witnessed the publication of new variants. It is also quite clear that research is still very much active in this area. One evidence is seen in Figure 3.5a which shows that the years 2014 and 2015 witnessed the highest level of activity, next to the year 2005. Furthermore, grouping the works in the collection by decades shows an increasing trend in the number of variants published. A similar trend is observed when the works are grouped by half-decade intervals.

The implication of these observations is that although much progress has been made in this research area, there is still much activity going on. It is worth noting at this point, that many of the works discussed in this survey were published under the auspices of major conferences and journals in the field of computer vision. This can easily be verified by a glance through

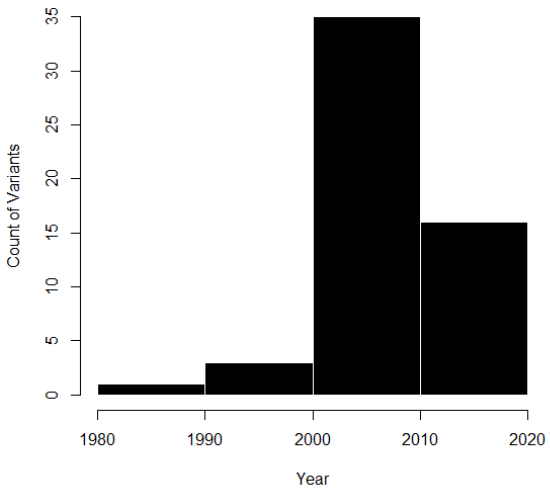
the references section. This should reduce the probability of merely high activity without much significant contributions. A plausible inference is that the consistently high (even rising) level of research activity suggests the presence of gaps. This may imply the persistence of some old problems or the emergence of new ones, or both.



(a) Yearly



(a) 5-year Interval



(c) Ten-year Interval

Figure 2.7: Research Activity in RANSAC literature from 1981 to 2015 measured by count of variants

2.11 Methodology for Identification of Classics

In software production settings, there may be a need for in-depth study of original works or review of specific variants in greater detail, than can be provided in a survey. Software makers and other practitioners, who are supposed to benefit from existing works, are faced with the challenge posed by the vastness and high-paced evolution of this literature. This section suggests ways to tackle this challenge, considering the infeasibility of studying all works in detail, within reasonable time. Qualitative as well as quantitative metrics are proposed.

As rules of thumb, this section suggests classifying as classic:

1. any work that is among the most popular in the entire collection of publications
2. any work (variant) that is preferred among those developed to solve the same problems
3. a pioneering work along the direction of a specific functional theme

The first suggested rule for identifying classics, should apply in most fields. The metric adopted in this survey to measure popularity of a work is the total number of citations it has attracted. This is based on the reasoning that a publication that presents a novel variant, is cited for one or more reasons, some of which are identified as the following:

1. The algorithm is used in an application.
2. The algorithm is influential in developing another variant presented in the work that cites it.
3. The cited work is included in discussion of related works.
4. For some reason, the variant published in the cited work is involved in the experiments of the work that cites it. This often happens when the author(s) see(s) the need to compete with the cited variant, usually because it is a popular choice for some specific performance requirement.

Besides the popularity of variants, measured by absolute number of citations attracted, another related metric is adopted in this survey: the citation rate. This provides a kind of normalization for fair comparison of works in terms of their impact factor as well as current rate of attracting attention in the community. This metric is computed as the ratio of total number of citations to the number of years since publication.

The second rule suggested for identifying important works is to go for the preferred variant under each functional-theme, that is, those developed to solve the same problems. An objective judgment in each case will require series of well-designed experiments for appropriate performance evaluations. This is definitely a major task that will require several months if not years, and possibly collaborations among several experts, to complete. A second best option that is expected to be suffice for the dominant themes, is suggested. The judgement of notable experts in the RANSAC community – the five authors of USAC – is relied on. As discussed in the section on USAC, each of the authors had been involved in RANSAC research for years and each had developed some of the most popular variants. Moreover, USAC itself, though published just about two years ago, has become quite successful and popular. The idea of USAC was to develop a unified framework composed of modules each taking care of a specified functional requirement that may come up under practical and computation consideration. Each module implements a variant preferred by the authors for the specific purpose. The reader is referred to the discussion provided in this survey on USAC in section 2.9.5 or the original paper [17] for details.

Lastly, while there is no guarantee that the collection of works in this survey is exhaustive, it is noted that a pioneering work would have been cited by most works that are along the same direction. This means it is unlikely that this survey would have missed a pioneering work in the process of collecting the original publications for such a large collection of variants. Therefore, looking up the discussion provided in this survey on any theme of interest and picking the earliest published, should be a good way to identify such works.

The suggested rules can be applied by researchers and practitioners to identify works that are important to their specific purposes. Priorities will vary from application to application.

In the section that follow, the suggested metrics for measuring popularity and popularity rate are put to use in analysis of the RANSAC literature, to seek answer to a few interesting questions.

2.12 Observations and Discussion

A number of interesting findings result from aggregating observations from the spatial and chronological analysis of the literature carried out in this survey. These findings are discussed in this section.

2.12.1 'Old' works with low popularity score

The entire collection of variants is categorized into three groups: high popularity (top 33%, about 18 in number), average popularity (middle 33%) and low popularity (bottom 33%). Another classification is created according to age. Since research into development of variants really became active from the year 2000, any work published on or before the middle of that decade (2005 or earlier) is classified as old. Any work published after 2005 is classified as recent. A variant that is old yet having low popularity score (total number of citations since it was published) is judged not to have attracted much attention. Such works may represent themes that have not been given much attention.

By these classifications, it is observed from table 2.3 that two variants, Feng and Hung's MAPSAC and AMLESAC, fall into the category of old works with low popularity. Interestingly, they both represent the same theme – user independence. A closer observation of the same table reveals that almost all the other user-independent variants (uMLESAC, StaRSaC, Sequential AC-RANSAC, MAC-RANSAC) which are of relatively average age are also not very popular. It is interesting to see that they all rank closely on the popularity table. This may be an indication that this theme is still relatively unpopular. At first glance, two automatic tuning works, AC-RANSAC and Sequential RANSAC, seem to skew this conclusion a bit, making it to the top 33%, but they lie somewhere at the bottom of this group. Moreover, they are both quite old and this may have introduced some bias in their popularity score. A plausible inference from these observations is that full user independence, though very advantageous is still not very popular. Such a conclusion is given further validation by the fact that modern software like all existing releases of MATLAB's computer vision toolbox, including the 2015b release, still use implementations of robust estimation functions based on variants that depend on user-supplied values. One possible reason may be that existing fully automatic techniques come at significantly higher computational cost and traded off simplicity. This would amount to forfeiting the very advantages that RANSAC offers over many other robust estimation techniques. Besides, an appropriate value for the distance threshold for example, can be determined through some experimentation. But these are just hypothetical conclusions: the observation may simply be a pointer to the fact that the priorities of research efforts so far lie in other themes than user-independence. Clearly from the table, most of the works that made it to the top 33% on the popularity table are related to either accuracy or efficiency. Any reader of RANSAC literature knows that these are popular themes.

A word caution is worth being chipped in here. Current popularity should not be used to conclusively judge the importance or future prospects of the success of variants. This is because there are a number of factors that affect the popularity of a work. These include the advantage of age and the fact that research efforts easily follow the direction of earlier works. But popularity still holds some value for judgment: a popular work is more likely to have undergone much scrutiny and review, so the claims made should be more reliable. Little wonder the variants that are in common use in computer vision software are found at the top of the table.

2.12.2 Works with High Popularity Rate

The list of variants at the top of Table 2.4, in which variants are ranked according to popularity rate rather than absolute total citations, covers a balance of all three performance themes, the top-most being gold-standard RANSAC which achieves a balance of accuracy and efficiency. Although it may be argued that its relatively old age is responsible for its popularity, and while this may be a factor, it is observed that there are several older works that are not nearly as popular. Another plausible argument calling for caution in judgment, is that this algorithm was published in a book – one of the most popular books in computer vision – which would have attracted citations because of other subjects than just this algorithm. However, it is still true that the gold-standard algorithm is highly favoured in software implementations till date. The stopping criterion, which distinguishes it from the original RANSAC, is widely used.

Some relatively recent works are found that have a high popularity rate. This may be a good indication of growing attention being paid to that work and the theme it represents. Recent works with high citation rate may represent a fast growing theme while older works with low citation rate may indicate that a theme or work is no longer receiving attention. Competitively high popularity rate is observed for USAC and ARRSAC. USAC has the higher rate.

USAC is an integrated framework representing improvement along multiple themes. ARRSAC is also ‘multi-theme’. This shows that there is remarkably fast growing interest in contributions that achieve multidirectional or balanced improvements, rather than simply trading off one performance measure to achieve another that is of emphasis in the concerned application. In fact, USAC probably represents the most comprehensive improvement coverage in a single work, ever in RANSAC literature. Clearly, multidirectional improvement should be considered by researchers who want to develop successful algorithms.

Table 2-2: Variants Sorted By Age/Year of Publication

S/N	Name	Year	Age (Years)	Reference
1	RANSAC (original)	1981	35	[10]
2	K-RANSAC	1995	21	[70]
3	MLESAC	2000	16	[22]
4	MSAC	2000	16	[23]
5	Sequential RANSAC	2001	15	[65]
6	R-RANSAC	2001	15	[71]
7	MAPSAC	2002	14	[23]
8	R-RANSAC with Tdd Test	2002	14	[72]
9	NAPSAC	2002	14	[50]
10	gold-standard RANSAC	2003	13	[42]
11	Preemptive RANSAC	2003	13	[13]
12	LO-RANSAC	2003	13	[40]
13	AC-RANSAC	2003	13	[37]
14	IMPSAC	2003	13	[58]
15	Feng and Hung's MAPSAC	2003	13	[73]
29	uMLESAC	2003	13	[74]
16	PROSAC	2005	11	[75]
17	MultiRANSAC algorithm	2005	11	[66]
18	guided-MLESAC	2005	11	[51]
19	DEGENSAC	2005	11	[60]
20	R-RANSAC with SPRT Test	2005	11	[45]
21	RANSAC with bail-out test	2005	11	[47]

22	KALMANSAC	2005	11	[68]
23	AMLESAC	2005	11	[34]
24	QDEGSAC	2006	10	[61]
25	GASAC	2006	10	[54]
26	Lee and Kim's Fuzzy RANSAC	2007	9	[38]
27	ARRSAC	2008	8	[18]
28	Optimal R-RANSAC	2008	8	[76]
30	SwarmSAC	2008	8	[77]
31	1-point RANSAC	2009	7	[78]
32	SCRAMSAC	2009	7	[43]
33	GroupSAC	2009	7	[79]
34	BaySAC	2009	7	[57]
35	SimSAC	2009	7	[57]
36	StaRSaC	2009	7	[80]
37	MAC-RANSAC	2010	6	[37]
38	Sequential AC-RANSAC	2010	6	[37]
39	BetaSAC	2010	6	[56]
40	CC-RANSAC	2011	5	[33]
41	LO+-RANSAC	2012	4	[41]
42	USAC	2013	3	[17]
43	Reliable RANSAC	2013	3	[48]
44	recursive RANSAC	2013	3	[69]
48	MC-RANSAC	2013	2	[49]
50	fuzzy RANSAC	2013	2	[39]
45	NCC-RANSAC	2014	2	[63]

46	FT-RANSAC	2014	2	[67]
47	Optimized BaySAC	2014	2	[81]
49	ANTSAC	2014	2	[82]
51	FSC	2015	1	[53]
52	Distributed Robust Consensus	2015	1	[83]
53	FRANSAC	2015	1	[53]
54	2-point RANSAC	2015	1	[62]
55	PURSAC	2015	1	[59]

Table 2-3: Variants Sorted By Popularity Score

Ranking (Total = 55)	Variant	Year	Popularity Score (Total Citation)
1	gold-standard RANSAC	2003	17475
2	RANSAC	1981	13595
3	MLESAC	2000	978
4	PROSAC	2005	518
5	Preemptive RANSAC	2003	502
6	LO-RANSAC	2003	303
7	ARRSAC	2008	211
8	MSAC	1998	205
9	MAPSAC	2002	201
10	Optimal R-RANSAC	2008	200
11	MultiRANSAC algorithm	2005	173

12	guided-MLESAC	2005	138
13	AC-RANSAC	2003	135
14	R-RANSAC with Tdd Test	2002	134
15	Sequential RANSAC	2001	115
16	NAPSAC	2002	108
17	IMPSAC	2003	105
18	DEGENSAC	2005	99
19	QDEGSAC	2006	98
20	R-RANSAC with SPRT Test	2005	95
21	1-point RANSAC	2009	94
22	SCRAMSAC	2009	79
23	GroupSAC	2009	65
24	RANSAC with Bail out test	2005	62
25	K-RANSAC	1995	55
26	GASAC	2006	50
27	CC-RANSAC	2001	44
28	USAC	2013	41
29	LO+-RANSAC	2012	40
30	KALMANSAC	2005	34
31	BaySAC	2009	28
31	SimSAC	2009	28
33	Sequential AC-RANSAC	2010	24
33	MAC-RANSAC	2010	24

33	Reliable RANSAC	2013	24
36	Feng and Hung's MAPSAC	2003	22
36	Lee and Kim's Fuzzy RANSAC	2007	22
36	StaRSaC	2009	22
39	AMLESAC	2005	19
40	uMLESAC	2003	17
41	BetaSAC	2010	13
42	recursive RANSAC	2013	12
43	R-RANSAC	2001	7
43	FSC	2015	7
45	SwarmSAC	2008	6
46	NCC-RANSAC	2014	4
47	FT-RANSAC	2014	2
47	Optimized BaySAC	2014	2
47	Distributed Robust Consensus	2015	2
50	MC-RANSAC	2013	2
50	ANTSAC	2014	1
50	fuzzy RANSAC	2014	1
50	FRANSAC	2015	1
54	2-point RANSAC	2015	0
54	PURSAC	2015	0

Table 2-4: Variants Ranked By Average Citation Rate

Ranking (Total = 55)	Variant	Year	Age	Popularity Score (Total Citation)	Popularity Rate $(\frac{Total\ Citation}{Age})$
1	gold-standard RANSAC	2003	13	17475	1344.23
2	RANSAC	1981	35	13595	388.43
3	MLESAC	2000	16	978	61.13
4	PROSAC	2005	11	518	47.09
5	Preemptive RANSAC	2003	13	502	38.62
7	ARRSAC	2008	8	211	26.38
10	Optimal R-RANSAC	2008	8	200	25
6	LO-RANSAC	2003	13	303	23.31
11	MultiRANSAC algorithm	2005	11	173	15.73
9	MAPSAC	2002	14	201	14.36
28	USAC	2013	3	41	13.67
21	1-point RANSAC	2009	7	94	13.43
12	guided-MLESAC	2005	11	138	12.55
8	MSAC	1998	18	205	11.39
22	SCRAMSAC	2009	7	79	11.29
13	AC-RANSAC	2003	13	135	10.38
29	LO+-RANSAC	2012	4	40	10
19	QDEGSAC	2006	10	98	9.8
14	R-RANSAC with Tdd Test	2002	14	134	9.57

23	GroupSAC	2009	7	65	9.29
18	DEGENSAC	2005	11	99	9
20	R-RANSAC with SPRT Test	2005	11	95	8.64
17	IMPSAC	2003	13	105	8.08
33	Reliable RANSAC	2013	3	24	8
16	NAPSAC	2002	14	108	7.71
15	Sequential RANSAC	2001	15	115	7.67
43	FSC	2015	1	7	7
42	recursive RANSAC	2013	2	12	6
24	RANSAC with Bail out test	2005	11	62	5.64
26	GASAC	2006	10	50	5
31	BaySAC	2009	7	28	4
31	SimSAC	2009	7	28	4
33	Sequential AC-RANSAC	2010	6	24	4
33	MAC-RANSAC	2010	6	24	4
36	StaRSaC	2009	7	22	3.14
30	KALMANSAC	2005	11	34	3.09
27	CC-RANSAC	2001	15	44	2.93
25	K-RANSAC	1995	21	55	2.62
36	Lee and Kim's Fuzzy RANSAC	2007	9	22	2.44
41	BetaSAC	2010	6	13	2.17
40	uMLESAC	2003	8	17	2.13
46	NCC-RANSAC	2014	2	4	2

47	Distributed Robust Consensus	2015	1	2	2
39	AMLESAC	2005	11	19	1.73
36	Feng and Hung's MAPSAC	2003	13	22	1.69
47	FT-RANSAC	2014	2	2	1
47	Optimized BaySAC	2014	2	2	1
50	FRANSAC	2015	1	1	1
45	SwarmSAC	2008	8	6	0.75
50	MC-RANSAC	2013	2	1	0.5
50	ANTSAC	2014	2	1	0.5
50	fuzzy RANSAC	2014	2	1	0.5
43	R-RANSAC	2001	15	7	0.47
54	2-point RANSAC	2015	1	0	0
54	PURSAC	2015	1	0	0

2.12.3 Identifying the Most fundamental Research Question

The observations discussed seem to produce a clear revelation on the dominant concerns in RANSAC literature. While there are varied directions along which contributions have been made, the most fundamental research question appears to be the following:

How can high accuracy be achieved in as few trials as possible?

Clearly, this is statement of the combined quest for speed and accuracy. These two performance criteria are combined in one word: efficiency. Given the foundation of this fundamental objective, most of the further important research concerns are captured thus:

How can this primary objective be achieved with as much simplicity, generality, non-randomness of results, and robustness, as possible?

The presented discussion of variants and analysis of literature reveals the validity of this work's statement of the most fundamental research question. The fundamental nature of the problem is evident in the fact that if RANSAC can be given an infinite amount of time, the probability of finding a good solution approaches unity. The statement is further validated by the fact that improvement of efficiency as attracted the most research attention. Interestingly, as the observation of the most recent works show, the quest is still very much on and there is still room for contribution.

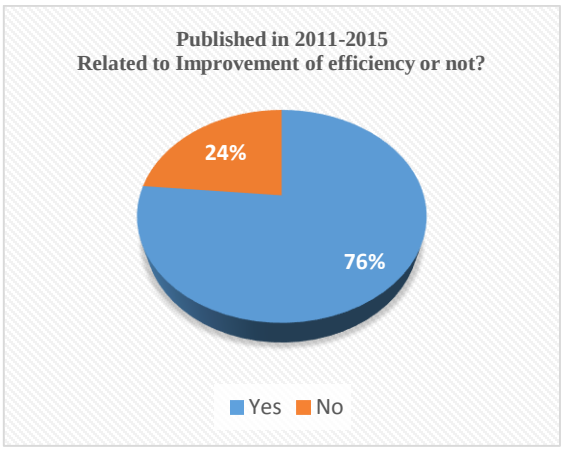
2.12.4 Trends in the Current Half-Decade and Forecasts of the Immediate Future

A study of Table 2.2, in which works are sorted by year of publication reveals that the current half-decade consists mostly of works that relate to improvement of efficiency. A closer study shows that a significantly high proportion of these works are concerned with search strategy development. Still along the same lines, the pie charts in Figure 3.6 show that about three-quarters of the research attention in the last half decade was given to developing variants that improve on RANSAC's efficiency. Also, a bit over half of the works published during this period are concerned with development of more effective search strategies.

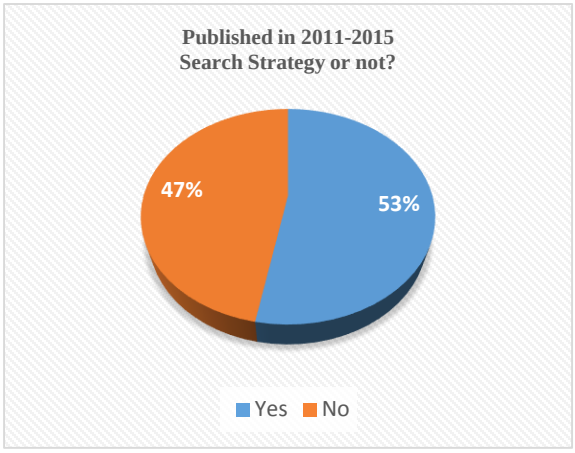
All the above observations show that improvement of efficiency is still the 'hottest' research topic in RANSAC literature, which is again consistent with the statement of the most fundamental research question in this work. This statement appears valid for the overall literature as well as for the last five years. It is seen that a prominent direction along which improved efficiency is pursued, is the development of effective search strategies to replace serial uniformly random sampling.

In view of the recent trends discussed and the literature understanding gained as a result of this survey, an attempt is made to forecast the direction of research activity in immediate future. It is foreseen that more works will be published along these directions in the immediate future. The reason for this statement is that, with regards to these themes, it does not appear that the community has come close to a 'plateau' such as is the case with some themes like local optimization, adopted optimization objectives, techniques for handling degeneracy, and a few others for which the state of the art has been existing for relatively long period. A good example is LO-RANSAC, published in 2003, which has generally remained the standard for local optimization. However, the literature keeps evolving, with respect to search strategies. As for many of these other themes, recent improvements, if at all, seem to be relatively marginal. Recently, some attention is being given to those that do not depend on problem-dependent priors while still possessing competitive efficiency. This is because the most popular technique along the lines of efficient sampling consensus, PROSAC, depends on problem-dependent priors. This trend is likely to continue in the nearest future.

New themes are also expected to continue to emerge to extend RANSAC to more complicated applications than those for which it was originally designed. Some of the recent ones have been discussed under the heading 'Other themes'.



(a)Proportion of variants related to improvement of speed and efficiency



(b) Proportion of variants that involve exploration of new search strategies

Figure 2.8: Trends in the last half-decade

2.13 Gaps Summary and Suggestions for Future Works

There is no doubt that all the discussed efforts of various researchers have resulted in a family of algorithms that is quite successful in the robust estimation front. It is a mark of success that members of this family of algorithms are widely used in practice and in software implementations. While an attempt has been made to forecast the immediate future of RANSAC research on the basis of recent trends, this section takes on a more proactive tone. Suggestions are provided on directions that should be pursued for significant gains in RANSAC research in the near future. The purpose is to help drive forward the state of the art in a well-rounded way.

First, researchers should note that any of the functional themes discussed is a prospective research direction. The entire RANSAC literature is barely three decades old. Some themes only emerged even more recently. The point is that there is definitely room for improvement in the direction of most of the themes discussed in this survey.

Perhaps, a more important point to stress is the fact that there are several other paradigms for robust estimation in the computer vision field, but there are specific properties that have directed much preference to RANSAC-like algorithms. Such properties include simplicity and the resulting ease of implementation; efficiency that results from non-exhaustive search; non-dependence on problem-specific information, and applicability over a wide range of practical situations. In fact, RANSAC is still preferred in software implementations over many techniques with guarantees of globally optimal solutions. While we acknowledge that any improvement to the algorithm is a valid research contribution, it is advocated that the strengths that have made it a preferred choice for robust estimation, should be preserved as much as possible. This is said in light of the fact that many variants that have been developed have achieved their improvements over the original algorithm by trading off some the cherished strengths, two of which are simplicity and generality. Simplicity is often traded off by introduction of new operations while generality is often traded off when algortihms leverage on problem-specific priors. This observations probably explains why popular computer vision software, as mentioned before, have stuck with relatively older variants.

One way in which balanced multifaceted improvement can be achieved is the development of integrated frameworks like USAC. Although simplicity will be traded off with such an approach, gains in many other directions will be produced by more research along this direction. It should be noted that there are still a number of functional themes discussed in this survey that are not yet covered in USAC. An example is that USAC is limited to the single-model case. More comprehensive comparative studies to decide the best choices among variants for each module of such an integrated framework, are also valuable for continuous improvement of USAC.

