OUTLIER DETECTION FOR ROBUST PARAMETER ESTIMATION AGAINST MULTI-MODELED/STRUCTURED DATA

by

Ngo Thanh Trung

A dissertation submitted to
THE GRADUATE SCHOOL OF ENGINEERING SCIENCE
OSAKA UNIVERSITY
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY IN ENGINEERING.

Osaka

March 2010

Abstract

Model parameter estimation and automatic outlier detection is a fundamental and important problem in computer vision. Vision data is noisy and usually contains multiple structures, models. RANSAC has been proven to be the most popular and effective solution for such problem, however it requires some user-defined threshold to discriminate inliers/outliers. It is then improved by the adaptive-scale robust estimators, which do not require the user-defined threshold and detect inliers automatically. However, there still remain two problems. The first problem is that these adaptive-scale robust estimators do not focus on the accurate inlier detection. The second problem is that for the data with multiple models and structures, a robust estimator is to extract only one structure out of them and all the data points of the other structures become the outliers. This produces a very high outlier rate situation for a robust estimator to work efficiently or the inliers/outliers may not be distinguished.

In this thesis, we present some research works that tackle to these two problems. First, we propose several adaptive-scale robust estimators which can detect inliers accurately. There are two reasons for the idea of accurate inlier detection. The first reason is that if a robust estimator detects inliers better, then the robustness of the estimation can be improved. The second reason is that, in many real applications such as motion segmentation and range image segmentation, if the inlier detection is not very well, then a structure can be broken into smaller structures or united with the other structures.

In the second research work that tackles to the second problem as stated above, we propose an idea to deal with problem of multiple models/structures. We think

about the outlier models, which are not the models that we want to extract the data for, and pre-filtering the data points of these models. This helps a robust estimator work more effective, robust and also save the computational cost. We demonstrate the idea in some specific situation with a new egomotion estimation algorithm.

In the experiments, various analytic simulations in many aspects have shown the advantage of the proposed robust estimators compared to several latest robust estimators. The real experiments were also performed to prove the validation of the proposed estimators in real applications. We also carried out the experiments for the new egomotion estimation algorithm, it has the advantage for the situation of fast camera motion compared to the state of the art algorithm.

Acknowledgements

It is a great pleasure to mention the people who have made the creation of this dissertation possible.

First of all, I would like to express my deepest gratitude to my advisors, Yasushi Yagi, Yasuhiro Mukaigawa, Hajime Nagahara and Ryusuke Sagawa for their invaluable guidance. I have received enthusiasm, knowledge and experience and I have learned a lot from them. Especially, I would like to give my honest thanks to Hajime Nagahara, who has been guiding me for more than six year since I first entered his laboratory in November 2003 and professor Yasushi Yagi, who accepted me in Department of Intelligent Media. I would like to thank my former advisor, professor Masahiko Yachida, for his kind support and advice when I was in the master course.

I am really lucky to belong to two laboratories, one in Toyonaka campus and the other in Suita campus, where I spent most of my time. I would like to thank all the staffs, colleagues and secretaries in the two laboratories. They have been very nice to me and make my campus life happy.

I would like to thank all reviewers, professor Hiroshi Ishiguro, professor Kosuke Sato and professor Tatsuo Arai, for their kind efforts to review my dissertation.

I would like to thank all my friends in Japan who have helped me in every matters I encountered.

I would like to send my warm thanks to my family for encouraging me for all my staying in Japan.

Finally, I would like to express my thanks to all financial supporters. My life in Japan was financially supported by Matsuda Scholarship and Japan Students Service Organization (JASSO). It is my honor to receive these supports. I would have hardly

finished the dissertation without them.

Contents

Abstract ii

Acknowledgements iv

List of Figures iii

List of Tables 1

1 Introduction 2
- 1.1 Problem Statement . . . 3
- 1.2 Related Works . . . 4
  - 1.2.1 Improvements in Computational Speed . . . 5
  - 1.2.2 Improvements in Adaptability . . . 6
  - 1.2.3 Improvements in Hypothesis Evaluation Accuracy . . . 7
- 1.3 Motivations of the Research in This Thesis . . . 9
- 1.4 Contributions . . . 12
- 1.5 Structure of the Thesis . . . 13

2 Adaptive-scale Robust Estimators using Distribution Model Fitting 14
- 2.1 Preliminaries . . . 14
- 2.2 Inlier Detection Methods . . . 16
  - 2.2.1 Fitting for One Parameter . . . 17
  - 2.2.2 Fitting for Two Parameters . . . 18
- 2.3 Bin-width for Density Estimation . . . 20
  - 2.3.1 Fixed Bin-width Computation . . . 20
  - 2.3.2 Adaptive Bin-width Computation . . . 20
- 2.4 Case-dependent Residual Distribution Analysis . . . 23
  - 2.4.1 Linear Residual . . . 24
  - 2.4.2 Non-linear Residual . . . 25
- 2.5 Hypothesis Evaluation Function . . . 27
- 2.6 Summary of Proposed Robust Estimators: FITSAC1, FITSAC2 . . . 28
- 2.7 Experiments with Adaptive-scale Robust Estimators . . . 28

2.7.1 Linear Residual 30
2.7.2 Non-linear Residual 37
2.7.3 Computational Cost 40
2.8 Chapter Conclusion 41

3 Outlier Detection by Pre-filtering before Fitting Model 42
3.1 Introduction 42
3.2 Egomotion Estimation by Separating Feature Set for Rotation and Translation 43
3.2.1 Overview of the Proposed Algorithm 44
3.2.2 Compound Omnidirectional Vision Sensor 45
3.2.3 Classification of Near and Far Features 46
3.2.4 Separate Estimation of Rotation and Translation 49
3.2.5 RANSAC-based Methods to Estimate Rotation and Translation 52
3.2.6 Optimizing the Solution Using Epipolar Constraint and All Supporting Pairs 54
3.3 Motion Estimation Experiments 55
3.3.1 Error Definitions 57
3.3.2 Experiments with Different Ratios of Near/Far Features . . . 57
3.3.3 Overall Performance Experiments 60
3.3.4 Discussion 62
3.4 Chapter Conclusion 63

4 Conclusions and Future Works 65

Bibliography 67

Vita 75

List of Figures

1.1 Different situations of the vision data. Vision data may contain single structure, multiple structures of the same model, or multiple structures from multiple models and random noise. We should note that the data points for each structure are also contaminated by some small unknown source of noise. 3

1.2 Residual distribution is contributed by different structures. In a), data contains a single noisy structure with random data points. In b), a more complex data contains two structures with different noise levels and random data points. In case c), a very complex data contains different models and structures. For the last data, a threshold to effectively separate inliers/outliers is impossible since the actual inlier and outlier distribution heavily overlap. 4

1.3 Flowchart of RANSAC and its generalized descendant. 4

1.4 Loss functions for RANSAC, LS, MSAC, MLESAC and adaptive-scale robust estimators. 8

1.5 Range image segmentation: the task is to segment all the planes that make the chair on the left. The results of some robust estimators are displayed on the right. Under or over estimation of inliers will cause the undesirable structures. 10

1.6 Estimation error curves in simulation for line fitting with different Gaussian noise levels. The graph shows the statistical relation between the estimation error and the ratio between the inlier scale used and the true inlier scale. The results confirm that a robust estimator should use an inlier scale estimate as close to the true inlier scale as possible. Too small or too large an inlier scale estimate results in less robustness. 10

1.7 The relationship between accuracy and adaptability of robust estimators. 11

2.1 Decomposition of residual density distribution: inlier density distribution and outlier (other) density distributions. The outlier distribution may consist of a distribution of the other structure and a distribution of random outliers. 16
2.2 The general flowchart for the proposed adaptive-scale robust estimators. 16
2.3 Fitting for One Parameter: inlier distribution and outlier distribution are assumed to be separated, therefore we use limited SDM for matching. 17
2.4 Demonstration of finding the inlier bound. Data contains two parallel lines, and the SDM in this case is the Gaussian. The residual histogram is computed given the estimate $\hat{\theta}$, which has two actual modes for the two lines. The inlier scale is obtained by finding the smallest fitting error, and then the inlier bound is computed as $t_{\hat{\theta}} = \kappa\sigma^*_{\hat{\theta}}$. 18
2.5 Fitting for Two Parameters: residual distribution is assumed to consist of inlier distribution, outlier distribution and ground (outlier) distribution. We use unlimited SDM for fitting both inlier distribution parameter and average ground distribution. 19
2.6 Demonstration of finding the σ for inlier distribution. Data contains two parallel lines similar to that in Fig.2.4, and the SDM in this case is the Gaussian. The residual histogram, ***a)***, is computed given the estimate $\hat{\theta}$, which has two actual modes for the two lines. The inlier scale is obtained by finding the smallest fitting error, ***b)***. The corresponding μ and h for the best σ, $\sigma^*_{\hat{\theta}}$, are given in ***c)*** and ***d)***, respectively. 19
2.7 Demonstration of finding scale estimate for adaptive bin-width computation through various situations of residual distribution. 23
2.8 Standardized Residual Distribution Model (SDM) of fundamental matrix estimation and line fitting problem with Gaussian noise on data points. 27
2.9 Experiments with varying outlier rates for single-line data: (a) estimation error, (b) ratio between the scale of the estimated inlier residuals and the scale of residuals of true inliers. 31
2.10 Example of inlier detection for robust estimators. 31
2.11 Random data sets with an outlier rate of 60% and (a) $\sigma_G = 8.0$ and (b) $\sigma_G = 50$. 32
2.12 Experiments with varying Gaussian noise scales and outlier rate fixed at 60%. Proposed estimator is highly resistant to high noise levels. . . 32
2.13 Parallel Lines: estimation by each estimator using a random data set with d=70. ALKS and the pbM are confused, since the two lines are extracted as one. ASKC and ASSC extract a small part of the actual line. LMedS estimates a line with a large number of inliers belonging to *Line* 1 and a few inliers belonging to *Line* 2. The proposed methods (FITSAC1 and FITSAC2) extract one of the two lines correctly and neatly. 33

2.14 Parallel Lines: (a) estimation error, (b) ratio between the number of estimated inliers and the number of true inliers. 33
2.15 An example of a step data set with four planes and various random points. Each point on a plane is contaminated by Gaussian noise with $\sigma_G = 5.0$. 34
2.16 Results from using data consisting of steps with different Gaussian noise levels σ_G = 1,2,...,10. a) shows the average estimation error, while b) shows the ratio between the estimated number of inliers and number of true inliers. 34
2.17 An example of the roof data set consisting of two planes and various random points. Each point on a plane is contaminated by Gaussian noise with $\sigma_G = 13.0$. 35
2.18 Estimation error using roof data with different Gaussian noise levels σ_G = 5,7,...,17. 35
2.19 Range image segmentation demonstrates the ability of robust estimator for segmentation problem. The robust estimator must properly detect the inliers. The over-detection or under-detection results in a undesirable segmentation. 36
2.20 Fundamental matrix estimation in a simulation using the GRAD residual function: (a) estimation error, and (b) ratio between the number of estimated inliers and the number of true inliers with various outlier rates. 37
2.21 A pair of images in a sequence: inliers (image features in red) and outliers (image features in green) are output by the proposed estimator. 38
2.22 Visualization of the estimated inlier bounds for estimators using the GRAD residual function with three videos sequences (from left to right) $Video\_4deg$, $Video\_14deg$ and $Video\_18deg$. An average of the histograms of residuals from the estimated solutions was made for each estimator to visualize how tightly the estimated inlier bound fits the residual distribution. 40
2.23 Processing time for all estimators 41

3.1 Algorithm flowchart: Detected features are classified as near or far features; rotation is estimated using the far features while translation is estimated using the near features; finally, optimization is performed to tune the estimated motion parameters. 44
3.2 An example of a compound omnidirectional sensor: a single orthographic camera with seven compound paraboloidal mirrors. 45
3.3 The ray directions reflected on one pair of compound paraboloidal mirrors (i and j) and the epipolar line. 47
3.4 Projection of ray directions and epipolar lines on the image plane for one pair of compound mirror i and j. 47

3.5 The disparity δ is computed using a gradient-based method after smoothing with a mean filter. Two intensity functions along the epipolar lines p_i and p_j are aligned at $p_{\boldsymbol{x_i}}$ and $p_{\boldsymbol{x_j}}$, $p_{\boldsymbol{x_i}} = p_{\boldsymbol{x_j}}$. 48
3.6 Camera coordinate system with origin at the center of central mirror. 49
3.7 Estimate rotation from motion of 2 points. 50
3.8 O and O' are located on the intersection of two epipolar planes. . . . 51
3.9 Number of required iterations for the seven-point algorithm without feature correspondence and the proposed algorithm. 54
3.10 The evaluation system. Rotary stages and the vision sensor are mounted on the translation stage. 55
3.11 Example input images (each of them is the big omnidirectional image at center of a sensor image) of FAR, MID, $NEAR$ (from left to right) with increasing near/far ratios. 57
3.12 Comparison of the convergence of estimating rotation with/without feature classification. 59
3.13 Comparison of the convergence of translation estimation with/without feature classification using ground-truth rotation. 59
3.14 Indoor scene 1. 60
3.15 Indoor scene 2. 60
3.16 Indoor scene 1: histogram of feature distances. 60
3.17 Indoor scene 2: histogram of feature distances. 60
3.18 Indoor scene 1: Frobenius error for rotation estimation. 60
3.19 Indoor scene 1: directional translation error for translation estimation. 60
3.20 Indoor scene 2: Frobenius error for rotation estimation. 61
3.21 Indoor scene 2: directional translation error for translation estimation. 61
3.22 Outdoor scene. 61
3.23 Outdoor scene: histogram of feature distances. 61
3.24 Outdoor scene: Frobenius error for rotation estimation. 61
3.25 Outdoor scene: directional translation error for translation estimation. 61

List of Tables

2.1 Summary of two estimators, FITSAC1 and FITSAC2. 28
2.2 Fundamental matrix estimation for real video sequences using GRAD residual function: estimation error. 39
2.3 Fundamental matrix estimation for real video sequences using GRAD residual function: number of estimated inliers. 39

3.1 Parameters of the compound sensor after calibration. 56
3.2 Results of misclassification of near/far features. 58

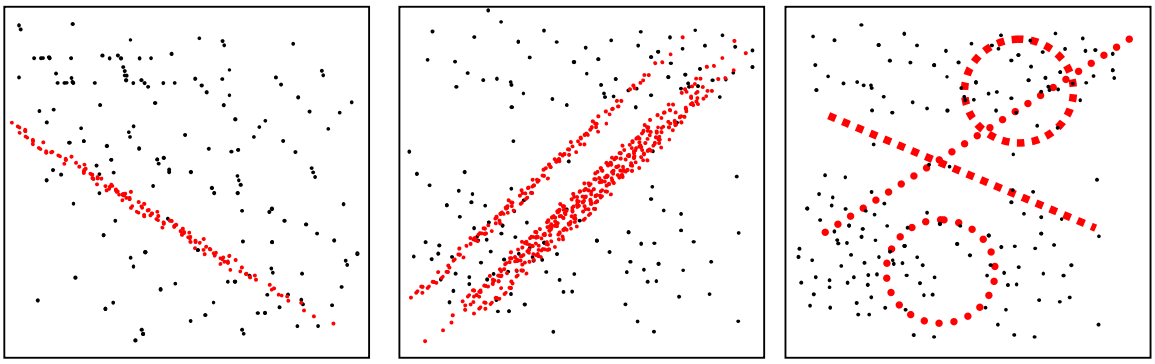
Chapter 1

Introduction

Vision supplies us plenty of information and various computer vision algorithms have been being proposed to exploit. We can now extract the geometric, photometric, semantic information and so on from all types of image such as intensity, infrared or range image. The applications of computer vision may include the extracting the geometric primitives from images like lines, planes, surfaces of known mathematic model [1], multiple view image transformation [3], motion estimation of the camera that is attached to a robot or an autonomous vehicle [3, 28], camera parameter calibration [2], image recognition [4], searching for the existance of a pattern in the image database, and in varieties of problems where the model that describes the mathematical formulation is known but the parameters of the model are unknown.

Vision data like intensity image or range image always contains large number of pixels and each of them is captured with unknown amount of noise caused by the sensor. Therefore, the computer vision algorithms usually have to work with the heavily over-constrained problem. Vision data usually contains several structures of the same model such as different lines of same 2D line model or different circles of the same 2D circle model. Moreover, multiple models may appear at the same time. The different situations of vision data are illustrated in Fig.1.1.

Therefore, in most computer vision problems, to extract an interested structure, the parameter estimation methods must distinguish the data points of this structure from other structures and other models. RANSAC [12] that was proposed by M.A.



Single structure. | Multiple structures with different noise levels of the same model. | Multiple structures of multiple models.

Figure 1.1: Different situations of the vision data. Vision data may contain single structure, multiple structures of the same model, or multiple structures from multiple models and random noise. We should note that the data points for each structure are also contaminated by some small unknown source of noise.

Fischler and R.C. Bolles has been proven the most robust and effective method for such problems. Although, in computer vision, Hough transform or randomized Hough transform [43, 44] are also very robust estimators, they are only efficient for low dimensional parameter vector estimation. For a high dimensional parameter vector, it is is inefficient to manage the huge voting space.

In the following sections, we briefly describe the estimation problem using RANSAC, then about the improvements of RANSAC as the related works and their issues. Then we propose our solutions and state the contributions.

1.1 Problem Statement

The estimation problem for a robust estimator is stated as follows. We have a data set (such as one in Fig.1.1), which contains the noisy data points of one or many interested structures, the model of which is known (such model as line in Fig.1.1). The data also contains the uninterested data points with unknown model. A robust estimator must extract a structure with supporting data points, which are called *inliers*, its parameter vector $\boldsymbol{\theta}$ (a line parameter vector in Fig.1.2, for example), and

discard all the other points, which are called *outliers*. An error is defined for each data point and it is called *residual*. The distribution of the residuals from inliers is called *inlier distribution* and *outlier distribution* for the outlier residuals. The standard deviation of the inlier residuals is called *inlier residual scale* or *inlier scale*. Since the inlier distribution depends on the definition of residual, in our research we call its model a *case-dependent distribution model*. In the real estimation problem, the difficulty is to discriminate the inliers and outliers since we only can compute the total distribution of all residuals. The examples for residual distribution decomposition, assuming the ground-truth inliers, outliers being known, are shown in Fig.1.2.

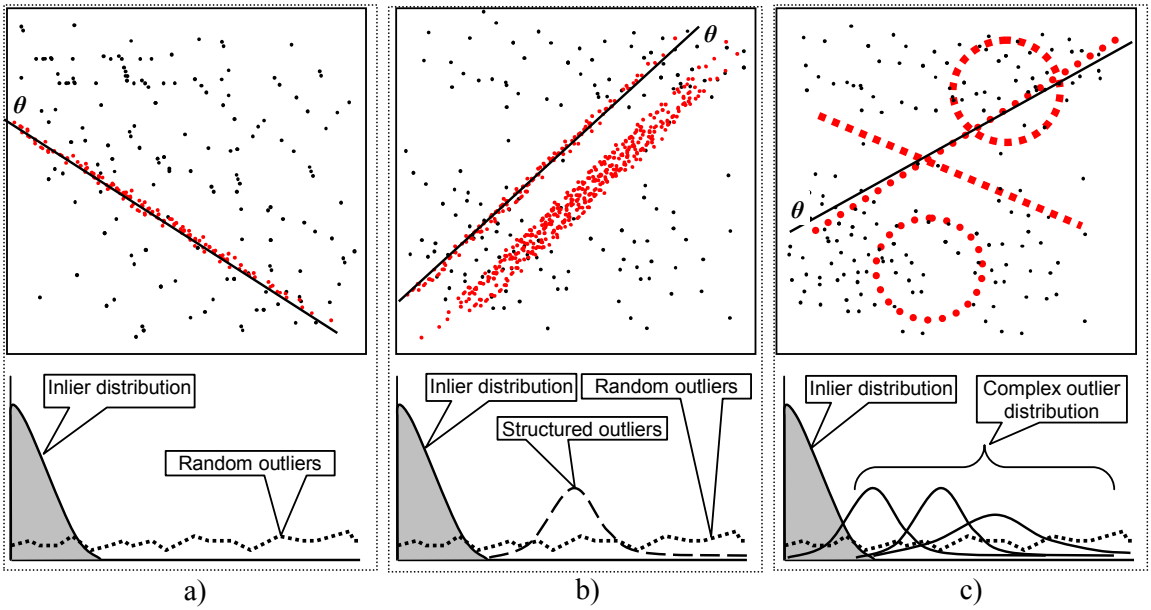


Figure 1.2: Residual distribution is contributed by different structures. In a), data contains a single noisy structure with random data points. In b), a more complex data contains two structures with different noise levels and random data points. In case c), a very complex data contains different models and structures. For the last data, a threshold to effectively separate inliers/outliers is impossible since the actual inlier and outlier distribution heavily overlap.

Having computed all the residuals, RANSAC requires the information about the inlier scale to discriminate the inliers. The same procedure is repeated several times, and the output of RANSAC is the parameter vector $\boldsymbol{\theta}^*$ that have the maximum number of inliers. The framework of RANSAC, which is summarized on the left of

Fig.1.3, is simple and open for varieties of improvements. These improvements of RANSAC are described in the following sections.

1.2 Related Works

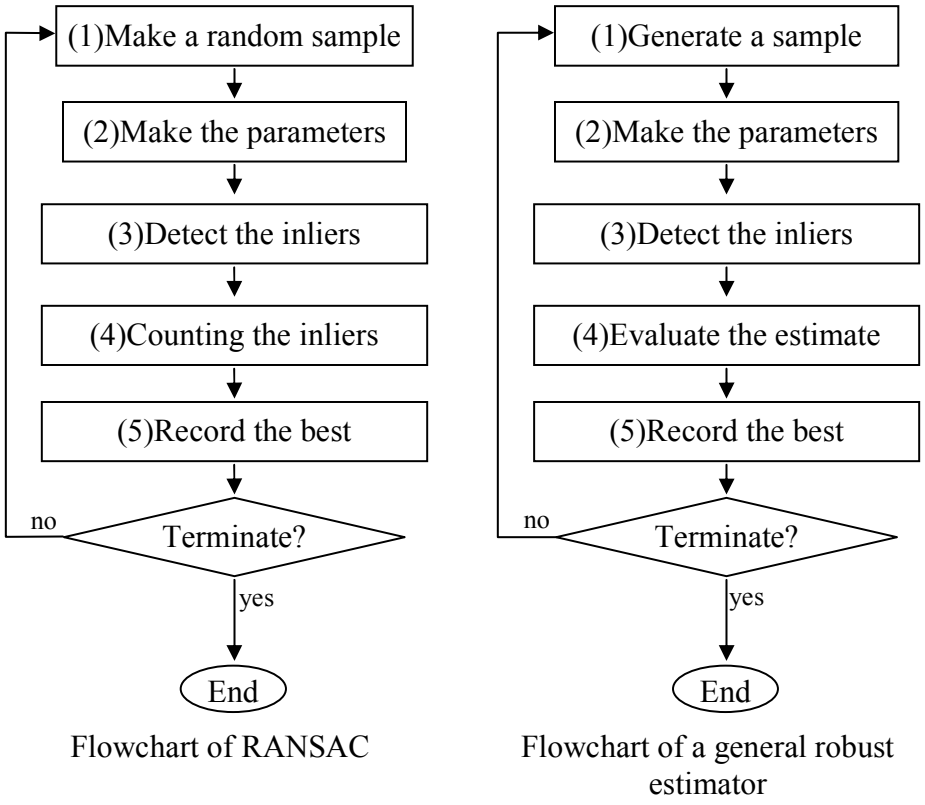


Figure 1.3: Flowchart of RANSAC and its generalized descendant.

In this section, the full image about the RANSAC family is described in order to give readers the whole context where our research works are addressed. The simple flowchart of RANSAC for estimating the parameter is described on the left of Fig.1.3. The RANSAC descendants just follow the framework to refine and improve each step for the different purposes. The general flowchart of these robust estimators is described on the right of Fig.1.3.

These improvements can be classified into several criteria relying on each of the original steps and setup of the estimator: improvements in sampling strategy (for

computational speed), in scoring function (for the accuracy of hypothesis evaluation), in adaptability to detect inliers or the ability to work independently. The other useful reviews, evaluation and comparison on robust estimators can be found in [9, 47, 48, 49, 50, 51]

1.2.1 Improvements in Computational Speed

Due to the strategy of random consensus, an estimator must try a large number of random samples to have the estimation conversed, the minimum number of required samples can be computed [28]. Therefore, a large number of these samples come from outliers that consequently produce the wrong estimates. There are many research works to improve the sampling strategy. These improvements in the sampling strategy is made in the Step (1) of flowchart in Fig.1.3.

A series of randomized RANSAC (R-RANSAC) which improve the execution time of standard RANSAC by adding verification for each random sample. The core idea of these improvements [13, 14, 15] is that most model hypotheses evaluated are influenced by outliers. For such erroneous hypotheses, it is sufficient to test only a small number of data points.