Many of the drawbacks of RANSAC are direct consequences of the uniform serial random-sampling search strategy it adopts. Concluding from the collection of works studied, perhaps, there is a limit to how much improvement is achievable as long as the fundamental sampling strategy of RANSAC is retained. This is especially true if dependence on problem-specific priors, is to be avoided. The exploration and development of suitable search strategies is a necessary research direction. This work suggests that research efforts look in such directions as the field of

metaheuristics – a field dedicated to developing effective search strategies for solving optimization problems. Very few works are found to have moved in this direction: SwarmSAC, GASAC, ANTSAC, and the achievements are encouraging. Multidirectional improvements are achieved as inherent properties of the search strategies without leveraging any problem-dependent information. Of course, such algorithms can still benefit from many other enhancements proposed in literature like use of various objective functions, preprocessing, local optimization, partial hypothesis evaluation, handling degeneracy, and several others, for further improved performance as deemed necessary.

Lastly on suggestions for future research, it should be noted that absolute repeatability remains an elusive property in RANSAC literature, though a few works have successfully reduced the randomness of solutions to remarkable levels. While this is also a direct consequence of the random search strategy, it is worth emphasizing. Software makers and practitioners will probably gladly opt for an algorithm that leaves every attribute of the gold standard RANSAC as it is, while being reliable enough to produce the same solution for every run on the same problem. Future research efforts are encouraged to explore the possibility of coming as close as possible to this ideal.

2.14 Chapter Summary

An introduction to the working principles and drawbacks of the RANSAC algorithm is presented followed by a brief overview of robust estimation techniques in general. These lead to the main contribution of this chapter: a survey which is to the best of our knowledge, the most comprehensive survey published on RANSAC variants. Analysis of literature lead to provision of answers to questions of interest. An attempt is made to identify the dominant themes, the most fundamental question(s) in RANSAC research and the most pressing concern(s) of research efforts, as well as recent trends. Also, to aid software production process which would typically involve studying variants in much more detail, quantitative and qualitative approaches are proposed to guide prioritization of original works to be studied. The survey concludes with identification of gaps and recommended directions for future research efforts.

CHAPTER THREE

THEORETICAL CONTRIBUTIONS AND PROPOSED ALGORITHMS

3.0 Chapter Introduction

This chapter contains much of the main contributions of this thesis. It is logically divided into six main parts. The first main part, Section 3.2, presents the main contribution to theory. A theorem is proposed, which provides the central idea upon which much of the novel content of this thesis is based. The theorem is stated and its potential for developing the desired alternative to RANSAC's search strategy, is described. A concise proof is provided to establish the theorem's validity. A few complementary propositions are also included. In section 3.3, the first novel algorithm, named Consecutive Instance Sample Consensus (CISAC), is presented. Its properties are discussed and a few preliminary experiments are performed to verify the properties beyond mere theoretical expectations. Section 3.4 presents a study of the problem of automatic estimation of distance threshold, an important parameter in RANSAC-like algorithms. The study reveals interesting insights on the problem and shows the new possibilities that CISAC represents. These insights are harnessed to develop a fully automatic algorithm, named Automatic CISAC (AutoCISAC), presented in section 3.5. Section 3.6 presents another algorithm, named Shuffle-and-Sweep Consensus (SASSAC), which is a refinement of CISAC for better reliability. The last two algorithms, M-estimator CISAC (MCISAC) and M-estimator SASSAC (MSASSAC), are M-estimate descendants of CISAC and SASSAC respectively. They are both described in section 3.7.

3.1 Revisiting RANSAC's Heuristic: Seeking All-inlier Samples

As discussed in chapter 2, RANSAC approaches robust estimation as a combinatorial optimization problem. It searches the space of hypothetical models constructed from minimal-size samples, to optimize its measure of model quality. Precisely, models are selected by random sampling of data instances, followed by evaluation of model constructed from each sample. This continues iteratively, until the termination criterion is satisfied. Then the model with the maximum number of inliers, is returned as the final estimate. A stopping criterion is used, which computes a lower bound for the number of trials required to give a certain probability of finding good solutions. The stopping criterion seeks to ensure that an all-inlier sample is chosen. Although there is no

guarantee that a particular all-inlier sample will produce the globally optimal solution, the heuristic of seeking all-inlier samples proves to be quite effective in finding good solutions. While an all-inlier sample does not necessarily result in an optimal model, it is definitely true that the optimal solution will be constructed using an all-inlier sample.

In summary, the RANSAC family of algorithms are fundamentally based on the following principle:

An optimal model is constructed from an all-inlier sample.

In place of random sampling, the consecutive inliers theorem, proposed in this thesis and stated in the next section, reveals a direct way to find all-inlier samples. Interestingly, the theorem applies to the generic robust estimation problem. That is, it holds for any given data, even those from non-computer-vision applications.

3.2 Theorem 1: A Deterministic Way To Find All-inlier Samples

This theorem is hereafter referred to as 'the consecutive inliers theorem'.

The theorem is stated thus:

Given ANY dataset containing m instances, for any natural number, $n < m$, if with respect to ANY model the number of outliers $C < \left\lfloor \frac{m}{n} \right\rfloor$, then there is at least one set of n consecutive instances that are all inliers.

In other words, the theorem states that, if the condition $C < \left\lfloor \frac{m}{n} \right\rfloor$ is satisfied, it is impossible to shuffle the data in such a way that there are no single cluster of *n* inliers that are consecutive.

3.2.1 Proof

The validity of Theorem 1 is quite easy to establish.

Let V be a set $B \cup W$, such that *B* and *W* are mutually exclusive, and *n* any natural number that is less than *m* which is another natural number.

In the context of robust estimation problem, each $v_i \in V$ is a data instance, each $b_j \in B$ is an outlier, and each $w_k \in W$ is an inlier. *n* and *m* are the minimal sample size and the data size respectively.

Since *B* and *W* are mutually exclusive, $B \cap W = 0$.

Therefore $|V| = |B| + |W|$.

In an attempt to ensure that no *n*-tuple made up purely of elements of W are consecutive, a single $b_j \in B$, is used to contaminate a cluster of (*n*-1) w_k's to complete an *n*-tuple. It is easy to see that any different approach may result in an inefficient use of the $b_j's$, causing shortage (none to use for contaminating $w_k's$) in some portions of V.

Now, if $|B| = \left\lfloor \frac{m}{n} \right\rfloor$, then concatenating all $\left\lfloor \frac{m}{n} \right\rfloor$ n-tuples that have been created to use up all $b_j \in B$, creates a set V':$|V'| = |V| - (m \bmod n)$. The remaining *m mod n* slots can only be filled with the unused $w_k's$, which are of course, exactly *m mod n* in number.

Clearly, replacing any of the $b_j's$ in the final set V with a w_k creates at least one cluster of $w_k's$ made up of $n + (m \bmod n)$ elements.

Hence, Theorem 1 is validated. □

An example should make the concept much clearer. Let *m* = 100, *n* = 2, $\left\lfloor \frac{m}{n} \right\rfloor$ = 50. Then let *C* = 49 in order to satisfy the condition $C < \left\lfloor \frac{m}{n} \right\rfloor$. An attempt to ensure that no two inliers are consecutive will have an outlier between any two inliers. The last outlier will have been included in the arrangement at index 97 if we start with an outlier, or 98 if we start with an inlier. Either way, we are left with at least 2 slots yet to be filled, having only inliers to fill them. Therefore at least two inliers must be consecutive in such a dataset. Clearly, if the outlier rate is reduced, then there must be even more consecutive sets of inliers.

Useful Analogy

An analogy that may prove useful in communicating the concept more easily is as follows:

If we have a sequence having for example, 4 (m) balls in total, each of which can either be black (outlier) or white (inlier), if the black balls are fewer than 2 i.e. $\left\lfloor \frac{m}{n} \right\rfloor$, then it is impossible that there are no 2 white balls that lie side by side in the sequence. Same is true if m is 5 balls, since $\left\lfloor \frac{m}{n} \right\rfloor = \left\lfloor \frac{5}{2} \right\rfloor = 2$.

3.2.2 Significance of Theorem 1

If the condition $C < \left\lfloor \frac{m}{n} \right\rfloor$is satisfied, where n is the minimal sample size required to construct a model of interest, then an efficient algorithm can be constructed to look only among hypotheses that are constructed from instances that follow each other in the data table. The algorithm is sure to find at least one all-inlier sample. Hence, it represents an alternative to RANSAC's approach to finding all-inlier samples, which are in turn used to construct good solutions. One reason the approach is expected to be effective is that the condition $C < \left\lfloor \frac{m}{n} \right\rfloor$ is likely to be fulfilled in many practical datasets since $n \ll m$. This drastically narrows search to $(m - n + 1)$ alternatives instead of the exhaustive space of $\binom{m}{n}$ alternatives. Interestingly, this subset size reduces by Δn as n, determined by the model, increases by Δn, for any given dataset. One implication of this is that such an algorithm should possess good complexity in high-model-dimension cases, for which RANSAC has been found to deteriorate in performance [50].

The value of seeking all-inlier samples is already stated in Proposition 1.

3.3 Extension

What if the condition $C < \left\lfloor \frac{m}{n} \right\rfloor$ is not satisfied, resulting in a situation where it is possible that there are no n inliers that are consecutive?

The above question is addressed in proposition 2.

3.3.1.1 Proposition 2

In a typical dataset, the likelihood is high that there will be at least one all-inlier sample containing purely or mostly consecutive data instances.

Proposition 2 simply points out a vital observation that can be gleaned from the proof of Theorem 1. In many cases, it will take a deliberate effort rather than chance, to efficiently arrange outliers is such a way that no *n* inliers are consecutive, even when $C \geq \left\lfloor \frac{m}{n} \right\rfloor$. Should C become so high that such likelihood becomes significant, it will still be quite likely that there is a cluster of *n* consecutive instances that are mostly inliers. Therefore in most practical cases, constructing an algorithm directly from the theorem will usually result in good solutions. The condition $C < \left\lfloor \frac{m}{n} \right\rfloor$is only necessary for an absolute guarantee of finding an all-inlier sample.

3.4 The CISAC (Consecutive Instances Sample Consensus) Algorithm

The algorithm is a direct application of Theorem 1. Therefore, practically all of its working principle has been explained in the section 3.3. As the consecutive inliers theorem dictates, CISAC only enumerates hypotheses constructed from all possible selections of *n* consecutive data instances, where *n* is the minimal sample size or model dimensionality. This is clearly a simple algorithm that is very easy to implement. It's outline is presented in algorithm 3.1.

Outline of the CISAC Algorithm

Input: threshold distance d

1. Generate all possible sets of n consecutive numbers between 1 and n. The procedure below achieves this for any given value of n:

```
m = number of rows in data;
n = minimal sample size corresponding to model type
k = 1
for i = 1 to (m-n+1)
    for d = 1 to n
        samplingIndices(k, d) = i + (d-1)
    end
    k = k + 1
end
```

2. Construct k models such that model(i) is fitted to n data instances whose indices are the n elements in the ith row of samplingIndices.

3. Return model with maximum number of inliers using threshold d to distinguish inliers from outliers.

Algorithm 3.1: The CISAC Algorithm

In algorithm 3.1, samplingIndices is a matrix whose kth row represents the kth sample, and each element d on a that row represents the the indices of the rows that are sampled in the dataset.

3.4.1 Properties of CISAC

Before looking into experimental results in section 3.4.2, certain properties are expected on the basis of the foregoing theoretical discussions. Some of these are discussed below.

Accuracy and Robustness: Theoretically, CISAC is guaranteed to find all-inlier samples when outlier rate is less than $\left\lfloor \frac{m}{n} \right\rfloor$. This means that for fitting a line or a similarity transformation, its guaranteed breakdown point is about 50%, 33% for affine-model fitting, 25% for projective homography, 14% for 7-point geometric models, and so on. However, in practical situations, as the experiments reported in section 4.4.2 show, CISAC maintains good accuracy under fairly high contamination even for projective homography. This implies robustness that in practice, should be competitive with RANSAC's.

Deterministic run time: Since the number of hypotheses generated and evaluated is fixed, unlike RANSAC, CISAC's runtime is completely deterministic.

Space and time Complexity: CISAC performs a fixed-sized sequence of hypothesis generation and evaluations, so it exhibits space and time complexity of $(m - n + 1)$. This makes it an efficient algorithm with increasing advantage as model dimensionality increases, a situation that causes RANSAC performance to deteriorate [50].

Repeatability: No matter how many times CISAC is run, it will generate and evaluate exactly the same set of hypotheses. Therefore it will always arrive at exactly the same solution. This represents a breakthrough in RANSAC literature and a rare property for non-exhaustive-search algorithms in general.

3.4.2 Preliminary Experiments

The goal here is to evaluate the effectiveness of CISAC's simple search strategy, and to assess its advantages if any, over RANSAC. In-depth comparative studies are reserved for chapter 4. The experiments in this section are conducted using simulated datasets and a real-life image pair. The simulated data are used in order to cover a sufficiently wide range of data conditions in terms of data size and contamination level.

3.4.2.1 Experiments with Line-fitting Problem Set

This series of experiments study a simple generic problem – the line fitting problem. The goal is to automatically fit a linear model to data containing outliers. 3 datasets are simulated with outlier rates 10%, 30% and 50% respectively. The total number of data instances in all cases is 100.

The values of the independent variables are randomly generated from a uniform distribution, and the corresponding values of the dependent variable are computed according to the true model:

$$y_i = \begin{cases} 10 + 5x_i + e: & i \leq t\ (inliers) \\ 50 + e: & otherwise\ (outliers) \end{cases}$$

$e\ is\ random\ error: e \in (0,1), e \sim U$

$i \in \{1,2,3, \dots, m\}$

$m\ is\ the\ data\ size$

$t\ is\ the\ true\ number\ of\ inliers$

Each algorithm (CISAC and RANSAC) is run on every problem five times each, to evaluate consistency or repeatability.

Discussion

Results from this experiment are presented in table 3.1. The table shows the full results of these experiments. For each problem, the data size and true number of inliers are recorded. The estimates of the model parameters and the corresponding values of the quality measures are then recorded for each run of the algorithms on each problem. The number of iterations taken by each algorithm is also recorded. Lastly, for fair comparison of efficiency, between both algorithms, the number of calls to the objective function by each algorithm, is recorded.

Table 3-1: Performance of CISAC and RANSAC on the Line Fitting Set

Algorithm	**Problem**	**m**	**T**	**Run**	**B_0**	**B_1**	**ME**	**I**	**iter**	**f**
CISAC	1	100	90	1	10.7326	4.8773	7.8156	90	1	99
CISAC	1	100	90	2	10.7326	4.8773	7.8156	90	1	99
CISAC	1	100	90	3	10.7326	4.8773	7.8156	90	1	99
RANSAC	1	100	90	1	9.6027	5.198	10.6602	90	3	3
RANSAC	1	100	90	2	10.3375	5.1423	8.4333	90	3	3
RANSAC	1	100	90	3	10.4077	4.8983	9.8464	90	3	3
CISAC	2	100	70	1	10.2142	5.2046	12.1647	70	1	99
CISAC	2	100	70	2	10.2142	5.2046	12.1647	70	1	99
CISAC	2	100	70	3	10.2142	5.2046	12.1647	70	1	99
RANSAC	2	100	70	1	10.1367	5.2097	11.9455	70	7	7
RANSAC	2	100	70	2	11.0525	4.8238	12.0723	70	7	7
RANSAC	2	100	70	3	10.3283	5.0762	10.2866	70	7	7
CISAC	3	100	50	1	11.3022	4.8334	17.0681	50	1	99
CISAC	3	100	50	2	11.3022	4.8334	17.0681	50	1	99
CISAC	3	100	50	3	11.3022	4.8334	17.0681	50	1	99

RANSAC	3	100	50	1	10.8263	4.8656	14.8883	50	17	17
RANSAC	3	100	50	2	9.8368	5.2784	15.9395	50	17	17
RANSAC	3	100	50	3	10.6101	4.9628	14.6079	50	17	17

Key: m = data size; **B_0 =** estimate of intercept; **B_1 =** estimate of slope; **ME = M-estimate** error; **I** = number of inliers detected; **iter** = number of iterations; **f** = number of calls to objective function.

Accuracy

The goal of both algorithms is to maximize the number of inliers. So, a higher number of inliers detected implies better accuracy. Since the ground truth (true number of inliers) is known in all cases, it is used to benchmark the performances of both algorithms. As an additional measure of model quality, for each model returned, the M-estimate error is computed. This is the error function that MSAC, another popular variant, seeks to minimize. The point is that two models may have the same quality with respect to the number of inliers, but differ in M-estimate error, which measures the bounded error within the threshold, rather than merely polarizing data points as inliers or outliers. Precisely, the error is computed as:

$$\sum_{i=1}^{m} \rho_1(e)$$

$$\rho_1(e) = \begin{cases} |e|, & |e| < |T| \\ |T|, & otherwise \end{cases}$$

As seen in Table 3.1, in all cases tested, CISAC returns the exact number of inliers. Same is true for RANSAC for nearly all cases, but with a few exceptions where it returns slightly poorer results.

This validates the effectiveness of the sample consensus paradigm in general, especially when compared to other robust linear regression techniques that are popular in statistics. The paradigm produces algorithms with robustness that is better than or equal to the likes of LMedS, in terms of breakdown point.

In terms of the M-estimate error, CISAC and RANSAC seem to compete closely. Again, recall that the emphasis of this experiment is validation of CISAC's effectiveness rather than comparative study which is reserved for chapter 4.

Solution Consistence/Repeatability

Perhaps a more important performance measure which reveals CISAC's advantage over RANSAC is the consistence of solutions returned. The theoretical discussion provided earlier on this subject is validated by the experiments. No matter how many times it is run on a given problem, CISAC returns exactly the same model, not just the same model quality. This is not true for RANSAC, nor has it ever been claimed for any RANSAC variant, to the best of our knowledge.

The implication of this is that owing to the consecutive inliers theorem, CISAC represents a breakthrough in addressing the long-standing reliability problem of RANSAC. CISAC is perfectly repeatable. In other words, although CISAC adopts non-exhaustive search like every other RANSAC variant, it is deterministic.

Run time Consistence

Another interesting behavior of CISAC, revealed in the experiments, although already discussed theoretically, is the fact that its run time is completely deterministic. It can easily be computed ahead for any problem, as a function of the data size for a given model type. RANSAC's run time varies when it is run on the same problem.

Speed

Considering the processing power of today's computers, both algorithms proved quite fast and efficient. This is shown by the number of function calls undertaken by each algorithm for each problem. While CISAC's runtime is perfectly deterministic and consistent, RANSAC is generally faster. In any case, the number of CISAC's function calls on each problem will take fractions of a second to compute, on today's average computer. So although slower than RANSAC on this problem set, CISAC is still fast enough.

Also unlike RANSAC, CISAC's runtime is not affected by outlier rate. This experiment as well the one reported for affine transformations in section 3.4.2.2 show that RANSAC speed deteriorates gradually as outlier rate is increased.

3.4.2.2 Experiment on Simulated Affine Transformation Problems

The methodology here is identical to that of Experiment 1, except for the model type being studied. Here the problems involve affine transformations. The minimal sample size corresponding to an

affine transformation is 3, that is, at least three instances are required to define it. Both the input and output in the dataset are bivariate. This is typical of robust estimation datasets in computer vision, where each row of the input matrix represents the spatial coordinates of matched features in the base image and the output matrix represents the corresponding coordinate in the transformed image. The true model used in each case studied is generated according to:

$$y_{ij} = \begin{cases} \theta^T x_{ik} + R_{ip}: & i \leq t\ (inliers) \\ 100 + 10R_{ip}: & otherwise\ (outliers) \end{cases}$$

$$\theta = \begin{bmatrix} 5 & 3 & 0 \\ 3 & 4 & 0 \\ 1 & 4 & 1 \end{bmatrix}$$

$$i = 1, 2, 3, \dots, m,$$

$$j, k, p \in \{1,2\}$$

$$x_{i1} = 5 \times r_1,\ x_{i2} = 3 \times r_2$$

$$real\ random\ numbers\ R_{ip}, r_1, r_2 \in (0,1) \sim U$$

The distance threshold is chosen through experimentation as 1.5.

Due to space constraints, only performance measures are included in table 3.2. Actual estimates of model parameters are not included in the table, since they are not required for performance evaluation. The observations are similar to those seen for the line fitting problem set. But for this problem set, CISAC's advantages becomes slightly more pronounced. It is clearly more accurate than RANSAC on both measures of accuracy: number of inliers *I* and M-estimate error *ME*. Interestingly, still in all cases CISAC returns exactly the true number of inliers.

Another interesting observation is the effect of increased model dimensionality. The line-fitting model has dimensionality of 2 while the affine model's is 3. CISAC becomes faster while RANSAC becomes slower as the model dimensionality increases, thereby reducing the speed gap between the two algorithms.

Table 3-2: Performance of CISAC and RANSAC on the Affine Transformation Set

Algorithm	Problem	m	t	ME	I	t	f
CISAC	1	100	90	59.593	90	1	98
CISAC	1	100	90	59.593	90	1	98
CISAC	1	100	90	59.593	90	1	98
RANSAC	1	100	90	68.8916	90	5	5
RANSAC	1	100	90	65.6811	90	6	6
RANSAC	1	100	90	64.1086	90	5	5
CISAC	2	100	70	82.018	70	1	98
CISAC	2	100	70	82.018	70	1	98
CISAC	2	100	70	82.018	70	1	98
RANSAC	2	100	70	85.3102	70	12	12
RANSAC	2	100	70	81.9344	70	12	12
RANSAC	2	100	70	86.6514	69	13	13
CISAC	3	100	50	102.2057	50	1	98
CISAC	3	100	50	102.2057	50	1	98
CISAC	3	100	50	102.2057	50	1	98
RANSAC	3	100	50	107.4615	49	38	38
RANSAC	3	100	50	104.0805	50	36	36
RANSAC	3	100	50	111.1799	44	53	53

Key: all symbols in the table have the same meaning as defined for Table 3.1.

3.4.2.3 Projective Homography on Aerial Photos of UKZN Campus

The goal of this experiment is to automatically estimate homography between matched features in two overlapping real-life aerial photographs. The two maps used in this experiment have different details, view, and zoom, of the same scene – the Westville campus of UKZN and its neighbourhood. Since the campus is found on both images, they are sure to have several features in common, such as buildings, road junctions, trees, and so on. Common features are first automatically detected and their coordinates extracted using the Speeded-Up Robust Features (SURF) technique. The extracted coordinate sets constitute the data to which the model is to be fit. This is a typical robust estimation problem because only a subset of the matched feature coordinate sets will conform to the best fitting model. In this context, others are treated as outliers. In the sample consensus approach, the best model is the one with the highest number of inliers.

(a) Image 1 (b) Image 2

Figure 3.1: Aerial Photos of UKZN Campus and its neighbourhood (Source: Google maps)

The appropriate threshold is determined empirically by seeking the smallest threshold that results in a stable solution for this image pair, varying its value from 0 in steps of 0.5. Its appropriateness is verified visually, by applying the inverse of the resulting transformation on image 2, to stitch the images, such that common features are aligned. A threshold of 4 was found to be appropriate. Using this threshold, CISAC and RANSAC are both run10 times each, to estimate the 2-D transformation necessary to align the image pair.

Table 3-3: Performance Evaluation on the UKZN Aerial Photo Pair

	Run												
Algorithm	1	2	3	4	5	6	7	8	9	10	Worst	Best	Median
CISAC	133	133	133	133	133	133	133	133	133	133	133	133	133
RANSAC	108	123	130	130	130	110	120	131	117	128	108	131	125.5

Table 3.4: Runtime Evaluation on the UKZN Aerial Photo Pair, measured by number of hypotheses tested

Run	RANSAC	CISAC
1	10	132
2	5	132
3	5	132
4	8	132
5	5	132
6	9	132
7	6	132
8	4	132
9	7	132
10	11	132

As shown in table 3.3, CISAC exhibited superior accuracy over RANSAC on this real-life problem. RANSAC is not only worse in accuracy, but also not stable, unlike CISAC that is totally deterministic. Concerning runtime, again the number of CISAC function calls can be performed by the average computer today in fractions of a second. So although RANSAC is faster, CISAC possesses practically satisfactory speed.

3.4.3 Summary of Experimental Results Comparing CISAC and RANSAC

The purpose of the experiments reported in this section is to verify the effectiveness of constructing a robust estimator based purely on the consecutive inliers theorem. Although some comparison is done with RANSAC, more in-depth comparative analysis is reserved for chapter 4.

The algorithm proposed and studied in this section, CISAC, is perfectly repeatable (deterministic) and observed to be more accurate than RANSAC in the cases tested. To the best of our knowledge, perfect repeatability has never been claimed for any RANSAC variant. It is a rare property indeed for non-exhaustive search algorithms in general. Concerning speed, although the cases tested, RANSAC performs less objective function calls, a measure of run time, an advantage of CISAC is that the number of function calls is completely deterministic. It can be computed ahead for any given dataset exactly as $m-n+1$, m being the data size and n the minimal sample size defined by the model of interest. This predictability holds much value for practical applications. Moreover, in the light of the computing power of modern computers, in practically all of the cases tested, especially the real-life image problem, the number of CISAC's function calls will execute in fractions of a second. Also, CISAC's efficiency advantage becomes increasingly pronounced as model dimensionality increases, because of its constant time complexity, which is not affected at all by data contamination. It should also be noted that in practical situations, a random algorithm like RANSAC may need to be run multiple times, before reliable conclusions can be drawn about its solutions. Such does not arise with a perfectly repeatable algorithm like CISAC.

CISAC will likely be considered by readers familiar with RANSAC literature as one of the simplest RANSAC variants ever developed. It is remarkably easy and straightforward to implement. It requires no prior information for biasing sampling. Informed by the consecutive inliers theorem, it simply compares models constructed from instances that follow themselves in the data table.

3.5 Study of Automatic Threshold Estimation Problem: The Value of Determinism

The performance of RANSAC and its variants depends on the choice of error threshold used in the algorithm to distinguish inliers from outliers. It has been shown that the optimal threshold range is associated with the range in which model parameters estimates is stable: the range that minimizes variance of parameters (VoP) [28]. While such an approach works, it is difficult to

estimate the threshold and model parameters simultaneously. However, this approach is necessitated by the stochastic behaviour of RANSAC. The completely non-random nature of CISAC creates new possibilities. This part of the chapter shows how it simplifies the problem of threshold learning. Minimization of VoP is simplified into search for the closest threshold range to zero where the solution does not change. Such an approach is remarkably simple and is capable of simultaneous estimation of threshold and model parameters.

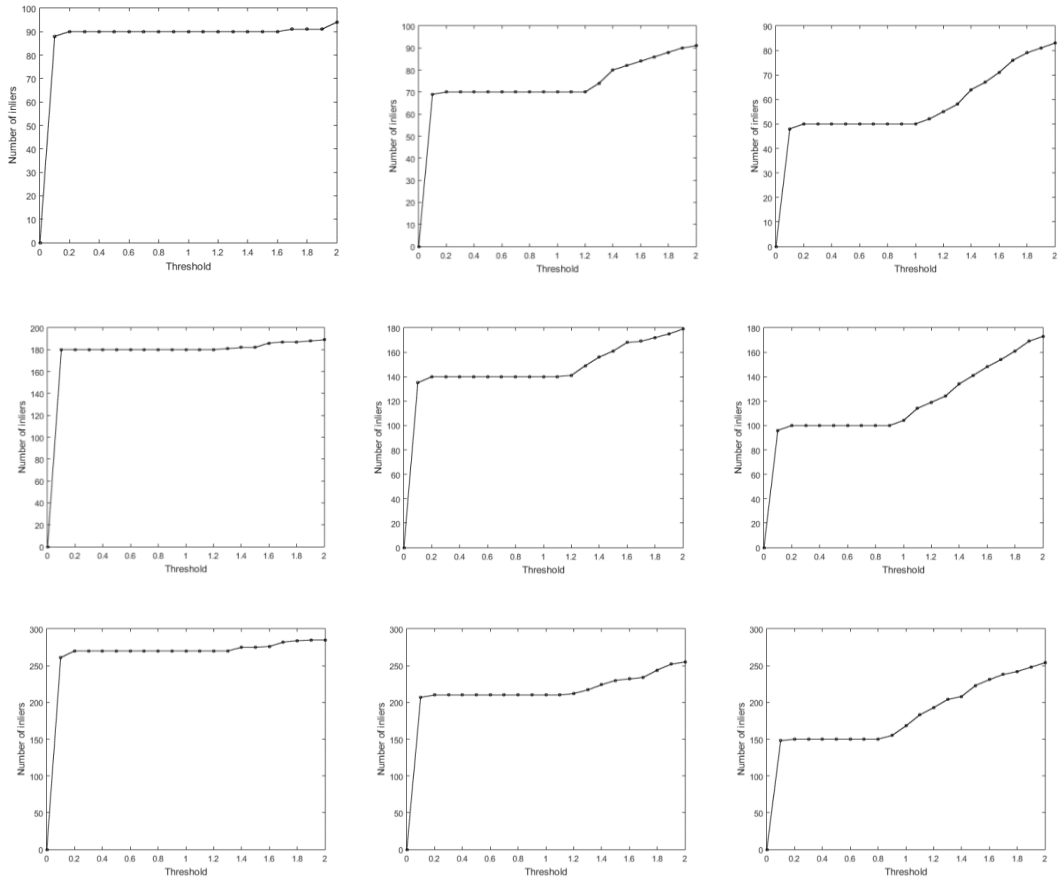
While perfect repeatability eliminates the possibility of adopting the VoP-based approach, since we would now have zero variance in solutions for each threshold value, the behavior over varying threshold values shows that the problem can be approached in a much simpler way. As the experiments in section 3.5.1 reveal, in the optimal threshold range, CISAC's solution remains steady and unchanging. Furthermore, over this range, the detected number of inliers is exactly equal to the true number of inliers.

3.5.1 Experimental Results

The purpose of the series of experiments is to verify the effectiveness of the proposed automatic threshold learning approach. Using 18 different datasets, the trend in solutions produced by CISAC over defined range of threshold values, is observed. The first 9 are line-fitting problems while the other 9 involve affine homographies. Simulated data is used so that for each problem, the ground truth on the model, inlier rate and error range are known. The observed outcomes can therefore be accurately benchmarked. For reasonable generalization, various data sizes and outlier rates are studied.

3.5.1.1 Experiment 1 (Line-fitting Problem Sets)

The datasets studied for line-fitting are simulated according to the same model of experiment in section 3.4.2.1. Figure 3.4 shows the observed trends.



Top row (left to right): Data size = 100 (i) outliers = 10% (ii) outliers = 30% (iii) outliers = 50%

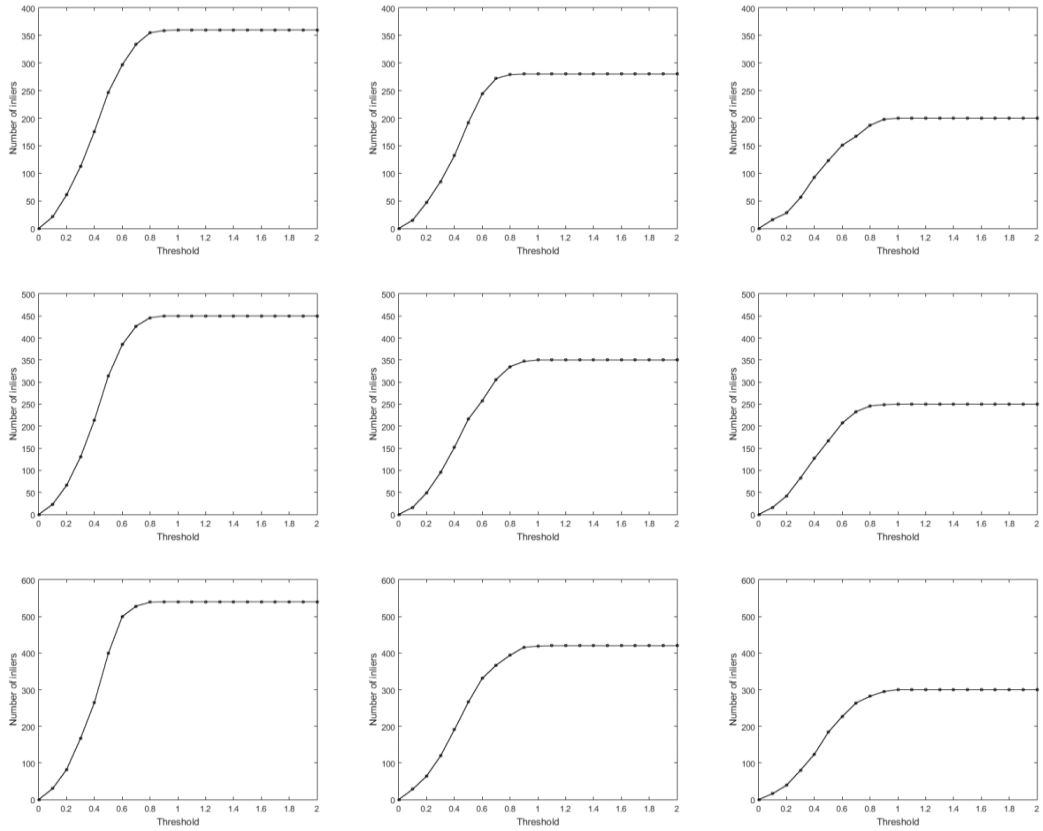
Middle row (left to right): Data size = 200 (i) outliers = 10% (ii) outliers = 30% (iii) outliers = 50%

Bottom row (left to right): Data size = 300 (i) outliers = 10% (ii) outliers = 30% (iii) outliers = 50%

Figure 3.2: Plots of number of inliers detected by CISAC for each threshold value used

3.5.1.2 Experiment on Affine Transformation Problem Set

The datasets studied in this series of experiments are simulated according to the same model of experiment in section 3.4.2.2. Figure 3.5 shows the observed trends.



Top row (left to right): Data size = 100 (i) outliers = 10% (ii) outliers = 30% (iii) outliers = 50%
Middle row (left to right): Data size = 200 (i) outliers = 10% (ii) outliers = 30% (iii) outliers = 50%
Bottom row (left to right): Data size = 300 (i) outliers = 10% (ii) outliers = 30% (iii) outliers = 50%

Figure 3.3: Plots of number of inliers detected by CISAC for each threshold value used

Figure 3.2 shows the number of inliers detected by CISAC for each threshold value used on each of the nine line-fitting problems. There are two interesting observations that are easily seen from these graphs. The first is that CISAC returns the exact number of inliers for each problem over a contiguous range of threshold values. Interestingly, this is the stable range being sought: the range in which the number of inliers does not change. The second observation is that the stable range is very wide when inlier rate is higher and narrows gradually as the outlier rate is increased. The narrowing of the stable region is caused by be the fact that higher outlier rate shifts the starting point of the stable range to the right (higher threshold value).

The same observations hold for all the affine transformation problems as shown in Figure 3.3. It can be concluded, that the observations about the stable range is general and independent of problem, model, data size, and contamination level. The observations suggest an approach to

automatic threshold estimation, which could be adopted in developing fully automatic robust estimators. The generalization is further supported by tests on real-life problems carried out in section 3.6 for the fully automatic algorithm presented in that section.

3.5.2 Summary of Study on Threshold Estimation

The problem of automatic estimation of error threshold, a parameter upon which RANSAC and many of its variants depend, is investigated. The existing approach of minimization of variance of parameters (VoP) which involves multiple runs of RANSAC per threshold value is simplified into single runs of CISAC per threshold value. Hence, the search for the stable range is reduced to a search for the range in which the solution does not change. Such an approach is made possible by the determinism of CISAC: a facility made possible by insights from the consecutive inliers theorem proposed in this thesis. The empirical studies reported in this section show that this stable range corresponds to the optimal threshold range. The corresponding model estimates are accurate and the inlier rates are exact, with respect to available ground truth, in all cases tested. So the approach is not only simple but precise. An additional advantage is that it affords simultaneous estimation of the threshold and model.

To spell out the observations in clear terms, the following are concluded from the study:

- Given a perfectly repeatable algorithm like CISAC, the solution quality (number of inliers) is a monotonically non-decreasing function of the distance threshold used in distinguishing inliers from outliers.
- As the threshold is increased from zero, the solution quality rises steadily until it smoothens out into steady state.
- It stays steady over a range of threshold values that is generally wide.
- The lower the outlier-contamination, the more quickly the steady state is reached, hence the wider the steady range.
- Interestingly, this steady value is found in all tested cases to be the true number of inliers

3.6 AutoCISAC: Deterministic, Fully Automatic Algorithm

A fully automatic robust estimator is presented. This is the second novel algorithm proposed in this thesis. It is based on CISAC and the parameter estimation technique established in section 3.5 of this chapter. The proposed framework is dubbed AutoCISAC. Experiments show that it is accurate and just like CISAC, it is also perfectly repeatable. For simulated problems with known ground truth, AutoCISAC returns the exact number of inliers. Real-life applications involving fully automatic image stitching, are also presented in which AutoCISAC is used in the transformation learning step.

3.6.1 The Algorithm

Starting from 0, the threshold is gradually increased and CISAC is run to compute model parameters and the corresponding number of inliers. A step size of 0.5 should be fine for most image-based applications while a smaller step size may be used if the algorithm is to be applied on regular real-valued numerical data. This process running CISAC per threshold value continues until the same number of inliers is reported by consecutive iterations. This implies that the appropriate threshold as well as the solution has been found. Any of the threshold-and-model-parameters set is appropriate, since they yield the same model quality measure. To avoid the need to store previous solutions, the latter one is chosen. This also avoids choosing one that is on the border of the range. For both reasons, the larger threshold is a better choice.

As shown by results of the experiments reported in this section, the estimated threshold and model parameters are optimal and the number of inliers is accurate.

3.6.2 Experiments and Applications

The performance of AutoCISAC is evaluated through a few experiments described in this subsection. The goal is to evaluate the effectiveness of its simple search approach, and to assess its advantages if any, over the gold-standard RANSAC. The experiments are conducted using simulated data and real-life images. The simulated data are used in order to cover a sufficiently wide range of data conditions in terms of data size and contamination level.

3.6.2.1 Experiment on Simulated Line-fitting Problem Set

Again, the line-fitting problems are simulated similar to that of section 3.4.2.1. Fifteen datasets are used in this study. Just for the sake of variety, the parameters of the true model, are changed. The model used is as follows:

$$y_i = \begin{cases} 20 + 4x_i + e: & i \leq t\ (inliers) \\ 50 + e: & otherwise\ (outliers) \end{cases}$$

$e\ is\ random\ error: e \in (0,1), e \sim U$

$i \in \{1,2,3, \dots, m\}$

$m\ is\ the\ data\ size$

$t\ is\ the\ true\ number\ of\ inliers$

The algorithm is run on each problem three times, to evaluate repeatability.

Table 3.5 showing the experimental results show that the automated mechanism of AutoCISAC is very effective. In all cases tested, the algorithm reports exactly the true number of inliers.

Table 3-4: AutoCISAC's Estimates and Corresponding Number of Inliers

Problem	Data size	True num of inliers	Run	B_0 (AutoCISAC's estimate)	B_1 (AutoCISAC's estimate)	AutoCISAC's detected num of inliers
1	100	90	1	19.9552	4.1607	90
1	100	90	2	19.9552	4.1607	90
1	100	90	3	19.9552	4.1607	90
2	100	70	1	20.0444	4.0201	70
2	100	70	2	20.0444	4.0201	70
2	100	70	3	20.0444	4.0201	70
3	100	50	1	20.5887	3.9114	50
3	100	50	2	20.5887	3.9114	50

3	100	50	3	20.5887	3.9114	50
4	200	180	1	20.5947	4.0296	180
4	200	180	2	20.5947	4.0296	180
4	200	180	3	20.5947	4.0296	180
5	200	140	1	20.8991	3.8127	140
5	200	140	2	20.8991	3.8127	140
5	200	140	3	20.8991	3.8127	140
6	200	100	1	20.4056	4.1682	100
6	200	100	2	20.4056	4.1682	100
6	200	100	3	20.4056	4.1682	100
7	300	270	1	21.0449	3.9763	270
7	300	270	2	21.0449	3.9763	270
7	300	270	3	21.0449	3.9763	270
8	300	210	1	20.8924	3.8452	210
8	300	210	2	20.8924	3.8452	210
8	300	210	3	20.8924	3.8452	210
9	300	150	1	20.8047	3.9461	150
9	300	150	2	20.8047	3.9461	150
9	300	150	3	20.8047	3.9461	150
10	400	360	1	20.6975	3.9143	360
10	400	360	2	20.6975	3.9143	360
10	400	360	3	20.6975	3.9143	360
11	400	280	1	20.1568	4.0886	280

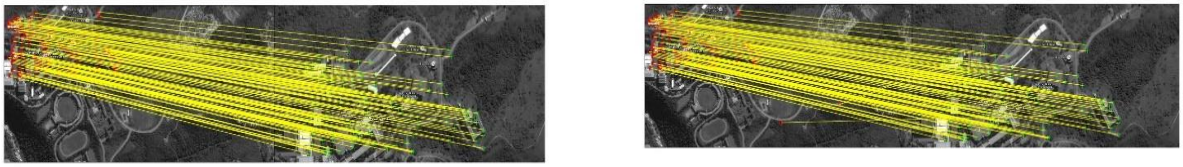
11	400	280	2	20.1568	4.0886	280
11	400	280	3	20.1568	4.0886	280
12	400	200	1	20.0114	4.1192	200
12	400	200	2	20.0114	4.1192	200
12	400	200	3	20.0114	4.1192	200
13	500	450	1	20.3967	3.9833	450
13	500	450	2	20.3967	3.9833	450
13	500	450	3	20.3967	3.9833	450
14	500	350	1	20.0674	4.0479	350
14	500	350	2	20.0674	4.0479	350
14	500	350	3	20.0674	4.0479	350
15	500	250	1	20.9058	3.9198	250
15	500	250	2	20.9058	3.9198	250
15	500	250	3	20.9058	3.9198	250

3.6.2.2 Application 1: Projective Homography on Aerial Photo of UKZN Campus

In this application, AutoCISAC is used to automatically estimate homography between matched features in the same pair of aerial photographs used in section 4.4.2.3. Common features are first automatically detected and their coordinates extracted using the SURF technique, before the transformation is estimated. The effectiveness of AutoCISAC is judged visually, by applying the inverse of the estimated transformation to the second image and displaying the transformed image on the same coordinate system as the first image. The outcome is a final stitched image in which common features are aligned.

(a) Image 1 (b) Image 2

Figure 3.4: Aerial Photos of UKZN Campus and its neighbourhood (Source: Google maps)



(a) Matched Features (b) Matches treated as inliers by AutoCISAC

(c) Final stitching obtained by applying the AutoCISAC-estimated transformation on Image 2 and displaying the transformed pixels on the same coordinate system as Image 1. The stitched image combines details from both images with common features aligned quite seamlessly. Empty spaces are zero padded (filled with black)

Figure 3.5: Stitching the UKZN aerial photo pair 1

3.6.2.3 Application 2: Projective Homography on an Outdoor Bus Scene

This follows the same procedure as application 1 in section 3.6.2.2, using the pair of images in Figure 3.8. Figure 3.9c shows the final stitched image.

(a) Image 1 (b) Image 2

Figure 3.6: Bus image pair

(a) Matched Features

(b) Matches treated as inliers by AutoCISAC

(c) Final stitching obtained by applying the AutoCISAC-estimated transformation on Image 2 and displaying the transformed pixels on the same coordinate system as Image 1. The stitched image combines details from both images with common features aligned quite seamlessly. Empty spaces are zero-padded (filled with black)

Figure 3.7: Stitching the Bus Scene Pair

3.6.3 Summary of Study on AutoCISAC

A fully automatic robust estimator named AutoCISAC, is proposed. Although based on the sample consensus paradigm of RANSAC, it eliminates all dependence on user-supplied parameters. It is perfectly repeatable, that is, its solutions are non-random. The algorithm basically extends CISAC by incorporating the automatic threshold estimation technique presented in section 4.5.

AutoCISAC returns number of inliers exactly equal to the ground truth in all simulated cases tested, and the solutions are perfectly repeatable. Real-life automatic image stitching applications are presented in which AutoCISAC is used in the homography estimation stage. These applications show the algorithm to be effective in such practical situations. It represents another innovation made possible by the consecutive inliers theorem.

3.7 SASSAC: Safeguarding Against Risk in the CISAC Algorithm

While for many practical problems, CISAC maintains its accuracy even under higher outlier-contamination, there is inherent risk of breakdown in the algorithm when the outlier rate is higher than the contamination threshold defined for a given problem by the consecutive inliers theorem. The reason CISAC is often accurate is that this threshold is high enough for many practical datasets. Moreover, having outlier rate that exceeds this threshold only creates a risk rather than certainty of breakdown. But this risk may be higher when outlier rate and model dimensionality are both very high. The algorithm proposed in this section is still based on CISAC but includes an additional mechanism for mitigating the risk. The algorithm is named Shuffle-and-Sweep Sample Consensus (SASSAC). SASSAC eliminates the need for data to meet the theoretical condition that guarantees CISAC's accuracy. As the comparative study presented in chapter 4 shows, SASSAC does not only overcome the earlier mentioned limitation but has an additional advantage of improved robustness to poor choice of error threshold used to distinguish inliers from outliers. This is a crucial parameter which is typically supplied by the user to algorithms belonging to the RANSAC family. The beauty of the mechanism is that it is indeed simple and easy to implement, so SASSAC still preserves RANSAC's simplicity.

3.7.1 The Algorithm

SASSAC is based on the logic that even if the outlier rate is greater than the threshold defined by the consecutive inliers theorem, at least one all-inlier sample and ultimately a good solution should eventually be constructed if CISAC runs iteratively on different random permutations of the data. This iterative process continues until the solution quality (number of inliers) found at an iteration is equal to that found in a previous iteration. This logic is based on the observation that the solution quality is likely to exhibit such 'stability' over different permutations of the data, for the best solution, and is likely to be the true number of inliers. The effectiveness of SASSAC's strategy is validated through experiments reported in chapter 4 in which a comparative study is carried out

on six algorithms. These include CISAC, SASSAC, RANSAC, MSAC and two others –MSASAC and MCISAC - described in that chapter. SASSAC improves CISAC's reliability and accuracy at a cost: a bit of randomness is introduced. The study in chapter 4 shows that SASSAC quite consistent, but it is not totally deterministic like CISAC. This behavior comes from the random permutation component.