Unlike RANSAC, which treats all correspondences equally and draws random samples uniformly from the full set, some other research works try to compute the priority of each data point and perform the random sampling relying on that priority. In the guided maximum likelihood estimation sample consensus (guided MLESAC) [18], the likelihood of each data point to be an inlier is used to guide the sampling. The data point with higher likelihood has higher priority in the sampling. In this work, the likelihood is computed from the feature matching score such as normalized cross-correlation between feature patches. In progressive RANSAC (PROSAC), samples are drawn progressively from sets of top-ranked correspondences. Experiments demonstrate that it is often significantly faster (up to more than hundred times) than RANSAC.

Different from randomized RANSAC or progressive RANSAC, the preemptive RANSAC [20] tries to compare all the generated hypotheses to find the best remaining

one. The idea is to avoid excessive scoring of useless hypotheses generated by random sampling. A hierarchical refinement scheme is applied to prune the bad hypotheses. The best hypothesis is remained through the entire test for all hierarchical levels. This method is suitable in real-time application when the processing time is fixed.

Besides the random search of the standard RANSAC, some algorithms [32, 23, 24, 25] apply the strategic random search by using a genetic algorithm [76]. This strategic search also improves significantly the execution time of the RANSAC.

1.2.2 Improvements in Adaptability

The adaptability we mention here is firstly about the ability to detect the inliers against various situation such as different outlier rates and problems with different inlier distribution models, these improvements are addressed in the Step (3). Secondly, it is about the ability of a robust estimator to work without the setup by user.

RANSAC is simple but powerful, however it requires the user to supply threshold to distinguish the outliers. Then it is fixed, which restrains RANSAC from working in a dynamic situation or in multi-structural data where the threshold should be adaptively changed due to the distribution of inliers. Several robust estimators also based on random sample consensus have been proposed to avoid the need for the prior knowledge about the inliers. In these estimators, the information about inliers is driven by the statistics of data points.

These robust estimators can be classified into two groups according to whether they make use of residual density estimation. The first group consists of estimators [29][30][31] that search the sorted residuals for the boundary between inliers and outliers without using density estimation. They distinguish the inliers and outliers relying on the statistical relationship between inliers and outliers. On the other hand, the estimators [32, 33, 35, 34, 36] in the second group detect the inliers using the residual density estimation. In these estimators, inlier distribution is detected by mean shift algorithm[45, 46], or by seeking the tail of Gaussian distribution that has low density [32]. A problem for the first group is that the estimators are sensitive to small pseudo-structures in the data and are less robust in real applications. The

methods used in the second group apply a smoothing parameter to estimate the residual density, and they are therefore not so sensitive to small structures, but instead they have to deal with the well-known problem of density smoothness. Since the smoothing parameter for the density estimation is computed before the inlier scale is known, the density of the residual is usually over- or under-smoothed, which results in a corresponding over- or under-estimate of the proportion of inliers. The other reason that the robust estimators do not detect correct inliers is that they [32] assume to work with a Gaussian distribution of residuals. However, in the actual residual distribution may differ from Gaussian distribution, which results in the over or under-detection of inliers.

Some adaptive-scale estimators do not require the prior knowledge of inliers but they need users to tune some other parameter to help them work well [35, 36, 32, 37]. Eliminating the user-defined parameters is one of the directions for RANSAC research to get a fully adaptive-scale estimator. There has been existing a trade-off between the robustness and the user-independency. Some robust estimators that work fully user-independently however the robustness is not very high such as least of median squares (LMedS) [11] and adaptive least k-order squares (ALKS) [30], minimize the probability of randomness (MINPRAN) [31], minimum unbiased scale estimator (MUSE) [29]. Some other robust estimators such as adaptive scale sample consensus (ASSC) [35], adaptive scale kernel consensus (ASKC) [36], pbM [33, 38] that rely on a smoothing parameter to compute the residual (error) probability density, however their performance is very robust.

Our proposed robust estimators are located in this category.

1.2.3 Improvements in Hypothesis Evaluation Accuracy

The accuracy of a robust estimator rely firstly on how well it detects the inliers in Step (3), as mentioned above for the adaptability, and secondly on how well it evaluates the given putative hypothesis with given detected inliers. These improvements are addressed in the Step (4).

In general, a robust estimator searches for a hypothesis $\hat{\theta}^*$ to minimize the objec-

tive function based on the sum of loss for all data points:

$$\hat{\theta}^* = \underset{\hat{\theta}}{argmin} \sum_{i=1}^{n} \rho(r^i_{\hat{\theta}}), \tag{1.1}$$

where $\hat{\theta}$ is a putative hypothesis, $r^i_{\hat{\theta}}$ is an error computed for data point i and n is number of data points. The score for each hypothesis depends on how the loss function evaluates the errors in step 3.

There are varieties of loss functions to be used have been used in robust estimators so far. In the original RANSAC, the loss function for each data point's error, r, is a simple function:

$$\rho_{RANSAC}(r) = \begin{cases} 0 & \text{if } |r| < t \\ 1 & \text{otherwise.} \end{cases} \tag{1.2}$$

where t is user-defined threshold. In the simple least squares method (LS) [9], the impact of each data point is just its square of error. It was later improved by the more complex nonlinear loss function instead of square of error, such as in M-estimator, L-estimator, R-estimator and MSAC [9, 10, 27]. In these estimators, the loss function are carefully designed to alleviate the small error and intensify the large error. For example, the loss function of MSAC [27] is:

$$\rho_{MSAC}(r) = \begin{cases} r^2 & \text{if } |r| < t \\ t^2 & \text{otherwise.} \end{cases} \tag{1.3}$$

Another improvement in the loss function is based on the maximum likelihood as proposed in MLESAC [16] and Guided-MLESAC [18]:

$$\rho_{MLESAC}(r) = -\log\{(\frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2})\lambda + k(1-\lambda)\}, \tag{1.4}$$

where σ is user-defined deviation of Gaussian error distribution, k is some constant and λ is mixing parameter that is tuned for the maximum likelihood. The difference of loss function for MLESAC from those of RANSAC, LS and MSAC is that the loss value can change for same error r to get the maximum likelihood. The difference can

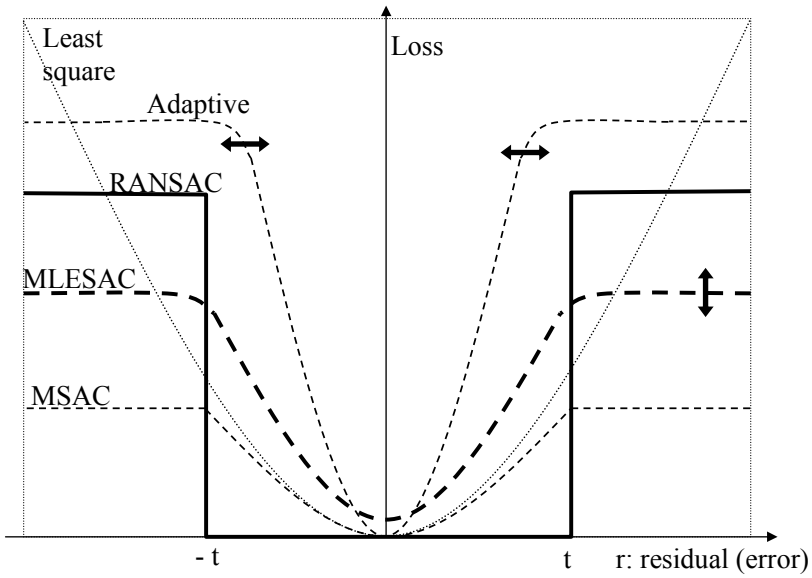


Figure 1.4: Loss functions for RANSAC, LS, MSAC, MLESAC and adaptive-scale robust estimators.

be described in Fig.1.4.

However, all these above loss functions require some prior knowledge about the inliers, such as the threshold t of RANSAC, MSAC, and error deviation of inliers, σ, for MLESAC.

In the adaptive-scale robust estimators, inliers are detected adaptively and the error deviation of inliers σ or the threshold t can be adaptively estimated. Obviously, they can apply these above loss functions to evaluate the hypothesis, however, the latest robust estimators [36, 33, 37, 38] tend to apply the kernel density estimation function. In these objective functions, the loss function can be:

$$\rho_{Adaptive}(r) = -\frac{1}{h_{\hat{\theta}}} K(\frac{r}{h_{\hat{\theta}}}), \tag{1.5}$$

where $h_{\hat{\theta}}$ is a window, the size of which is related to the detected inliers, such as the deviation of inlier residuals and K is the kernel for computing the density, such as Gaussian kernel or Epanechnikov kernel [74]. This loss value for each data point is smaller if $h_{\hat{\theta}}$ is smaller, which means the detected inlier distribution is denser. The adaptive loss function is also plotted in Fig.1.4, where the shape of it changes

adaptively to fit the inlier distribution.

The other research works focus on the improvement of the quality of the putative hypothesis, this improvement is called the local optimization. These research works are made after the model parameters are estimated from Step (2). The robust estimators [21, 22, 33] try to perform the local optimization which is nested in RANSAC to improve the quality of the estimated parameters also the number of inliers of the temporally recorded best estimate. The idea of this improvement is that the estimate from minimal sample is always the approximation of the ground-truth solution.

1.3 Motivations of the Research in This Thesis

From the above surveying about the robust estimators, we have found some issues of the current robust estimators about the adaptability. The first issue of the current robust estimators is that they do not precisely detect inliers (over- or under-detect the inliers), although the current adaptive-scale robust estimators [30, 32, 39, 31, 36, 38] can work automatically without prior knowledge about the inliers. The reasons are either the sensitivity to small pseudo-structures, the smoothness issue of residual density computation or the assumption of a Gaussian distribution, as pointed out in Section 1.2.2. In our observation and experience in robust estimation in computer vision, we understand that it is important that an robust estimator distinguish inliers and outliers correctly for two reasons. The first reason is that in many applications, such as range image segmentation [32, 5, 30, 40, 6], motion segmentation [7, 73, 8], the inliers of a structure should be detected correctly, otherwise the structure is divided into several smaller undesirable structures or united with the other structure, this is illustrated in Fig.1.5. The second reason is that if the robust estimator over- or under-detects the inliers, then the robustness is also reduced. This comes from the fact that the robustness of the estimator depends on the size of the proportion of inliers that can be detected. If the estimator detects only a few inliers or too many inliers, the robustness of the estimator is reduced, as shown in the simulation in Fig.1.6.

Understanding this issue, in the thesis we propose several adaptive-scale robust estimators which improve the adaptability (to robustly detect inlier and to be user-

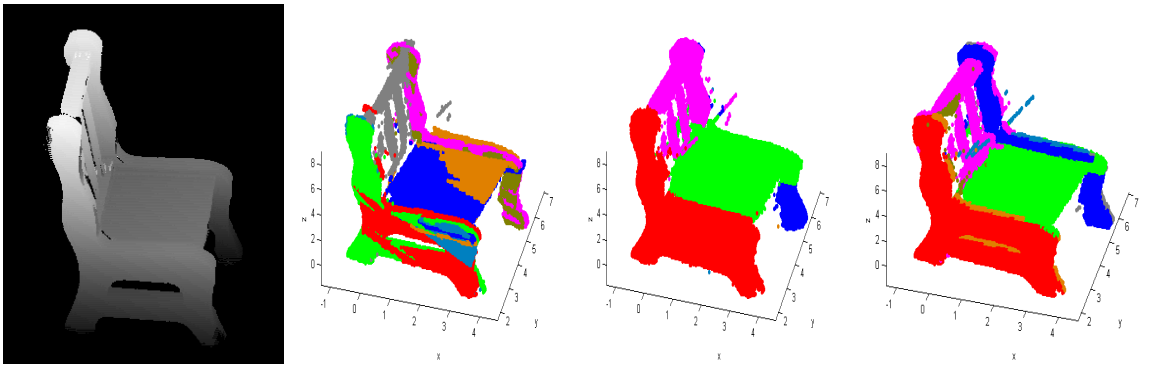


Figure 1.5: Range image segmentation: the task is to segment all the planes that make the chair on the left. The results of some robust estimators are displayed on the right. Under or over estimation of inliers will cause the undesirable structures.

independent). Different from previous adaptive-scale estimators, we focus on the accuracy of inlier detection to improve the adaptability and robustness of the estimators. We also compute the distribution of residuals using some smoothing parameter. However, since the smoothness of the density estimation is inevitable, we do not detect the inliers directly from the roughly estimated density. Instead, we apply a matching method to detect inliers by globally searching the estimated density of the residual to find the most likely inlier distribution. The inlier distribution is assumed to be known or it can be modeled using a case-dependent but known constraint, the residual function, which importantly constrains the inlier distribution. This has not been used in any previous works. To compute the residual density, we use histogram and the bin-width can be assigned manually or adaptively. We also propose an solution to eliminate all the user-defined parameter in one of the proposed estimators. The advantage of these proposed estimators is that they adaptively fit the inlier residual distribution well so that the robustness and accuracy is improved and they work well for multi-structural data.

We plot the intuitive relationship of the robust estimators compared to the proposed method in Fig.1.7.

The second issue of current robust estimators is about the data that contains multiple structures from multiple models. In such situation, a robust estimator can only

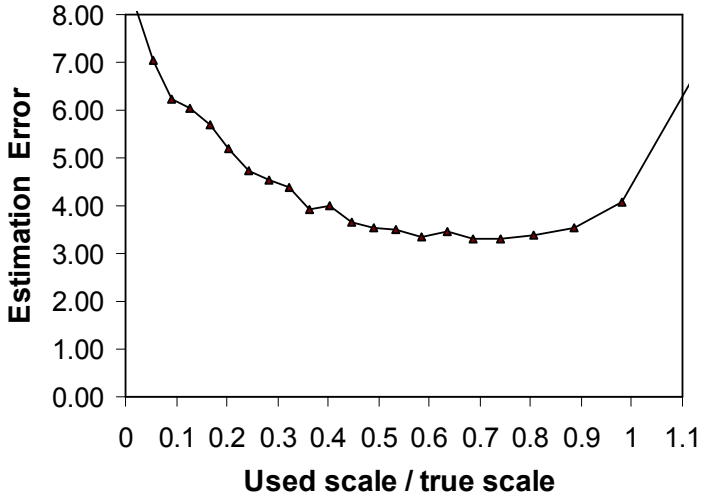


Figure 1.6: Estimation error curves in simulation for line fitting with different Gaussian noise levels. The graph shows the statistical relation between the estimation error and the ratio between the inlier scale used and the true inlier scale. The results confirm that a robust estimator should use an inlier scale estimate as close to the true inlier scale as possible. Too small or too large an inlier scale estimate results in less robustness.

extract one structure at each execution, then the data points for all the other structures become outliers. This makes a extremely high outlier rate estimation problem, and reduces the efficiency of the robust parameter estimation problem. Moreover, the inlier distribution and outlier distribution of some structure may not be separated, which is demonstrated in Fig.1.2.c, the inlier distribution and outlier distribution may be overlapped heavily. In such case, the definition of inlier scale, inlier threshold can not be used to distinguish inliers and outliers. However, if we consider about structured outliers, we can distinguish them. In searching for the solution, we have not found any solution for this problem in the literatures. Therefore, the second goal of the thesis is about an efficient method for detecting outliers to reduce the workload and improve the efficiency for robust estimators. We have thought about the model-based pre-filtering technique to separate the data from uninterested models. In this thesis, we demonstrate an efficient solution for camera egomotion estimation.

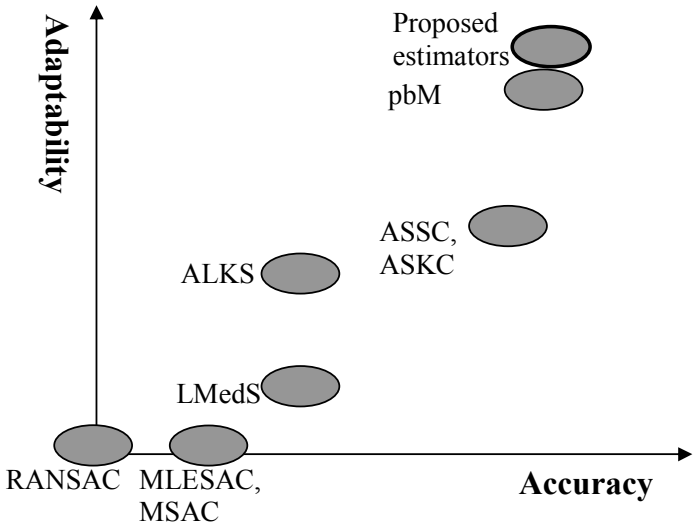


Figure 1.7: The relationship between accuracy and adaptability of robust estimators.

1.4 Contributions

The first contribution is about the inlier detection method. We proposed several algorithms, which are described in Section 2.2, for outlier detection to apply for robust estimators: an algorithm to estimate inlier distribution by matching with the distribution model to find the inliers and an algorithm to search for inliers by fitting with both inlier distribution model and the ground distribution of outliers. The corresponding robust estimators (FITSAC1, FITSAC2) have the advantage that they can adaptively, correctly detect the inliers and do not rely much on the smoothness of the density estimation like previous works. The advantage of the second proposed estimator compared to the first one is that it works fully adaptively, all its parameter setup is data driven, meanwhile the smoothness of density estimation in the first is manually set, like most previous works.

Bin-width (smoothing parameter) for computing residual histogram (or probability density distribution by kernel density estimation (KDE) method) is always a serious problem for robust estimators that rely on the density estimation technique. In this thesis, we also propose an algorithm to estimate an adaptive bin-width (adaptive smoothing parameter), which is described in Section 2.3.2, for robust estimators.

It is effectively used in FITSAC2.

In previous works, theoretically, most robust estimators assume that the inlier distribution is normally distributed. However in practice, they are possible to work with any unimodal distribution. Since they assume to work with a Gaussian distribution, therefore, they do not detect the inliers accurately in case the actual distribution is not Gaussian. In this thesis, we tackle to this problem by carefully analyzing the distribution, which is presented in Section 2.4, and hence the proposed robust estimators can detect inliers correctly.

We demonstrate the idea of pre-filtering the outliers in the camera egomotion estimation, which is described in Section 3.2. We propose a new egomotion estimation algorithm for a special type of stereo camera.

1.5 Structure of the Thesis

The thesis is organized as follows. Firstly, In the Chapter 2, the research work about the robust estimators will be presented. Here, we describe about all the constituents of our robust estimators. A novel method for inlier detection by distribution model fitting is given in Section 2.2. Our method relies on the density computation, then the bin-width selection is given in Section 2.3. The distribution model is discussed in Section 2.4. Then we describe the experimental results for our robust estimators in Section 2.7.

Secondly, the pre-filtering technique will be given in Chapter 3. Here, we demonstrate the idea of pre-filtering by proposing a new egomotion estimation algorithm, which is given in Section 3.2. Then, we show its validation by experiments in Section 3.3.

Finally, Chapter 4 will present our conclusions and future works.

Chapter 2

Adaptive-scale Robust Estimators using Distribution Model Fitting

2.1 Preliminaries

In this section, we describe the estimation problem and some definitions that are used in the thesis.

Assume the estimation of a structure model with the constraint:

$$g(\boldsymbol{\theta}, \boldsymbol{X}) = 0, \tag{2.1}$$

where $\boldsymbol{\theta}$ is the parameter vector of the structure, and $\boldsymbol{X}$ is an explanatory data point. Our estimation problem is then described as:

- *Input*: N observed data points $\boldsymbol{X_i}, i = 1..N$, including both inliers and outliers.
- *Output*: Parameter $\boldsymbol{\theta}$ that describes the data.

In a real problem, each inlier $\boldsymbol{X^t}$ is affected by an unknown amount of noise $\boldsymbol{n}$:

$$\boldsymbol{X} \;\; = \;\; \boldsymbol{X^t} + \boldsymbol{n}. \tag{2.2}$$

Therefore, the actual parameters $\boldsymbol{\theta}$ cannot be recovered, and some approximation of $\boldsymbol{\theta}$ needs to be estimated. A robust estimator based on random sampling like RANSAC

solves the problem by trying many random trial estimates $\hat{\boldsymbol{\theta}}$, with the best estimate $\hat{\boldsymbol{\theta}}^*$ being the approximation of $\boldsymbol{\theta}$. In evaluating whether an estimate $\hat{\boldsymbol{\theta}}$ is good or bad, the estimator can only rely on the statistics of the error for each data point; this error is called the residual, which is a non-negative measure in the proposed method. For each model estimation problem, there are numerous ways of defining the residual function, including using the original constraint function (2.1). Generally, however, the residual is defined as:

$$r_{\hat{\theta}} = f(\hat{\boldsymbol{\theta}}, \boldsymbol{X}). \tag{2.3}$$

A good definition of the residual is that proposed by Luong et al. [68]:

$$r_{\hat{\theta}} = \frac{g(\hat{\boldsymbol{\theta}}, \boldsymbol{X})}{\parallel \bigtriangledown g(\hat{\boldsymbol{\theta}}, \boldsymbol{X}) \parallel}, \tag{2.4}$$

where $\bigtriangledown g(\hat{\boldsymbol{\theta}}, \boldsymbol{X})$ is the gradient of g with respect to variable $\boldsymbol{X}$.

In a real problem, the inlier residual is not zero. The standard deviation of these inlier residuals is called the *"inlier scale"*, and is denoted by $\sigma_{\hat{\theta}}$. The problem is that $\sigma_{\hat{\theta}}$ is not known, and therefore, an inlier scale estimator tries to estimate it. This estimate is denoted by $\sigma^*_{\hat{\theta}}$. Once the inlier scale has been found, the threshold $t_{\hat{\theta}} = \tau\sigma^*_{\hat{\theta}}$ can be decided to distinguish inliers from outliers.

Given an estimate $\hat{\boldsymbol{\theta}}$, and an inlier scale $\sigma_{\hat{\theta}}$, the probability density function for all residuals is denoted as $P_{\hat{\theta}}(r)$, which is the sum of density functions for inliers and outliers. The proposed estimator works with data with multiple structures, and therefore the residual distribution may have multiple modes. A segment of the distribution that has a mode near the origin is assumed to belong to the inlier structure, whereas the others belong to the outlier structures. The decomposition of the residual distribution is illustrated in Fig.2.1. The outlier distribution is usually complicated and unpredictable. However, the inlier distribution can be well modeled in most problems. In our method, the inlier distribution model is made using the residual function. The density function for the standardized distribution model (SDM), with the sample deviation of 1, is denoted as $P(\xi)$, $\xi \geq 0$. Then, the inlier distribution is estimated by matching the residual distribution $P_{\hat{\theta}}(r)$ with SDM.

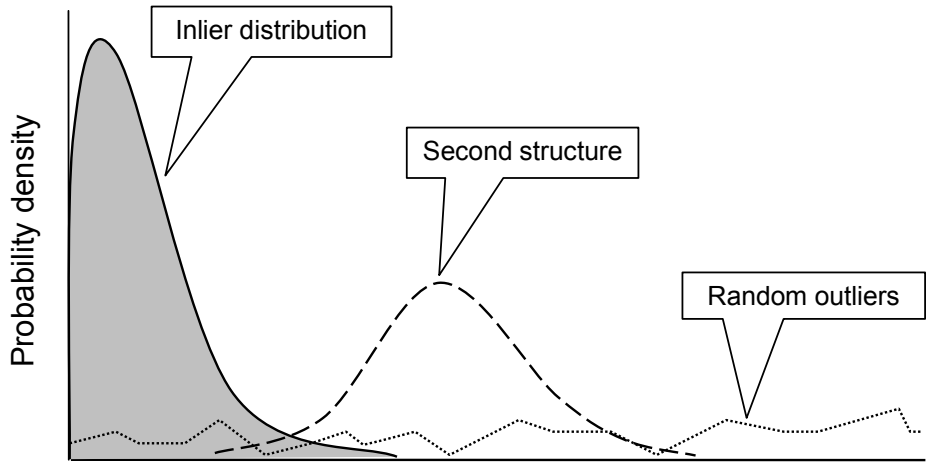


Figure 2.1: Decomposition of residual density distribution: inlier density distribution and outlier (other) density distributions. The outlier distribution may consist of a distribution of the other structure and a distribution of random outliers.

An adaptive-scale robust estimator consists of two constituents [32, 30, 29, 31, 35, 42]: a scale estimator and a hypothesis evaluator. For a given putative hypothesis, the scale estimator has to detect the inliers and estimate the inlier scale. Then the hypothesis evaluator computes the score using a objective function for this detection based on the estimated inlier scale. This is different from that of pbM [33, 37, 38], an adaptive-scale robust estimator in which inliers are detected finally after the solution is outputted.

The general flowchart for the proposed adaptive-scale robust estimators is presented in Fig.2.2, the details are described in the following sections. The inlier residual distribution model, SDM, is described in Section 2.4. A histogram of residuals is computed using the bin-width that is discussed in Section 2.3. Searching for an inlier scale, inlier detection method, is presented in Section 2.2. And finally, the hypothesis is evaluated by an objective function in Section 2.5.

2.2 Inlier Detection Methods

In this section, we describe about the robust methods to detect inliers by fitting the actual residual distribution of putative hypothesis with the SDM to find the inlier

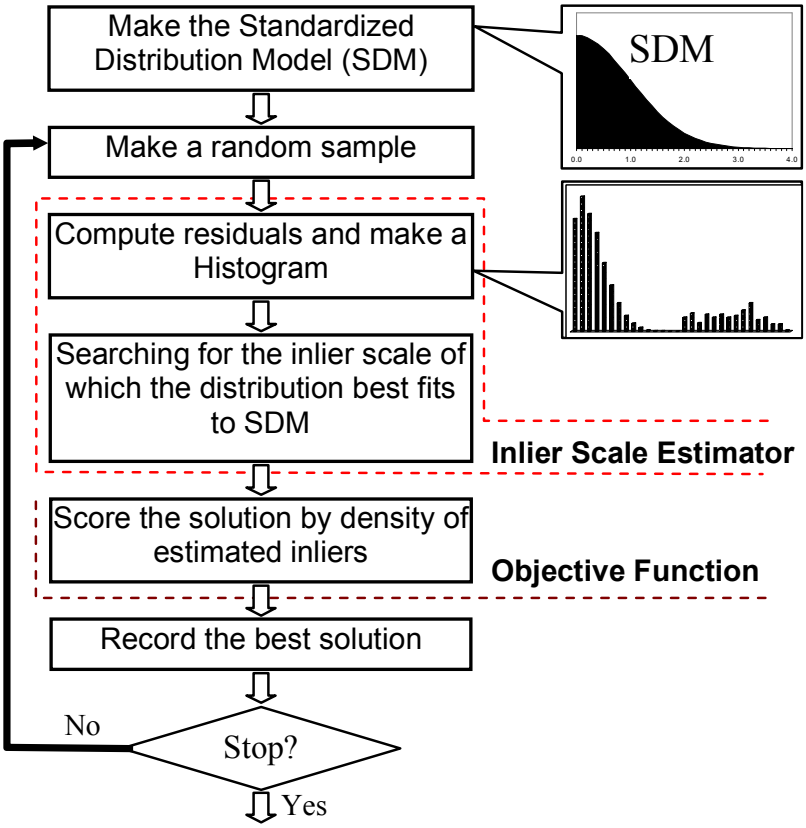


Figure 2.2: The general flowchart for the proposed adaptive-scale robust estimators.

distribution. The first method uses only a dense segment of SDM for fitting. This is fast but it requires some parameter, even thought it is easy to setup in practice. The second method does not limit the SDM and uses fitting the whole SDM. In this method, both inlier distribution parameter (inlier residual deviation) and also the average of ground distribution (additive outlier distribution). Which helps the second method works more robust though it is a bit slower.

2.2.1 Fitting for One Parameter

Since the tail, with low density, of the inlier distribution is usually heavily overlapped with the outlier distribution, we do not use the whole SDM for matching. Only the dense segment of $P(\xi)$ with $0 \leq \xi \leq \kappa$ that contains most of population of SDM is used for matching. κ is selected so that the range $0 \leq \xi \leq \kappa$ contains more

than 97% of the population. For example, when the SDM is the standard Gaussian distribution, we set $\kappa = 2.5$.

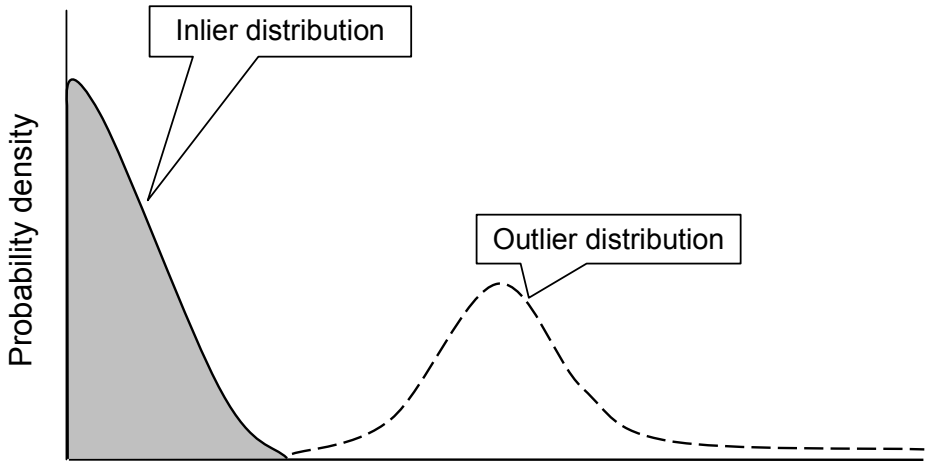


Figure 2.3: Fitting for One Parameter: inlier distribution and outlier distribution are assumed to be separated, therefore we use limited SDM for matching.

The inlier scale is estimated by searching the best fit between a segment of the residual distribution and the SDM. The segment of the residual distribution used for matching starts from zero. Then, the residual scale of the first structure is detected regardless of the outlier structures. The fitting error between the density function $P_{\hat{\theta}}(\rho)$ with assumed inlier scale σ and the SDM density function $P(\frac{\rho}{\sigma})$ is:

$$e_{\hat{\theta}}(\sigma) = \min_{\mu} \int_0^{\kappa\sigma} \Big(P_{\hat{\theta}}(\rho) - \mu P(\frac{\rho}{\sigma}) \Big)^2 \, d\rho, \tag{2.5}$$

where μ is some scale of the SDM density function, ρ is the residual variable and κ indicates the part of the SDM used in the matching as discussed in Section 2.1. The minimization (2.5) with respect to μ is solved when it is assigned:

$$\mu = \frac{\int_0^{\kappa\sigma} P_{\hat{\theta}}(\rho) P(\frac{\rho}{\sigma}) \, d\rho}{\int_0^{\kappa\sigma} P(\frac{\rho}{\sigma})^2 \, d\rho}. \tag{2.6}$$

Then, the best scale of inlier residuals $\sigma^*_{\hat{\theta}}$ is estimated by searching the scale that gives the smallest fitting error. This is summarized as

$$\sigma^*_{\hat{\theta}} = \underset{\sigma}{argmin}\{e_{\hat{\theta}}(\sigma)\}. \tag{2.7}$$

Inliers are then distinguished using the threshold $t_{\hat{\theta}} = \kappa\sigma^*_{\hat{\theta}}$. The inlier scale $\sigma^*_{\hat{\theta}}$ is refined for later use in the objective function, by being replaced by the standard deviation of estimated inliers:

$$\hat{\sigma}^*_{\hat{\theta}} = \sqrt{\int_0^{t_{\hat{\theta}}} \rho^2 P_{\hat{\theta}}(\rho)\, d\rho}, \tag{2.8}$$

In our algorithm, we compute the probability density of the residual from an estimate $\hat{\theta}$ by applying the well-known histogram method, although the KDE can also be used. A histogram is simple and as residual sorting is not required, in contrast to most previous estimators, it can be computed with low computational cost. Then, (2.5) and (2.6) are converted into histogram-based form, ρ is replaced by the bin variable $b_i = ib_{\hat{\theta}}$, which is the location of the i^{th} bin, with $b_{\hat{\theta}}$ the bin-width. The refined inlier scale in (2.8) is replaced by the sample deviation of inlier residuals $r_i \leq t_{\hat{\theta}}$. In addition, $P_{\hat{\theta}}(b_i)$ is the count of residuals belonging to the i^{th} bin. Searching for the best inlier scale $\sigma^*_{\hat{\theta}}$ and $t_{\hat{\theta}}$ is graphically depicted in Fig.2.4.

2.2.2 Fitting for Two Parameters

In the above method, we model the actual residual distribution as the separated inlier distribution and outlier distribution and use limited SDM for the fitting to find the inlier distribution. Even though, this algorithm works quick and quite well against the data with more than 80% of outlier rate in experiments, theoretically, this is true when the outlier distribution does not accommodate within the inlier distribution.

To realize the drawback of the above model for actual residual distribution, we propose another model for residual distribution. Residual distribution is assumed to consist of inlier distribution, outlier distribution and ground (outlier) distribution, which is illustrated in Fig.2.5.

The fitting error between the density function $P_{\hat{\theta}}(\rho)$ with assumed inlier scale σ

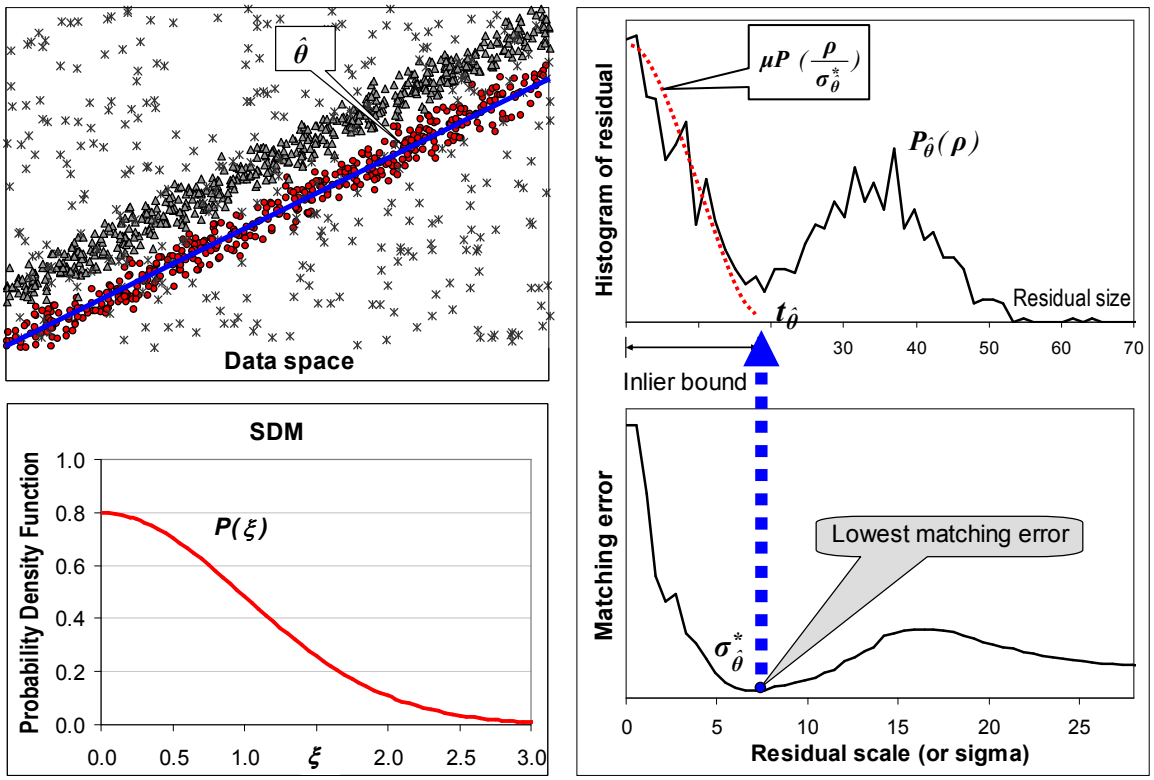


Figure 2.4: Demonstration of finding the inlier bound. Data contains two parallel lines, and the SDM in this case is the Gaussian. The residual histogram is computed given the estimate $\hat{\theta}$, which has two actual modes for the two lines. The inlier scale is obtained by finding the smallest fitting error, and then the inlier bound is computed as $t_{\hat{\theta}} = \kappa\sigma^*_{\hat{\theta}}$.

and the unlimited SDM density function $P(\frac{\rho}{\sigma})$ is:

$$e_{\hat{\theta}}(\sigma) = \min_{\mu,h} \int_0^{+\infty} \left(P_{\hat{\theta}}(\rho) - \mu P(\frac{\rho}{\sigma}) - h \right)^2 d\rho, \tag{2.9}$$

where h is the ground distribution which is added to inlier distribution. Compared to the model fitting error function (2.5), in (2.9), parameter h is to compensate for the high ground distribution. The minimization (2.9) can also be solved straightforwardly:

$$\mu = \frac{\int_0^{+\infty}(ab^2)d\rho \int_0^{+\infty} b d\rho - \int_0^{+\infty}(ab)d\rho \int_0^{+\infty}(b^2)d\rho}{\int_0^{+\infty}(b^3)d\rho \int_0^{+\infty}(b)d\rho - \left(\int_0^{+\infty}(b^2)d\rho\right)^2}, \tag{2.10}$$

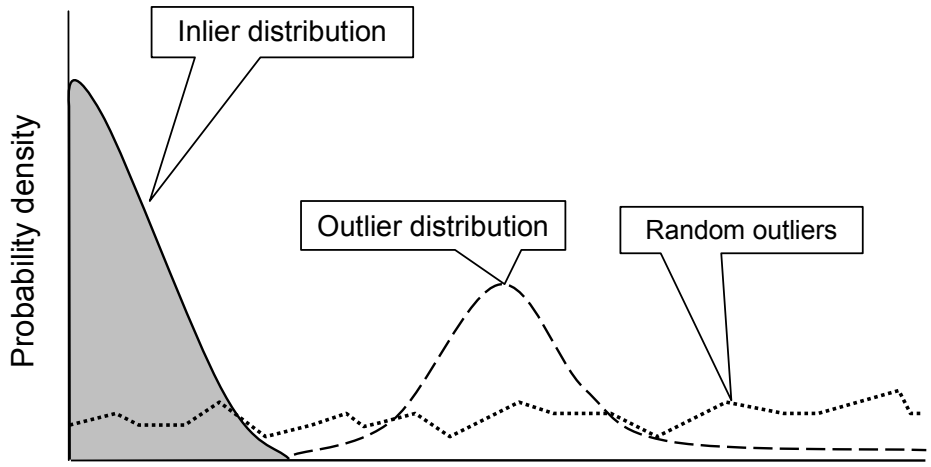


Figure 2.5: Fitting for Two Parameters: residual distribution is assumed to consist of inlier distribution, outlier distribution and ground (outlier) distribution. We use unlimited SDM for fitting both inlier distribution parameter and average ground distribution.

and

$$h = \frac{\int_0^{+\infty}(b^3)d\rho \int_0^{+\infty}(ab)d\rho - \int_0^{+\infty}(ab^2)d\rho \int_0^{+\infty}(b^2)d\rho}{\int_0^{+\infty}(b^3)d\rho \int_0^{+\infty}(b)d\rho - \left(\int_0^{+\infty}(b^2)d\rho\right)^2}, \tag{2.11}$$

where

$$a = P_{\hat{\theta}}(\rho), b = P(\frac{\rho}{\sigma}).$$

Searching for the best inlier scale $\sigma^*_{\hat{\theta}}$ is graphically depicted in Fig.2.6. We can clearly see that, the fitting error function $e_{\hat{\theta}}(\sigma)$ in this method is smoother than that of the previous method, which is promising to produce more robustness in practice.

2.3 Bin-width for Density Estimation

Bin-width (smoothing parameter or bandwidth for KDE) is the size of a bin in the residual histogram mentioned in Section 2.2. In this section, we decide the bin-width to be used in our algorithm. Bin-width (or bandwidth in previous works) affects the smoothness of the density distribution and consequently influences the detection of local peak or valley. Setting the bin-width is usually a difficult problem for those

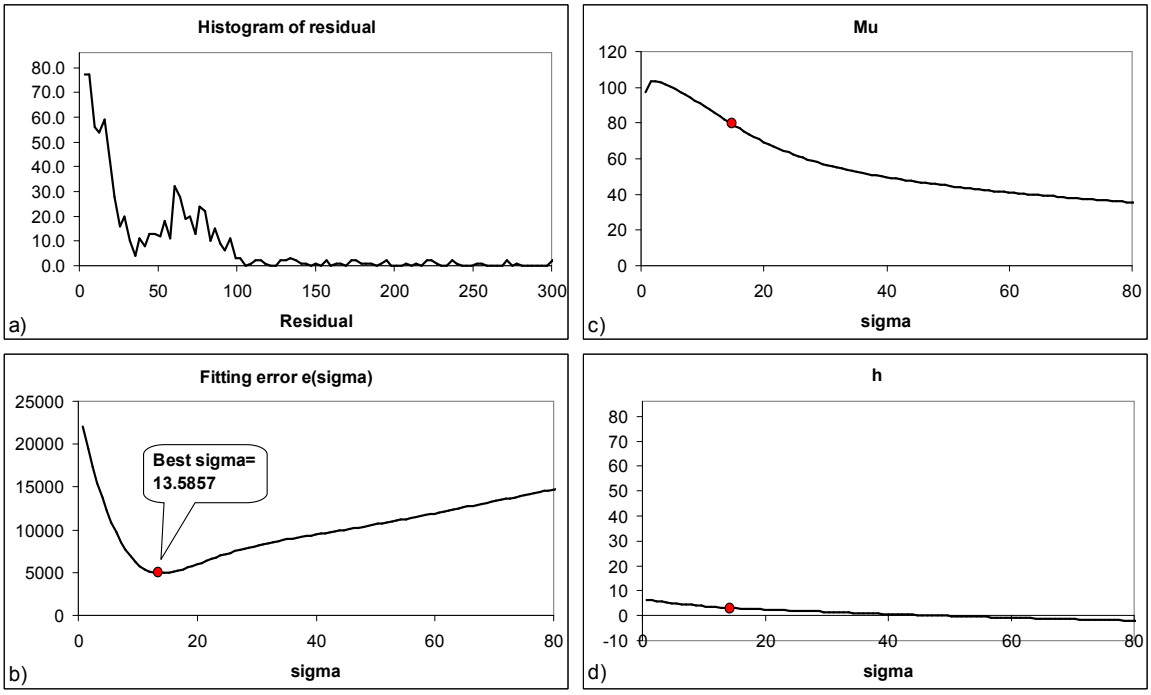


Figure 2.6: Demonstration of finding the σ for inlier distribution. Data contains two parallel lines similar to that in Fig.2.4, and the SDM in this case is the Gaussian. The residual histogram, ***a)***, is computed given the estimate $\hat{\theta}$, which has two actual modes for the two lines. The inlier scale is obtained by finding the smallest fitting error, ***b)***. The corresponding μ and h for the best σ, $\sigma^*_{\hat{\theta}}$, are given in ***c)*** and ***d)***, respectively.

methods that rely on the probability density of residuals.

In this section, we describe about two choices for the smoothing parameter. The first is fixed and is widely used in robust estimators. It needs some experience to produce the smoothness of density estimation. The second smoothing parameter is totally data driven, and consequently it does not require any experience of user for all the situation.

2.3.1 Fixed Bin-width Computation

A bin-width that produces good smoothness of the density estimation is required in such situation, and a widely used bin-width [74] for robust estimators is:

$$b_{\hat{\theta}} = \left(\frac{243 \int_{-1}^{1} K(\zeta)^2 d\zeta}{35N(\int_{-1}^{1} \zeta^2 K(\zeta) d\zeta)^2} \right)^{\frac{1}{5}} \hat{s}_{\hat{\theta}}, \tag{2.12}$$

where K is some kernel, such as the popular Gaussian kernel or the Epanechnikov kernel, $\hat{s}_{\hat{\theta}}$ is some scale estimate, such as the standard deviation of residuals, median scale estimate [9] or MAD estimate [9], and N is the number of data points. In our method, $\hat{s}_{\hat{\theta}}$ is the smallest window containing 15% of the smallest residuals.

2.3.2 Adaptive Bin-width Computation

In the previous Section 2.3.1, we use a fast computation for bin-width similar to most of previous adaptive-scale robust estimators. It requires some experience of user for the smoothness of the computed density. It can not be adaptively changed for a dynamic situation such as when the outlier rate is varied while capturing the data. Therefore, an adaptive bin-width is desirable for a robust estimator. There exist number of previous works for binwidth computation [74, 52, 53, 54, 55], however these methods for computing the density is not suitable since they do not focus only on the inlier distribution.

In searching for a solution, we has become fascinated by the one of previous robust estimators, ALKS (adaptive least k^{th} order squares)[30] which is an improvement of MUSE (minimum unbiased scale estimator) [29].

Given an hypothesis $\hat{\theta}$, all the residual r_i for n data points are computed. In ALKS, all residuals are then sorted increasingly. ALKS proposes an *normalized error* [30]:

$$\epsilon_k^2 = \frac{1}{q_k - p} \sum_{i=1}^{q_k} \left(\frac{r_i}{\hat{s}_k} \right)^2 = \frac{\hat{\sigma}_k^2}{\hat{s}_k^2}, \tag{2.13}$$

where p is the size of random sample for random sampling, $\hat{\sigma}_k^2$ is the standard deviation of first k residuals, $\hat{s}_k$ is unbiased scale estimate [29]:

$$\hat{s}_k = \frac{r_k}{\Phi^{-1}\left(0.5(1 + \frac{k}{n+1})\right)}, \tag{2.14}$$

and $\Phi(u, 1)$ is the cumulative density function for Gaussian distribution. $\Phi^{-1}(.)$ is the argument of the normal cumulative density function having the value inside the bracket, which is used as a compensation factor for (2.13). This compensation factor is just an approximation since in practice the data is contaminate with unknown proportion of outliers. q_k is the number of inliers decided by the threshold $2.5\hat{s}_k$. In our experience, ALKS has an ability to detect the inlier distribution quite well under the complicate multi-structural data. However, the inliers/outliers dichotomy is usually located outside the inlier distribution, which means ALKS usually over-estimates the inliers. Another problem for ALKS is that it needs the minimum size of a meaningful structure, however it is reasonable in practice where the size of data is perceivable. If there exists extreme outliers in the data, $\hat{s}_k$ can be come so large that minimum of the criterion is produced at an incorrect k due to the relative relation between inlier and outlier distributions.

Taking the advantage of the excellent ability to detect the structure, we solve the drawbacks of ALKS and use this algorithm not for a robust estimator but for computing an adaptive bin-width. We replace (2.13) by the following term:

$$\zeta_k^2 = \frac{1}{k-p} \sum_{i=1}^{k} \left(\frac{r_i}{r_k} \right)^2. \tag{2.15}$$

In this normalized error function, the compensation factor is removed for its ineffective performance. From(2.15), we can see that ζ_k is limited by 1: $0 < \zeta_k < 1$. We use this function not to find the inlier but to find most inliers to use in the bin-width computation (2.12). The algorithm is described as follows:

1. Find the location k_{max} of global maximum ζ_{max} of ζ_k

$$k_{max} = \underset{p<k\leq n}{argmax}(\zeta_k).$$

2. Find the min value of ζ_k:

$$\zeta_{min} = \min_{k_{max}<k\leq n} (\zeta_k).$$

3. Find the first k_1, $k_1 > k_{max}$, that produces

$$\zeta_{k_1} = \frac{1}{2}(\zeta_{max} + \zeta_{min}).$$

4. The scale estimate $s_{\hat{\theta}}$ for bin-width computation function (2.12) is set:

$$s_{\hat{\theta}} = r_{k_1}.$$

5. Finaly the adaptive bin-width is computed as follows:

$$b_{\hat{\theta}} = \left(\frac{243 \int_{-1}^{1} K(\zeta)^2 d\zeta}{35N(\int_{-1}^{1} \zeta^2 K(\zeta) d\zeta)^2} \right)^{\frac{1}{5}} r_{k_1}, \tag{2.16}$$

The demonstration of finding the scale estimate $s_{\hat{\theta}}$ for adaptive bin-width computation is described in Fig.2.7:

Having obtained the bin-width, a histogram of the estimate can be built. Since the bin-width is small for outlier residuals, especially in case of high outlier-rates, the number of bins may be large and therefore, large number of bins for outliers should be ignored. For a specific inlier unimodal distribution with deviation σ of N residuals, the bin-width is computed, the densest bin at the mode contains a limited number of residuals, which is many greater than 1. Then, the number of bins for that inlier distribution is limited, which is many less than N even when the distribution becomes a uniform distribution. However, if there are more than two component distributions (an inlier distribution and some outlier distributions), the number of bins may be many greater than N due to the large scale of outlier residuals. In order to search only for the inlier distribution, in practice, we limit the number of bins, for example, by N.