Outline of the SASSAC Algorithm

Input: threshold distance d

1. Initialize I1 = 0, T1 = model with parameters equal 0
2. Generate all possible sets of *n* consecutive numbers between 1 and *m*. The procedure below achieves this for any given value of n:

```
m = number of rows in data;
n = minimal sample size corresponding to model type
k = 1
for i = 1 to (m-n+1)
    for d = 1 to n
        samplingIndices(k, d) = i + (d-1)
    end
    k = k + 1
end
```

3. Construct k models such that model(i) is fitted to n data instances whose indices are the n elements in the ith row of samplingIndices.
4. Compare the models and store the best in T2
5. I2 = max number of inliers (corresponding to T2)
6. If I1 == I2,

   terminate and return T2 as final model

   else I1 = I2; T1 = T2

   Shuffle data rows randomly

   Go to 2

Algorithm 3.2: Outline of the SASSAC algorithm

3.8 Two More Algorithms: Novel M-estimators

Torr and Zisserman [21] proposed an alternative to RANSAC's measure of model quality which takes into account the magnitudes of individual errors of inlier points rather than RANSAC's binary polarization. The alternative model quality measure is an error function to be minimized, given as:

$$\boldsymbol{\rho_1(e^2)} = \begin{cases} |\boldsymbol{e}|, & |\boldsymbol{e}| < \boldsymbol{T} \\ \boldsymbol{T}, & |\boldsymbol{e}| \geq \boldsymbol{T} \end{cases}$$

The resulting variant is named M-estimator Sample Consensus (MSAC). This is the variant that the MATLAB computer vision toolbox uses for its homography estimation function: *estimateGeometricTransform.* MSAC adopts the same search strategy as RANSAC: only difference between both is the model quality measure used as the optimization objective.

All three algorithms proposed so far in this chapter are based on the consensus set maximization objective of RANSAC. Two more algorithms are proposed in this section, which adopt the M-estimator objective of MSAC. The resulting algorithms are named MCISAC and MSASSAC. Just as MSAC is the M-estimator version of RANSAC, MCISAC and MSASSAC are the M-estimator versions of CISAC and SASSAC respectively. Other than the change of model quality measure, MCISAC is identical with CISAC; same goes between SASSAC and MSASSAC. An important clause here is that although MSASSAC replaces SASSAC's optimization objective, it still maintains the concept of terminating when the number of inliers of the best solution found through the sweep phase of one iteration is same as that found in a previous iteration. This choice is made because termination based on a difference in a real-valued quantity such as M-estimate error will result in an algorithm that runs in unnecessarily long time.

3.9 Chapter Summary

Much of the novel content of this thesis is presented in this chapter. The consecutive inliers theorem is proposed, which reveals a straightforward way to achieve RANSAC's goal: finding all-inlier samples in order to find good solutions. This theorem leads to the development of five algorithms. Three are totally deterministic, which is a property that has never been claimed in RANSAC literature to the best out our knowledge. A set of preliminary experiments and real-life applications show that the proposed algorithms are not only efficient, but very accurate, often reporting number of inliers that is equal to ground truth. While in-depth comparative studies are reserved for chapter 4, some comparison is done in this chapter that reveal the superiority of the proposed algorithms to RANSAC in accuracy and consistence. Other interesting properties achieved are tractable complexity even for high-model-dimension problems; predicatble and consistent runtime that is unaffected by outlier rate.

All proposed algorithms meet the design requirements set at the start of this research: no extra operations, no dependence on problem-specific priors; only a replacement of RANSAC's random-sampling search strategy to achieve all improvements.

CHAPTER FOUR

COMPARATIVE STUDY OF THE NEW ALGORITHMS AND THEIR RANDOM-SAMPLING COUNTERPARTS

4.0 Chapter Introduction

Chapter 3 presents the consecutive inliers theorem proposed in this work. This theorem reveals a direct way to find all-inlier samples used to construct good solutions. The algorithms, CISAC, SASSAC, MCISAC and MSASSAC are borne out of application of this theorem. The comparative study presented in this chapter is valuable for two main reasons. One is the fact that the algorithms being compared represent two different search strategy categories: random sampling and the consecutive-inliers strategy. This helps to evaluate the effectiveness of the latter, which is novel to this thesis. Another value is in the fact that the random-sampling algorithms, RANSAC and MSAC, have been favoured in software implementations such as functions in MATLAB's computer vision toolbox and the OpenCV library. Given the remarkable properties achieved in the design of the new algorithms, if any of them can be shown to offer performance advantages over either of the two existing algorithms, then such will be a good candidate for future software implementations.

The study compares the performances of all six algorithms on the most general 2-D linear geometric transformation problem: projective homography. Simulated data as well as real-life images are used. Two real-life image pairs are used in the study to ensure that conclusions drawn from the study apply to practical problems. Both are aerial photos of the Westville campus of UKZN and its neigbourhood. Simulation helps to cover a wide range of data conditions and also affords the possibility of benchmarking performance with known ground truth. Moreover, all the simulated datasets are generic, so conlusions drawn in this study can be relied on for model-fitting problems from any field.

Sections 4.1, 4.2 and 4.3 describe the methodology, performance criteria of interest and data analysis approach respectively while section 4.4 presents experimental results, analysis and discussion.

4.1 Methodology

A total of 18 simulated datasets are used in this study, to cover a wide range of data conditions. They are categorized into two groups: the first 9 are datasets having outlier rates between 10% and 30%, while the other 9 have outlier rates between 40% and 60%. The datasets are generated according to the following model:

$$y_{ij} = \begin{cases} \theta^T x_{ik} + R_{ip}: & i \leq t\ (inliers) \\ 100 + 10R_{ip}: & otherwise\ (outliers) \end{cases}$$

$$\theta = \begin{bmatrix} 5 & 3 & 2 \\ 3 & 4 & 6 \\ 1 & 4 & 1 \end{bmatrix}$$

$$i = 1, 2, 3, \dots, m,$$

$$j, k, p \in \{1,2\}$$

$$x_{i1} = 5 \times r_1$$

$$x_{i2} = 3 \times r_2$$

$$real\ random\ numbers\ R_{ip}, r_1, r_2 \in (0,1) \sim U$$

The model represents projective homography transformation. It is the most general form of linear 2-D transformation, with minimal sample size of 4. Each dataset is passed to each of the six algorithms for an estimate of the true model to be computed. An input parameter required by all six algorithms is the distance threshold. Additional inputs to RANSAC and MSAC are the confidence level and the maximum number of trials or iterations, both of which are related to the stopping criterion adopted by both algorithms. As described in chapter 3, the new algorithms do not require this termination criterion, so these parameters do not apply to them. An optimal value of the distance threshold is empirically determined according to the process described and justified in section 3.5. The confidence level used by RANSAC and MSAC for computing the lower time bound in the stopping criterion is set to 99% while the maximum number of iterations is set to 5000.

The problem sets are described thus:

Problem set 1: Estimation of projective homography for datasets in category 1 (outlier rate 10%, 20% and 30%).

Problem set 2: Estimation of projective homography for datasets in category 2 (outlier rate 40%, 50% and 60%).

The algorithms are also evaluated on real-life image pairs. The images were sourced from Google maps. They are aerial photographs of the Westville campus of UKZN and its neighbourhood. The two images that make up a pair contain overlapping details including some difference in zoom and orientation. Common features are automatically identified using the SURF technique implemented in the *detectSURFFeatures* function of MATLAB's computer vision toolbox (2015b). The coordinates of the matched features are extracted as the dataset from which all six algorithms attempt to learn the geometric transformation relating both images.

4.2 Performance Criteria

The algorithms are evaluated and compared in terms of accuracy, consistence of solution and runtime, and speed. Accuracy is measured using two alternative metrics: the consensus maximization objective of RANSAC (the number of inliers) and the M-estimate error of MSAC. Repeatability is measured by the amount of randomness (variation) observed when an algorithm is run multiple times on the same problem. Adopting a popular practice in the field of optimization, a field that has much to do with development of search techniques, speed is measured by the number of times an algorithm calls the objective function. This measure is independent of hardware platform, so it is more objective than measuring runtime in seconds.

Note:

The two measures of accuracy follow opposing directions. Number of inliers is to be maximized while M-estimate error is to be minimized. Therefore in interpreting the results, when accuracy is measured using the former, higher values imply better accuracy, while the reverse is the case when accuracy is measured by M-estimate error.

4.3 Data Analysis Approach

Preliminary analyses are carried out using informative visualizations. Specifically, boxplots are used to compare at a glance, the accuracies and variation in solutions produced by each algorithm.

Statistical hypothesis testing techniques are used to evaluate the significance of differences observed between the set of results produced by each algorithm per problem set. The goal is to

make inferences or generalizations on the overall ranking of each algorithm relative to others. Statistical significance is tested using the Friedman test. Where the difference is confidently concluded to be significant (at 95% confidence level), additionally, a posthoc analysis is performed for pairwise comparison of the algorithms, using the Nemenyi test.

Interpreting the boxplots

The boxplot uses box and whiskers to represent the minimum, first quartile, median, third quartile and maximum in a set of numerical data values. The bottom and top ends of the whiskers represent the minimum and maximum respectively, the bottom and top edges of the boxes are the first and third quartile, and the middle line is the median. The vertical length of the boxes (interquartile range) and the distance between the ends of the whiskers (range) give a measure of variation in the solutions produced over multiple runs on the same problem. Extreme values are exempted from the process of arriving at these statistics, and displayed as isolated points.

For easier visualization, a consistent colour code is adopted: green for SASSAC, brown for CISAC, yellow for RANSAC, red for MSASSAC, gray for MCISAC, blue for MSAC. The larger the colour area, the larger the interquartile range, which is a measure of variation. High variation implies poor repeatability.

Interpreting Results from Friedman tests

The hypotheses of the test are formulated as follows:

Null Hypothesis: There is no statistically significant difference in the mean ranks of the number of inliers detected by all six algorithms over all the problems.

Alternate Hypothesis: There is significant difference in the ranks.

The test statistics is given as:

$$F = \frac{12}{rc(c+1)} \sum_{i=1}^{c} {R_j}^2 - 3r(c+1)$$

${R_j}^2$ = square of the total of the ranks for group j. In the reported experiments, a group is an algorithm.

r = number of blocks, that is number of problem instances for which the comparisons are being done.

c = number of groups.

F follows the Chi-square distribution.

The test is performed on the results of the experiments in this chapter using facilities in the PMCMR package in the R software. The p-value is returned. A p-value less than 0.5 indicates that at the standard confidence level of 95%, the null hypothesis can be confidently rejected and the alternate hypothesis accepted. Otherwise, we fail to reject the null hypothesis due to the absence of sufficient evidence.

Interpreting the Nemenyi tests

Upon rejection of the null hypothesis in the Friedman test, Nemenyi pairwaise comparison is performed and the outcomes are displayed in a table. Each cell *ij* represents p-values for test of significant difference between algorithms *i* and *j*. Similar to the Friedman test, significant difference can only be confidently concluded when p-value is less than 0.5.

Interpreting Mean and Median Ranks

Since there are six algorithms being compared in the study, the highest mean and median rank score is 6, the lowest being 1. A value of 6 is assigned to the algorithm with the highest value of the measure of interest. That is, in the case where accuracy is measured using number of inliers, the best algorithm gets a rank of 6. The best algorithm on the M-estimate error criterion will get a ranking of 1, since it has the lowest value of the measure. Where there are ties the same rank is assigned to the tied algorithms.

4.4 Experimental Results for Simulated Problem Sets

This section presents analysis of the results from the experiments. The full experimental data recorded from the experiments are presented in Appendix 1.

4.4.1 Accuracy Measured by Number of Inliers

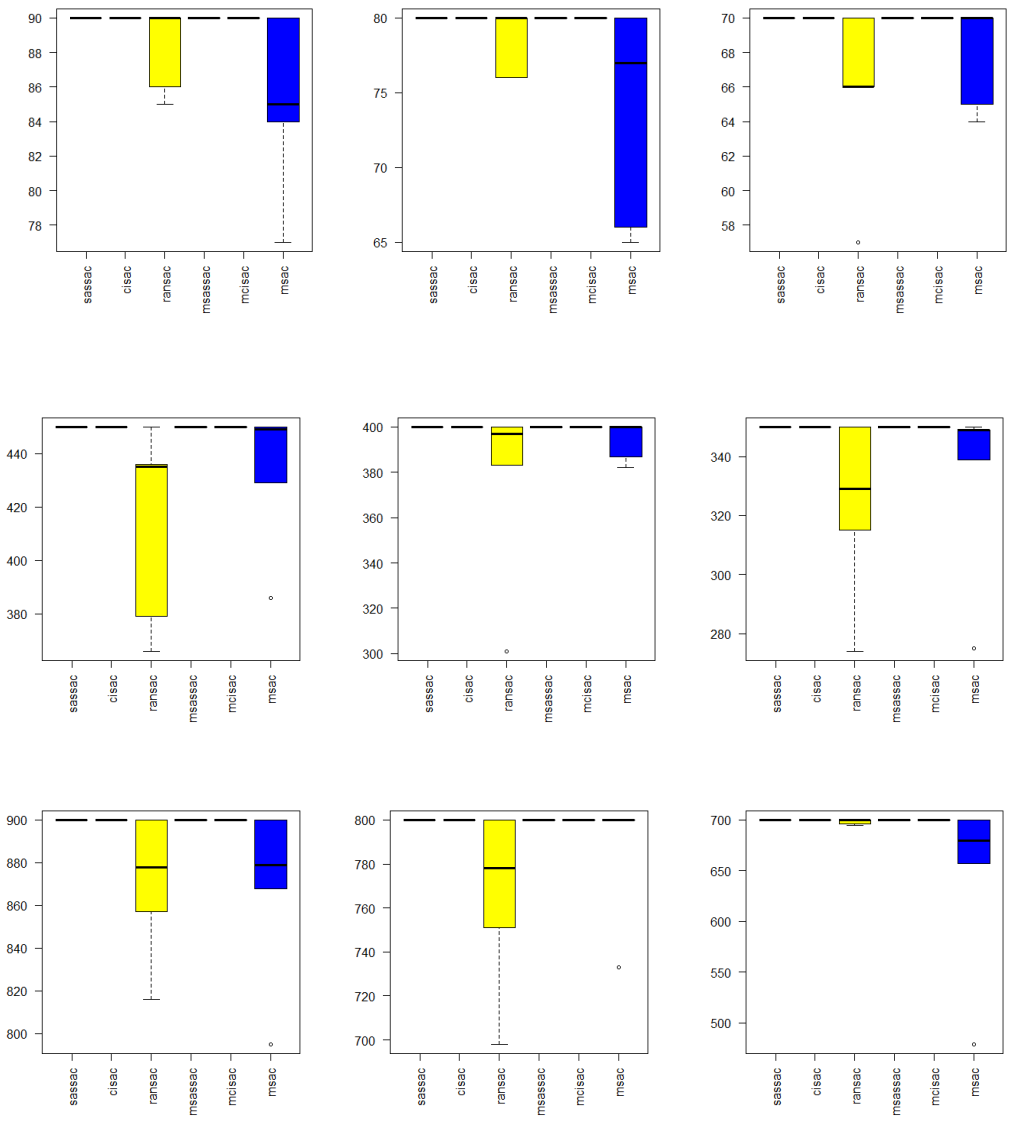


Figure 4.1: Problem set 1 (outlier rates 10%, 20%, 30%), comparing accuracies meausred by number of inliers

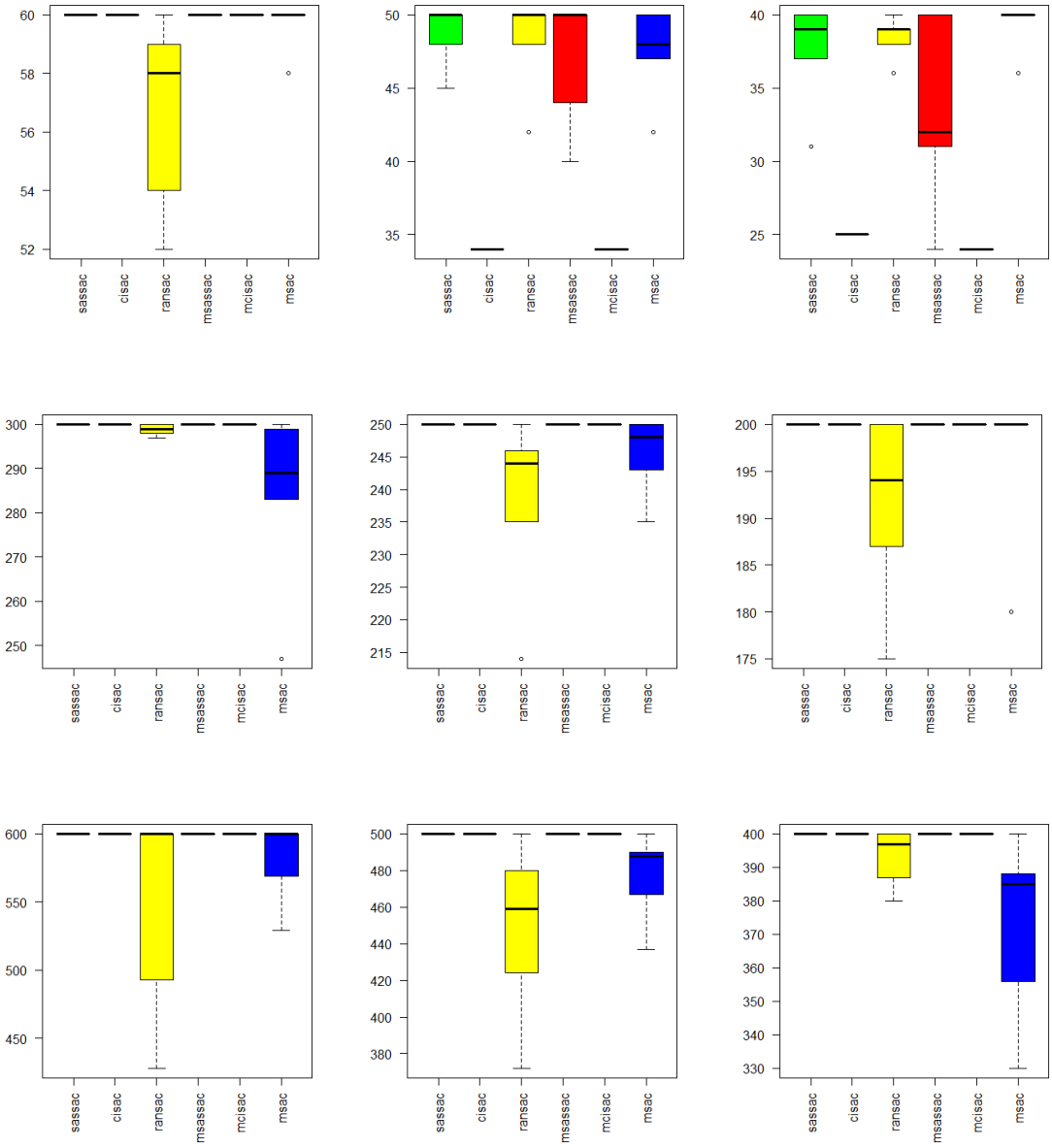


Figure 4.2: Problem set 2 (outlier rates 40%, 50%, 60%), comparing accuracies measured by the number of inliers

4.4.1.1 Statistical Test for Difference in Overall Ranking

Problem set 1: outlier rates 10%, 20%, 30%

```
Friedman chi-squared = 106.99, df = 5, p-value < 2.2e-16
```

Table 4-1: Nemenyi Pairwise comparison for Problem Set 1

	SASSAC	CISAC	RANSAC	MSASSAC	MCISAC
CISAC	1.000000	-	-	-	-
RANSAC	0.000300	0.000300	-	-	-
MSASSAC	1.000000	1.000000	0.000300	-	-
MCISAC	1.000000	1.000000	0.000300	1.000000	-
MSAC	0.000800	0.000800	1.000000	0.000800	0.000800

Table 4-2: Median Ranks for Problem Set 1

SASSAC	CISAC	RANSAC	MSASSAC	MCISAC	MSAC
6	6	2	6	6	2

Table 4-3: Mean Ranks for Problem Set 1

SASSAC	CISAC	RANSAC	MSASSAC	MCISAC	MSAC
6.000000	6.000000	3.355556	6.000000	6.000000	3.488889

Interpretation: The Friedman test reveals significant difference, since the p-value is less than 0.5. That is, at least two of the six algorithms are significantly different in overall ranking on the problem set. From the Nemenyi comparison table, SASSAC, MSASSAC, CISAC and MCISAC, are not significantly different in performance. RANSAC and MSAC are also not significantly different from each other but they differ from the first group. The tables of mean and median ranks show that RANSAC and MSAC form the worse group.

An interesting observation is that on every problem in this set, all the four new algorithms reported exactly the true number of inliers. Moreover, none of the two that have some elements of

randomness in their mechanism, SASSAC and MSASSAC, exhibited any randomness in the solution quality produced on this problem set.

Problem set 2: outlier rates 40%, 50%, 60%

```
Friedman chi-squared = 41.411, df = 5, p-value = 7.75e-08
```

Table 4-4: Nemenyi Pairwise comparison for Problem Set 2

	SASSAC	CISAC	RANSAC	MSASSAC	MCISAC
CISAC	0.650500	-	-	-	-
RANSAC	0.001400	0.168100	-	-	-
MSASSAC	0.999800	0.817400	0.004200	-	-
MCISAC	0.445500	0.999600	0.310400	0.632000	-
MSAC	0.218000	0.594700	0.974000	0.050600	0.787500

Table 4-5: Mean Ranks for Problem Set 2

SASSAC	CISAC	RANSAC	MSASSAC	MCISAC	MSAC
5.800000	5.133333	3.355556	5.688889	5.022222	3.822222

Table 4-6: Median Ranks for Problem Set 2

SASSAC	CISAC	RANSAC	MSASSAC	MCISAC	MSAC
6	6	2	6	6	6

On this problem set, as table 1A in the appendix and the boxplots in Figure 4.2 show, in seven out of all nine cases tested, all four algorithms returned inlier number equal to ground truth, without any variation. The relative ranking of the algorithms remain same as for problem set 1, in the order SASSAC, MSASSAC, CISAC, MCISAC, MSAC and RANSAC.