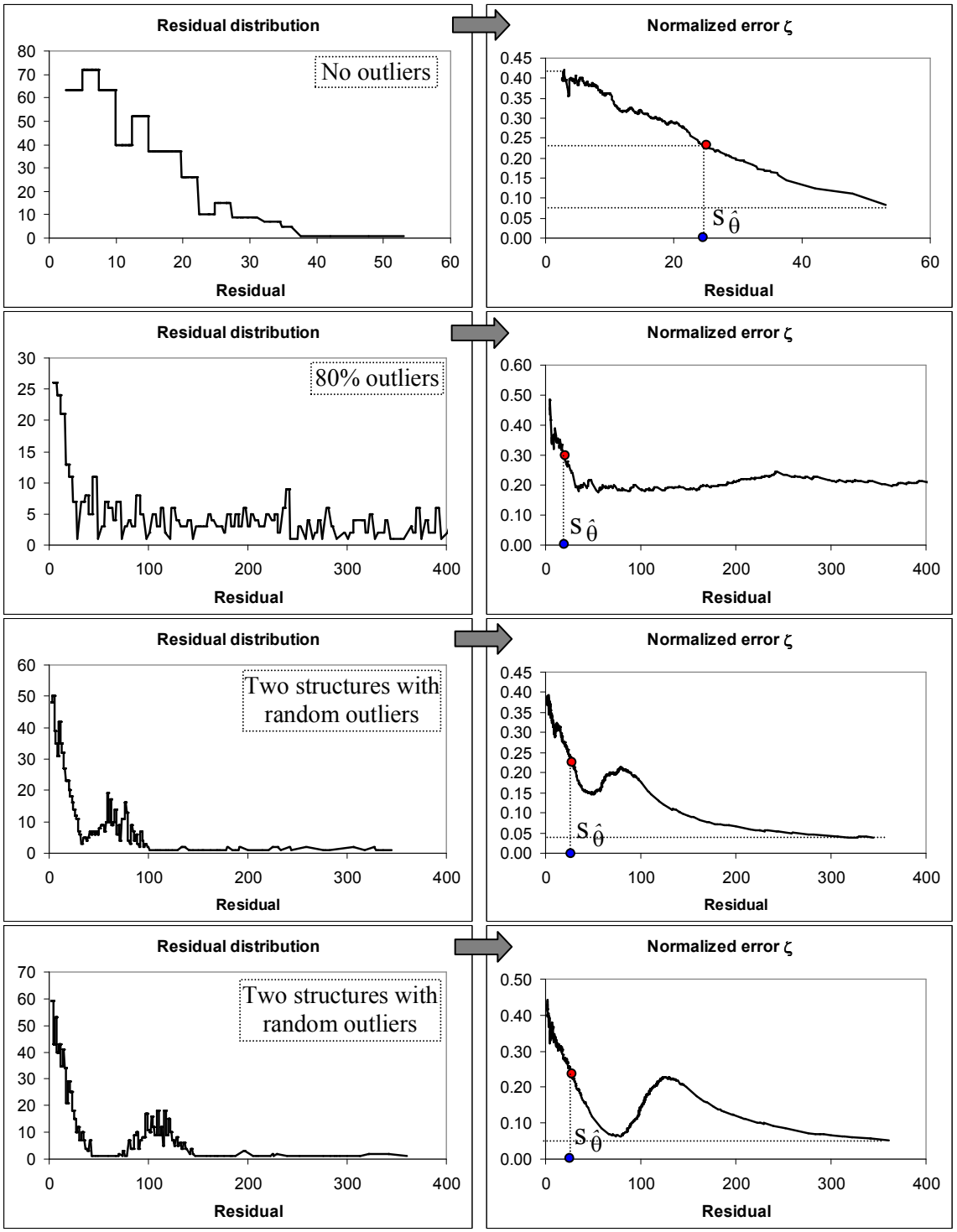


Figure 2.7: Demonstration of finding scale estimate for adaptive bin-width computation through various situations of residual distribution.

2.4 Case-dependent Residual Distribution Analysis

In most previous works [32, 30, 35, 36, 42], the theory of the algorithms is made for the Gaussian distribution of inlier residuals. However, it is not always true in practice. Therefore, we would like to formulate a residual distribution closer to the actual inlier residual distribution, in order to extract the inliers better. Our idea is that the distribution of inlier residuals depends on the residual function.

In this section, we carry out an analysis of the residual distribution for various estimation problems. It is better to assume Gaussian noise on the data points than to assume a Gaussian distribution of residuals. Firstly, this is because the noise on data points originates from physical sensors such as a camera in which noise distribution is usually modeled by a Gaussian distribution. Secondly, it is because the residual distribution is constrained by the residual function. Therefore, we assume that the noise model for the data points is known and it is a Gaussian of unknown variance in this thesis. However, due to the residual function (2.3), the distribution of residuals is generally different from that noise distribution. Then, we analyze the distribution model for residuals. Two examples are presented in this section: line fitting and fundamental matrix estimation.

2.4.1 Linear Residual

We start the analysis with a well-known problem for a robust estimator, the line fitting problem, in which the residual function is a linear function of the parameters. We have a set of N points (x, y), and the parameters of the true line l are slope (a, b) and intercept c, where a and b are normalized so that $a^2 + b^2 = 1$. We denote these parameters as $\boldsymbol{\theta} = (a, b, c)$. In most computer vision problems, the data points are limited within some bound. Inliers are contaminated by noise with a noise model such that:

$$\begin{aligned} x &= x^t + n_x, \\ y &= y^t + n_y, \end{aligned} \tag{2.17}$$

where (x^t, y^t) is the true point and (n_x, n_y) is noise added to the point. The noise scale is assumed to be much smaller than the bound of the data points.

Given an estimate for the estimation of the line fitting problem: $\hat{\boldsymbol{\theta}} = (\hat{a}, \hat{b}, \hat{c})$, where $\hat{a}^2 + \hat{b}^2 = 1$, the fit of this estimate to the data set is analyzed by the residuals of all points. We focus on the analysis of the distribution of residuals. Signed residual r for data point (x, y) is computed as:

$$r = \hat{a}x + \hat{b}y + \hat{c}. \tag{2.18}$$

This is actually the signed point-line distance from (x, y) to the estimated line. For outliers, regardless of whether the estimate $\hat{\boldsymbol{\theta}}$ is correct or not, the residual r is still large and is bounded by the same limit $[r_{min}, r_{max}]$.

For inliers, r can be decomposed as follows.

$$\begin{aligned} r &= (\hat{a}x^t + \hat{b}y^t + \hat{c}) + (\hat{a}n_x + \hat{b}n_y) \\ &= r^t + r^n, \end{aligned} \tag{2.19}$$

where $r^t = \hat{a}x^t + \hat{b}y^t + \hat{c}$ and $r^n = \hat{a}n_x + \hat{b}n_y$. It can be seen that r is the sum of two different variables with different properties. r^t is the linear combination of x^t and y^t given the estimation parameters $\hat{a}, \hat{b}, \hat{c}$, and depends strictly on the accuracy of the estimation. r^n is the linear combination of the noise on the data points. If the noise on the data points is Gaussian noise, with some standard deviation and zero mean, $n_x \in G(\sigma_x, 0), n_y \in G(\sigma_y, 0)$. Then r^n is also a variable that comes from a Gaussian with standard deviation $\sigma_n = \sqrt{\hat{a}^2\sigma_x^2 + \hat{b}^2\sigma_y^2}$ and is bounded $\sigma_n < \sqrt{2(\sigma_x^2 + \sigma_y^2)}$. r^n does not really depend on the accuracy of the estimation. The better the estimate, the smaller r^t becomes and in the ideal case when the estimate is perfect, r^t =0, and the distribution of $r = r^n$ is entirely a Gaussian distribution.

This analysis can also be extended to any multiple linear regression problem in

which the residual is a linear function of the variables:

$$r = \sum_{k=1}^{p} \hat{a}_k x_k + \hat{a}_0, \tag{2.20}$$

where $\hat{a}_k$ is a parameter of the estimation, and $(x_1..x_p)$ is a data point. As the estimate improves, so the distribution of inlier residuals matches the Gaussian distribution more closely. In this case, the residual distribution model is a Gaussian distribution. The SDM is then the standard Gaussian distribution for the absolute of the variable.

2.4.2 Non-linear Residual

Similar to Section 2.4.1, in this section we analyze the problem when the residual is a non-linear function or general function (2.3) of a data point. In this case, it is difficult to analyze the distribution mathematically. However, such a function constrains the distribution of residuals helping us to analyze it statistically by simulation, and then the ideal distribution of the residuals can be modeled. Implementation of this step can be done online.

Assuming a certain noise model on the data points, such as Gaussian noise on the data point $\boldsymbol{X}$, we can model how the residuals from inliers are distributed in the ideal case. In a complicated problem such as fundamental matrix estimation, it is easier to analyze by simulation. For a fundamental matrix estimation the constraint function of the data points is [67][68]:

$$g(\boldsymbol{F}, \boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}'^{\boldsymbol{T}} \boldsymbol{F} \boldsymbol{x} = 0, \tag{2.21}$$

where $\boldsymbol{F}$ is the fundamental matrix and $\boldsymbol{X} = (\boldsymbol{x}, \boldsymbol{x}')$ is a single pair of point correspondences on two consecutive images. Several residual definitions exist, such as those in [68]. Two non-linear residual functions are selected to simulate how the residuals are distributed.

- The first residual function, which is called GRAD in this thesis, is based on a

gradient criterion:

$$r = f(\boldsymbol{F}, \boldsymbol{x}, \boldsymbol{x}') = \frac{\left|\boldsymbol{x}'^{T}\boldsymbol{F}\boldsymbol{x}\right|}{\sqrt{\| \boldsymbol{F}\boldsymbol{x} \|^2 + \| \boldsymbol{F}^T\boldsymbol{x}' \|^2}}. \tag{2.22}$$

- The second residual function, which is called DIST in this thesis, uses symmetric distance from points to epipolar lines:

$$r = f(\boldsymbol{F}, \boldsymbol{x}, \boldsymbol{x}') = \left|\boldsymbol{x}'^{T}\boldsymbol{F}\boldsymbol{x}\right| \sqrt{\frac{1}{\| \boldsymbol{F}\boldsymbol{x} \|^2} + \frac{1}{\| \boldsymbol{F}^T\boldsymbol{x}' \|^2}}. \tag{2.23}$$

The simulation is performed with an exceptionally large number of data points, and the statistical results are shown in Fig.2.8. For the ideal case in this simulation, residuals are calculated with a known fundamental matrix, zero-mean Gaussian noise is assumed on data point $\boldsymbol{X}$, and no outliers appear. The distribution of residuals is standardized so that the sample standard deviation, denoted by σ, is 1. Fig.2.8 shows the standardized residual distributions together with the standard Gaussian distribution for comparison. For the distribution of GRAD residuals, about 97.7% of the population is found within the range 2.5σ, and about 99.9% of residuals within 5σ. For the distribution of DIST residuals, about 97.6% of the population is found within the range 1.5σ, and about 99.7% of residuals within 5σ. For the Gaussian distribution, 97% of the population are within 2.5σ.

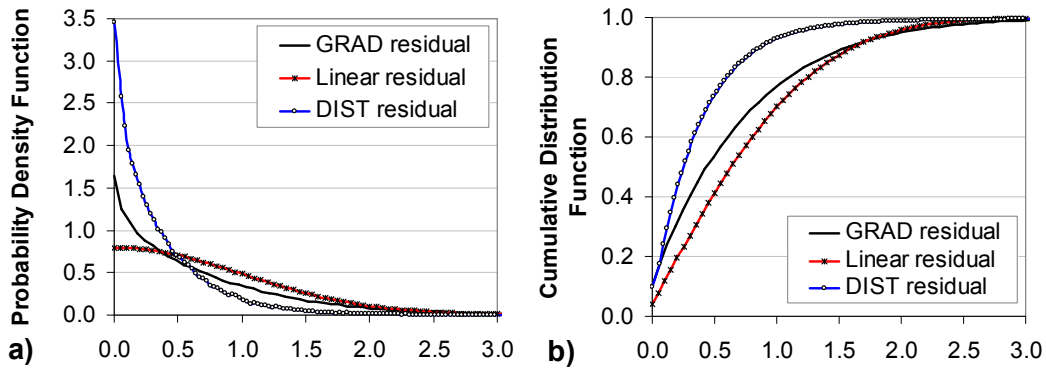


Figure 2.8: Standardized Residual Distribution Model (SDM) of fundamental matrix estimation and line fitting problem with Gaussian noise on data points.

2.5 Hypothesis Evaluation Function

Inspired by the use of the KDE in the pbM-Estimator [37][38], and ASKC [36], we also apply it in our adaptive objective function to evaluate the putative hypothesis:

$$F(\hat{\theta}) = \frac{1}{N\kappa\hat{\sigma}^*_{\hat{\theta}}} \sum_{i=1}^{N} K(\frac{r_{i,\hat{\theta}}}{\kappa\hat{\sigma}^*_{\hat{\theta}}}), \tag{2.24}$$

where $\hat{\sigma}^*_{\hat{\theta}}$ is adaptively estimated by the proposed inlier scale estimator as shown in Section 2.2 and κ has been defined in Section 2.2; K is a kernel such as Gaussian or Epanechnikov kernel. The KDE objective function evaluates how densely the residuals are distributed at zero using a kernel's window. In our case, the window of kernel K is $\kappa\hat{\sigma}^*_{\hat{\theta}}$, which covers all the estimated inliers, therefore the objective function gives the density measured at zero only for inliers. Similar to the M-estimators, a large residual makes a small contribution, whereas a small residual makes a large contribution to the overall score. However, this objective function is different to that in the conventional M-Estimators in two aspects. First, the scale estimate $\hat{\sigma}^*_{\hat{\theta}}$ is estimated for only inlier residuals, and is adaptively estimated. Second, the sum of weights on the residuals is scaled by $\frac{1}{\hat{\sigma}^*_{\hat{\theta}}}$, which intensifies the score when the estimated inlier scale is small and reduces the score when the estimated inlier scale is large. In summary, the KDE objective function declares a solution to be better under the following conditions:

(a) Larger number of estimated inliers,

(b) Smaller scale of inliers,

(c) Smaller residuals of inliers.

Reviewing previous robust estimation algorithms, RANSAC applies only criterion (a) to evaluate a solution; the M-Estimators use criteria (a) and (c); the modified pbM-Estimator [38] uses (a) and (c) and weakly uses (b), since its bandwidth for KDE is not guaranteed to be the same as the inlier scale; while ASSC [35] only applies (a) and (b).

In our methods, the inlier distribution type is automatically figured out for specific problem using the residual function. We realize that the Gaussian kernel or Epanechnikov kernel do not fit the inlier distribution well, therefore, we replace them by our case-dependent kernel, which is modeled in Section 2.4.

2.6 Summary of Proposed Robust Estimators: FITSAC1, FITSAC2

The flowchart for proposed estimators is generalized as shown in Fig.2.2. The details for FITSAC1 and FITSAC2 are described in Table 2.1.

The criterion for terminating the random sampling depends on the applications. It can be the excess of an amount of running time, or a number of iterations that assures a good estimate [28]. In our experiments, we fix the same number of iterations for the proposed method as well as the compared methods.

2.7 Experiments with Adaptive-scale Robust Estimators

In this section, we describe the experiments carried out to validate our algorithms in both linear and non-linear estimation problems: plane fitting, line fitting and fundamental matrix estimation. For each problem, a simulation is first used to understand the various aspects of the algorithm and then a real experiment with real data is carried out to validate the algorithm in a real situation. For the plane and line fitting problems, we compared our algorithms with several popular robust estimators: the pbM-Estimator , LMedS, ALKS, ASSC, and ASKC. For the fundamental matrix estimation, we used LMedS, ASSC, ASKC, and ALKS for comparison since the pbM-Estimator was originally proposed for linear robust regression problems only. In the experiments using ALKS, since it is very unstable when the normalized error function accumulates only small number of residuals, we started using this error function only when it accumulated a number of residuals greater than 15% of the total number of

Step	**FITSAC1**	**FITSAC2**
1.	Make the standardized residual distribution model (SDM) using the residual function. This can be done online or offline.	
2.	Create a random sample and then estimate the putative parameters $\hat{\boldsymbol{\theta}}$	
3.	Estimate all the residuals of the data points given the parameters $\hat{\boldsymbol{\theta}}$	
4.	Estimate the bin-width as described in **2.3.1**, and then compute the residual histogram $P_{\hat{\theta}}$.	Estimate the bin-width as described in **2.3.2**, and then compute the residual histogram $P_{\hat{\theta}}$.
5.	Estimate the inlier scale according to **2.2.1.**	Estimate the inlier scale according to **2.2.2.**
6.	Estimate the score using the objective function in 3.5 with Gaussian kernel.	Estimate the score using the objective function in 3.5 with case-dependent kernel.
7.	Update the currently best solution.	
8.	Repeat from step **2**. if not terminated	

Table 2.1: Summary of two estimators, FITSAC1 and FITSAC2.

data points. For the pbM-Estimator, we used the program from the authors[75]. The Epanechnikov kernel was used for all kernel density estimations including the related objective functions such as in the proposed objective function. All algorithms were supplied with the same set of random sampling trial hypotheses and no estimation optimization was done in any of the algorithms. In the proposed estimator, FITSAC1, the value of κ is chosen according to the SDM. κ is selected so that the section of SDM for matching contains about 97% of the population. In the experiments, $\kappa = 2.5$ for the line fitting problem and fundamental matrix estimation using the GRAD function, while $\kappa = 1.5$ for the fundamental matrix estimation using the DIST function. For FITSAC2, the inlier scale, or the standard deviation of inlier residuals, is estimated. The performance of FITSAC2 does not rely on any user-define parameters. However, when the exact inlier detection is necessary, we also use κ to distinguish inliers, but this is done after the execution of random sampling. The criteria for validating the proposed estimators are:

- robustness with various outlier rates and noise scales,
- accuracy of the inlier bound (threshold to distinguish the inliers), and
- the ability to work with data with multiple structures.

In data with the appearance of multiple structures, it is important that an estimator estimates a tight bound and outputs as many inliers as possible for a particular structure, otherwise the actual structure may be broken into many smaller structures or several structures may be estimated as a single one.

2.7.1 Linear Residual

In this problem, a estimator must extract the correct line or plane from a data set that contains single or multiple structures with the appearance of random outliers. The experiments were carried out by various popular and analytic simulations for a robust estimator as previous works. For data with a single structure, the evaluation was carried out with various outlier rates and noise scales. For data with multiple

structures, we validated the proposed estimators using the various types of data with multiple structures frequently used for testing robust estimators: that is, data with parallel lines, data with steps and roof data.

Given an estimate $\hat{\boldsymbol{\theta}} = (\hat{a}, \hat{b}, \hat{c}, \hat{d})$, the residual function is defined as:

$$r_i = |\hat{a}x_i + \hat{b}y_i + \hat{c}z_i + \hat{d}|, \tag{2.25}$$

where (x_i, y_i, z_i) is a data point. The estimation error is defined as follows.

$$Error_{\hat{\theta}} = \sqrt{(a-\hat{a})^2 + (b-\hat{b})^2 + (c-\hat{c})^2 + (d-\hat{d})^2}, \tag{2.26}$$

where (a, b, c, d) are ground-truth parameters. The normal vector of each plane is normalized so that $\sqrt{a^2+b^2+c^2} = 1$, $\sqrt{\hat{a}^2+\hat{b}^2+\hat{c}^2} = 1$.

Single Structure with Various Outlier Rates

A 3D plane with 500 points was randomly generated for each trial data set. Gaussian noise with a mean of zero and noise scale σ_G was added to the inliers. Random outliers were generated to replace inliers, and therefore, the total data set always contained 500 points. All the points were located within the 3D volume [0, 0, 0, 1000, 1000, 1000]. 100 data sets were randomly generated, and for each data set, the same 10000 iterations of random sampling were supplied to each estimator. The graphs shown below use the averages of the results for all 100 data sets.

We evaluated both the estimation error and inlier bound with various outlier rates. The ratio between the number of estimated inliers and the number of true inliers, and the ratio between the scale of the estimated inlier residual and the scale of the true inlier residual should be about 1 for any estimator.

In the first experiment for 3D plane fitting, we tested the outlier rate factor for all estimators with the same noise scale $\sigma_G = 8$. The average results are shown in Fig.2.9. Fig.2.9.a describes the break-down point and the accuracy of the robust estimators, while Fig.2.9.b shows the ratio between the estimated and true inlier scales. The example of inlier detection for robust estimators is shown in Fig.2.10, in which

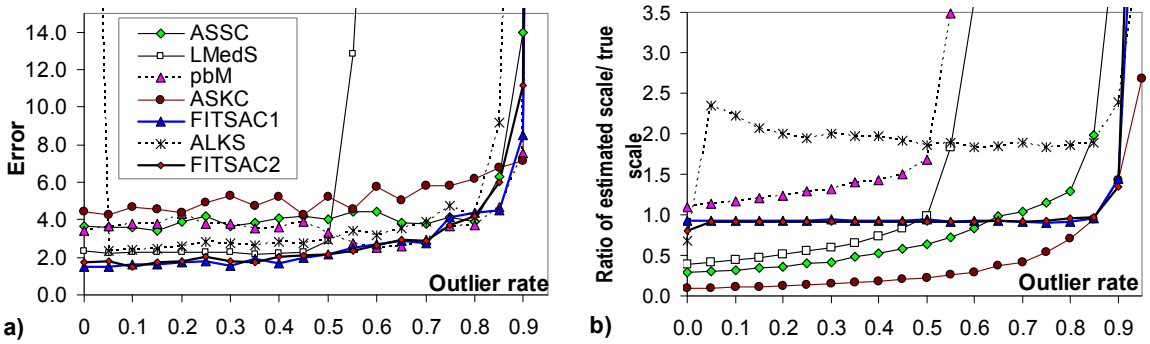


Figure 2.9: Experiments with varying outlier rates for single-line data: (a) estimation error, (b) ratio between the scale of the estimated inlier residuals and the scale of residuals of true inliers.

FITSAC1 and FITSAC2 have the similar results. We can see that in this experiment our proposed algorithm yields the best overall performance for accuracy and estimated inlier scale of all the algorithms. At low outlier rates, less than 50%, LMedS is accurate, but for higher outlier rates, LMedS fails to estimate. The performance of ALKS is unstable for very low or high outlier rates; the estimated inlier scale ratio is about 2, which means that ALKS overestimates the inlier scale. ASSC, ASKC, pbM and the proposed algorithm have similar breakdown points allowing these to retain good performance up to an outlier rate of 90%. ASSC and ASKC show similar performance, since their estimated inlier scales and KDE bandwidths correlate, but they usually underestimate the inlier scale. On the contrary, the performance of the pbM and proposed estimators for estimating residual density does not really depend on the bandwidth (or bin-width), and thus the accuracy of the pbM and proposed estimator remains high for the various outlier rates. In addition, as the proposed estimators always estimates an accurate inlier scale, the estimated inlier scale closely matches the true inlier scale. However, it should also be noted that the pbM estimates the solution first and then estimates the inlier scale and consequently the inlier scale is not important for the accuracy of the estimated solution.

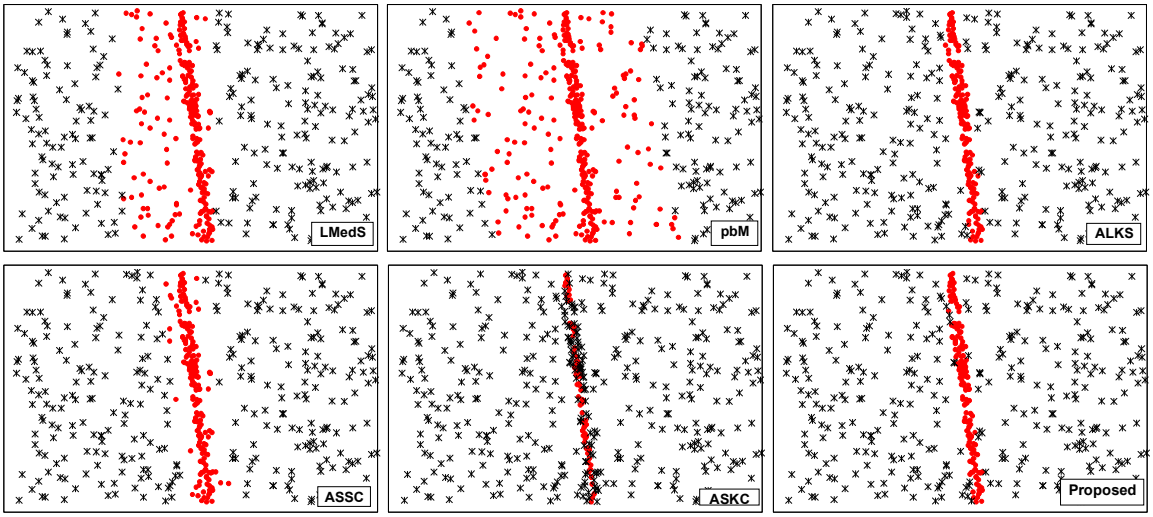


Figure 2.10: Example of inlier detection for robust estimators.

Single Structure with Varying Noise Levels on Inliers

A second experiment was carried out to test all estimators with various noise scales. The data was set up similar to the experiment for 2D line fitting, except that the Gaussian noise scale σ_G on inliers varied between 1 and 52, while the outlier rate was fixed at 60%. Examples of the noise scales are shown in Fig.2.11, while the average results are shown in Fig.2.12. Fig.2.12.a describes the estimation error, while Fig.2.12.b describes the ratio between the estimated inlier scale and true inlier scale. Since the outlier rate is 60%, LMedS fails to estimate correctly, giving a much larger estimated number of inliers than the number of true inliers. The performance of ALKS is unstable with the higher noise levels on inliers. All the other estimators have lower accuracy with higher noise levels, although the proposed estimators gives the most robust performance. FITSAC1 and FITSAC2 have quite similar performance. These results confirm that our proposed estimators have the best accuracy and robustness of all the estimators, and the estimated inlier bound is quite close to the ground-truth.

Parallel Lines with Different Distances

Here we demonstrate the ability of the estimators with the appearance of multiple structures in the data.

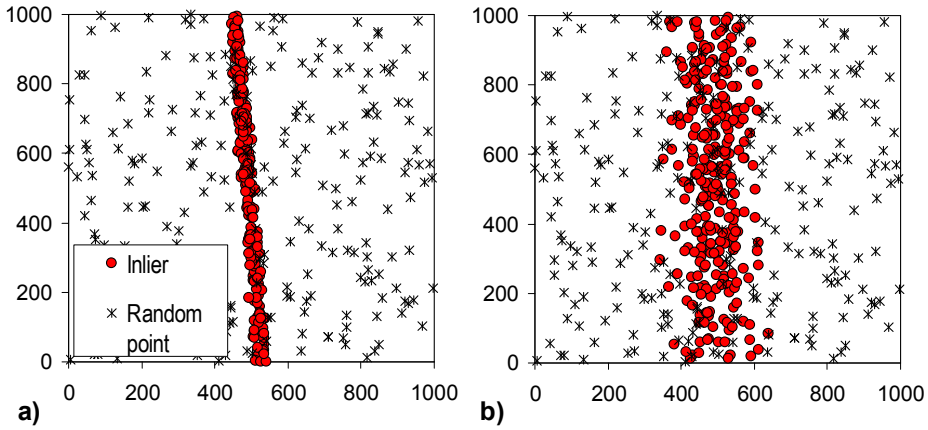


Figure 2.11: Random data sets with an outlier rate of 60% and (a) $\sigma_G = 8.0$ and (b) $\sigma_G = 50$.

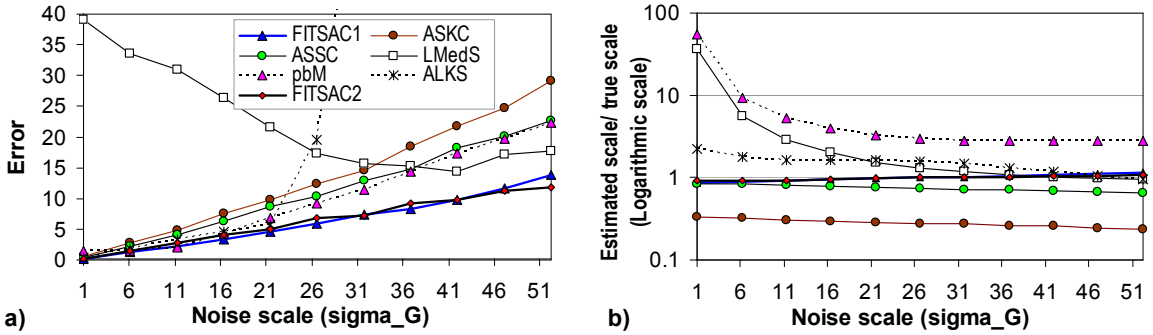


Figure 2.12: Experiments with varying Gaussian noise scales and outlier rate fixed at 60%. Proposed estimator is highly resistant to high noise levels.

A data set containing two parallel lines was used in this experiment. Each estimator was required to estimate one of the two lines correctly with a precise inlier bound. The experiment was carried out with different distances between the two parallel lines:

$$\begin{aligned} Line1 : 2x - y + d &= 0, \ where\ d = 20, 30, 40, ...210 \\ Line2 : 2x - y &= 0. \end{aligned}$$

Various random data sets were used, with each data set containing 270 random outliers, 420 random points on $line2$, and 210 random points on $line1$. Gaussian noise σ_G=8.0 was added to each point on each line, and the coordinates of all points

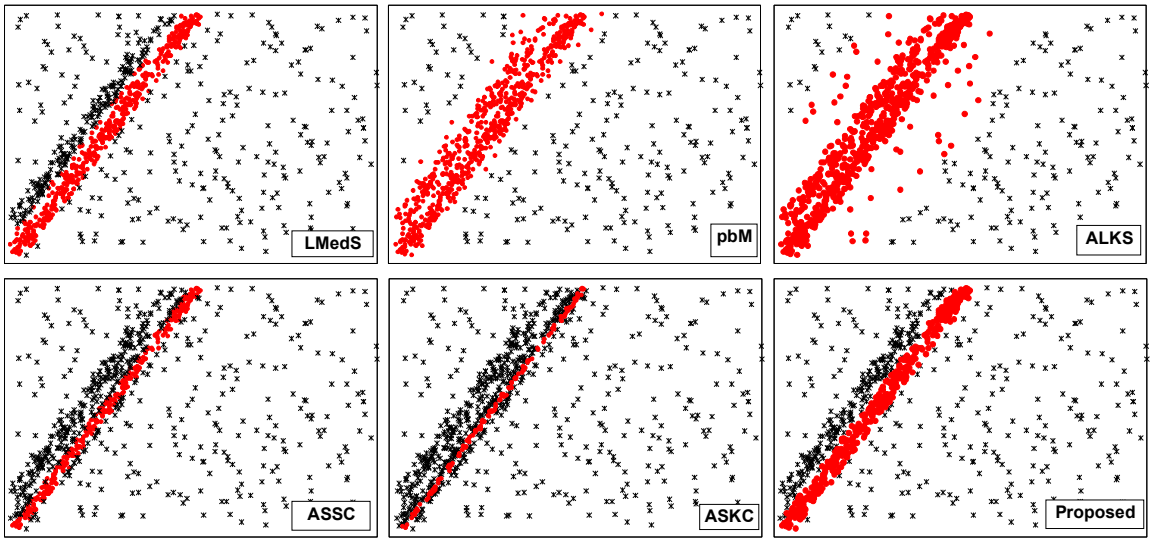


Figure 2.13: Parallel Lines: estimation by each estimator using a random data set with d=70. ALKS and the pbM are confused, since the two lines are extracted as one. ASKC and ASSC extract a small part of the actual line. LMedS estimates a line with a large number of inliers belonging to *Line* 1 and a few inliers belonging to *Line* 2. The proposed methods (FITSAC1 and FITSAC2) extract one of the two lines correctly and neatly.

were within the rectangle (0, 0, $62.5\sigma_G$, $62.5\sigma_G$). The estimations of the robust estimators using an example data set are shown in Fig.2.13, in which FITSAC1 and FITSAC2 have similar results. In this example, all estimators estimated the correct line, but LMedS, the pbM and ALKS overestimated the population of inliers, ASSC and ASKC underestimated the inliers, while the proposed estimator estimated the inliers correctly. The average results for 100 random data sets are shown in Fig.2.14. Fig.2.14.a shows the estimation error for the robust estimators, while Fig.2.14.b shows the ratio between the number of estimated inliers and the number of true inliers. When the two lines are close together with $d = 20$, they are almost mistaken for being one line, with all estimators having a similar accuracy. When the lines are further apart, the performance of ALKS is the worst, as it only manages to estimate correctly once the two lines are very far apart with $d > 170$. This is understandable since it is claimed [30] that ALKS only estimates correctly step signals with a height greater than $8\sigma_G$. Because the actual outlier rate of estimating any line is greater

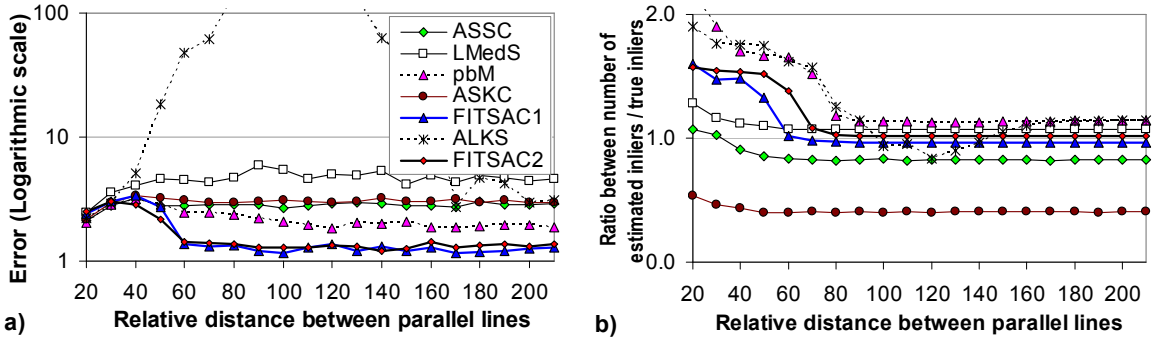


Figure 2.14: Parallel Lines: (a) estimation error, (b) ratio between the number of estimated inliers and the number of true inliers.

than 50%, LMedS produces worse results as the two lines move further apart. ASSC and ASKC have a similar performance, but the number of inliers is underestimated in both cases and remains similar since it is only related to their KDE bandwidth. The proposed algorithm starts to estimate the line correctly for both solution parameters and inliers when $d = 60$, that is, when the distance between the lines is about $3.3\sigma_G$. With regards the bound on the estimated inliers, our proposed estimator gives the best results, since the number of estimated inliers is relatively close to the number of true inliers; in fact it is slightly smaller since leverage true inliers were also judged as outliers. FITSAC1 can detect the inliers better than FITSAC2 when two lines are close since it ignores the long tail of the distribution model, while FITSAC2 use the whole model for matching. However, Here, the results of FITSAC2 has shown that eventhough it does not limit the distribution model for matching, FITSAC2 still resist to the multiple outlier distribution quite well. The reason for this resistant is that the structure with high density is always detected first in the proposed algorithm, outlier distribution from other structures has only small effect on the whole matching function and the ground distribution parameter compansates for the whole outlier distribution well.

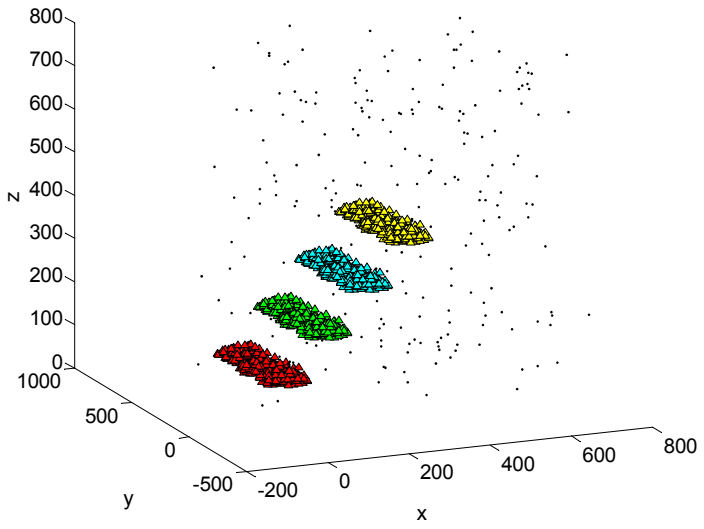


Figure 2.15: An example of a step data set with four planes and various random points. Each point on a plane is contaminated by Gaussian noise with $\sigma_G = 5.0$.

Multiple Structures: Steps with Varying Noise Levels

In this experiment, the step data consisted of four planes, set up as shown in Fig.2.15. The parameters of the actual planes are:

$$\begin{aligned} Plane\ 1: &\quad z - 100 = 0 \\ Plane\ 2: &\quad z - 200 = 0 \\ Plane\ 3: &\quad z - 300 = 0 \\ Plane\ 4: &\quad z - 400 = 0 \end{aligned}$$

The data set used in the evaluation consisted of 240 random points for each plane and 240 random outliers. Each data point on a plane was contaminated by Gaussian noise with σ_G. The experiment was carried out to test all the estimators with different values of σ_G. For larger values of σ_G, the four planes move closer and may become fused. The results are illustrated in Fig.2.16, which gives the average of the results for 100 such randomly generated data sets.

In this experiment, the pbM-Estimator did not perform well since it mistook the four planes for the same structure, and consequently the estimated number of inliers is about four times the number of true inliers for each plane, as shown inFig.2.16.b.

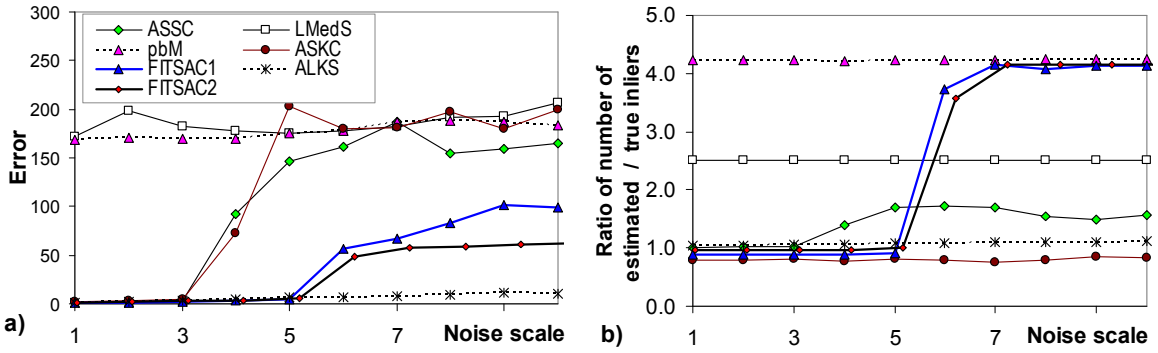


Figure 2.16: Results from using data consisting of steps with different Gaussian noise levels σ_G = 1,2,...,10. a) shows the average estimation error, while b) shows the ratio between the estimated number of inliers and number of true inliers.

LMedS also did not perform adequately since the outlier rate is high for the estimation of any plane. ASSC and ASKC succeeded in estimating correctly with low noise levels only. The number of estimated inliers for the two methods remained similar regardless of whether they failed or succeeded. The proposed methods were able to work correctly with slightly higher noise levels but then became confused, and the four planes were estimated as a single plane. In this comparison, ALKS worked correctly with much higher noise levels. However, since ALKS is well-known for its instability and sensitivity to small pseudo structures, we limited the size of possible structures, such that the estimated structure for ALKS was larger than 15% of the data. Therefore, it was able to estimate these steps correctly. In this case, its sensitivity was an advantage.

Multiple Structures: Roof with Varying Noise Levels

In this experiment, two planes were set up as shown in Fig.2.17. The parameters of the actual planes are:

$$\begin{aligned} Plane\ 1 : x - y &= 0 \\ Plane\ 2 : x + y + 500 &= 0 \end{aligned}$$

The data set used in the evaluation consisted of 350 random points for *Plane* 1,

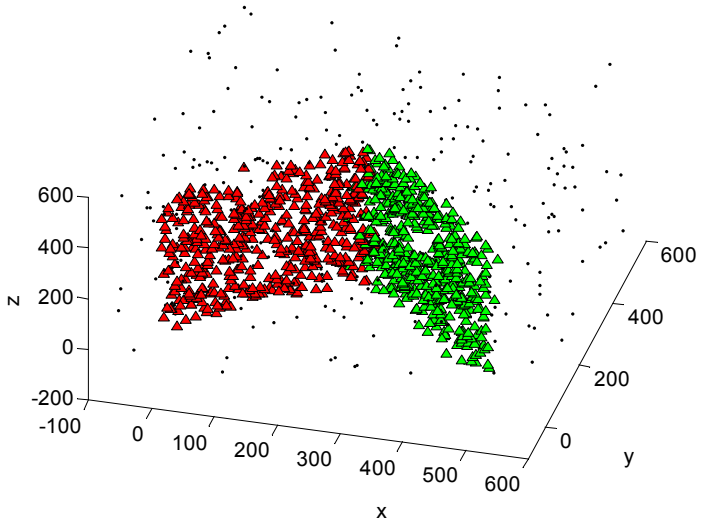


Figure 2.17: An example of the roof data set consisting of two planes and various random points. Each point on a plane is contaminated by Gaussian noise with $\sigma_G = 13.0$.

350 random points for *Plane* 2 and 300 random outliers. Each data point on a plane was contaminated by Gaussian noise with σ_G. The experiment was carried out to test all the estimators with different values of σ_G. The results are depicted in Fig.2.18, which gives the average results for 100 such randomly generated data sets.

The results show that most of the estimators worked well with this type of data except the LMedS since the actual outlier rate for estimating any plane was higher than 50%. The proposed estimator and pbM-Estimator outperformed the others. The pbM-Estimator performed slightly better with a low noise level, whereas the proposed estimators performed slightly better with high noise levels.

The above results clearly show that our proposed algorithms work well with data containing multiple structures. FITSAC2 are comparable to FITSAC1 even without user-defined parameter.

Range Image Segmentation

In this experiment, we demonstrate the ability of robust estimators for segmentation problem, for example, range image segmentation. In this problem, robust

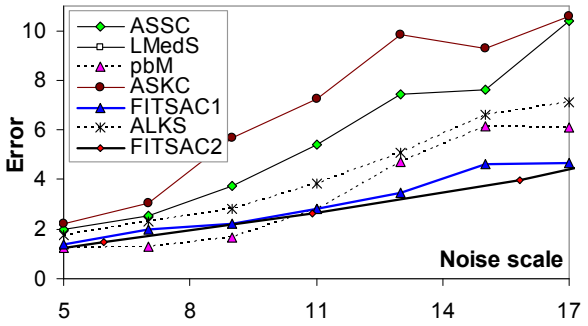


Figure 2.18: Estimation error using roof data with different Gaussian noise levels σ_G = 5,7,...,17.

estimator must extract all the planes that make the object, a chair in our experiment. The segementation for each robust estimator is done as follows. First, the robust estimator extracts one segment, then the inliers for that segment are removed from the data. The same procedure is repeated with a fixed number of iterations or until there is no remaining data points. We set this number to 8 for this chair image.

The segmentation results are shown in Fig.2.19. The results show that pbM over-detected the inliers then some structures were combined. ASKC and ASSC under-detected the inliers then some planes were divided into smaller parts. FITSAC1 and FITSAC2 gave the most proper results, in which planes were segmented clearly.

2.7.2 Non-linear Residual

For this experiment, we first carried out a simulation to validate various aspects of the proposed algorithm, and then performed the experiment with real data to show the effectiveness in a real situation. The GRAD and DIST residual definitions described in Section 2.4.2 were used for the fundamental matrix estimation. These residual definitions are not linear, and therefore, the pbM-Estimator is not applicable, because it was originally designed for linear residual problems only. Thus we compared the proposed algorithm with ASSC, ASKC, LMedS and ALKS, even though the non-linear residual function could have been linearized for use by the pbM.

Since it is not possible to compare the estimated fundamental matrix with a ground-truth fundamental matrix, we computed the error as the standard deviation

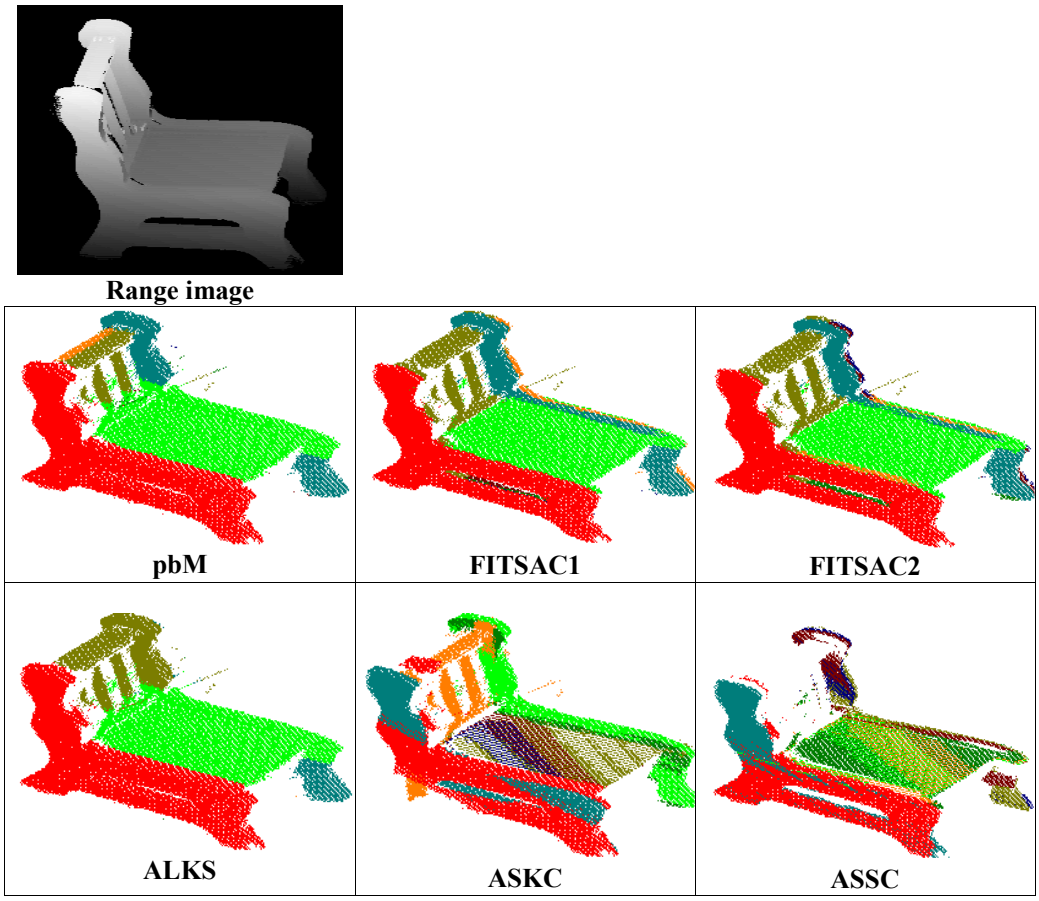


Figure 2.19: Range image segmentation demonstrates the ability of robust estimator for segmentation problem. The robust estimator must properly detect the inliers. The over-detection or under-detection results in a undesirable segmentation.

of only the inlier residuals of the estimated fundamental matrix $\hat{\boldsymbol{\theta}}^* = \hat{\boldsymbol{F}}^*$:

$$Error_{\hat{\boldsymbol{F}}^*} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(r_{i,\hat{\boldsymbol{F}}^*})^2}, \tag{2.27}$$

where M is the number of inliers. This error computation relies on how the solution fits the motion data: a better fit produces smaller residuals for inliers, and vice versa. In the simulation, we know the true inliers and thus M is known. In the real experiment, the error is computed for the M smallest residuals (which are considered inliers), with M assigned manually after checking the actual data.

Fundamental Matrix Estimation in a Simulation

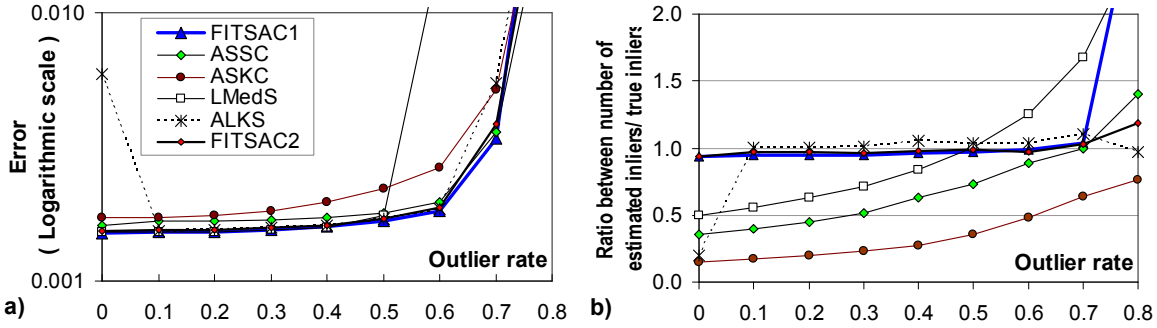


Figure 2.20: Fundamental matrix estimation in a simulation using the GRAD residual function: (a) estimation error, and (b) ratio between the number of estimated inliers and the number of true inliers with various outlier rates.

We simulated points on a unit sphere, with 500 points randomly distributed on a unit sphere. Altering the view point slightly causes the points on the sphere to move, thus creating 500 pairs of point correspondences. Some of these pairs were then replaced by outlying pairs with random point coordinates, thus keeping the total number of pairs as 500. Coordinates (x, y, z) for each inlier point on the unit sphere, before and after being moved, are contaminated by Gaussian noise with zero mean and noise scale σ_G. The fundamental matrix was estimated using the seven point algorithm [69]. Experiments were carried out for robustness under various outlier rates and the average results of 100 randomly generated data sets are shown in Fig.2.20 with $\sigma_G = 0.005$ for the GRAD residual function. The results of these experiments are similar to those in the plane fitting problem described above. These results prove that our proposed algorithm gives the highest robustness under various outlier rates and estimates a reasonable number of inliers which is close to the number of true inliers. ALKS performs quite well in these experiments and also produces an estimate or the number of inliers which is close to the number of true inliers, however it is unstable for very low or high outlier rates. ASSC and ASKC usually underestimate the population of inliers as in the previous experiments.

We carried out a similar simulation using the DIST residual function. However, as the results are similar to those using the GRAD residual function they are not

shown here.