4.4.2 Accuracy Measured by M-estimate Error

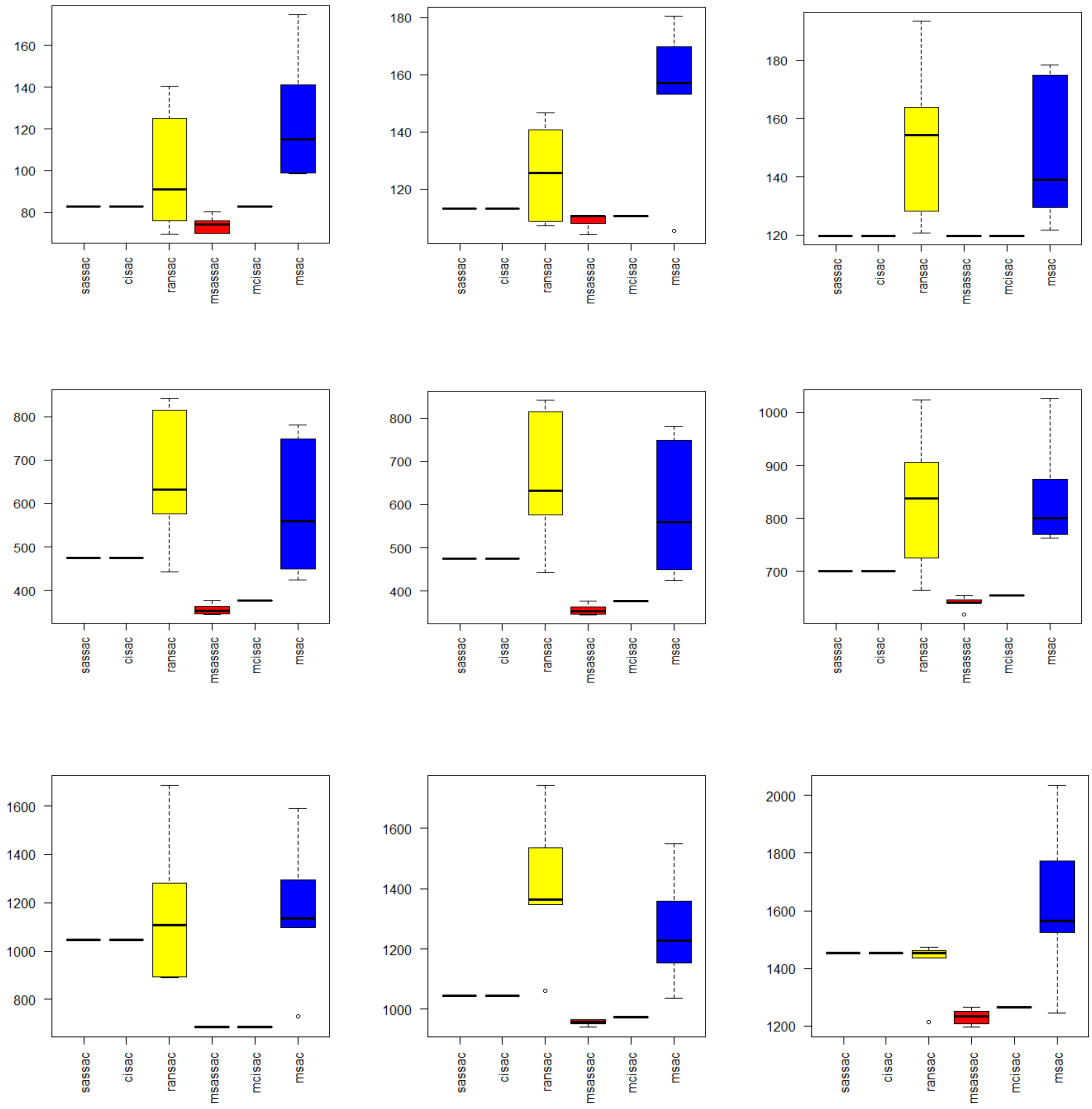


Figure 4.3: Problem set 1 (outlier rates 10%, 20%, 30%) - comparing accuracies measured by M-estimate error

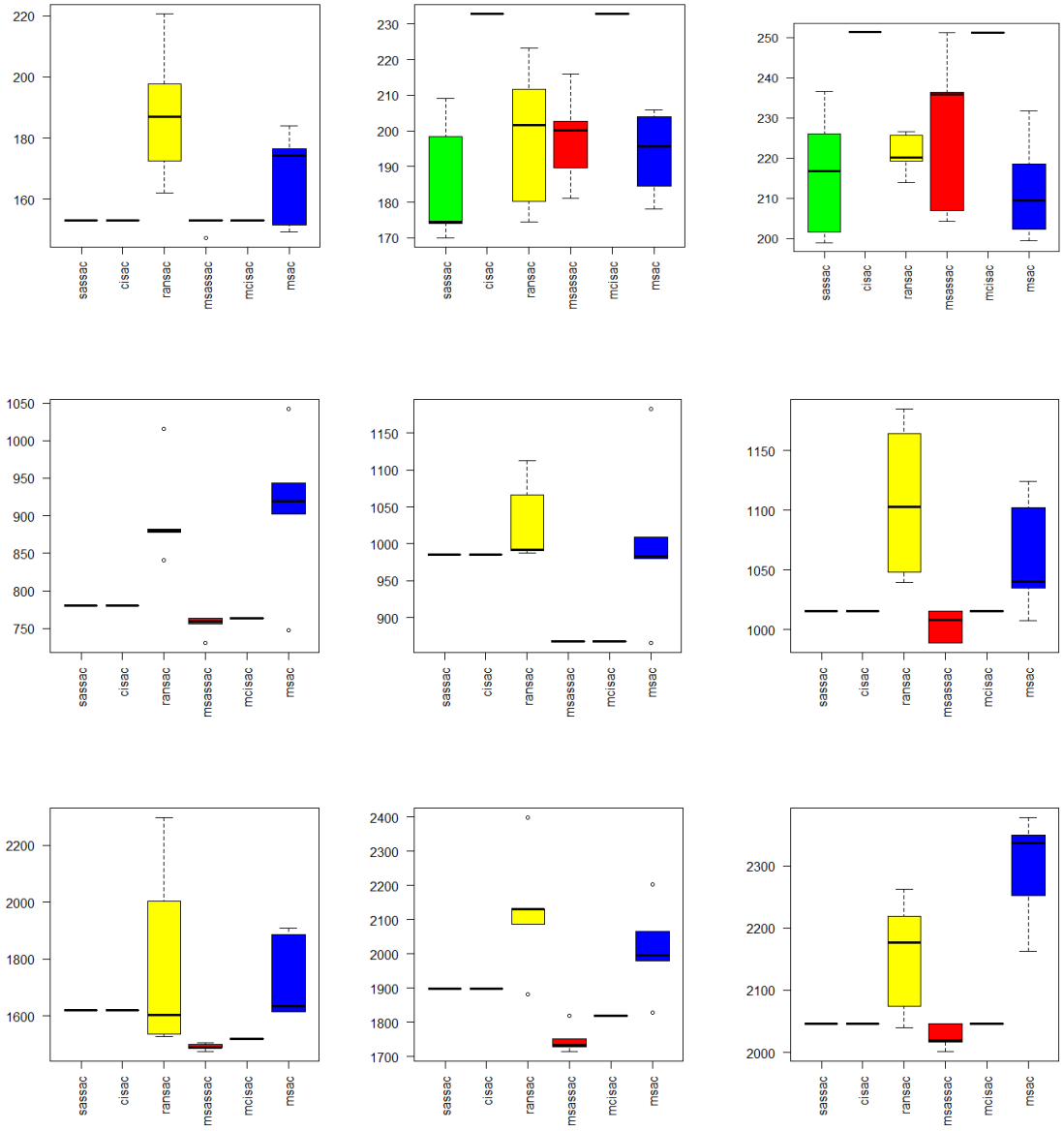


Figure 4.4: Problem set 2 (outlier rates 40%, 50%, 60%) - comparing accuracies measured by M-estimate error

4.4.2.1 Statistical Test for Difference in Overall Ranking

Problem set 1: outlier rates 10%, 20%, 30%

```
Friedman chi-squared = 147.53, df = 5, p-value < 2.2e-16
```

Table 4-7: Nemenyi Pairwise comparison for Problem Set 1

	SASSAC	CISAC	RANSAC	MSASSAC	MCISAC
CISAC	1.000000	-	-	-	-
RANSAC	0.130200	0.130200	-	-	-
MSASSAC	0.000000	0.000000	0.000000	-	-
MCISAC	0.000800	0.000800	0.000000	0.392400	-
MSAC	0.004200	0.004200	0.870400	0.000000	0.000000

Table 4-8: Mean Ranks for Problem Set 1

SASSAC	CISAC	RANSAC	MSASSAC	MCISAC	MSAC
4.422222	4.422222	4.733333	1.733333	2.600000	5.177778

Table 4-9: Median Ranks for Problem Set 1

SASSAC	CISAC	RANSAC	MSASSAC	MCISAC	MSAC
4	4	5	1	2	6

On this problem set, the order from best to worst, that is lowest to highest error, is MSASSAC, MCISAC, SASSAC, CISAC, RANSAC and MSAC. MSASSAC and MCISAC seem to form the best group, SASSAC and CISAC another. Then come RANSAC and MSAC in the rank.

Problem set 2: outlier rates 40%, 50%, 60%

```
Friedman chi-squared = 70.433, df = 5, p-value = 8.329e-14
```

Table 4-10: Nemenyi Pairwise comparison for Problem Set 2

	SASSAC	CISAC	RANSAC	MSASSAC	MCISAC
CISAC	0.375250	-	-	-	-
RANSAC	0.011440	0.738920	-	-	-
MSASSAC	0.003060	0.000000	0.000000	-	-
MCISAC	0.969510	0.068950	0.000560	0.043110	-
MSAC	0.295210	0.999999	0.817390	0.000000	0.046740

Table 4-11: Mean Ranks for Problem Set 2

SASSAC	CISAC	RANSAC	MSASSAC	MCISAC	MSAC
3.955556	4.777778	4.622222	2.222222	3.577778	4.133333

Table 4-12: Median Ranks for Problem Set 2

SASSAC	CISAC	RANSAC	MSASSAC	MCISAC	MSAC
4	5	5	2	4	5

Similar to problem set 1, on this problem set, the order from best to worst, that is lowest to highest error, is MSASSAC, MCISAC, SASSAC, CISAC, RANSAC and MSAC. MSASSAC and MCISAC seem to form the best group, SASSAC and CISAC another. RANSAC and MSAC come behind.

4.5 Experimental Results for Real Life Images

All six algorithms are run 5 times each on the matched features dataset from the UKZN Aerial Photo Pair studied in section 3.4.2.3. Table 4.13 shows the experimental results.

Table 4-13: Experimental Results on the UKZN Aerial Photo Pair

Algorithm	Run	I	ME	iter	f
SASSAC	1	133	151.34387	2	264
CISAC	1	133	151.34387	1	132
RANSAC	1	127	225.6481	5	5
MSASSAC	1	133	128.01881	2	264
MCISAC	1	133	133.52167	1	132
MSAC	1	130	512.93524	3	3
SASSAC	2	133	151.34387	2	264
CISAC	2	133	151.34387	1	132
RANSAC	2	130	187.9812	6	6
MSASSAC	2	132	132.22241	2	264
MCISAC	2	133	133.52167	1	132
MSAC	2	131	374.40723	3	3
SASSAC	3	133	151.34387	2	264
CISAC	3	133	151.34387	1	132
RANSAC	3	132	146.37769	5	5
MSASSAC	3	133	133.52167	2	264
MCISAC	3	133	133.52167	1	132
MSAC	3	132	321.01636	2	2
SASSAC	4	133	151.34387	2	264

CISAC	4	133	151.34387	1	132
RANSAC	4	114	224.25392	9	9
MSASSAC	4	133	126.51538	2	264
MCISAC	4	133	133.52167	1	132
MSAC	4	131	334.45541	3	3
SASSAC	5	133	151.34387	2	264
CISAC	5	133	151.34387	1	132
RANSAC	5	132	146.65369	5	5
MSASSAC	5	133	133.01285	3	396
MCISAC	5	133	133.52167	1	132
MSAC	5	133	402.25391	8	8

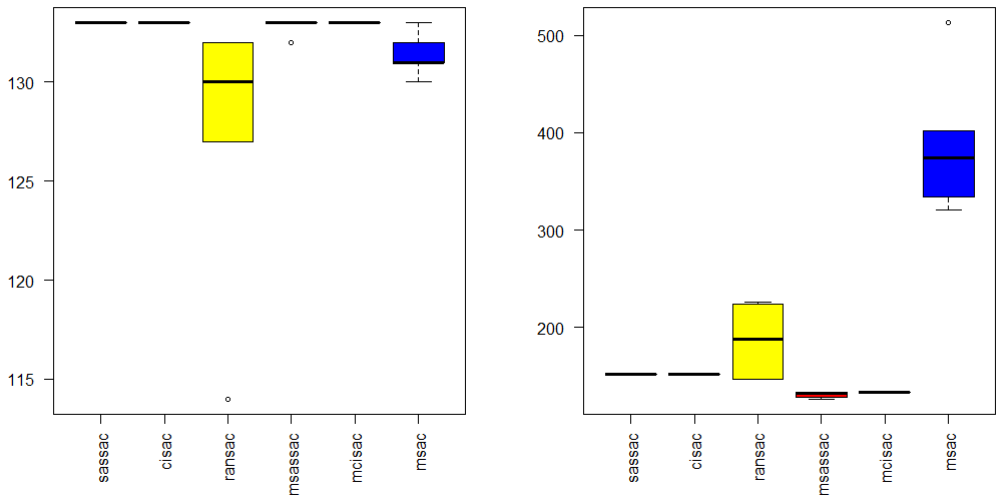


Figure 4.5: Comparing accuracies on the UKZN Pair 1 - Left: measured by number of inliers, right: measured by M-estimate error

On this real-life problem, all four new algorithms exhibited superior accuracy and consistence to either RANSAC or MSAC on both measures of accuracy. All four new algorithms exhibited equal

accuracies when evaluated on the basis of number of inliers. On the M-estimate error ciriterion, MSASSAC is the best.

4.6 Summary of Findings

The outcomes of the series of experiments that make up this comparative study are summarized in this subsection.

Accuracy

In the analysis of experimental results presented in this chapter, accuracy of the algorithms is measured using two different criteria: number of inliers detected which RANSAC seeks to maximize, and the M-estimate error that MSAC seeks to minimize. The six algorithms compared are those that adopt the random-sampling strategy - RANSAC and MSAC – and those proposed in this work – SASSAC, CISAC, MSASSAC, MCISAC.

From the various observations discussed, there is sufficient evidence that on the consensus maximization criterion, SASSAC is generally the most accurate of the six algorithms. MSASSAC competes closely with SASSAC. On each problem set, the difference detected between the performance of MSASSAC and that of SASSAC with respect to this criterion, is not statistically significant. When accuracy is measured using the M-estimate error criterion, MSASSAC is significantly superior to any of the other five. MCISAC is second, SASSAC third, CISAC fourth. On both criteria, RANSAC and MSAC perform significantly worse than any of the four new algorithms. The study shows that these generalizations apply to both low contamination and extreme outlier-rate problems.

Still on accuracy, an interesting observation in the experiments is the fact that all four new algorithms often returned the exact true number of inliers on the various problems. It is worth pointing out that the conditions covered in the experiments range from simple to extreme conditions that are found in practical applications, in terms of contamination level and data size. In fact, it is not rare to find practical applications that involve contamination level much less than the extreme 40% studied or data size much less than 1000. The real-life problem studied validates the statement. The implication is that the algorithms have been quite rigorously tested and compared.

Repeatability

Clearly, the four consecutive-inlier algorithms are generally more consistent than their random-sampling counterparts. CISAC and MCISAC are perfectly repeatable exhibiting absolute zero randomness. SASSAC and MSASSAC put up a similar behavior on many of the problems tested, but they possess some inherent randomness which is manifested on a few of the problems tested. In any case, any of these algorithms exhibit far less randomness or variation than either RANSAC or MSAC, on practically all problems tested.

Simplicity

As described in chapter 1, a major design goal in this work is simplicity. Unlike RANSAC and MSAC, all four new algorithms do not make use of the popular confidence-level-based stopping criterion, described in chapter 2. They also involve very simple procedures and computations. Informed by the consecutive inliers theorem, CISAC and MCISAC simply go straight for the $m-n+1$ possible models that are constructed from consecutive rows in the data, and choose the best. That is all there is to the algorithms. SASSAC and MSASSAC perform this simple procedure iteratively over a few different random permutations of the data. Due to these new strategies, the only parameter required to be supplied to all four algorithms is the distance threshold, which is also required by RANSAC and MSAC, as well as most other variants in literature.

Speed

RANSAC and MSAC are generally faster than the new algorithms. However, as can be seen in experimental results in this chapter as well as section 3.5, the typical runtime of the new algorithms, particularly CISAC and MCICSAC, measured by the number of calls to the objective function, is still low and will typically run on today's computers in fractions of seconds or at most a few seconds for large-sized problems. One interesting advantage of CISAC and MCISAC is that there runtime is deterministic function of data size and minimal sample size. So, for any given problem the runtime can be exactly computed ahead of time. This property holds some value for planning in practical applications. MSASSAC and SASSAC, also exhibit very predictable runtime, although not exactly deterministic. They are also not as fast CISAC and MCISAC.

Another interesting observation is that as either contamination or model dimensionality is increased, the runtime of RANSAC and MSAC deteriorates that of CISAC and MCISAC are unaffected by contamination level. It even gets slightly better as model dimensionality is

increased. Although this improvement is typically insignificant, it is remarkable enough that the runtime is not worsened.

Run time Variation

CISAC and MCISAC exhibit totally deterministic run time. From the experimental results, the runtime of SASSAC and MSASSAC is quite consistent, almost non-random. The runtimes of RANSAC and MSAC are more random than those of the other algorithms.

4.7 Chapter Summary

A comparative study of four of the algorithms proposed in this thesis (CISAC, MICSAC, SASSAC and MSASSAC) and their random sampling counterparts (RANSAC and MSAC) is presented in this chapter. The algorithms are evaluated on simulated datasets covering an extensive range of data conditions, as well as a real-life problem. The study reveals quite clearly that all four new algorithms are more accurate and more consistent than either RANSAC or MSAC. This holds for problems with low outlier rate as well as those with extreme contamination. Two of the new algorithms, CISAC and MCISAC are deterministic while the other two exhibit minimal randomness. Interestingly, the four algorithms do not just merely offer improvements over the accuracies of RANSAC and MSAC, but detect the true number of inliers in most of the cases tested. Although they are generally slower than RANSAC and MSAC, the new algorithms, particularly CISAC and MCISAC are still competitively fast. Moreover, their runtimes can be computed ahead of time. On the consensus maximization criterion, SASSAC is the most accurate of the six, while MSASSAC which is the most accurate on the M-estmate error criterion. MSASSAC also competes closely with SASSAC on the former criterion. Note that the maximization of consensus is the very objective that the authors of RANSAC designed it to achieve, while minimization of M-estimate error is the objective that the authors of MSAC designed it to achieve. But the random sampling strategy with which both algorithms explore the solution space limits them. The outcomes of this study show that the consecutive inliers strategy proposed in this thesis is a better way to optimize both objectives. The implication of having MSASSAC and SASSAC perform better than either RANSAC and MSAC on any of the two criteria, is that the proposed strategy optimizes both objectives more accurately, even when optimization of one of the objectives is not directly sought.

The implication of all these study outcomes is that the objectives of this research work have been realized quite satisfactorily.

CHAPTER FIVE

SOFTWARE CONTRIBUTIONS

5.0 Preamble

This chapter takes the contributions of this work a step further. Presented are two alternative implementations of the homography estimation function in MATLAB's computer vision toolbox, which is a widely used software library. The contributed functions are two alternatives, based on MCISAC and MSASSAC respectively, shown in chapter 4 to offer performance advantages over MSAC, the variant used in the official implementation. MSAC has been used in all releases of the toolbox, including the latest 2015b. The syntax of the proposed functions are consistent with the official MATLAB pattern. This chapter includes a user guide written in the official MATLAB documentation style which includes description of syntax, input arguments and output arguments for both functions. The documentation includes brief practical demonstrations. In order to benefit from code optimization expertise of the professionals at *Mathworks*, the proposed implementations are achieved by editing the source code of the official function, replacing the MSAC components with the proposed algorithms. A few additional output arguments that are useful for performance evaluation are included in the proposed implementations.

5.1 MATLAB's Geometric Transform Estimation Function

Due to the ubiquity of robust estimation problems in the field of computer vision, software in this field include robust estimation functions as an important component. Many applications involve estimation of homography, that is, 2-D geometric transformation between image pairs. MATLAB's computer vision toolbox, which is a very popular software library in this field, provides a function for this purpose, named *estimateGeometricTransform*. The current release (2015b), as well as past releases of this toolbox implement this function based on MSAC. This chapter focuses on improving the state of the art by contributing alternative implementation of MATLAB's *estimateGeometricTransform*. The main idea is to replace the MSAC component of the official implementation with MCISAC and MSASSAC to produce the two proposed alternative functions.

5.2 The Original MSAC Algorithm

As discussed in chapter 2, MSAC was proposed by Torr and Zisserman in 1998. It is one of the earliest variants of RANSAC. It replaces the optimization objective. MSAC's error function is stated thus:

$$\rho_1(e^2) = \begin{cases} |e|, & |e| < T \\ T, & |e| \geq T \end{cases}$$

Where T is the error bound used in distinguishing inlier from outliers as in RANSAC, that is, the maximum distance between an inlier point and the projection.

5.2.1 Drawbacks of MSAC

MSAC's optimization objective could be a better measure of model accuracy in many cases, since it considers the actual errors within the distance threshold bound, rather than the binary polarization adopted RANSAC. Apart from the advantages that result from replacing RANSAC's optimization objective with MSAC's, MSAC shares other drawbacks of the former in chapter 2. The reason is not far-fetched: MSAC adopts the uniform random-sampling strategy of RANSAC, which is noted to be the root of many of RANSAC's drawbacks.

5.3 MATLAB's MSAC: Stopping Criterion for Balance of Efficiency and Accuracy

MATLAB implements a refined version of MSAC. In the official implementation of the estimateGeometricTransform function, the stopping criterion discussed in chapter 3, is incorporated into MSAC. The purpose is to achieve improved efficiency. The number *k*, of samples that have to be drawn for a given probability P_I of drawing an all-inlier sample is given by:

$$k = \frac{\log(\eta)}{\log(1 - P_I)}$$

$$P_I = \frac{\binom{I}{n}}{\binom{m}{n}}$$

$$= \prod_{j=0}^{n-1} \frac{I - j}{m - j}$$

$$\approx \left(\frac{I}{m}\right) n$$

where *I* is the number of inliers and *m* is the number of points in the full data, and *n* is minimal sample size.

5.4 Pros and cons of MCISAC and MSASSAC

The comparative study of chapter 4 clearly established the superiority of either MCISAC or MSASSAC over MSAC, on quite a number of performance measures.

The main advantage of MCISAC over MSASSAC is that it is totally deterministic while MSASSAC has some elements of randomness to it. However, MCISAC shares the 'small' risk of breakdown associated with CISAC, for very high model dimensionality and extreme contamination, as discussed in chapters 3 and 4. Nevertheless, MCISAC is still a good choice for homography estimation since this risk as been shown to be quite small and breakdown is unlikely to occur in many practical situation. Although MSASSAC introduces a bit a randomness, it offers improvement over MCISAC's exploration, resulting in better accuracy. As shown in the experiments of chapter 4, this randomness is generally small, and the variation in solution quality could be zero in many practical cases. In any case, the inherent randomness as well as increased runtime, are the costs incurred by MSASSAC, to achieve better accuracy and better robustness to poor choice of distance threshold, than MCISAC.

In summary, MCISAC and MSASSAC compete closely and are both good choices for the MATLAB function in question. The clear advantages of MSASSAC over MCISAC is the elimination of the risk of breakdown (implying better reliability), improved accuracy and robustness to poor choice of distance threshold. However, if the priority is either repeatability or speed, then MCISAC assures the user better.

5.5 User Guide for Proposed Functions

As earlier described, the function *estimateGeometricTransform* estimates 2-D geometric transformations between image pairs. In this section, the syntax is described and examples of use are presented. The pattern of MATLAB's documentation style is followed, so users who are familiar with the original MATLAB function, should find the presentation intuitive. Both implementations that are proposed share exactly the same syntax.

5.5.1 Syntax and Description

This subsection shows the various syntaxes that are legal in calling the functions. The variety results from the fact that some input and output arguments are optional.

Syntax 1

```
tform = estimateGeometricTransformMCISAC(X,Y,transformType);

tform = estimateGeometricTransformMSASSAC(X,Y,transformType);
```

where X and Y are the matrices of matched points in the first and second image respectively.

In each case, the command returns a 2-D geometric transform object tform by detecting inlier pairs in the matched set and computing the appropriate transformation.

Optional output are specified in Syntax 2 to 4.