Fundamental Matrix Estimation in Real Video Sequences

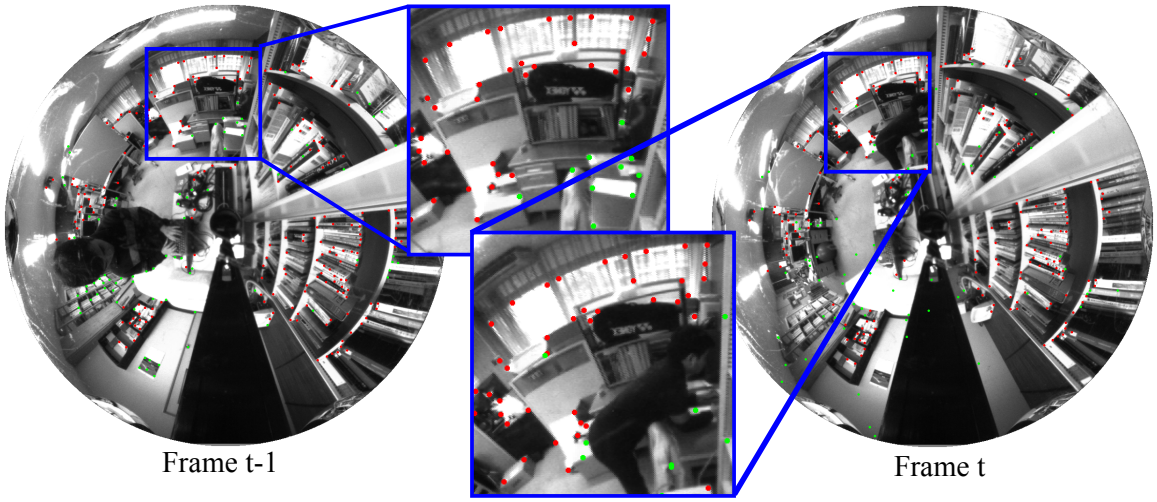


Figure 2.21: A pair of images in a sequence: inliers (image features in red) and outliers (image features in green) are output by the proposed estimator.

In this experiment, real video sequences were captured in an indoor environment with an omnidirectional vision sensor. Examples of the captured images are shown in Fig.2.21. The sensor consisted of an omnidirectional mirror, a telecentric lens and an imaging sensor. The camera was mounted on a rotary stage and controlled by a PC, which translated the camera whilst it was being rotated. For each pair of images, 200 Harris image features were detected on the first image and tracked on the second image to obtain the feature correspondence pairs using the KLT feature tracker[56] implemented in OpenCV [72]. Features for each image were mapped to the unit sphere. The fundamental matrix between a pair of consecutive images was computed using the seven point algorithm with these feature correspondence pairs. For each video sequence, about 50 images were captured whilst ensuring the same rotation between consecutive images. The performance of all the estimators tends to deteriorate with a greater degree of rotation, since the KLT tracker is less accurate under greater rotation. Therefore, we used three video sequences with different rotation settings. These video sequences are referred to as $Video\_4deg$, $Video\_14deg$ and $Video\_18deg$

for rotation speeds of 4 degrees/frame, 14 degrees/frame, and 18 degrees/frame, respectively. We computed the error by (2.27) and M was set independently for each video sequence after randomly checking five pairs of images within each video sequence. The average number of true inliers and the assigned value of M for each video sequence are given in Table 2.2. From this table, we can see that the outlier rate for $Video\_4deg$ is low, about 10%. For $Video\_14deg$, the outlier rate is about 50%, and for $Video\_18deg$, the outlier rate is about 65%. For each image pair, 20000 iterations of random sampling were provided for each estimator. In this case, the true noise model on the feature points was not known. However, it was assumed to be a Gaussian model with zero mean and thus the residual distribution models for the GRAD and DIST residual were known. In this experiment, the results for GRAD and DIST residual function are similar, the only description of experiment for GRAD is shown in this section.

The average error and number of estimated inliers for 100 executions of each video sequence are given in Table 2.2 and Table 2.3, respectively. The results show that the FITSAC1 has the best accuracy for various outlier rates. The number of estimated inliers correlates with the outlier rate; it is slightly larger than the number of true inliers. FITSAC2 is slightly less accurate than FITSAC1 but the inlier detection is similar to that of FITSAC1. FITSAC2 has the similar accuracy compared to ASSC. ASSC and ASKC estimate a similar number of inliers for the various outlier rates as in the previous experiments. ALKS performs the worst of all these estimators in this real experiment.

In addition, we also evaluated the inlier bound for all estimators using visualization. For each estimator, the average of all the histograms of residuals from the estimated solutions for all executions was calculated for comparison with the average of the estimated thresholds. This averaging for visualization purposes can be done within the same video sequence and with the same control speed only, since the performance of the KLT feature tracking is similar. The visualization is shown in Fig.2.22 (a), (b) and (c) for the three video sequences $Video\_4deg$, $Video\_14deg$ and $Video\_18deg$, respectively. It can be seen from these figures that the proposed estimator output the most reasonable results, with the estimated threshold adaptively

Table 2.2: Fundamental matrix estimation for real video sequences using GRAD residual function: estimation error.

Video sequence	Video_4deg	Video_14deg	Video_18deg
Average number of true inliers	187.70/200	102.75/200	72.25/200
Assigned M	150	90	60
Fitting error			
FITSAC1	0.000615	0.001493	0.001692
FITSAC2	0.000638	0.001690	0.001765
ASSC	0.000731	0.001673	0.001756
ASKC	0.000926	0.002350	0.002426
ALKS	0.004123	0.008205	0.008013
LMedS	0.000625	0.001676	0.002536

Table 2.3: Fundamental matrix estimation for real video sequences using GRAD residual function: number of estimated inliers.

Video sequence	Video_4deg	Video_14deg	Video_18deg
Average number of true inliers	187.70/200	102.75/200	72.25/200
FITSAC1	182.307	110.847	87.935
FITSAC2	183.306	109.913	82.925
ASSC	68.079	65.534	65.534
ASKC	23.122	24.073	24.073
ALKS	66.057	94.977	70.133
LMedS	101.000	101.000	101.000

located in the tail of the actual distribution, and in which the density of outliers was low and the density of inliers within the inlier bound was high. This means that it was able to separate the inliers and outliers effectively. ASKC and ASSC output a solution in which the threshold was not located in the tail of the distribution, and in which the density of inliers was very high since the number of inliers was underestimated. ALKS did not work well resulting in a low density of histograms and very large inlier bounds. For the sake of giving only informative comparisons, the estimated inlier bounds for ALKS are not shown.

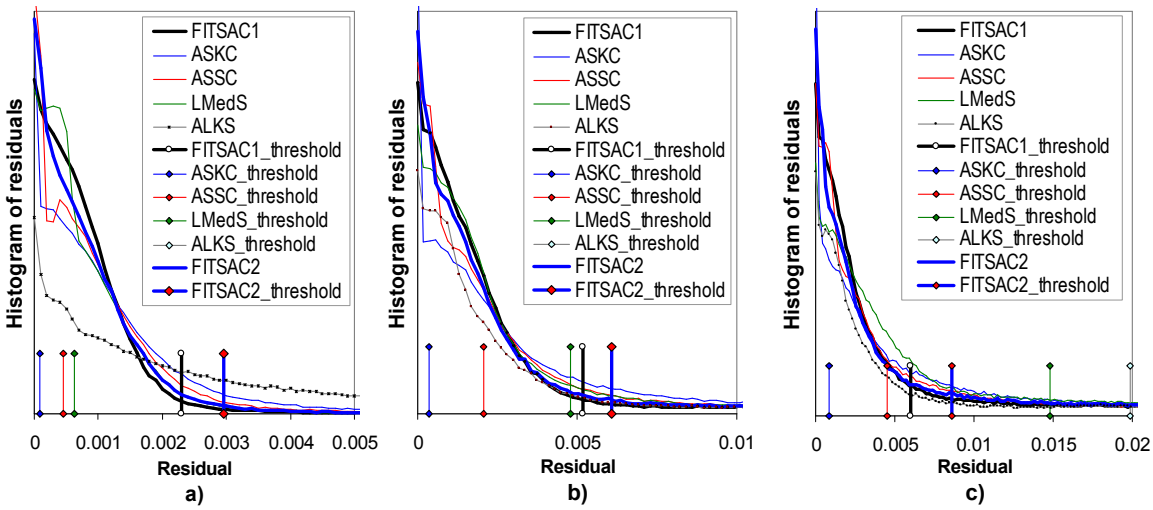


Figure 2.22: Visualization of the estimated inlier bounds for estimators using the GRAD residual function with three videos sequences (from left to right) $Video\_4deg$, $Video\_14deg$ and $Video\_18deg$. An average of the histograms of residuals from the estimated solutions was made for each estimator to visualize how tightly the estimated inlier bound fits the residual distribution.

2.7.3 Computational Cost

We simulated the relation between processing time and the number of data points, the average results of which are shown in Fig.2.23. The graph shows that overall the proposed estimator gives the fastest computational time, especially for large data. For the FITSAC1, the residuals are not needed to be sorted, therefore it is fast in comparison with the others, especially when the number of data points increases. FITSAC2 is the second slowest estimator. For the other estimators, the residuals have to be sorted first. LMedS is the simplest algorithm among the sorting-based methods, it takes the second fastest place in this comparison. After sorting the residuals, ALKS needs more cost to find the separation between inliers and outliers. ASKC and ASSC have the same procedure to locate the inlier distribution using mean-shift algorithm, the only difference is that ASKC uses the smaller window (bandwidth) for searching the local peaks of residual density then it consumes less computational cost than ASSC. The slowest estimator is pbM since it consumes heavy cost to find a global peak of residual density.

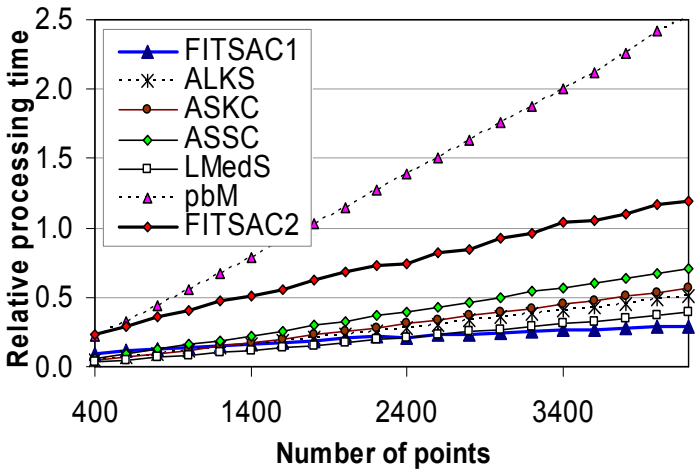


Figure 2.23: Processing time for all estimators

2.8 Chapter Conclusion

In this thesis, we proposed two novel highly robust estimators (FITSAC1 and FITSAC2) for the estimation problem in computer vision that deals with data with high outlier rates and multiple structures. Our algorithms do not need any prior information about the inlier scale, as this is estimated adaptively.

Depending on the specific problem, the distribution model of residuals is analyzed using that useful constraint, the residual function. The analysis is feasible and simple, and simulation of the residual distribution model can always be performed. The advantage of this approach is that it estimates the inlier scale correctly and therefore improves robustness.

The adaptive smoothing parameter efficiently help FITSAC2 work robustly in various situation without any support from user.

The proposed robust estimators were positively validated through experiments with various conditions and real estimation problems. The use of the constraint from the residual function in the robust estimator is effective for improving the robustness and detection of inliers.

The proposed estimator can be applied to any problem in which the residual function is properly defined. Furthermore, it is especially useful when the inlier scale needs to be estimated accurately.

Chapter 3

Outlier Detection by Pre-filtering before Fitting Model

3.1 Introduction

As stated in earlier in Chapter 1, in case of multi-models and multi-structures, the workload for a robust estimator is extremely high to work efficiently and when the inlier distribution and outlier distribution overlap the conventional threshold can not be used to distinguish inliers/outliers. We propose an idea to filter out the data from uninterested (outlier) models to reduce the workload and improve the robustness for robust estimator.

Although, the general solution has now not been proposed, we find out it can be solved heuristically in specific problem such as with the support of the sensor that captures the data. For examples, when we estimate the camera egomotion using a video camera and image feature tracking. If the environment is rigid and no moving objects such as people, cars, the state of the art egomotion estimation algorithms [67, 68, 69] work well in real time for real applications. However, if the environment is dynamically changed such as moving car, people...then there is a large number of outlier optical flows comes from different objects. This makes the egomotion estimation algorithm can not converge in real-time. In such case, an inertial sensor [64, 65] can be used for improving the performance by quickly eliminating the outliers of the

camera motion. Although inertial sensor does not supply accurate egomotion, we can use this information to roughly and quickly classify the inlier and outlier optical flows.

In the following section, we demonstrate the idea with a specific problem of egomotion using a stereo camera system. A novel egomotion estimation algorithm for fast motion of a compound omnidirectional vision sensor will be presented.

3.2 Egomotion Estimation by Separating Feature Set for Rotation and Translation

Egomotion, which consists of both rotation and translation, is an attractive research topic in computer vision and robotics. Using a camera is a common option in estimating egomotion. The egomotion of the camera is recovered by observing the motion on images. In realistic applications such as a wearable system, an unmanned aerial or land vehicle, fast motion usually occurs, especially for rotation. Previous research works can be classified into either local search or global search approaches for finding correspondences between consecutive frames and solving the egomotion estimation problem.

In the local search approach, a camera is assumed to move smoothly and slowly. Using this assumption, image feature points can be tracked for correspondence by a feature tracker [56] or an optical flow computation [58] through a video sequence. The camera egomotion is then estimated from the feature correspondence [59, 61, 60, 68, 63]. However, the assumption is too restrictive and is not effective for realistic applications such as in aerial vehicles and wearable cameras where the motion is fast.

In the global search approach, the random sample consensus (RANSAC) methods [12, 16, 20] solve the correspondence and camera motion estimation simultaneously. This approach searches globally for the combinations that fit the motion hypotheses given by random sampling. Hence, there is no motion restriction. However, it is well-known that the computational cost of RANSAC increases exponentially according to the number of corresponding points. For 5-DOF egomotion estimation, we need five

or seven feature correspondences between consecutive frames. Therefore, it is difficult to compute this problem in real-time using RANSAC methods.

Therefore, in this chapter, we propose real-time egomotion estimation algorithm with a compound omnidirectional vision sensor. The proposed algorithm tries to tackle the fast camera motion therefore it falls in the global search algorithm category, where the feature point correspondence is not supplied. Therefore, if we directly apply RANSAC to simultaneously estimate the motion and feature correspondences, it is not efficiently due to the outlier rate is too high. In this algorithm, all feature points are considered to belong to rotation model or translation model. The proposed method quickly classifies the image features into far and near features by the characteristics of the sensor. The feature set for rotation model consists of only far feature points and translation model consists of only near feature points. Then we estimate the camera rotation and translation separately using far and near features, respectively. The idea comes from the observation that motion of far features on the image is almost rotation, and near feature points describe the translation clearly. The proposed method is realized to reduce the computational cost and improve the robustness of the estimation.

3.2.1 Overview of the Proposed Algorithm

The proposed algorithm estimates the camera egomotion with 5 degrees of freedom, 3D rotation and 2D translation (the direction of translation without magnitude). The flowchart for the algorithm is presented in Fig.3.1. We used a compound omnidirectional sensor, described in the next section. The sensor provided a compound image that consists of all mirror's omnidirectional images. Corner feature points are detected in the center omnidirectional image, and are then classified as near or far features. We estimate the egomotion from two successive video frames. Rotation is estimated using only the far features, while translation is estimated using only the near features after eliminating the rotational motion using the estimated rotation. Since we are dealing with large camera motion, tracking image features is not helpful. Therefore, we use the RANSAC search [12] to find the global correspondences and to

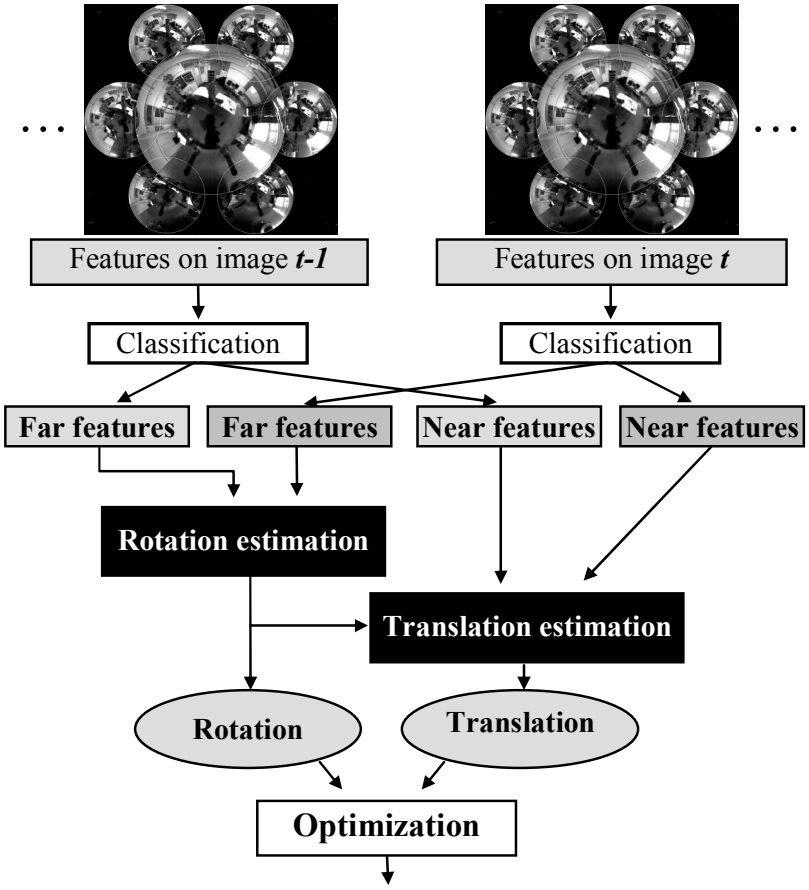


Figure 3.1: Algorithm flowchart: Detected features are classified as near or far features; rotation is estimated using the far features while translation is estimated using the near features; finally, optimization is performed to tune the estimated motion parameters.

estimate the egomotion simultaneously. During this process, we could estimate rotation and translation separately to reduce the computation complexity, because we have already classified the feature as either far or near. Finally, rotation and translation parameters are optimized under epipolar constraints using all the supporters from the RANSAC estimation.

3.2.2 Compound Omnidirectional Vision Sensor

A compound omnidirectional vision sensor consists of M paraboloidal mirrors, one large mirror at the center with $M-1$ small surrounding mirrors, and a single

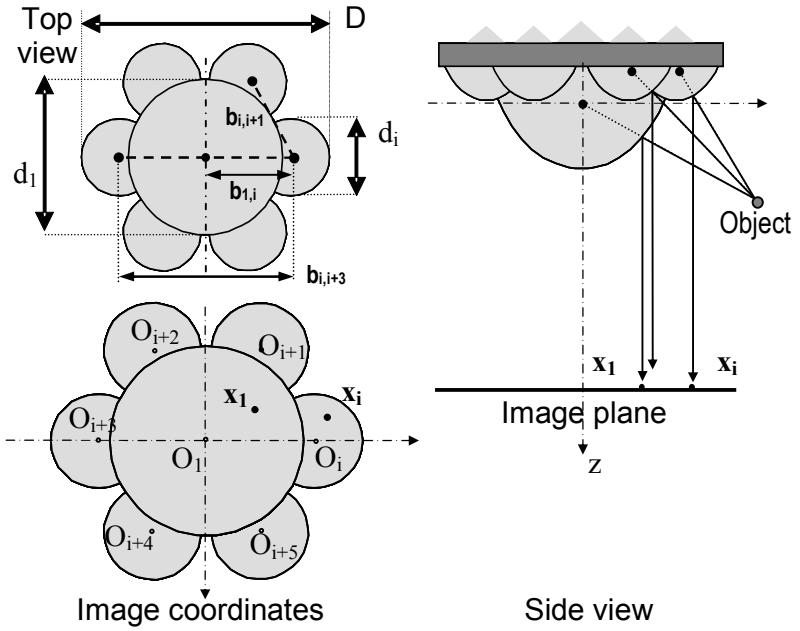


Figure 3.2: An example of a compound omnidirectional sensor: a single orthographic camera with seven compound paraboloidal mirrors.

camera. In a 3D coordinate system, each mirror i has the parameter (r_i, O_i), where r_i is its radius of curvature at the top and O_i is its location. The baseline between a pair of mirrors (i, j) is defined as $b_{i,j} = \sqrt{||O_i - O_j||}$. Fig.3.2 shows one example of our compound sensor, in which $M = 7$ paraboloidal mirrors and a single orthographic camera are used. Mirror i, $i = 1..7$, has the diameter d_i, and the total diameter of the compound mirror is D.

A light ray from an object is projected onto the image plane by reflection from the mirrors. Since the position of each mirror is different, the distance of an object can be computed by triangulation. However, the baseline of triangulation is very narrow since it is the distance of the reflected points on the mirrors. Hence, it is not practical to use this sensor to compute with accuracy the distance of an object. Instead, we classify objects into two categories, near and far objects.

For a mirror i, $i = 1, ..., M$, the mirror coordinate system originates at the optical center, the vertical axis z_i points towards its top along the symmetrical axis of the mirror, and the whole camera coordinate system coincides with the coordinate system

of the center mirror, see Fig.3.6. The shape of the surface of a paraboloidal mirror is described as the function of the spherical coordinate system:

$$\tau = \frac{r_i}{1 + \cos(\phi)}, \tag{3.1}$$

where r_i is the radius of curvature of the mirror i at its top, (τ, ϕ, θ) is a point on the surface of the paraboloidal mirror i in the spherical coordinate system, τ is the distance from the origin to the surface point, $0 \leq \phi \leq \pi$, $-\pi \leq \theta \leq \pi$. The top of the mirror has the coordinates $(\tau, \phi, \theta) = (\frac{r_i}{2}, 0, 0)$. An object point P from the ray direction (ϕ, θ), which is represented by $\boldsymbol{P} = (\sin(\phi)\cos(\theta), \sin(\phi)\sin(\theta), \cos(\phi))$, has the projection on the image plane with the coordinates:

$$\boldsymbol{x_i} = \{O_i^x + \frac{r_i \sin(\phi)\cos(\theta)}{1 + \cos(\phi)}, O_i^y + \frac{r_i \sin(\phi)\sin(\theta)}{1 + \cos(\phi)}\}, \tag{3.2}$$

where (O_i^x, O_i^y) is the center of an omnidirectional image from the mirror i, measured in pixel units is the result of the calibration process. Since we use the orthographic image sensor, the geometric parameter calibration of the paraboloidal mirrors is simple and has been reported in previous work [66]. Therefore it is not shown in this thesis.

Since the total size of the constructed compound mirror is less than 50 mm, the stereo baseline is very narrow, which means that the resolution of the computing distance is low. Therefore, we propose classifying the distance of the image features as either near or far, instead of computing an accurate distance. We have found this method to be useful and a small system is sufficient for estimating egomotion.

3.2.3 Classification of Near and Far Features

In this section, we describe the classification for an object point on one pair of mirrors, say mirror i and j, then describe the classification for multiple pairs of mirrors, since the sensor consist of seven mirrors.

We consider a situation in which an object is placed at an infinite distance from the sensor and observed in the images of two mirrors. As the object gets close to the

sensor along the ray of one of the mirrors, the projected image on the other mirror shifts along the epipolar line. This shift is called disparity. In this thesis, we consider an object to be far if the disparity is less than a given threshold.

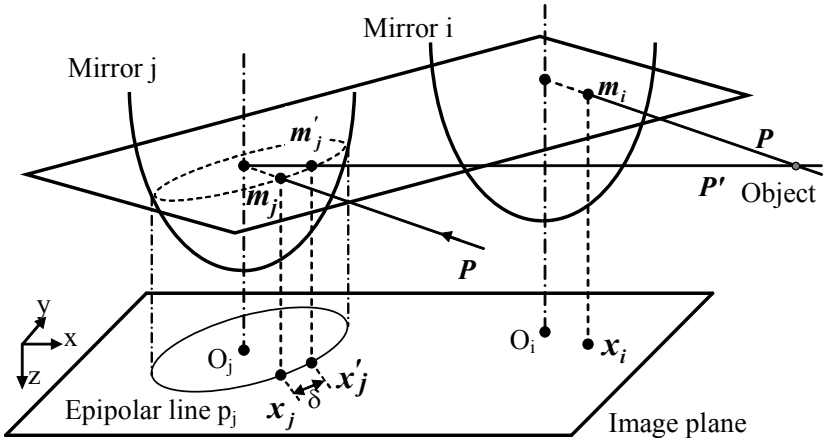


Figure 3.3: The ray directions reflected on one pair of compound paraboloidal mirrors (i and j) and the epipolar line.