Syntax 2

```
 [tform,inlierPoints1,inlierPoints2]
= estimateGeometricTransformMCISAC(X,Y,transformType);


[tform,inlierPoints1,inlierPoints2]
= estimateGeometricTransformMSASSAC(X,Y,transformType);
```

In addition to the 2-D transform object, this command returns the corresponding inlier pairs: `inlierPoints1` and `inlierPoints2.`

Syntax 1 and 2 above are identical to that of the original MATLAB implementation. Additional outputs which may be useful for performance evaluation are included in the implementation presented in this work: the M-estimate error to evaluate accuracy and the number of iterations to track run time. It is unnecessary to return the number of inliers since it is simply the number of rows of the matrix `inlierPoints1` or `inlierPoints2.` The added outputs are reflected in syntax 3.

Syntax 3

```
[tform,inlierPoints1,inlierPoints2,mError,iter]
= estimateGeometricTransformMCISAC(X,Y,transformType);

[tform,inlierPoints1,inlierPoints2,mError,iter]
= estimateGeometricTransformMSASSAC(X,Y,transformType);
```

`mError` is the magnitude of the M-estimate error corresponding to the returned transform. This is the same error that MSAC seeks to minimize. `iter` is the number of iterations.

Syntax 4

```
[__,status] = estimateGeometricTransformMCISAC(X,Y,transformType);
[__,status] = estimateGeometricTransformMSASSAC(X,Y,transformType);
```

The output `status` comes from the official MATLAB implementation. It is returns a status code 0, 1 or 2. The purpose is to report solution conditions in case there are conditions that cannot produce results. If the status is not requested such as in syntax 4, the function will produce an error under such problematic conditions. This component is left as provided in the official implementation.

Syntax 5

```
[__] = estimateGeometricTransformMCISAC(X,Y,transformType, 'MaxDistance',
Value);
[__] = estimateGeometricTransformMSASSAC(X,Y,transformType, 'MaxDistance',
Value);
```

Syntax 5 points out the availability of optional input arguments that can be passed to the function. Again, the pattern of the official MATLAB implementation is followed. These input arguments are discussed below.

Input Arguments

In the official implementation, there are three optional arguments. Two are specific to MSAC but are not required by the new algorithms. The latter do not require a specification of maximum

number of trials and confidence level, both of which are related to MSAC's stopping criterion. The only argument left, which applies to the new algorithms is the error bound (distance threshold) used for distinguishing inliers from outliers. It is named 'MaxDistance'. If the user does not specifiy a value for this argument, a default of 1.5 is used just like in the official implementation.

Table 5-1: Summary of Input Arguments

Argument	Definition	Value
matchedPoints1	Matched points from image 1	cornerPoints object | SURFPoints object | MSERRegions object | M-by-2 matrix of [x,y] coordinates
matchedPoints2	Matched points from image 2	cornerPoints object | SURFPoints object | MSERRegions object | M-by-2 matrix of [x,y] coordinates
transformType	Tranform/model type	'similarity' | 'affine' | 'projective'

Table 5-2: Name-Value Pair Input Argument

Name	Definition	Value
'MaxDistance'	Error bound for distinguishing inliers from outliers, i.e. maximum distance from point to projection	positive numeric scalar (default of 1.5 is used if not specified)

Table 5-3: Summary of Output Arguments

Argument	Definition	Value
tform	Geometric tranformation	affine2d object | projective2d object
status	status code	0 (No error) | 1 (matchedPoints and matchedPoints2 do not contain enough points) | 2 (Not enough inliers have been found)
inlierPoints1	inliers in image 1	cornerPoints object | SURFPoints object | MSERRegions object | M-by-2 matrix of [x,y] coordinates
inlierPoints2	inliers in image 2	cornerPoints object | SURFPoints object | MSERRegions object | M-by-2 matrix of [x,y] coordinates

5.5.2 Demo Examples

In this subsection, examples are presented to demonstrate the use of the contributed functions. The examples, one for each function involve recovering an image that has undergone distortion, by using the proposed functions to estimate the transformation. Matched feature pairs are automatically detected using the SURF technique after which the proposed functions are used to estimate the geometric transform in other to recover the distortion parameters, as well as the original image. The problem is actually taken from MATLAB's official documentation, while the function used in the geometric transform estimation stage is replaced with the proposed ones. The goal here is demonstration rather than performance evaluation.

5.5.2.1 Automatic Recovery of Image Rotation and Scale with Proposed estimategeometrictransformMCISAC Function

This example is identical to a similarly titled example in the official MATLAB documentation. The only difference is that the official *estimateGeometricTransform* function is replaced by the proposed implementation in this work named *estimateGeometricTransformMCISAC*. As the name implies, it is based on MCISAC.

The example uses detectSURFFeatures and vision.GeometricTransformEstimator System object to recover rotation angle and scale factor of a distorted image. Then the distorted image is

transformed to recover the original image. In the example presented, the true rotation and scale factor are 30 degrees and 0.7 respectively.

Step 1: Read Image

Read an image into the MATLAB workspace.

```
original = imread('cameraman.tif');
imshow(original);
text(size(original,2),size(original,1)+15, ...
    'Image courtesy of Massachusetts Institute of Technology', ...
    'FontSize',7,'HorizontalAlignment','right');
```

Figure 5.1: Image Read and Displayed

Step 2: Resize and Rotate the Image

```
scale = 0.7;
J = imresize(original, scale); % Try varying the scale factor.

theta = 30;
distorted = imrotate(J,theta); % Try varying the angle, theta.
figure, imshow(distorted)
```

Figure 5.2: Image Resized and Rotated

Step 3: Find Matching Features between Images

Detect common features in both the original and transformed images, using the SURF technique

```
ptsOriginal  = detectSURFFeatures(original);
ptsDistorted = detectSURFFeatures(distorted);
```

Extract feature descriptors from detected features

```
[featuresOriginal,   validPtsOriginal]  = extractFeatures(original,
ptsOriginal);
[featuresDistorted, validPtsDistorted]  = extractFeatures(distorted,
ptsDistorted);
```

Use the descriptors to match features

```
indexPairs = matchFeatures(featuresOriginal, featuresDistorted);
```

Retrieve coordinates of corresponding points for each image.

```
matchedOriginal  = validPtsOriginal(indexPairs(:,1));
matchedDistorted = validPtsDistorted(indexPairs(:,2));
```

Show point matches. Notice the presence of outliers.

```
figure;
showMatchedFeatures(original,distorted,matchedOriginal,matchedDistorted);
title('Putatively matched points (including outliers)');
```

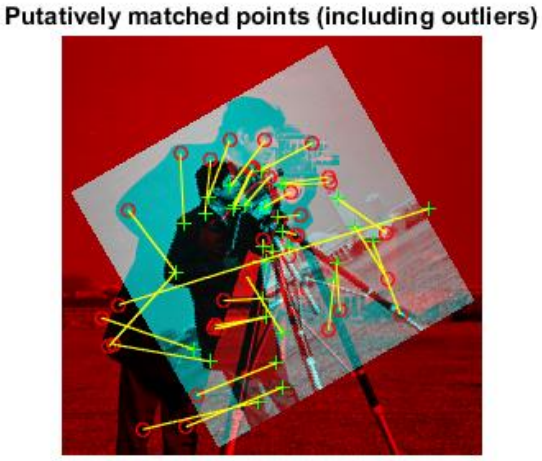


Figure 5.3: Matched Features between the Image Pair

Step 4: Estimate Transformation and solve for angle

Estimate transformation corresponding to the matching point pairs using the proposed function.

```
[mcisactform, mcisacinlierDistorted, mcisacinlierOriginal] =
estimateGeometricTransformMCISAC(...
    matchedDistorted, matchedOriginal, 'similarity');
```

Step 5: Solve for Scale and Angle

Let $sc = scale \times cos\theta$

Let $sc = scale \times sin\theta$

Then, Tinv = $\begin{pmatrix} sc & -ss & 0 \\ ss & sc & 0 \\ tx & ty & 1 \end{pmatrix}$

where *tx* and *ty* are *x* and *y* translations, respectively.

Compute the inverse transformation matrix.

```
mcisacTinv = mcisactform.invert.T;

mcisacss = mcisacTinv(2,1);
mcisacsc = mcisacTinv(1,1);
mcisacscale_recovered = sqrt(mcisacss*mcisacss + mcisacsc*mcisacsc)
mcisactheta_recovered = atan2(mcisacss,mcisacsc)*180/pi
```

```
mcisacscale_recovered =
    0.6979

mcisactheta_recovered =
   30.3193
```

Notice that the estimated values compare quite accurately with the true values of 0.7 and 30 degrees respectively.

Repeat Steps 4 and 5 to evaluate repeatability

Second Trial

```
[mcisactform, mcisacinlierDistorted, mcisacinlierOriginal] =
estimateGeometricTransformMCISAC(...
    matchedDistorted, matchedOriginal, 'similarity');

mcisacTinv = mcisactform.invert.T;

mcisacss = mcisacTinv(2,1);
mcisacsc = mcisacTinv(1,1);
mcisacscale_recovered = sqrt(mcisacss*mcisacss + mcisacsc*mcisacsc)
mcisactheta_recovered = atan2(mcisacss,mcisacsc)*180/pi
```

```
mcisacscale_recovered =
    0.6979

mcisactheta_recovered =
   30.3193
```

Third Trial

```
[tform, inlierDistorted, inlierOriginal] = estimateGeometricTransform(...
    matchedDistorted, matchedOriginal, 'similarity');
[mcisactform, mcisacinlierDistorted, mcisacinlierOriginal] =
estimateGeometricTransformMCISAC(...
    matchedDistorted, matchedOriginal, 'similarity');

mcisacTinv = mcisactform.invert.T;

mcisacss = mcisacTinv(2,1);
mcisacsc = mcisacTinv(1,1);
mcisacscale_recovered = sqrt(mcisacss*mcisacss + mcisacsc*mcisacsc)
mcisactheta_recovered = atan2(mcisacss,mcisacsc)*180/pi
```

```
mcisacscale_recovered =
    0.6979

mcisactheta_recovered =
   30.3193
```

5.5.2.2 Demonstration of estimateGeometricTransformMSASSAC Function

This example is identical to the one presented for *estimateGeometricTransformMCISAC*. Here second proposed function is used: *estimateGeometricTransformMSASSAC*.

Step 1: Read Image

Bring an image into the workspace.

```
original = imread('cameraman.tif');
imshow(original);
text(size(original,2),size(original,1)+15, ...
    'Image courtesy of Massachusetts Institute of Technology', ...
    'FontSize',7,'HorizontalAlignment','right');
```

Figure 5.4: Image Read and Displayed

Step 2: Resize and Rotate the Image

```
scale = 0.7;
J = imresize(original, scale); % Try varying the scale factor.
theta = 30;
distorted = imrotate(J,theta); % Try varying the angle, theta.
figure, imshow(distorted)
```

Figure 5.5: Resized and Rotated Image

Step 3: Find Matching Features between Images

Detect common features in both images.

```
ptsOriginal  = detectSURFFeatures(original);
ptsDistorted = detectSURFFeatures(distorted);
```

Extract feature descriptors using the SURF method.

```
[featuresOriginal,   validPtsOriginal]  = extractFeatures(original,
ptsOriginal);
[featuresDistorted, validPtsDistorted]  = extractFeatures(distorted,
ptsDistorted);
```

Match features by using their descriptors.

```
indexPairs = matchFeatures(featuresOriginal, featuresDistorted);
```

Retrieve locations of corresponding points for each image.

```
matchedOriginal  = validPtsOriginal(indexPairs(:,1));
matchedDistorted = validPtsDistorted(indexPairs(:,2));
```

Show putatitive matches

```
figure;
showMatchedFeatures(original,distorted,matchedOriginal,matchedDistorted);
title('Putatively matched points (including outliers)');
```

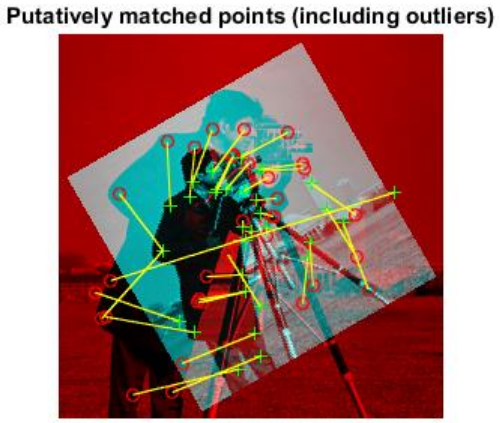


Figure 5.6: Matched Features between Images

Step 4: Estimate Transformation and solve for angle

```
[msassactform, msassacinlierDistorted, msassacinlierOriginal] =
estimateGeometricTransformMSASSAC(...
    matchedDistorted, matchedOriginal, 'similarity');
```

Step 5: Solve for Scale and Angle

```
[msassactform, msassacinlierDistorted, msassacinlierOriginal] =
estimateGeometricTransformMSASSAC(...
    matchedDistorted, matchedOriginal, 'similarity');
msassacTinv = msassactform.invert.T;
msassacss = msassacTinv(2,1);
msassacsc = msassacTinv(1,1);
msassacscale_recovered = sqrt(msassacss*msassacss + msassacsc*msassacsc)
msassactheta_recovered = atan2(msassacss,msassacsc)*180/pi
```

```
msassacscale_recovered =
    0.6991

msassactheta_recovered =
   29.9552
```

Second trial

```
[msassactform, msassacinlierDistorted, msassacinlierOriginal] =
estimateGeometricTransformMSASSAC(...
    matchedDistorted, matchedOriginal, 'similarity');

msassacTinv = msassactform.invert.T;
msassacss = msassacTinv(2,1);
msassacsc = msassacTinv(1,1);
msassacscale_recovered = sqrt(msassacss*msassacss + msassacsc*msassacsc)
msassactheta_recovered = atan2(msassacss,msassacsc)*180/pi
```

```
msassacscale_recovered =
    0.6992

msassactheta_recovered =
   29.9701
```

Third Trial

```
[msassactform, msassacinlierDistorted, msassacinlierOriginal] =
estimateGeometricTransformMSASSAC(...
    matchedDistorted, matchedOriginal, 'similarity');

msassacTinv = msassactform.invert.T;
msassacss = msassacTinv(2,1);
msassacsc = msassacTinv(1,1);
msassacscale_recovered = sqrt(msassacss*msassacss + msassacsc*msassacsc)
msassactheta_recovered = atan2(msassacss,msassacsc)*180/pi
```

```
msassacscale_recovered =
    0.6991

msassactheta_recovered =
  29.9552
```

5.6 Chapter Summary

Two functions are contributed in this chapter. Both are proposed alternatives to the official homography estimation function, named *estimateGeometricTransform*. The proposed implementations are based on two of the algorithms proposed in this dissertation, MCISAC and MSASSAC, which have both been shown in chapter 5 to offer several advantages over MSAC, the variant that has been used in MATLAB's official implementation till date. The advantages include improved accuracy, repeatability and more tractable complexity that makes them perform well even under very high outlier-contamination, robustness to poor choice of distance threshold. The algorithms are also simpler in the sense that they do not require two of the input parameters of MATLAB's MSAC: the maximum number of iterations and confidence level. These parameters are used to compute the appropriate termination criterion, which is eliminated in the proposed algorithms. Unlike many RANSAC variants that offer some improvements over MSAC and RANSAC, the new algorithms completely retain the generic nature of these older algorithms. While both of them produce results that are quite consistent, one of them – MCISAC – is completely non-random. Its run time is also totally deterministic. To the best of the author's knowledge, these last two properties have never been claimed by any algorithm in RANSAC literature. They have been made possible by insight from the consecutive inliers theorem.

Having already established these properties empirically in chapter 5, the contribution in this particular chapter is motivated by the need to make software implementations of these algorithms available, to indeed drive forward the state of the art. It is hoped that future releases of MATLAB's computer vision toolbox, which is widely used in the field, will benefit from these innovations. For the purpose user-friendliness, the syntax of the proposed implementations and the documentation provided are consistent with the official MATLAB pattern.

CHAPTER SIX

SUMMARY, CONCLUSION AND RECOMMENDATIONS

6.0 Chapter Introduction

This is the concluding chapter of this dissertation. The research work is summarized along with evaluation of achievement of its research objectives. Then directions for future work are highlighted.

6.1 Research Summary

This study set out to drive forward the state of the art in robust estimation, particularly in computer vision, by contributing a comprehensive survey of RANSAC variants and by developing new algorithms that address gaps in literature. One motivation is the fact that, although much progress has been made in development of RANSAC variants that offer improvements with respect to various performance measures, popular software libraries have stuck with a few relatively older variants. The implication is that the high activity and progress in RANSAC research especially in the last two decades is not reflected commensurately in software libraries that are widely used in this field. It is observed that the few variants that have remained the favourite choice in popular software libraries, possess some attractive properties that many of the newer ones compromise, to achieve the improvements they offer. Two of these properties are simplicity and generality.

This research identified the need to develop a search strategy that could replace RANSAC's uniform random sampling, without dependence on problem-specific prior information. The value of such a strategy is that improvements can be achieved purely by a change in search strategy: no extra operations. Algorithms designed in this way preserve the two earlier mentioned properties and should be better candidates for software implementation.

An additional effort of this work in driving forward the state of the art, is a comprehensive survey of existing variants. The survey presented is the most comprehensive and up-to-date document published on this subject, to the best of the author's knowledge. The motivation for such an effort is that providing an organized discussion of existing works in such a vast literature may be equally as valuable as embarking on new ones. Also of value is the included analysis of literature aimed towards holistic understanding and identification of high-priority problems in RANSAC research.

Specifically, the study sought to provide answers to a few questions highlighted as follows.

i. Taking a holistic look at RANSAC literature, what are the dominant themes; what is/are the most fundamental question(s) in RANSAC research and what is/are the most pressing concern(s) of research efforts?
ii. What directions should future research efforts pursue to achieve algorithms with higher odds of being considered as good candidates for implementation in popular software?
iii. How can the drawbacks of RANSAC be overcome purely by a change in search strategy, without introducing new steps nor dependence on problem-specific priors?

The first two questions were answered through the instrumentality of the comprehensive survey conducted and presented in chapter 2, while the last constituted the main technical problem addressed in the rest of the dissertation.

6.2 Summary of Findings

The study collected and reviewed a total of 55 variants, by far the most comprehensive and up-to-date collection in literature. It is found that in the last one-and-half decade, at least one variant is published nearly every year. The activity level is still high and appears to be increasing, the last two years alone witnessing the introduction of more than ten variants. The most dominant themes in literature are found to be the pursuit of improved accuracy, and speed. Typically it is not desired to have one while losing much of the other, implying that a major concern is efficiency. About 76% of works found in the survey are related to improvement of efficiency. While some works seek improvements through diverse techniques that often lead to introduction of new operations or steps into the original algorithm, 53% of the entire collection pursue improvement by developing alternatives to RANSAC's search strategy. Still gaps exist. Many of the strategies developed leverage on problem-specific prior information. While several directions were identified in the survey as good directions for future works, the rest of the dissertation was dedicated to address this apparently most fundamental problem: developing search strategies that neither introduce new operations nor require problem-specific priors.

Central to this dissertation is a theorem proposed in chapter three, referred to as the 'consecutive inliers theorem'. The theorem reveals a straightforward strategy for finding good solutions in any robust estimation problem. The direct outcome is a search strategy that possesses the properties that this work sought to achieve: simplicity and non-dependence on problem-specific prior

information. This search strategy is the basis for the first novel algorithm presented in this work, named CISAC. CISAC does not only preserve the simplicity and generality of RANSAC, but is also shown to be generally more accurate. Perhaps, a more interesting outcome is that CISAC is completely non-random in both solutions produced as well as runtime. To the best of the author's knowledge, this property has never been claimed in RANSAC literature, and is not common among non-exhaustive search algorithms in general.

Harnessing the new possibilities created by CISAC, a study is presented on the problem of estimating optimal distance threshold, an important parameter required as input to RANSAC-like algorithms. Empirical observations shows that the number of inliers detected by such a perfectly repeatable algorithm, is a monotonically increasing function of distance threshold that reaches steady state at the true value and remains so for a wide range of threshold values. This shows how the properties of CISAC simplifies the problem. This insight is put to use in developing a mechanism for fully automatic robust estimation, leading to the second proposed algorithm, named AutoCISAC. Efforts to improve on CISAC's reliability and accuracy, especially under extreme outlier-contamination, resulted in another algorithm named SASSAC, which turned out to be even more accurate.

CISAC, AutoCISAC and SASSAC, adopt RANSAC's consensus maximization objective, differing from the latter only by the adoption of the search strategies proposed in this work. Taking a cue from the popular MSAC which replaces RANSAC's optimization objective with M-estimate error minimization, the 'consecutive inliers' strategy is explored for this objective, resulting in two more algorithms: MCISAC and MSASSAC.

Simulated problems that cover a wide range of conditions and real-life problems are used for testing. Comparative study on homography estimation show that all four proposed algorithms are generally more accurate than RANSAC and MSAC. Interestingly all the proposed algorithms are not just more accurate than their random-sampling counterparts, they detect the true number of inliers in many cases tested.

Two of them, MCISAC and MSASSAC are then implemented as contributed alternatives to the homography estimation function provided in MATLAB's computer vision toolbox. The official implementation in various releases including the current version (2015b) are based on MSAC. Documentation is provided to describe the syntaxes, including demonstrations. To ahieve user-

friendliness, both the syntax of the functions and the documentation format are consistent with the official MATLAB style.

6.3 Conclusion

It was hypothesized at the onset of this research, that many, if not all, of the drawbacks of RANSAC can be overcome by only a replacement of its random sampling strategy. The outcomes summarized in section 6.2 validate this argument and also validate the effectiveness of the strategies developed in this work. The implication is that contrary to the approach of many existing works discussed in chapter 2, RANSAC's simplicity is preserved without dependence on problem-specific prior information. The proposed algorithms are therefore likely to be considered by software makers as good candidates for future releases. Indeed, a proactive step has been taken to implement alternatives to a very useful function in MATLAB's computer vision toolbox, which is based on MSAC, over which the new algorithms are shown in this dissertation to exhibit performance improvements.

6.4 Recommendations for Future Work

There is much room for further work. Hopefully, this work is only the beginning of the exploration of the potentials of the newly proposed Consecutive Inliers Theorem as well as the insights provided by the study reported in section 3.5, on the threshold estimation problem. As other research efforts explore, refine and build upon these insights, much benefits will hopefully be realized for theory and practice of robust estimation.

The automatic threshold estimation mechanism that resulted in AutoCISAC can also be applied to the other proposed algorithms. One implication is that a fully automatic function will be achieved as a candidate for a future contribution to the MATLAB computer vision toolbox. This facility is currently unavailable, even in the 2015b release of the toolbox. One possible reason is the high computational cost and runtime of existing automatic techniques, which may be significantly reduced by the newly proposed mechanism in section 3.5. It is also noted that MATLAB's Computer Vision toolbox has another robust estimation function, specifically designed for estimation of the fundamental matrix. This is a higher dimensional model for which the advantages of the algorithms proposed in this work are expected to be even more pronounced, especially SASSAC and MSASSAC. Implementation of the corresponding improved function is a candidate

future project. It is also noted that the OpenCV library, another very popular software library also includes two different functions that implement RANSAC for homography and fundamental matrix estimation respectively. This library should also benefit from the proposed algorithms in the near future.

Finally, since all five proposed algorithms are completely generic, they can be applied to any model-fitting problem, from other scientific fields. Notice that the extensive set of simulated datasets studied in this work are generic. The author therefore recommends exploration of the new algorithms for research and application in diverse scientific fields with model-fitting needs, keeping in mind the fact that they claim outlier-robustness that is competitive with many of the most robust techniques found in modern statistical software. While such robustness is inherited from the general sample consensus paradigm rather than the specific contributions of this work, the other specific advantages offered should make the newly proposed algorithms even more attractive.