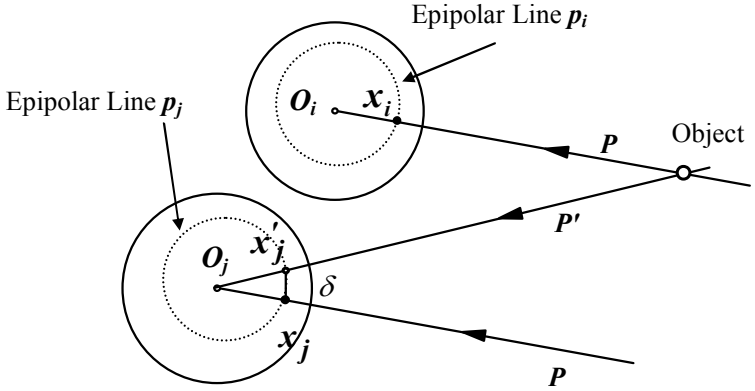


Figure 3.4: Projection of ray directions and epipolar lines on the image plane for one pair of compound mirror i and j.

Fig.3.3 shows an example of the epipolar line for the paraboloidal mirrors i and j. Their center points in the image plane are $\boldsymbol{O_i}$ and $\boldsymbol{O_j}$, respectively. If an object is at an infinite distance, the ray direction is $\boldsymbol{P}$. The rays are projected on $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ after being reflected, at $\boldsymbol{m_i}$ and $\boldsymbol{m_j}$ on the mirrors. If the object moves closer to mirror i along the direction $\boldsymbol{P}$, the ray direction to mirror j is $\boldsymbol{P'}$. Thus, the reflected and projected points become $\boldsymbol{m_j}'$ and $\boldsymbol{x_j}'$. $\boldsymbol{x_j}'$ is shifted δ pixels along the epipolar line.

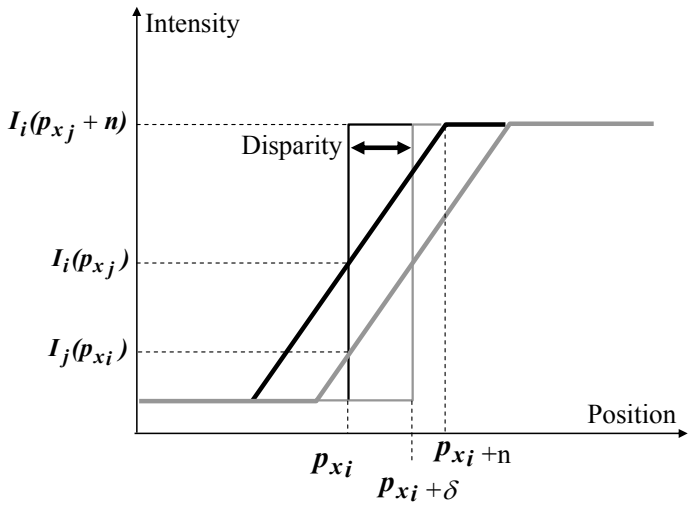


Figure 3.5: The disparity δ is computed using a gradient-based method after smoothing with a mean filter. Two intensity functions along the epipolar lines p_i and p_j are aligned at $p_{\boldsymbol{x_i}}$ and $p_{\boldsymbol{x_j}}$, $p_{\boldsymbol{x_i}} = p_{\boldsymbol{x_j}}$.

There are two such epipolar lines p_i and p_j for i and j, respectively. Fig.3.4 shows more detail of the projection on the image plane.

Since the proposed method is a narrow baseline stereo, the disparity δ is small if an object is at a practical distance. Therefore, we compute the disparity without searching corresponding points thus enabling real-time computation. Since we apply this method to feature points detected by a feature detector, we assume that the intensity around $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ along their epipolar lines are step functions defined as:

$$\begin{cases} I_i(p) = H(p - p_{\boldsymbol{x_i}}) \\ I_j(p) = H(p - p_{\boldsymbol{x_j}} - \delta), \end{cases} \tag{3.3}$$

where $H(x)$ is a step function, that is 1 if $x \geq 0$ and 0 otherwise, p indicates the position in the epipolar line and $p_{\boldsymbol{x_i}}$, $p_{\boldsymbol{x_j}}$ are the position of $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ in their epipolar lines, respectively.

The Lucas–Kanade method [70] computes disparity from the gradient of intensity without searching for correspondences. However, the gradient-based method cannot be applied directly to the case assumed in (3.3). Therefore, we filter the images

before computing disparity. Our method smoothes the intensity along the epipolar line using a 1D mean filter. Fig.3.5 shows an example of this. The thin lines are the original intensities of two images along the epipolar lines. The black and gray lines denote images i and j. The shift between the black and gray lines indicates the disparity. After applying a mean filter of window size $2n+1$ to $I_i(p)$ and $I_j(p)$, we obtain the smoothed functions indicated by the thick black and gray lines. Then, we can compute the disparity from the gradient of the smoothed functions as follows:

$$D_{i,j}(\boldsymbol{P}, n) = \frac{I_i(p\boldsymbol{x_i}) - I_j(p\boldsymbol{x_j})}{I_i(p\boldsymbol{x_i} + n) - I_i(p\boldsymbol{x_i})}. \tag{3.4}$$

If the disparity δ is less than n, $D_{i,j}(\boldsymbol{P}, n) < 1$, otherwise $D_{i,j}(\boldsymbol{P}, n) \geq 1$. Therefore, we classify a feature point in the direction $\boldsymbol{P}$ according to the following criterion:

$$\begin{cases} \text{Far feature} & \text{if} \quad D_{i,j}(\boldsymbol{P}, n) < 1 \\ \text{Near feature} & \text{otherwise.} \end{cases} \tag{3.5}$$

The classification criterion is adjusted by the window size of the mean filter n. If a user wants to discriminate features at a certain distance, the disparity d corresponding to the distance can be computed from the optical geometry of the sensor. Thus, the classification is achieved by setting $n = d$. Since d differs with respect to the position in the image of our sensor, we compute d that corresponds to the distance of discrimination for each pixel of the image. In implementation, all the positions in (3.4) can be computed off-line and stored in a look-up table. Then in run-time, the disparity can be checked quickly.

Since our sensor has seven mirrors, it can compute disparity by using different pairs (i, j) of mirrors. If a feature is observed by multiple pairs, it is determined to be a near one if it is classified as near by more than one of these pairs.

3.2.4 Separate Estimation of Rotation and Translation

Once the coordinate system has been located at the optical center O of the center mirror as shown in Fig.3.6, the surrounding scenery moves around the sensor. If the

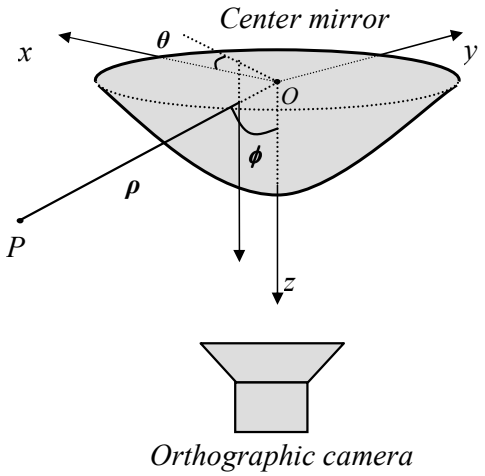


Figure 3.6: Camera coordinate system with origin at the center of central mirror.

position of object point P is $\rho\boldsymbol{P}$, where ρ is the distance from P to the origin and $\boldsymbol{P} = (\sin\phi\cos\theta, \sin\phi\sin\theta, \cos\phi)$ is the direction from the origin to P. The position $\rho'\boldsymbol{P'}$ after rigid transformation is given by:

$$\rho'\boldsymbol{P'} = \boldsymbol{R}\rho\boldsymbol{P} + \boldsymbol{T}, \tag{3.6}$$

where $\boldsymbol{P}$, $\boldsymbol{P'}$ are coordinates on the unit sphere or the directions of P before and after the motion, ρ and ρ' are its depths before and after the motion, $\boldsymbol{R}$ is the rotation matrix and $\boldsymbol{T}$ the translation vector.

It is noted that the depth, ρ and ρ', are not known from the captured image, while $\boldsymbol{P}$ and $\boldsymbol{P'}$ are known. By subdividing (3.6) by ρ', we obtain

$$\boldsymbol{P'} = \frac{\rho}{\rho'}\boldsymbol{R}\boldsymbol{P} + \frac{\boldsymbol{T}}{\rho'}. \tag{3.7}$$

From (3.7) we see that if the distance ρ' is much larger than $\boldsymbol{T}$, we can ignore the term $\frac{\boldsymbol{T}}{\rho'}$. Therefore, if P is a far feature, the motion is determined only by rotation and

$\frac{\rho}{\rho'} \approx 1$. Equation (3.7) is then simplified as follows:

$$\boldsymbol{P'} = \boldsymbol{R}\boldsymbol{P}. \tag{3.8}$$

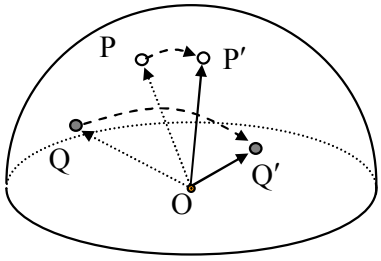


Figure 3.7: Estimate rotation from motion of 2 points.

Consequently, we can estimate rotation $\boldsymbol{R}$ separately by omitting the translation $\boldsymbol{T}$ from the equation. Translation is then estimated after eliminating the estimated rotation from the motion.

Computing Rotation Independently

In conventional methods, at least five pairs of corresponding feature points between two images are required to estimate rotation and translation. But since we can distinguish far feature points, we can compute rotation separately. We can thus reduce the required number of feature points to two pairs.

In Fig.3.7, $\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{P'}$ and $\boldsymbol{Q'}$ are the projected points of the two points P and Q before and after a rotation, respectively. If we consider the cross-product vector $\boldsymbol{n}$ of $\boldsymbol{P}$ and $\boldsymbol{Q}$, the cross-product vector $\boldsymbol{n'}$ of $\boldsymbol{P'}$ and $\boldsymbol{Q'}$ is given by applying the same rotation to $\boldsymbol{n}$ as follows:

$$\boldsymbol{P'} = \boldsymbol{RP} \quad \boldsymbol{Q'} = \boldsymbol{RQ} \quad \boldsymbol{n'} = \boldsymbol{Rn}, \tag{3.9}$$

where the lengths of $\boldsymbol{n}$ and $\boldsymbol{n'}$ are normalized to the unit length. Then the rotation matrix $\boldsymbol{R}$ is computed from the three pairs of vectors on the unit sphere:

$$\boldsymbol{R} = [\boldsymbol{P'}\ \boldsymbol{Q'}\ \boldsymbol{n'}]\,[\boldsymbol{P}\ \boldsymbol{Q}\ \boldsymbol{n}]^{-1}, \tag{3.10}$$

and $\boldsymbol{R}$ is normalized, $|\boldsymbol{R}| = 1$. In the estimation algorithm using RANSAC, this computation is used to initialize the rotation model, which needs only two points of correspondence in two consecutive frames.

Having finished the random sampling using the RANSAC method, the best rotation matrix $\boldsymbol{R}_{est}$ and a set S_R of k feature correspondence pairs between two video frames are outputted:

$$\boldsymbol{P'}^i = \boldsymbol{R}_{est}\boldsymbol{P}^i, \tag{3.11}$$

where $i = 1..k$, $k > 3$. We then solve the over-determined equation system (3.11) for three rotation angles. This equation system can be solved using the least mean squares method (LMS) with the minimization function:

$$\min_{\phi,\theta,\psi} \sum_{\boldsymbol{P}\in S_R} (\boldsymbol{P'} - \boldsymbol{R}(\phi,\theta,\psi)\boldsymbol{P})^T(\boldsymbol{P'} - \boldsymbol{R}(\phi,\theta,\psi)\boldsymbol{P}), \tag{3.12}$$

where $\boldsymbol{R}(\theta,\phi,\psi)$ is a rotation matrix built from three angles θ,ϕ,ψ. Since the minimization is nonlinear, we used the Levenberg-Marquardt minimization with the initial parameters given by (0,0,0). After the minimization, the output rotation matrix, $\boldsymbol{R}(\theta_c,\phi_c,\psi_c)$, clearly satisfies the conditions of a rotation matrix, the orthogonality condition and its determinant being +1.

Computing Translation after Eliminating Rotation

Once rotation has been estimated, we can eliminate the rotation of features. Therefore, in this section we assume that no rotation occurs between two succeeding views. The translation vector can now be estimated from the motion of two near feature points.

Consider a situation in which the camera moves while observing two near feature points P and Q as shown in Fig.3.8. These points are projected, onto $\boldsymbol{P}$ and $\boldsymbol{Q}$ in the previous video frame, and $\boldsymbol{P'}$ and $\boldsymbol{Q'}$ in the current video frame. Now, we consider two planes, π_P and π_Q. π_P is created by three points, O, O', and P. Similarly, π_Q is created by O, O', and Q. Since the translation vector $\boldsymbol{T}$ is the intersection of the two planes, $\boldsymbol{T}$ is computed as follows:

$$\boldsymbol{T} = (\boldsymbol{P} \times \boldsymbol{P'}) \times (\boldsymbol{Q} \times \boldsymbol{Q'}). \tag{3.13}$$

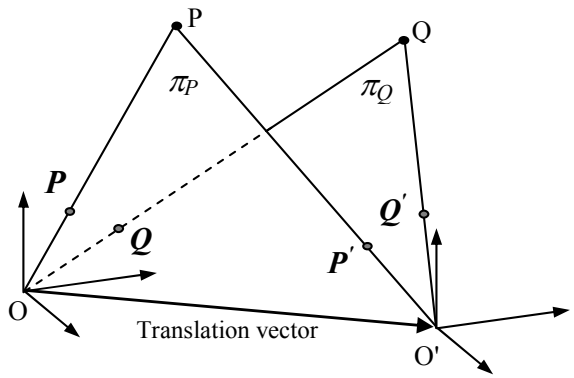


Figure 3.8: O and O' are located on the intersection of two epipolar planes.

Since the motion of the feature points in the image occurs as a result of the translation of the camera, the projection of the translation vector and the motion vector of the features on the image plane must be opposite. We use this criterion to adjust the direction of the translation vector.

3.2.5 RANSAC-based Methods to Estimate Rotation and Translation

The RANSAC-based algorithms are implemented similarly for both rotation and translation estimation. For both algorithms, a random sample is taken of two features in the previous frame and two more in the current frame. The algorithms simultaneously find the motion parameters and the correspondence of image features. The difference is the feature used for estimation. For rotation estimation, near features which do not hold the rotation represented by (3.8) are filtered out by our compound sensor. Since it is not feasible to use far features for estimating the translation, these should be excluded from the translation estimation and only near feature points should be used for this task.

The RANSAC estimation of both rotation and translation is summarized as follows:

1. Randomly select two features in the first video frame and two image features in the second video frame to assign two pairs of correspondence.

2. Calculate the motion parameters (rotation matrix $\boldsymbol{R}$, or translation vector $\boldsymbol{T}$).

3. Count the supporting pairs of correspondence that match the estimated parameters.

4. Record the current best solution with the maximum number of supporting pairs.

5. If not stopped, return to the first step.

In our implementation, the termination criterion for RANSAC sampling is processing time.

For estimating rotation, the rotation matrix $\boldsymbol{R}$ is computed as shown in Section 3.2.4. Counting supporting pairs for the rotation $\boldsymbol{R}$ is done by applying the rotation of far features in the previous frame and matching these features to the features in the current frame. If $\boldsymbol{P}$ is a far feature in the previous frame and $\hat{\boldsymbol{P}}$ is the position after applying the estimated rotation, we compute the angle between $\hat{\boldsymbol{P}}$ and all far features in the current frame located near $\hat{\boldsymbol{P}}$. If the angle between $\hat{\boldsymbol{P}}$ and $\boldsymbol{P'}$ is the smallest and less than a threshold, we count $(\boldsymbol{P}, \boldsymbol{P'})$ as a supporting pair of correspondence.

For estimating translation, the translation vector $\boldsymbol{T}$ is computed as shown in Section 3.2.4 for each random sample. Then the number of supporting pairs for each translation vector is counted to select the best translation vector. From a near feature point $\boldsymbol{P}$ in the previous frame and $\boldsymbol{T}$, an epipolar plane $\pi(\boldsymbol{P}, \boldsymbol{T})$ can be computed. Similarly, an epipolar plane $\pi(\boldsymbol{P'}, \boldsymbol{T})$ can be computed for each near feature $\boldsymbol{P'}$ of the current frame. If the angle between the normal vectors of $\pi(\boldsymbol{P}, \boldsymbol{T})$ and $\pi(\boldsymbol{P'}, \boldsymbol{T})$ is the smallest and less than a threshold, we count $(\boldsymbol{P}, \boldsymbol{P'})$ as a supporting pair of correspondence. Therefore, counting the supporting pairs for the translation vector is a problem of stereo matching.

Then we theoretically compare the computation cost of proposed method and the well-known seven-point algorithm. In the proposed approach, the correspondence and motion is determined in a RANSAC procedure. This approach cannot be applied to the well-known seven-point algorithm, because the computational cost is prohibitive. Assuming a problem that has no prior knowledge about feature correspondences, we formulate the computational cost of our proposed algorithm compared with that

of the seven-point algorithm which estimates the essential matrix without feature correspondence. For a RANSAC algorithm, the number of iterations required before the estimation obtains a correct sample is:

$$k = \frac{\log z}{\log(1 - w^n)}, \tag{3.14}$$

where z is the probability of seeing only bad samples, w is the fraction of inliers among all data points and n is the number of data points for one sample; refer to the book [28] for the details. Let p_{in} be the probability of selecting an inlier $\boldsymbol{P}$ in the previous frame, and p_{sup} be the probability of selecting a correct supporter $\boldsymbol{P'}$ in the current frame. $\boldsymbol{P'}$ can be found in the region around the location of $\boldsymbol{P}$ and the region contains about m features (in the current frame), then $p_{sup} = \frac{1}{m}$. The value of p_{sup} is similar for both the proposed algorithm and the seven-point algorithm, however the value of p_{in} is different. In our proposed method, the inliers are far features at a certain distance from the sensor, and far features are classified by the compound sensor. Therefore, not all far features classified by the sensor are inliers. Thus the average value of p_{in} in the proposed algorithm is smaller than that in the seven-point algorithm. The probability of selecting the correct correspondence pair $(\boldsymbol{P}, \boldsymbol{P'})$ is $w = p_{in}p_{sup}$. For the proposed algorithm, the probability of selecting two pairs of feature correspondence is $w_2 = (p_{2in}p_{2sup})^2$. For the seven-point algorithm the probability of selecting seven pairs of feature correspondence is $w_7 = (p_{7in}p_{7sup})^7$.

To ensure the same possibility of obtaining a correct sample, the value of z must be equal for both algorithms, therefore, the number of required iterations must vary to meet the requirement. We simulate how these two algorithms require the number of iterations in the case that $p_{7in} = 1.5p_{2in}$ and $z = 0.9$. Details of the required number of iterations are described in Fig.3.9, which shows that the seven-point algorithm requires many more iterations compared with the proposed algorithm. For example, if m=10, there are about 10 features in the current frame around the location of an inlier in the previous frame, and therefore $p_{sup} = 0.1$, and k_7= 294042.This can be compared with the proposed algorithm giving k_2= 16.4 only. From these analyses, the proposed method drastically decreases the computation cost owing to the separation

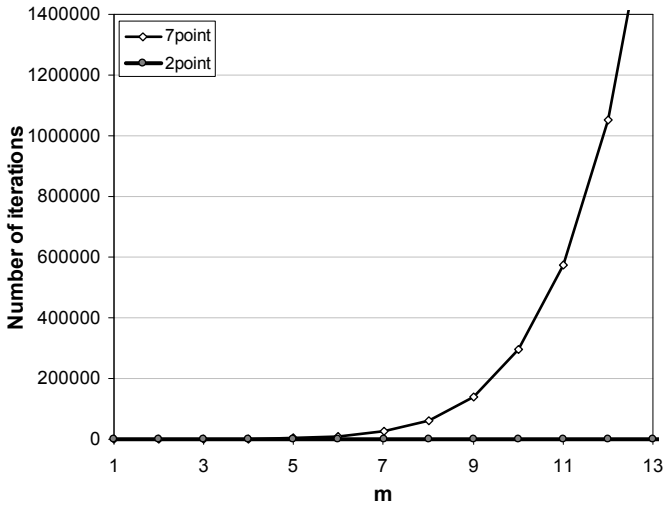


Figure 3.9: Number of required iterations for the seven-point algorithm without feature correspondence and the proposed algorithm.

of the camera motion.

3.2.6 Optimizing the Solution Using Epipolar Constraint and All Supporting Pairs

After the estimation using the RANSAC-based method, the approximate rotation matrix $\boldsymbol{R}(\phi_c, \theta_c, \psi_c)$ and translation vector $\boldsymbol{T}_{est}$ are acquired. An optimization is performed to find the best rotation angles and translation vector using all the supporting pairs that have been obtained from the RANSAC-based method. The optimized parameters are estimated by minimizing the following epipolar constraint function:

$$\min_{\theta,\phi,\psi,tx,ty,tz} \sum_{(\boldsymbol{P},\boldsymbol{P'})\in S} (\boldsymbol{P'}E\boldsymbol{P})^2, \tag{3.15}$$

where S is the set of all pairs of feature points that support the best solution estimated using the RANSAC-based method, and $(\boldsymbol{P},\boldsymbol{P'})$ is one of these pairs. E is the essential matrix computed as $E = [\boldsymbol{T}]_\times \boldsymbol{R}(\theta, \phi, \psi)$, where $\boldsymbol{R}(\theta, \phi, \psi)$ is the rotation matrix built from the rotation angles (θ, ϕ, ψ), and $[\boldsymbol{T}]_\times$ is the matrix representation of the cross product with $\boldsymbol{T} = (t_x, t_y, t_z)$; see [67] for further details. We also used the

Levenberg-Marquardt minimization with the initial parameters given by $\boldsymbol{R}(\theta_c, \phi_c, \psi_c)$ and $\boldsymbol{T}_{est}$. After the optimization, the rotation matrix is $\boldsymbol{R}(\theta_{opt}, \phi_{opt}, \psi_{opt})$, which obviously meets the conditions of a rotation matrix.

3.3 Motion Estimation Experiments

In our experiments, we used the compound omnidirectional sensor that is shown in Fig.3.2. The compound sensor is mounted on a system with two rotary stages and a 50 cm translation stage (Fig.3.10). The ω_θ rotation is controlled by one rotation controller on the z-axis, and the ω_ϕ rotation on the y-axis is controlled by the other. The dimensional translation of the camera system is controlled by a translation controller. The vision sensor is a 1600×1200 pixel CCD camera (Scorpion: Point Grey Research) with a telecentric lens. The parameters of the compound sensor and its parameters after the calibration are shown in Table 3.1. In the experiments, the maximum distance for classification by this compound sensor is about 3 m. The proposed method is processed by a PC with a Pentium D 3.2GHz processor. OpenCV [72] is used for image processing including the Harris [57] feature detection procedure.

Table 3.1: Parameters of the compound sensor after calibration.

Parameter		Design (mm)	Actual size in image (pixel)
Total diameter	D	43	813
Center mirror radius	d_1	25	473
Side mirror radius	d_i	13	239
Center mirror radius of curvature	r_1	17.5	331
Side mirror radius of curvature	r_i	8.7	165
Largest baseline (side mirrors)	$b_{i,i+3}$	30	567
Smallest baseline (side mirrors)	$b_{i,i+1}$	15	283
Side mirror - center mirror baseline	$b_{1,i}$	19.5	369

The experiments were carried out in various environments to evaluate accuracy with respect to processing time and camera motion. Our experimental results were compared with the results from the essential matrix-based solution. We implemented the seven-point algorithm based on the work of Torr [73] that estimates the essential matrix using RANSAC and the multi-resolution Kanade-Lucas-Tomasi (KLT) fea-

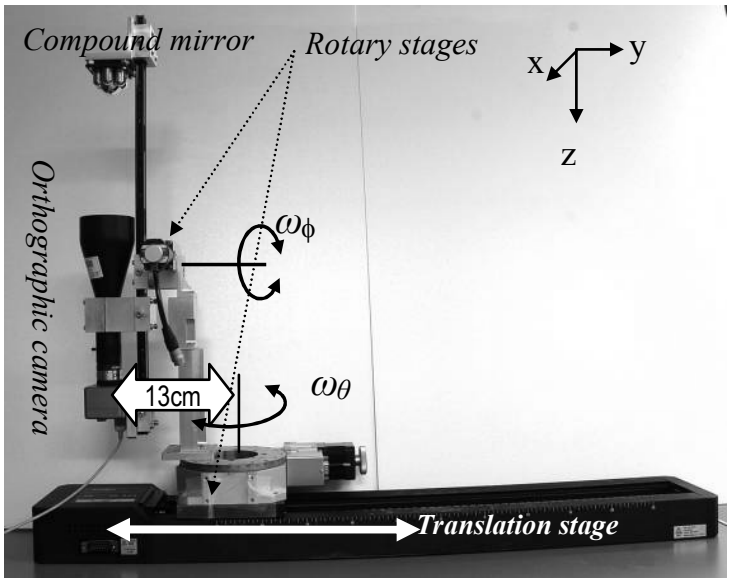


Figure 3.10: The evaluation system. Rotary stages and the vision sensor are mounted on the translation stage.

ture tracker [56] implemented in OpenCV [72]. Motion parameters are also tuned by using the same optimization method as our proposed algorithm. In this method, which we refer to as 7ALGRANSAC, the feature correspondences are given by the feature tracker including outlier correspondences. While it is possible to implement the seven-point algorithm using RANSAC without knowing the correspondences, it is very time-consuming to sample a set of 7 correspondences between two consecutive frames. Consequently, we do not cover this implementation in the thesis. We also compared the performance of the proposed algorithm with and without feature classification using the compound sensor to show the effectiveness of the near/far feature classification procedure. The detailed results of these experiments are described in the following sections, which show the averages of the frame-by-frame estimation errors for each video sequence.