REFERENCES

[1] P. Meer, D. Mintz, A. Rosenfeld, and D. Y. Kim, "Robust regression methods for computer vision: A review," *International journal of computer vision*, vol. 6, pp. 59-70, 1991.

[2] P. J. Huber, *Robust statistics*: Springer, 2011.

[3] F. R. Hampel, "Robust estimation: A condensed partial survey," *Probability Theory and Related Fields*, vol. 27, pp. 87-104, 1973.

[4] M. Zuliani, "RANSAC for Dummies," *Vision Research Lab, University of California, Santa Barbara*, 2009.

[5] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American statistical association*, vol. 79, pp. 871-880, 1984.

[6] J. Ramsay, "A comparative study of several robust estimates of slope, intercept, and scale in linear regression," *Journal of the American Statistical Association*, vol. 72, pp. 608-615, 1977.

[7] L. G. Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, pp. 325-376, 1992.

[8] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and vision computing*, vol. 21, pp. 977-1000, 2003.

[9] D. L. Hill, P. G. Batchelor, M. Holden, and D. J. Hawkes, "Medical image registration," *Physics in medicine and biology*, vol. 46, p. R1, 2001.

[10] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381-395, 1981.

[11] B. Micusik and T. Pajdla, "Using RANSAC for omnidirectional camera model fitting," 2003.

[12] D. Robinson and P. Milanfar, "Fundamental performance limits in image registration," *Image Processing, IEEE Transactions on*, vol. 13, pp. 1185-1199, 2004.

[13] D. Nistér, "Preemptive RANSAC for live structure and motion estimation," *Machine Vision and Applications*, vol. 16, pp. 321-329, 2005.

[14] P. H. S. Torr, "Motion segmentation and outlier detection," University of Oxford England, 1995.

[15] F. Mufti, R. Mahony, and J. Heinzmann, "Robust estimation of planar surfaces using spatio-temporal RANSAC for applications in autonomous vehicle navigation," *Robotics and Autonomous Systems*, vol. 60, pp. 16-28, 2012.

[16] S. Choi, T. Kim, and W. Yu, "Performance evaluation of RANSAC family," *Journal of Computer Vision,* vol. 24, pp. 271-300, 1997.

[17] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J. Frahm, "USAC: a universal framework for random sample consensus," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 35, pp. 2022-2038, 2013.

[18] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus," in *Computer Vision–ECCV 2008*, ed: Springer, pp. 500-513, 2008.

[19] P. Čížek and J. Á. Víšek, *Least trimmed squares*: Springer, 2000.

[20] C. V. Stewart, "MINPRAN: A new robust estimator for computer vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 17, pp. 925-938, 1995.

[21] P. Torr and A. Zisserman, "Robust computation and parametrization of multiple view relations," in *Computer Vision, 1998. Sixth International Conference on*, pp. 727-732, 1998.

[22] P. H. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding,* vol. 78, pp. 138-156, 2000.

[23] P. H. S. Torr, "Bayesian model estimation and selection for epipolar geometry and generic manifold fitting," *International Journal of Computer Vision,* vol. 50, pp. 35-61, 2002.

[24] J. V. Miller and C. V. Stewart, "MUSE: Robust surface fitting using unbiased scale estimates," in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pp. 300-306, 1996.

[25] C. Haifeng and P. Meer, "Robust regression with projection based M-estimators," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 878-885 vol.2, 2003.

[26] R. Toldo and A. Fusiello, "Robust multiple structures estimation with j-linkage," in *Computer Vision–ECCV 2008*, ed: Springer, pp. 537-547, 2008.

[27] T.-J. Chin, H. Wang, and D. Suter, "Robust fitting of multiple structures: The statistical learning approach," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 413-420, 2009.

[28] T.-J. Chin, J. Yu, and D. Suter, "Accelerated hypothesis generation for multi-structure robust fitting," in *Computer Vision–ECCV 2010*, ed: Springer, pp. 533-546, 2010.

[29] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *Robotics and Automation, IEEE Transactions on,* vol. 17, pp. 890-897, 2001.

[30] H. Li, "Consensus set maximization with guaranteed global optimality for robust geometry estimation," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1074-1080, 2009.

[31] C. Olsson, O. Enqvist, and F. Kahl, "A polynomial-time bound for matching and registration with outliers," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1-8, 2008.

[32] R. Litman, S. Korman, A. Bronstein, and S. Avidan, "Inverting RANSAC: Global Model Detection via Inlier Rate Estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5243-5251, 2015.

[33] O. Gallo, R. Manduchi, and A. Rafii, "CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data," *Pattern Recognition Letters,* vol. 32, pp. 403-410, 2011.

[34] A. Konouchine, V. Gaganov, and V. Veznevets, "AMLESAC: A new maximum likelihood robust estimator," in *Proceedings of the International Conference on Computer Graphics and Vision (GrapiCon)*, 2005.

[35] S. Choi, T. Kim, and W. Yu, "Robust video stabilization to outlier motion using adaptive RANSAC," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1897-1902, 2009.

[36] L. Moisan and B. Stival, "A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix," *International Journal of Computer Vision*, vol. 57, pp. 201-218, 2004.

[37] J. Rabin, J. Delon, Y. Gousseau, and L. Moisan, "MAC-RANSAC: a robust algorithm for the recognition of multiple objects," in *Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPTV 2010)*, pp. 051, 2010.

[38] J. jae Lee and G. Kim, "Robust estimation of camera homography using fuzzy RANSAC," in *Computational Science and Its Applications–ICCSA 2007*, ed: Springer, pp. 992-1002, 2007.

[39] T. Watanabe, "A fuzzy RANSAC algorithm based on reinforcement learning concept," in *Fuzzy Systems (FUZZ), 2013 IEEE International Conference on*, pp. 1-6, 2013.

[40] O. Chum, J. Matas, and J. Kittler, "Locally optimized RANSAC," in *Pattern Recognition*, ed: Springer, pp. 236-243, 2003.

[41] K. Lebeda, J. Matas, and O. Chum, "Fixing the locally optimized RANSAC–Full experimental evaluation," Research Report CTU–CMP–2012–17, Center for Machine Perception, Czech Technical University, Prague, Czech Republic, 2012. http://cmp. felk. cvut. cz/software/LO-RANSAC/Lebeda-2012-Fixing LORANSAC-tr. pdf. 16, 35, 412012, 2012.

[42] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*: Cambridge university press, 2003.

[43] T. Sattler, B. Leibe, and L. Kobbelt, "SCRAMSAC: Improving RANSAC's efficiency with a spatial consistency filter," in *Computer vision, 2009 ieee 12th international conference on*, pp. 2090-2097, 2009.

[44] M. Xu and J. Lu, "Distributed RANSAC for the robust estimation of three-dimensional reconstruction," *IET computer vision,* vol. 6, pp. 324-333, 2012.

[45] J. Matas and O. Chum, "Randomized RANSAC with sequential probability ratio test," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, pp. 1727-1732, 2005.

[46] J. Matas and O. Chum, "Randomized RANSAC with T d, d test," *Image and Vision Computing,* vol. 22, pp. 837-842, 2004.

[47] D. P. Capel, "An Effective Bail-out Test for RANSAC Consensus Scoring," in *BMVC*, 2005.

[48] X. Wang, H. Zhang, and S. Liu, "Reliable RANSAC Using a Novel Preprocessing Model," *Computational and mathematical methods in medicine,* vol. 2013, 2013.

[49] P. Trivedi, T. Agarwal, and K. Muthunagai, "MC-RANSAC: A Pre-processing Model for RANSAC using Monte Carlo method implemented on a GPU," in *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, pp. 1380-1383, 2013.

[50] D. Nasuto and J. B. R. Craddock, "Napsac: High noise, high dimensional robust estimation-it's in the bag," 2002.

[51] B. J. Tordoff and D. W. Murray, "Guided-MLESAC: Faster image transform estimation by using matching priors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 27, pp. 1523-1535, 2005.

[52] Y. Zhao, R. Hong, J. Jiang, J. Wen, and H. Zhang, "Image matching by fast random sample consensus," in *Proceedings of the fifth international conference on internet multimedia computing and service*, pp. 159-162, 2013.

[53] Y. Wu, W. Ma, M. Gong, L. Su, and L. Jiao, "A novel point-matching algorithm based on fast sample consensus for image registration," *Geoscience and Remote Sensing Letters, IEEE,* vol. 12, pp. 43-47, 2015.

[54] V. Rodehorst and O. Hellwich, "Genetic algorithm sample consensus (gasac)-a parallel strategy for robust parameter estimation," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pp. 103-103, 2006.

[55] A. S. Chernyavskiy, "Discrete attribute-based particle swarm optimization for robust parameter estimation.", 2008.

[56] A. Meler, M. Decrouez, and J. Crowley, "Betasac: A new conditional sampling for ransac," in *British Machine Vision Conference 2010*, 2010.

[57] T. Botterill, S. Mills, and R. D. Green, "New Conditional Sampling Strategies for Speeded-Up RANSAC," in *BMVC*, pp. 1-11, 2009.

[58] P. H. Torr and C. Davidson, "IMPSAC: Synthesis of importance sampling and random sample consensus," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, pp. 354-364, 2003.

[59] J. Wang and X. Luo, "Purposive Sample Consensus: A Paradigm for Model Fitting with Application to Visual Odometry," in *Field and Service Robotics*, pp. 335-349, 2015.

[60] O. Chum, T. Werner, and J. Matas, "Two-view geometry estimation unaffected by a dominant plane," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, pp. 772-779, 2005.

[61] J.-M. Frahm and M. Pollefeys, "RANSAC for (quasi-) degenerate data (QDEGSAC)," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, pp. 453-460, 2006.

[62] C. C. Chou and C.-C. Wang, "2-point RANSAC for scene image matching under large viewpoint changes," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 3646-3651, 2015.

[63] X. Qian and C. Ye, "NCC-RANSAC: A Fast Plane Extraction Method for 3-D Range Data Segmentation," *Cybernetics, IEEE Transactions on*, vol. 44, pp. 2771-2783, 2014.

[64] Y. Kanazawa and H. Kawakami, "Detection of Planar Regions with Uncalibrated Stereo using Distributions of Feature Points," in *BMVC*, pp. 1-10, 2004.

[65] E. Vincent and R. Laganiére, "Detecting planar homographies in an image pair," in *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*, pp. 182-187, 2001.

[66] M. Zuliani, C. S. Kenney, and B. Manjunath, "The multiransac algorithm and its application to detect planar homographies," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, pp. III-153-6, 2005.

[67] A. Barclay and H. Kaufmann, "FT-RANSAC: Towards robust multi-modal homography estimation," in *Pattern recognition in remote sensing (PRRS), 2014 8th IAPR workshop on*, pp. 1-4, 2014.

[68] A. Vedaldi, H. Jin, P. Favaro, and S. Soatto, "KALMANSAC: Robust filtering by consensus," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, pp. 633-640, 2005.

[69] P. C. Niedfeldt and R. W. Beard, "Recursive RANSAC: Multiple Signal Estimation with Outliers," in *9th IFAC Symposium on Nonlinear Control Systems*, pp. 45-50, 2013.

[70] Y. C. Cheng and S. C. Lee, "A new method for quadratic curve detection using K-RANSAC with acceleration techniques," *Pattern Recognition,* vol. 28, pp. 663-682, 1995.

[71] J. Matas and O. Chum, "Randomized ransac," *Center for Machine Perception, Czech Technical University, Prague, December,* 2001.

[72] O. Chum and J. Matas, "Randomized RANSAC with Td, d test," in *Proc. British Machine Vision Conference*, pp. 448-457, 2002.

[73] C. Feng and Y. Hung, "A Robust Method for Estimating the Fundamental Matrix," in *DICTA*, pp. 633-642, 2003.

[74] S. Choi and J.-H. Kim, "Robust regression to varying data distribution and its application to landmark-based localization," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, pp. 3465-3470, 2008.

[75] O. Chum and J. Matas, "Matching with PROSAC-progressive sample consensus," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, pp. 220-226, 2005.

[76] O. Chum and J. Matas, "Optimal randomized RANSAC," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 30, pp. 1472-1482, 2008.

[77] A. Chernyavskiy, "A robust scheme of model parameters estimation based on the particle swarm method in the image matching problem," *Journal of Computer and Systems Sciences International,* vol. 47, pp. 764-777, 2008.

[78] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 4293-4299, 2009.

[79] K. Ni, H. Jin, and F. Dellaert, "Groupsac: Efficient consensus in the presence of groupings," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2193-2200, 2009.

[80] J. Choi and G. Medioni, "StaRSaC: Stable random sample consensus for parameter estimation," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 675-682, 2009.

[81] Z. Kang, L. Zhang, B. Wang, Z. Li, and F. Jia, "An optimized BaySAC algorithm for efficient fitting of primitives in point clouds," *Geoscience and Remote Sensing Letters, IEEE,* vol. 11, pp. 1096-1100, 2014.

[82] S. Otte, U. Schwanecke, and A. Zell, "ANTSAC: A Generic RANSAC Variant Using Principles of Ant Colony Algorithms," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 3558-3563, 2014.

[83] E. Montijano, S. Martinez, and C. Sagues, "Distributed Robust Consensus Using RANSAC and Dynamic Opinions," *Control Systems Technology, IEEE Transactions on,* vol. 23, pp. 150-163, 2015.

APPENDIX 1.

Table 1: Results for Simulated Problem Sets 1 in Chapter 4

Algorithm	Problem	m	true I	Run	ME	detected I	t	f
SASSAC	1	100	90	1	82.89684	90	2	194
CISAC	1	100	90	1	82.89684	90	1	97
RANSAC	1	100	90	1	124.88872	86	7	7
MSASSAC	1	100	90	1	80.452403	90	2	194
MCISAC	1	100	90	1	82.89684	90	1	97
MSAC	1	100	90	1	98.657932	90	6	6
SASSAC	1	100	90	2	82.89684	90	2	194
CISAC	1	100	90	2	82.89684	90	1	97
RANSAC	1	100	90	2	69.640197	90	15	15
MSASSAC	1	100	90	2	70.028832	90	2	194
MCISAC	1	100	90	2	82.89684	90	1	97
MSAC	1	100	90	2	174.73525	77	12	12
SASSAC	1	100	90	3	82.89684	90	2	194
CISAC	1	100	90	3	82.89684	90	1	97
RANSAC	1	100	90	3	140.27772	85	14	14
MSASSAC	1	100	90	3	76.194084	90	2	194
MCISAC	1	100	90	3	82.89684	90	1	97
MSAC	1	100	90	3	141.23607	85	8	8
SASSAC	1	100	90	4	82.89684	90	2	194
CISAC	1	100	90	4	82.89684	90	1	97
RANSAC	1	100	90	4	91.19497	90	6	6

MSASSAC	1	100	90	4	74.280386	90	2	194
MCISAC	1	100	90	4	82.89684	90	1	97
MSAC	1	100	90	4	98.798896	90	6	6
SASSAC	1	100	90	5	82.89684	90	2	194
CISAC	1	100	90	5	82.89684	90	1	97
RANSAC	1	100	90	5	76.072187	90	6	6
MSASSAC	1	100	90	5	70.014016	90	2	194
MCISAC	1	100	90	5	82.89684	90	1	97
MSAC	1	100	90	5	115.0917	84	17	17
SASSAC	2	100	80	1	113.31558	80	2	194
CISAC	2	100	80	1	113.31558	80	1	97
RANSAC	2	100	80	1	125.54253	80	10	10
MSASSAC	2	100	80	1	110.46279	80	2	194
MCISAC	2	100	80	1	110.46279	80	1	97
MSAC	2	100	80	1	180.50108	65	25	25
SASSAC	2	100	80	2	113.31558	80	2	194
CISAC	2	100	80	2	113.31558	80	1	97
RANSAC	2	100	80	2	107.2246	80	18	18
MSASSAC	2	100	80	2	110.46279	80	2	194
MCISAC	2	100	80	2	110.46279	80	1	97
MSAC	2	100	80	2	153.24258	77	12	12
SASSAC	2	100	80	3	113.31558	80	2	194
CISAC	2	100	80	3	113.31558	80	1	97
RANSAC	2	100	80	3	140.7289	76	13	13
MSASSAC	2	100	80	3	104.13995	80	2	194

MCISAC	2	100	80	3	110.46279	80	1	97
MSAC	2	100	80	3	169.88386	66	23	23
SASSAC	2	100	80	4	113.31558	80	2	194
CISAC	2	100	80	4	113.31558	80	1	97
RANSAC	2	100	80	4	108.65162	80	11	11
MSASSAC	2	100	80	4	110.46279	80	3	291
MCISAC	2	100	80	4	110.46279	80	1	97
MSAC	2	100	80	4	105.50282	80	10	10
SASSAC	2	100	80	5	113.31558	80	2	194
CISAC	2	100	80	5	113.31558	80	1	97
RANSAC	2	100	80	5	146.77326	76	13	13
MSASSAC	2	100	80	5	107.91078	80	2	194
MCISAC	2	100	80	5	110.46279	80	1	97
MSAC	2	100	80	5	157.14596	80	10	10
SASSAC	3	100	70	1	119.73696	70	2	194
CISAC	3	100	70	1	119.73696	70	1	97
RANSAC	3	100	70	1	128.19693	70	18	18
MSASSAC	3	100	70	1	119.73696	70	2	194
MCISAC	3	100	70	1	119.73696	70	1	97
MSAC	3	100	70	1	174.85968	65	28	28
SASSAC	3	100	70	2	119.73696	70	2	194
CISAC	3	100	70	2	119.73696	70	1	97
RANSAC	3	100	70	2	163.79897	66	23	23
MSASSAC	3	100	70	2	119.73696	70	2	194
MCISAC	3	100	70	2	119.73696	70	1	97

MSAC	3	100	70	2	129.35824	70	18	18
SASSAC	3	100	70	3	119.73696	70	2	194
CISAC	3	100	70	3	119.73696	70	1	97
RANSAC	3	100	70	3	154.27638	66	24	24
MSASSAC	3	100	70	3	119.73696	70	2	194
MCISAC	3	100	70	3	119.73696	70	1	97
MSAC	3	100	70	3	138.99692	70	18	18
SASSAC	3	100	70	4	119.73696	70	3	291
CISAC	3	100	70	4	119.73696	70	1	97
RANSAC	3	100	70	4	120.69667	70	21	21
MSASSAC	3	100	70	4	119.73696	70	2	194
MCISAC	3	100	70	4	119.73696	70	1	97
MSAC	3	100	70	4	121.67552	70	20	20
SASSAC	3	100	70	5	119.73696	70	2	194
CISAC	3	100	70	5	119.73696	70	1	97
RANSAC	3	100	70	5	193.46522	57	45	45
MSASSAC	3	100	70	5	119.73696	70	2	194
MCISAC	3	100	70	5	119.73696	70	1	97
MSAC	3	100	70	5	178.28081	64	27	27
SASSAC	4	500	450	1	474.3711	450	2	994
CISAC	4	500	450	1	474.3711	450	1	497
RANSAC	4	500	450	1	442.62535	450	6	6
MSASSAC	4	500	450	1	363.75572	450	2	994
MCISAC	4	500	450	1	377.06159	450	1	497
MSAC	4	500	450	1	558.79645	449	16	16

SASSAC	4	500	450	2	474.3711	450	2	994
CISAC	4	500	450	2	474.3711	450	1	497
RANSAC	4	500	450	2	815.24973	379	13	13
MSASSAC	4	500	450	2	353.91678	450	2	994
MCISAC	4	500	450	2	377.06159	450	1	497
MSAC	4	500	450	2	748.06336	429	9	9
SASSAC	4	500	450	3	474.3711	450	2	994
CISAC	4	500	450	3	474.3711	450	1	497
RANSAC	4	500	450	3	632.157	436	7	7
MSASSAC	4	500	450	3	377.06159	450	2	994
MCISAC	4	500	450	3	377.06159	450	1	497
MSAC	4	500	450	3	448.75539	450	6	6
SASSAC	4	500	450	4	474.3711	450	2	994
CISAC	4	500	450	4	474.3711	450	1	497
RANSAC	4	500	450	4	841.49242	366	16	16
MSASSAC	4	500	450	4	346.69881	450	2	994
MCISAC	4	500	450	4	377.06159	450	1	497
MSAC	4	500	450	4	423.45978	450	6	6
SASSAC	4	500	450	5	474.3711	450	2	994
CISAC	4	500	450	5	474.3711	450	1	497
RANSAC	4	500	450	5	576.50131	435	15	15
MSASSAC	4	500	450	5	345.09398	450	2	994
MCISAC	4	500	450	5	377.06159	450	1	497
MSAC	4	500	450	5	779.79232	386	12	12
SASSAC	5	500	400	1	549.19484	400	2	994

CISAC	5	500	400	1	549.19484	400	1	497
RANSAC	5	500	400	1	663.97153	397	11	11
MSASSAC	5	500	400	1	487.26203	400	2	994
MCISAC	5	500	400	1	489.60448	400	1	497
MSAC	5	500	400	1	719.916	382	13	13
SASSAC	5	500	400	2	549.19484	400	2	994
CISAC	5	500	400	2	549.19484	400	1	497
RANSAC	5	500	400	2	584.73829	400	11	11
MSASSAC	5	500	400	2	489.60448	400	2	994
MCISAC	5	500	400	2	489.60448	400	1	497
MSAC	5	500	400	2	701.51557	387	12	12
SASSAC	5	500	400	3	549.19484	400	2	994
CISAC	5	500	400	3	549.19484	400	1	497
RANSAC	5	500	400	3	988.24219	301	34	34
MSASSAC	5	500	400	3	489.60448	400	2	994
MCISAC	5	500	400	3	489.60448	400	1	497
MSAC	5	500	400	3	539.76599	400	13	13
SASSAC	5	500	400	4	549.19484	400	2	994
CISAC	5	500	400	4	549.19484	400	1	497
RANSAC	5	500	400	4	518.3864	400	22	22
MSASSAC	5	500	400	4	467.09653	400	2	994
MCISAC	5	500	400	4	489.60448	400	1	497
MSAC	5	500	400	4	565.02423	400	10	10
SASSAC	5	500	400	5	549.19484	400	2	994
CISAC	5	500	400	5	549.19484	400	1	497

RANSAC	5	500	400	5	773.22821	383	15	15
MSASSAC	5	500	400	5	489.60448	400	2	994
MCISAC	5	500	400	5	489.60448	400	1	497
MSAC	5	500	400	5	605.18518	400	10	10
SASSAC	6	500	350	1	700.90715	350	2	994
CISAC	6	500	350	1	700.90715	350	1	497
RANSAC	6	500	350	1	664.80133	350	18	18
MSASSAC	6	500	350	1	646.93481	350	2	994
MCISAC	6	500	350	1	654.97612	350	1	497
MSAC	6	500	350	1	1026.363	275	49	49
SASSAC	6	500	350	2	700.90715	350	2	994
CISAC	6	500	350	2	700.90715	350	1	497
RANSAC	6	500	350	2	724.84957	350	18	18
MSASSAC	6	500	350	2	640.59631	350	2	994
MCISAC	6	500	350	2	654.97612	350	1	497
MSAC	6	500	350	2	769.1797	350	40	40
SASSAC	6	500	350	3	700.90715	350	2	994
CISAC	6	500	350	3	700.90715	350	1	497
RANSAC	6	500	350	3	1023.5465	274	50	50
MSASSAC	6	500	350	3	654.97612	350	2	994
MCISAC	6	500	350	3	654.97612	350	1	497
MSAC	6	500	350	3	762.45126	339	21	21
SASSAC	6	500	350	4	700.90715	350	2	994
CISAC	6	500	350	4	700.90715	350	1	497
RANSAC	6	500	350	4	906.32579	315	28	28

MSASSAC	6	500	350	4	640.41771	350	2	994
MCISAC	6	500	350	4	654.97612	350	1	497
MSAC	6	500	350	4	873.87299	349	28	28
SASSAC	6	500	350	5	700.90715	350	2	994
CISAC	6	500	350	5	700.90715	350	1	497
RANSAC	6	500	350	5	838.29208	329	24	24
MSASSAC	6	500	350	5	618.99361	350	2	994
MCISAC	6	500	350	5	654.97612	350	1	497
MSAC	6	500	350	5	799.80489	349	18	18
SASSAC	7	1000	900	1	1047.5428	900	2	1994
CISAC	7	1000	900	1	1047.5428	900	1	997
RANSAC	7	1000	900	1	1282.0084	857	7	7
MSASSAC	7	1000	900	1	685.99278	900	2	1994
MCISAC	7	1000	900	1	685.99278	900	1	997
MSAC	7	1000	900	1	1135.6823	900	6	6
SASSAC	7	1000	900	2	1047.5428	900	2	1994
CISAC	7	1000	900	2	1047.5428	900	1	997
RANSAC	7	1000	900	2	891.22164	900	6	6
MSASSAC	7	1000	900	2	685.99278	900	2	1994
MCISAC	7	1000	900	2	685.99278	900	1	997
MSAC	7	1000	900	2	728.15628	900	6	6
SASSAC	7	1000	900	3	1047.5428	900	2	1994
CISAC	7	1000	900	3	1047.5428	900	1	997
RANSAC	7	1000	900	3	893.45433	900	6	6
MSASSAC	7	1000	900	3	685.99278	900	2	1994

MCISAC	7	1000	900	3	685.99278	900	1	997
MSAC	7	1000	900	3	1295.7646	868	17	17
SASSAC	7	1000	900	4	1047.5428	900	2	1994
CISAC	7	1000	900	4	1047.5428	900	1	997
RANSAC	7	1000	900	4	1685.7172	816	9	9
MSASSAC	7	1000	900	4	685.99278	900	2	1994
MCISAC	7	1000	900	4	685.99278	900	1	997
MSAC	7	1000	900	4	1096.0524	879	7	7
SASSAC	7	1000	900	5	1047.5428	900	2	1994
CISAC	7	1000	900	5	1047.5428	900	1	997
RANSAC	7	1000	900	5	1107.0685	878	23	23
MSASSAC	7	1000	900	5	685.99278	900	2	1994
MCISAC	7	1000	900	5	685.99278	900	1	997
MSAC	7	1000	900	5	1591.4606	795	11	11
SASSAC	8	1000	800	1	1046.0696	800	2	1994
CISAC	8	1000	800	1	1046.0696	800	1	997
RANSAC	8	1000	800	1	1535.4666	698	34	34
MSASSAC	8	1000	800	1	941.89488	800	2	1994
MCISAC	8	1000	800	1	974.17485	800	1	997
MSAC	8	1000	800	1	1358.6011	800	10	10
SASSAC	8	1000	800	2	1046.0696	800	2	1994
CISAC	8	1000	800	2	1046.0696	800	1	997
RANSAC	8	1000	800	2	1347.2883	778	12	12
MSASSAC	8	1000	800	2	957.16911	800	2	1994
MCISAC	8	1000	800	2	974.17485	800	1	997

MSAC	8	1000	800	2	1226.9052	800	10	10
SASSAC	8	1000	800	3	1046.0696	800	2	1994
CISAC	8	1000	800	3	1046.0696	800	1	997
RANSAC	8	1000	800	3	1362.7538	800	13	13
MSASSAC	8	1000	800	3	964.80227	800	2	1994
MCISAC	8	1000	800	3	974.17485	800	1	997
MSAC	8	1000	800	3	1549.4192	733	15	15
SASSAC	8	1000	800	4	1046.0696	800	2	1994
CISAC	8	1000	800	4	1046.0696	800	1	997
RANSAC	8	1000	800	4	1741.9824	751	14	14
MSASSAC	8	1000	800	4	953.41494	800	2	1994
MCISAC	8	1000	800	4	974.17485	800	1	997
MSAC	8	1000	800	4	1153.5091	800	10	10
SASSAC	8	1000	800	5	1046.0696	800	2	1994
CISAC	8	1000	800	5	1046.0696	800	1	997
RANSAC	8	1000	800	5	1059.9889	800	13	13
MSASSAC	8	1000	800	5	966.74463	800	2	1994
MCISAC	8	1000	800	5	974.17485	800	1	997
MSAC	8	1000	800	5	1037.5314	800	10	10
SASSAC	9	1000	700	1	1453.7138	700	2	1994
CISAC	9	1000	700	1	1453.7138	700	1	997
RANSAC	9	1000	700	1	1472.1798	700	20	20
MSASSAC	9	1000	700	1	1250.0945	700	2	1994
MCISAC	9	1000	700	1	1266.3553	700	1	997
MSAC	9	1000	700	1	1774.1328	657	24	24

SASSAC	9	1000	700	2	1453.7138	700	2	1994
CISAC	9	1000	700	2	1453.7138	700	1	997
RANSAC	9	1000	700	2	1215.2537	700	18	18
MSASSAC	9	1000	700	2	1207.4707	700	2	1994
MCISAC	9	1000	700	2	1266.3553	700	1	997
MSAC	9	1000	700	2	1245.3406	700	37	37
SASSAC	9	1000	700	3	1453.7138	700	2	1994
CISAC	9	1000	700	3	1453.7138	700	1	997
RANSAC	9	1000	700	3	1461.3562	695	34	34
MSASSAC	9	1000	700	3	1197.4443	700	2	1994
MCISAC	9	1000	700	3	1266.3553	700	1	997
MSAC	9	1000	700	3	1565.8152	680	30	30
SASSAC	9	1000	700	4	1453.7138	700	2	1994
CISAC	9	1000	700	4	1453.7138	700	1	997
RANSAC	9	1000	700	4	1452.1023	696	27	27
MSASSAC	9	1000	700	4	1234.9436	700	2	1994
MCISAC	9	1000	700	4	1266.3553	700	1	997
MSAC	9	1000	700	4	1523.5976	700	19	19
SASSAC	9	1000	700	5	1453.7138	700	2	1994
CISAC	9	1000	700	5	1453.7138	700	1	997
RANSAC	9	1000	700	5	1436.59	700	18	18
MSASSAC	9	1000	700	5	1266.3553	700	2	1994
MCISAC	9	1000	700	5	1266.3553	700	1	997
MSAC	9	1000	700	5	2035.1112	479	44	44

Table 2: Results for Simulated Problem Sets 2 in Chapter 4

Algorithm	Problem	m	true I	Run	ME	detected I	t	f
SASSAC	1	100	60	1	152.99403	60	2	194
CISAC	1	100	60	1	152.99403	60	1	97
RANSAC	1	100	60	1	220.68039	52	62	62
MSASSAC	1	100	60	1	152.99403	60	2	194
MCISAC	1	100	60	1	152.99403	60	1	97
MSAC	1	100	60	1	151.67451	60	35	35
SASSAC	1	100	60	2	152.99403	60	2	194
CISAC	1	100	60	2	152.99403	60	1	97
RANSAC	1	100	60	2	186.98374	58	50	50
MSASSAC	1	100	60	2	152.99403	60	3	291
MCISAC	1	100	60	2	152.99403	60	1	97
MSAC	1	100	60	2	184.02039	60	39	39
SASSAC	1	100	60	3	152.99403	60	2	194
CISAC	1	100	60	3	152.99403	60	1	97
RANSAC	1	100	60	3	172.57921	59	41	41
MSASSAC	1	100	60	3	152.99403	60	2	194
MCISAC	1	100	60	3	152.99403	60	1	97
MSAC	1	100	60	3	174.38544	58	59	59
SASSAC	1	100	60	4	152.99403	60	3	291
CISAC	1	100	60	4	152.99403	60	1	97
RANSAC	1	100	60	4	197.83007	54	54	54
MSASSAC	1	100	60	4	147.39959	60	2	194

MCISAC	1	100	60	4	152.99403	60	1	97
MSAC	1	100	60	4	149.29141	60	62	62
SASSAC	1	100	60	5	152.99403	60	3	291
CISAC	1	100	60	5	152.99403	60	1	97
RANSAC	1	100	60	5	161.95442	60	79	79
MSASSAC	1	100	60	5	152.99403	60	3	291
MCISAC	1	100	60	5	152.99403	60	1	97
MSAC	1	100	60	5	176.44471	60	35	35
SASSAC	2	100	50	1	198.3337	48	2	194
CISAC	2	100	50	1	232.82974	34	1	97
RANSAC	2	100	50	1	174.45257	50	73	73
MSASSAC	2	100	50	1	181.06812	50	2	194
MCISAC	2	100	50	1	232.82974	34	1	97
MSAC	2	100	50	1	195.632	47	94	94
SASSAC	2	100	50	2	174.47959	50	2	194
CISAC	2	100	50	2	232.82974	34	1	97
RANSAC	2	100	50	2	211.57448	48	86	86
MSASSAC	2	100	50	2	200.16008	50	2	194
MCISAC	2	100	50	2	232.82974	34	1	97
MSAC	2	100	50	2	184.50655	50	73	73
SASSAC	2	100	50	3	209.1657	45	2	194
CISAC	2	100	50	3	232.82974	34	1	97
RANSAC	2	100	50	3	223.31354	42	147	147
MSASSAC	2	100	50	3	202.58942	44	3	291
MCISAC	2	100	50	3	232.82974	34	1	97

MSAC	2	100	50	3	178.21307	50	88	88
SASSAC	2	100	50	4	170.03042	50	2	194
CISAC	2	100	50	4	232.82974	34	1	97
RANSAC	2	100	50	4	180.35232	50	91	91
MSASSAC	2	100	50	4	215.8358	40	2	194
MCISAC	2	100	50	4	232.82974	34	1	97
MSAC	2	100	50	4	203.88146	48	118	118
SASSAC	2	100	50	5	173.96111	50	2	194
CISAC	2	100	50	5	232.82974	34	1	97
RANSAC	2	100	50	5	201.594	50	194	194
MSASSAC	2	100	50	5	189.67469	50	2	194
MCISAC	2	100	50	5	232.82974	34	1	97
MSAC	2	100	50	5	205.937	42	147	147
SASSAC	3	100	40	1	236.61729	31	2	194
CISAC	3	100	40	1	251.35911	25	1	97
RANSAC	3	100	40	1	226.50533	36	273	273
MSASSAC	3	100	40	1	235.88215	31	2	194
MCISAC	3	100	40	1	251.15514	24	1	97
MSAC	3	100	40	1	231.78606	36	273	273
SASSAC	3	100	40	2	201.54457	40	2	194
CISAC	3	100	40	2	251.35911	25	1	97
RANSAC	3	100	40	2	219.22604	40	179	179
MSASSAC	3	100	40	2	204.31704	40	2	194
MCISAC	3	100	40	2	251.15514	24	1	97
MSAC	3	100	40	2	218.51399	40	221	221

SASSAC	3	100	40	3	216.74428	39	2	194
CISAC	3	100	40	3	251.35911	25	1	97
RANSAC	3	100	40	3	213.88642	39	198	198
MSASSAC	3	100	40	3	206.9146	40	3	291
MCISAC	3	100	40	3	251.15514	24	1	97
MSAC	3	100	40	3	202.36666	40	179	179
SASSAC	3	100	40	4	225.94238	37	2	194
CISAC	3	100	40	4	251.35911	25	1	97
RANSAC	3	100	40	4	225.65677	38	441	441
MSASSAC	3	100	40	4	236.42906	32	3	291
MCISAC	3	100	40	4	251.15514	24	1	97
MSAC	3	100	40	4	199.38365	40	253	253
SASSAC	3	100	40	5	198.94105	40	2	194
CISAC	3	100	40	5	251.35911	25	1	97
RANSAC	3	100	40	5	220.16068	39	198	198
MSASSAC	3	100	40	5	251.15514	24	2	194
MCISAC	3	100	40	5	251.15514	24	1	97
MSAC	3	100	40	5	209.34699	40	179	179
SASSAC	4	500	300	1	779.8866	300	2	994
CISAC	4	500	300	1	779.8866	300	1	497
RANSAC	4	500	300	1	879.6941	297	36	36
MSASSAC	4	500	300	1	730.77668	300	2	994
MCISAC	4	500	300	1	763.88325	300	1	497
MSAC	4	500	300	1	901.82973	289	40	40
SASSAC	4	500	300	2	779.8866	300	2	994

CISAC	4	500	300	2	779.8866	300	1	497
RANSAC	4	500	300	2	1015.8159	299	35	35
MSASSAC	4	500	300	2	763.88325	300	2	994
MCISAC	4	500	300	2	763.88325	300	1	497
MSAC	4	500	300	2	747.50849	300	56	56
SASSAC	4	500	300	3	779.8866	300	2	994
CISAC	4	500	300	3	779.8866	300	1	497
RANSAC	4	500	300	3	877.54274	300	35	35
MSASSAC	4	500	300	3	756.19966	300	2	994
MCISAC	4	500	300	3	763.88325	300	1	497
MSAC	4	500	300	3	1042.1705	247	79	79
SASSAC	4	500	300	4	779.8866	300	2	994
CISAC	4	500	300	4	779.8866	300	1	497
RANSAC	4	500	300	4	840.72529	300	35	35
MSASSAC	4	500	300	4	759.50143	300	3	1491
MCISAC	4	500	300	4	763.88325	300	1	497
MSAC	4	500	300	4	919.63173	299	35	35
SASSAC	4	500	300	5	779.8866	300	2	994
CISAC	4	500	300	5	779.8866	300	1	497
RANSAC	4	500	300	5	881.73109	298	36	36
MSASSAC	4	500	300	5	763.88325	300	2	994
MCISAC	4	500	300	5	763.88325	300	1	497
MSAC	4	500	300	5	943.16384	283	44	44
SASSAC	5	500	250	1	984.85475	250	2	994
CISAC	5	500	250	1	984.85475	250	1	497

RANSAC	5	500	250	1	991.62215	246	78	78
MSASSAC	5	500	250	1	867.51711	250	2	994
MCISAC	5	500	250	1	867.51711	250	1	497
MSAC	5	500	250	1	983.0789	250	90	90
SASSAC	5	500	250	2	984.85475	250	2	994
CISAC	5	500	250	2	984.85475	250	1	497
RANSAC	5	500	250	2	1066.003	235	94	94
MSASSAC	5	500	250	2	867.51711	250	2	994
MCISAC	5	500	250	2	867.51711	250	1	497
MSAC	5	500	250	2	1008.5939	235	117	117
SASSAC	5	500	250	3	984.85475	250	2	994
CISAC	5	500	250	3	984.85475	250	1	497
RANSAC	5	500	250	3	990.65135	244	80	80
MSASSAC	5	500	250	3	867.51711	250	2	994
MCISAC	5	500	250	3	867.51711	250	1	497
MSAC	5	500	250	3	865.87391	250	73	73
SASSAC	5	500	250	4	984.85475	250	2	994
CISAC	5	500	250	4	984.85475	250	1	497
RANSAC	5	500	250	4	1112.2619	214	152	152
MSASSAC	5	500	250	4	867.51711	250	2	994
MCISAC	5	500	250	4	867.51711	250	1	497
MSAC	5	500	250	4	979.58798	248	75	75
SASSAC	5	500	250	5	984.85475	250	2	994
CISAC	5	500	250	5	984.85475	250	1	497
RANSAC	5	500	250	5	987.63909	250	98	98

MSASSAC	5	500	250	5	867.51711	250	2	994
MCISAC	5	500	250	5	867.51711	250	1	497
MSAC	5	500	250	5	1182.6985	243	106	106
SASSAC	6	500	200	1	1015.7682	200	2	994
CISAC	6	500	200	1	1015.7682	200	1	497
RANSAC	6	500	200	1	1102.5133	194	379	379
MSASSAC	6	500	200	1	988.99309	200	2	994
MCISAC	6	500	200	1	1015.7682	200	1	497
MSAC	6	500	200	1	1040.2145	200	179	179
SASSAC	6	500	200	2	1015.7682	200	3	1491
CISAC	6	500	200	2	1015.7682	200	1	497
RANSAC	6	500	200	2	1039.5718	200	577	577
MSASSAC	6	500	200	2	1008.4005	200	2	994
MCISAC	6	500	200	2	1015.7682	200	1	497
MSAC	6	500	200	2	1124.0208	200	192	192
SASSAC	6	500	200	3	1015.7682	200	2	994
CISAC	6	500	200	3	1015.7682	200	1	497
RANSAC	6	500	200	3	1164.2729	187	235	235
MSASSAC	6	500	200	3	1015.7682	200	2	994
MCISAC	6	500	200	3	1015.7682	200	1	497
MSAC	6	500	200	3	1007.2848	200	448	448
SASSAC	6	500	200	4	1015.7682	200	4	1988
CISAC	6	500	200	4	1015.7682	200	1	497
RANSAC	6	500	200	4	1048.2362	200	179	179
MSASSAC	6	500	200	4	1015.7682	200	2	994

MCISAC	6	500	200	4	1015.7682	200	1	497
MSAC	6	500	200	4	1101.8811	180	273	273
SASSAC	6	500	200	5	1015.7682	200	2	994
CISAC	6	500	200	5	1015.7682	200	1	497
RANSAC	6	500	200	5	1184.6736	175	306	306
MSASSAC	6	500	200	5	989.14609	200	2	994
MCISAC	6	500	200	5	1015.7682	200	1	497
MSAC	6	500	200	5	1034.6176	200	298	298
SASSAC	7	1000	600	1	1619.8645	600	2	1994
CISAC	7	1000	600	1	1619.8645	600	1	997
RANSAC	7	1000	600	1	1603.2725	600	35	35
MSASSAC	7	1000	600	1	1505.6346	600	2	1994
MCISAC	7	1000	600	1	1520.2328	600	1	997
MSAC	7	1000	600	1	1615.1748	600	58	58
SASSAC	7	1000	600	2	1619.8645	600	2	1994
CISAC	7	1000	600	2	1619.8645	600	1	997
RANSAC	7	1000	600	2	2003.9353	493	77	77
MSASSAC	7	1000	600	2	1500.087	600	2	1994
MCISAC	7	1000	600	2	1520.2328	600	1	997
MSAC	7	1000	600	2	1907.5858	529	66	66
SASSAC	7	1000	600	3	1619.8645	600	2	1994
CISAC	7	1000	600	3	1619.8645	600	1	997
RANSAC	7	1000	600	3	1528.2524	600	68	68
MSASSAC	7	1000	600	3	1489.212	600	2	1994
MCISAC	7	1000	600	3	1520.2328	600	1	997

MSAC	7	1000	600	3	1633.7031	600	39	39
SASSAC	7	1000	600	4	1619.8645	600	2	1994
CISAC	7	1000	600	4	1619.8645	600	1	997
RANSAC	7	1000	600	4	2296.9123	428	136	136
MSASSAC	7	1000	600	4	1476.0138	600	2	1994
MCISAC	7	1000	600	4	1520.2328	600	1	997
MSAC	7	1000	600	4	1614.3995	600	63	63
SASSAC	7	1000	600	5	1619.8645	600	2	1994
CISAC	7	1000	600	5	1619.8645	600	1	997
RANSAC	7	1000	600	5	1536.5452	600	35	35
MSASSAC	7	1000	600	5	1490.4184	600	2	1994
MCISAC	7	1000	600	5	1520.2328	600	1	997
MSAC	7	1000	600	5	1885.2831	569	43	43
SASSAC	8	1000	500	1	1898.8244	500	2	1994
CISAC	8	1000	500	1	1898.8244	500	1	997
RANSAC	8	1000	500	1	2398.6682	372	240	240
MSASSAC	8	1000	500	1	1729.1584	500	2	1994
MCISAC	8	1000	500	1	1818.0483	500	1	997
MSAC	8	1000	500	1	1979.1915	488	80	80
SASSAC	8	1000	500	2	1898.8244	500	2	1994
CISAC	8	1000	500	2	1898.8244	500	1	997
RANSAC	8	1000	500	2	2085.6419	424	142	142
MSASSAC	8	1000	500	2	1818.0483	500	2	1994
MCISAC	8	1000	500	2	1818.0483	500	1	997
MSAC	8	1000	500	2	2066.6587	467	96	96

SASSAC	8	1000	500	3	1898.8244	500	2	1994
CISAC	8	1000	500	3	1898.8244	500	1	997
RANSAC	8	1000	500	3	2131.4288	459	103	103
MSASSAC	8	1000	500	3	1750.877	500	2	1994
MCISAC	8	1000	500	3	1818.0483	500	1	997
MSAC	8	1000	500	3	2202.3849	437	125	125
SASSAC	8	1000	500	4	1898.8244	500	2	1994
CISAC	8	1000	500	4	1898.8244	500	1	997
RANSAC	8	1000	500	4	1882.3426	500	73	73
MSASSAC	8	1000	500	4	1733.7194	500	2	1994
MCISAC	8	1000	500	4	1818.0483	500	1	997
MSAC	8	1000	500	4	1994.8553	490	79	79
SASSAC	8	1000	500	5	1898.8244	500	2	1994
CISAC	8	1000	500	5	1898.8244	500	1	997
RANSAC	8	1000	500	5	2130.7171	480	92	92
MSASSAC	8	1000	500	5	1714.8072	500	2	1994
MCISAC	8	1000	500	5	1818.0483	500	1	997
MSAC	8	1000	500	5	1829.0406	500	73	73
SASSAC	9	1000	400	1	2046.2313	400	2	1994
CISAC	9	1000	400	1	2046.2313	400	1	997
RANSAC	9	1000	400	1	2039.413	400	179	179
MSASSAC	9	1000	400	1	2019.5921	400	3	2991
MCISAC	9	1000	400	1	2046.2313	400	1	997
MSAC	9	1000	400	1	2378.0379	356	357	357
SASSAC	9	1000	400	2	2046.2313	400	2	1994

CISAC	9	1000	400	2	2046.2313	400	1	997
RANSAC	9	1000	400	2	2218.7764	397	439	439
MSASSAC	9	1000	400	2	2001.5483	400	2	1994
MCISAC	9	1000	400	2	2046.2313	400	1	997
MSAC	9	1000	400	2	2349.5912	330	388	388
SASSAC	9	1000	400	3	2046.2313	400	3	2991
CISAC	9	1000	400	3	2046.2313	400	1	997
RANSAC	9	1000	400	3	2262.2181	380	229	229
MSASSAC	9	1000	400	3	2046.2313	400	2	1994
MCISAC	9	1000	400	3	2046.2313	400	1	997
MSAC	9	1000	400	3	2162.6089	400	179	179
SASSAC	9	1000	400	4	2046.2313	400	2	1994
CISAC	9	1000	400	4	2046.2313	400	1	997
RANSAC	9	1000	400	4	2177.0274	387	204	204
MSASSAC	9	1000	400	4	2017.2168	400	2	1994
MCISAC	9	1000	400	4	2046.2313	400	1	997
MSAC	9	1000	400	4	2336.9847	385	209	209
SASSAC	9	1000	400	5	2046.2313	400	2	1994
CISAC	9	1000	400	5	2046.2313	400	1	997
RANSAC	9	1000	400	5	2073.8845	400	179	179
MSASSAC	9	1000	400	5	2046.2313	400	2	1994
MCISAC	9	1000	400	5	2046.2313	400	1	997
MSAC	9	1000	400	5	2252.6734	388	202	202