Several types of environmental data were captured in our experiments to validate our algorithm. We extracted 200 features from each frame using a Harris feature detector. The whole sensor image is used for feature classification; and the big omnidirectional image at the center is used for feature detection. The experiment showed

that for each frame the Harris feature detector needed 0.066 sec to extract 200 features.

We also set up the parameters so that our algorithm could cope with a maximum rotation velocity of 31 degrees/frame and a translation velocity of 8.5 cm/frame. For our algorithm, the processing time includes feature extraction, feature classification and RANSAC motion estimation, and Levenberg Marquardt optimization of the motion parameters. By contrast the processing time for the 7ALGRANSAC process includes initial feature detection, frame by frame feature tracking, RANSAC estimation of the essential matrix, motion parameter extraction and optimization using the Levenberg Marquardt method.

3.3.1 Error Definitions

Errors of motion are defined for motion between a pair of video frames. To evaluate the rotation error, we first compute the residual rotation after eliminating the estimated motion $\hat{\boldsymbol{R}}$ with the true motion $\boldsymbol{R}_{tr}$ from the rotary stage controller:

$$\boldsymbol{R}_{er} = \hat{\boldsymbol{R}}\boldsymbol{R}_{tr}^{-1} \tag{3.16}$$

This is the error of the estimated rotation that is represented by a matrix. If the estimation is perfect, the matrix $\boldsymbol{R}_{er}$ is the identity rotation matrix. The difference between $\boldsymbol{R}_{er}$ and the identity rotation matrix $\boldsymbol{I}$ is assumed to be the error of estimation. We evaluate the rotation error by a Frobenius norm of the matrix $(\boldsymbol{R}_{er} - \boldsymbol{I})$ as follows:

$$\sqrt{\sum_{i=1,j=1}^{3,3} (\boldsymbol{R}_{er,ij} - \boldsymbol{I}_{ij})^2}. \tag{3.17}$$

If the error is small, it can be regarded as the angle error in radians.

The translation error is the angular difference between the normalized estimated translation vector and the normalized ground-truth translation vector, because our method estimates translation without magnitude. We call this the directional translation error, which is also measured in radians.

3.3.2 Experiments with Different Ratios of Near/Far Features

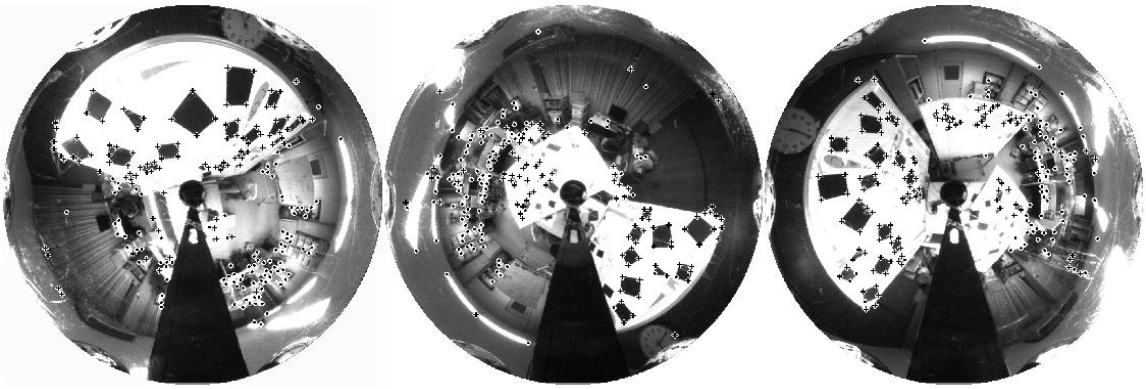


Figure 3.11: Example input images (each of them is the big omnidirectional image at center of a sensor image) of FAR, MID, $NEAR$ (from left to right) with increasing near/far ratios.

First experimental data were captured for three different ratios of near/far features. These data sets are labelled FAR, MID, $NEAR$, and are shown in Fig.3.11, with a decreasing number of far features (or an increasing number of near features). Near features were situated within 3 m of the sensor, whereas far features were located at distances ranging from 4 m to about 10 m. Motion of the sensor was controlled by only the rotary stage ω_θ on the Oz axis. While the sensor was rotated, it was also translated. The motion path was circular with a radius of 13 cm.

For these data sets, experimental results were obtained for feature classification, the convergence of rotation and translation estimation.

Feature Classification

First, we tested the accuracy of classification using the proposed sensor. We manually checked the classified results with the ground-truth and summarized the results of feature classification using 10 random frames. For the ground-truth, a feature is classified as near if the distance is less than 3 m; otherwise it is classified as

Table 3.2: Results of misclassification of near/far features.

	FAR	MID	$NEAR$
Near→ Far	4.4%	8.4%	9.8%
Far→ Near	2.1%	9.0%	5.3%
Actual number of near features	92	105	112
Actual number of far features	108	95	88

far. Table 3.2 gives a summary of feature classification for the three data sets, which shows that the accuracy of feature classification is more than 90%.

Convergence of RANSAC for Estimating Rotation

Next, we evaluate the accuracy of rotation estimation with respect to processing time. The processing time includes the Harris feature detection with and without feature classification and the RANSAC matching time for estimating rotation. The camera translation and rotation angles were fixed as the control rotation velocity $\omega_\theta =$ 10 degrees/frame. We compared the convergence of estimating rotation with feature classification (denoted as CLASSIFIED) and without feature classification (denoted as UNCLASSIFIED). The results are shown in Fig.3.12.

Because most outliers for estimating rotation were removed by classification, the processing time was reduced significantly for the CLASSIFIED case compared with that for the UNCLASSIFIED case. The processing time of 0.2 sec is reasonable for use in real applications with acceptable accuracy. Since the far features were not truly at infinity and the rotation matrix is computed from four random points on two images and no optimization was performed after random sampling, some error existed in the estimation regardless of the processing time.

Convergence of RANSAC for Estimating Translation

The experiments for the convergence of translation estimation were carried out with the same rotation estimation, which in this case was ground-truth rotation. The camera translation and rotation velocities were fixed with the control rotation velocity $\omega_\theta = 10$ degrees/frame. The processing time consisted of the Harris feature

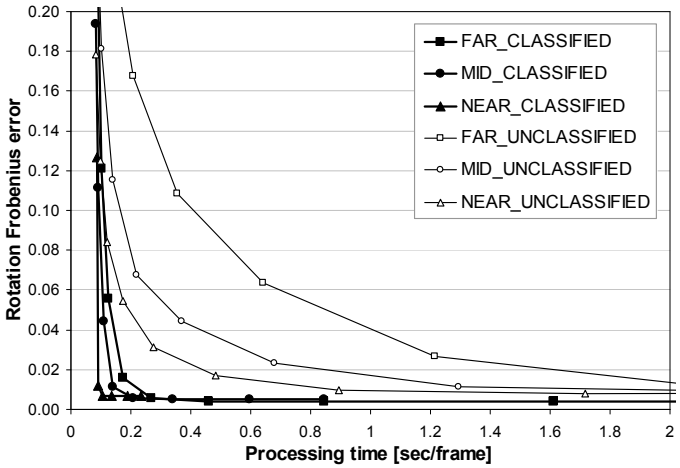


Figure 3.12: Comparison of the convergence of estimating rotation with/without feature classification.

detection with/without feature classification and the RANSAC matching time for estimating translation. Fig.3.13 shows the results of convergence for both classified and unclassified features. The results show that the translation estimation with only near features converged much faster than in the unclassified case. The results are similar to those in Fig.3.12 showing that the classification of features is effective. Since the translation vector is computed from four random points on two images and no optimization was performed after random sampling, some error existed in the estimation regardless of the processing time.

From the experiments on the convergence of rotation and translation estimation using classified and unclassified features, we can see that with classification of features, the egomotion (rotation and translation) computation is much faster than is the case without feature classification but with the same processing time.

3.3.3 Overall Performance Experiments

In these experiments, the performance of the proposed algorithm was tested with various real data. Two indoor video sequences and one outdoor scene were captured, as shown in Figs.3.14, 3.15 and 3.22, respectively. For these videos, the sensor was moved by both a translation stage controller and a rotation stage controller instead

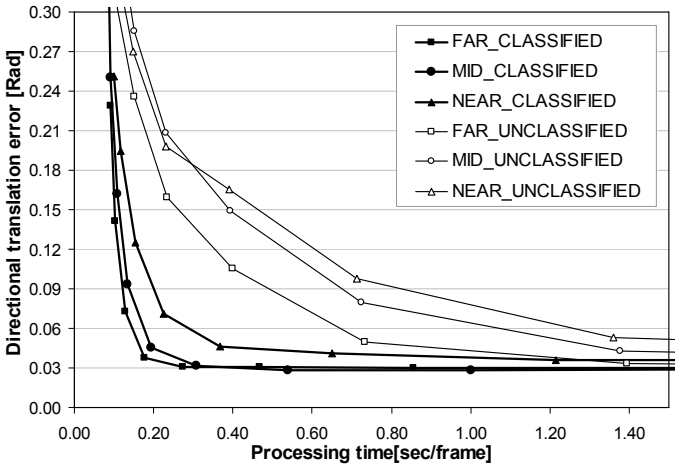


Figure 3.13: Comparison of the convergence of translation estimation with/without feature classification using ground-truth rotation.

of using only a rotation stage controller as in the previous data sets. The speed of the translation stage was fixed at 5 cm/frame, while the rotation speed ω_θ varied between 12 and 30 degrees/frame; one video was taken at each rotation speed. In the experiments, the processing time allowed for each algorithm was 0.5 sec/frame for terminating the RANSAC iteration.

Indoor Scenes

Examples of feature classification in the two scenes are shown in Figs.3.14 and 3.15, while the distributions of the distances from the sensor to the feature points are shown in Figs.3.16 and 3.17 for the two scenes, respectively. The distance distributions are presented as the distance histograms with the bin-width of 10 cm. The distributions of the far feature points for these two scenes are similar, whereas the distributions of near feature points differ. The near feature points in the first scene are closer to the sensor. The distances were computed using stereo matching for the central omnidirectional images and the baseline connecting two ends of the translation stage, the length of which is 50 cm. A series of captures provided us the average distribution as shown in Figs.3.16 and 3.17.

The experimental results are described in Figs.3.18 and 3.19 for the first indoor scene

Figure 3.14: Indoor scene 1.

Figure 3.15: Indoor scene 2.

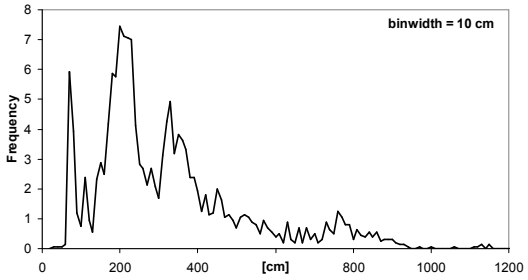


Figure 3.16: Indoor scene 1: histogram of feature distances.

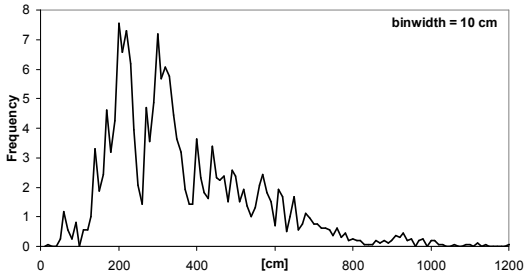


Figure 3.17: Indoor scene 2: histogram of feature distances.

and Figs.3.20 and 3.21 for the second indoor scene. For smaller motion, the performance of 7ALGRANSAC is better than the proposed algorithm; however for larger motion, the proposed algorithm gives the better results. Since 7ALGRANSAC relies on the feature correspondences from a feature tracker, the estimation accuracy of 7ALGRANSAC decreases with greater motion, as many outliers are included in the correspondences. Since our algorithm does not rely on correspondences from a feature tracker, the performance is robust for any amount of motion. Rotation estimation is a little less accurate with greater motion. This can be explained by the larger translation of the sensor, because our algorithm assumes that the distance to far features

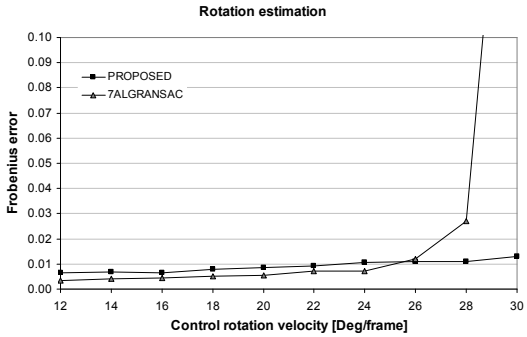


Figure 3.18: Indoor scene 1: Frobenius error for rotation estimation.

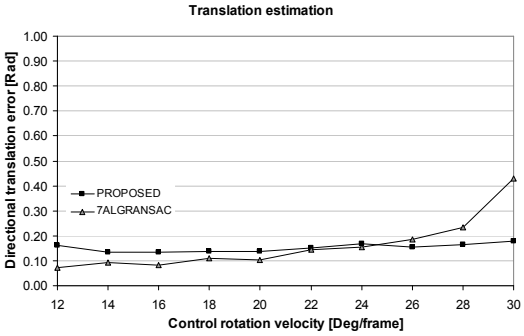


Figure 3.19: Indoor scene 1: directional translation error for translation estimation.

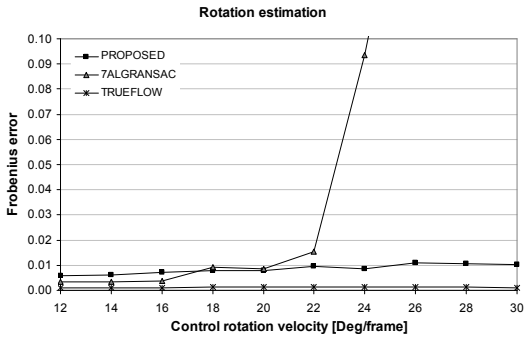


Figure 3.20: Indoor scene 2: Frobenius error for rotation estimation.

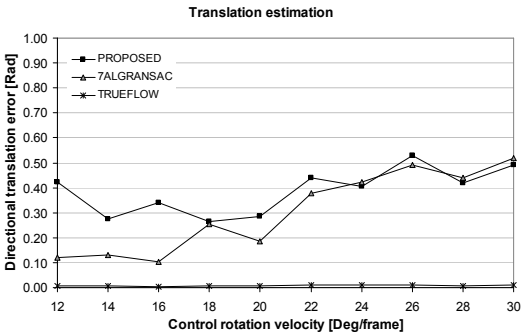


Figure 3.21: Indoor scene 2: directional translation error for translation estimation.

is much larger than the translation speed of the sensor.

We can also see that with the same camera motion, the translation estimation in the first scene in Fig.3.19 is better than that in Fig.3.21, while the rotation estimation accuracy is similar in two scenes. The reason for the difference is that the near feature points in the first indoor scene are distributed closer to the sensor, while the distributions of the far feature points are similar. A similar variation in the accuracy of 7ALGRANSAC was observed for these scenes.

+ Cross point, feature classified as near
• Dot point, feature classified as far

Figure 3.22: Outdoor scene.

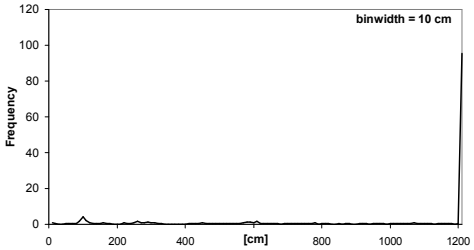


Figure 3.23: Outdoor scene: histogram of feature distances.

Outdoor Scene

In a outdoor scene, far feature points are significantly farther from the sensor than those in the indoor scenes, and there are fewer near feature points than in either of the indoor scene. One of the outdoor scenes is shown in Fig.3.22 and the distribution of feature distances from the sensor is shown in Fig.3.23 as a distance histogram with a bin-width of 10 cm. For this scene, most feature points are very far from the sensor, with few near feature points located around the sensor and on the ground. The experimental results are described in Figs.3.24 and 3.25. The results are similar to those of the indoor scenes. For slow motion, the proposed algorithm and 7ALGRANSAC produced similar results; however the proposed algorithm gave better results for fast motion. The proposed algorithm produced robust results for all variations in motion speed.

We can also see that for the outdoor scene, the far feature points are farther away, the approximation error is lower, and we have a more accurate rotation estimate. And since there are very few near feature points, the translation estimation accuracy for both algorithms is not as good as that in the indoor scenes. Due to the few near feature points in this scene, 7ALGRANSAC did not work well. 7ALGRANSAC

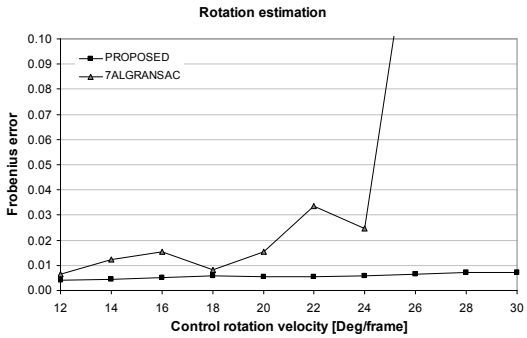


Figure 3.24: Outdoor scene: Frobenius error for rotation estimation.

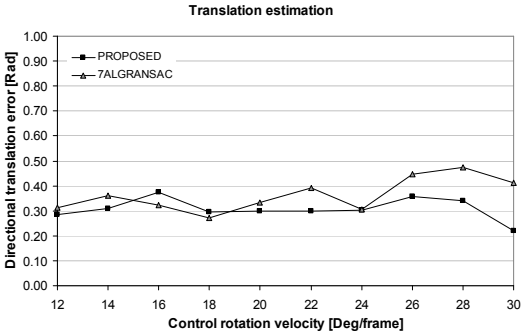


Figure 3.25: Outdoor scene: directional translation error for translation estimation.

simultaneously estimates rotation and translation, and therefore if translation is not accurate, rotation is directly affected.

3.3.4 Discussion

The proposed algorithm relies on separate camera motion estimations. The rotation is estimated using far feature points while the translation is estimated using near feature points. Far and near feature points are classified using the proposed compound omnidirectional sensor. There are some exceptional cases in which the proposed algorithm does not work well, but these are not seen as a disadvantage of the proposed method. The first situation arises when the scene is small and all feature points are classified as near features, with the results that we have no far features for estimating camera rotation. However, for fast and sudden camera motion in a small environment, the previous egomotion algorithms also have problems in computing the feature correspondence. This needs to be addressed in our future research. The second situation arises when all the feature points are very far from the sensor and are classified as far features. In this situation, the rotation can be accurately estimated by our algorithm, however, we have no near features for estimating camera translation. Previous egomotion algorithms also face the same problem because the translation vector is relatively too small to be estimated effectively. The algorithms that simultaneously estimate rotation and translation do not work well in this sit-

uation because an inaccurate translation estimation directly influences the rotation accuracy. However, algorithms that separate rotation and translation, such as the proposed algorithm, work better.

The distances of feature points can affect the accuracy of the estimation. For translation estimation, this influence is well-known for previous algorithms and the proposed algorithm. If feature points are relatively closer to the sensor, then higher accuracy of translation we can get and otherwise. However, for rotation estimation, we approximate the rotation by the motion of far feature points. The farther the distances of far features, the higher accuracy we have.

In the current algorithm, we only use the geometry constraint for computing the camera motion. Obviously, if we can apply the similarity constraint of feature points the results can be significantly improved. Some feature descriptor such as SIFT[71] can be used in such a situation.

3.4 Chapter Conclusion

We have proposed a new method for estimating egomotion which applies the RANSAC process. The algorithm has demonstrated our idea of pre-filtering the outlier model before applying a robust estimator perform the estimation when data contains multiple models. This preprocessing helps the robust estimators work efficiently. Using the compound omnidirectional sensor, image features are classified into near or far features. The rotation of the camera is estimated using only the far features, since the motion of far features in the images is modeled solely by rotation. After estimating the rotation, the translation is estimated using only the near features. Therefore, only two pairs of features are required to estimate either rotation or translation, whereas the seven-point algorithm requires seven pairs of features. Because of this reduction in computational complexity, the proposed method can work in real time without being given correspondences. Consequently, it can compute large camera motion since it does not assume small motion is required to find correspondences by a conventional feature tracker.

Although, we have not found the general framework to work with outlier models

like RANSAC to detect outlier data points, we found that it can be solved in some specific problems.

Chapter 4

Conclusions and Future Works

We propose two step framework for outlier detection and parameter estimation for multi-modeled/structured data. Firstly, the pre-filtering eliminates the outliers that are easily seen by some mean such as the sensor. The pre-filtering is used to reduce the workload for robust estimator, especially in the case the actual inlier distribution and outlier distribution overlap and it is impossible for a robust estimator to distinguish inliers/outliers. To demonstrate the pre-filtering technique, we have proposed a new egomotion estimation algorithm that works efficiently by classifying the near and far image features. For estimating camera rotation, near features become outliers and for estimating camera translation, far features are useless since they support any translation direction. The experiments show that the classification (or the pre-filtering technique) works well to improve the efficiency of RANSAC.

Secondly, we have proposed two novel adaptive-scale robust estimators (FITSAC1 and FITSAC2) for the estimation problem in computer vision that deals with data with high outlier rates and multiple structures. Depending on the specific problem, the distribution model of residuals is analyzed using that useful constraint, the residual function. The analysis is feasible and simple, and simulation of the residual distribution model can always be performed. The advantage of this approach is that it estimates the inlier scale correctly and therefore improves robustness. The adaptive smoothing parameter efficiently help FITSAC2 work robustly in various situation without any support from user. The proposed robust estimators were positively vali-

dated through experiments with various conditions and real estimation problems. The use of the constraint from the residual function in the robust estimator is effective for improving the robustness and detection of inliers.

In the future works, we would like to continue to look for a general and effective solution for pre-filtering technique for multi-modeled/structured data to help a robust estimator work better.

Bibliography

[1] A. Hornberg, Handbook of Machine Vision, Wiley VCH, 2006.

[2] A. J. Lacey, N. Pinitkarn, N. A. Thacker, An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration, British Machine Vision Conference, 2008.

[3] R.I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2nd Edition, 2004.

[4] T. Okabe, Y. Sato, Object Recognition Based on Photometric Alignment Using RANSAC, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 221–228, 2003.

[5] A. Hoover, J.B. Gillian, X. Jiang, P.J. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D.W. Eggert, A. Fitzgibbon, and R.B. Fisher, ?n Experimental Comparison of Range Image Segmentation Algorithms, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, no. 7, pp. 673–689, 1996.

[6] H. Wang, D. Suter, Mdpe: a very robust estimator for model fitting and range image segmentation, International Journal of Computer Vision (IJCV), vol. 59, no. 2, pp. 139–166, 2004.

[7] A.B. Hadiashar, D. Suter, Robust motion segmentation using rank ordering estimators, Asian Conference on Computer Vision, pp. 599–606, 2006.

[8] R. Tron, R. Vidal, A Benchmark for the Comparison of 3-D Motion Segmenta-

tion Algorithms, IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8, 2007.

[9] P.J. Rousseeuw and A. Leroy. Robust Regression and Outlier Detection. John Wiley & Sons, New York. 1987.

[10] P. J. Huber, Robust Statistics, John Wiley and Sons, 1981.

[11] P. J. Rousseeuw, Least Median of Squares Regression, Journal of the American Statistical Association, vol. 79, no. 388, pp. 871–880, 1984.

[12] M. A. Fischler and R. C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Comm. of the ACM 24, pp. 381–395, 1981.

[13] J. Matas and O. Chum, Randomized RANSAC with Sequential Probability Ratio Test, In Proc. International Conference on Computer Vision, vol. 2, pp. 1727–1732, 2005.

[14] O. Chum, J. Matas, Optimal Randomized RANSAC, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 8, pp. 1472–1482, 2008.

[15] J. Matas, O. Chum, Randomized RANSAC with Td,d Test, Image and Vision Computing, vol. 22, no. 10, pp. 837–842, 2004.

[16] P.H.S. 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.

[17] P.H.S. Torr and A. Zisserman, Robust Computation and Parametrization of Multiple View Relations, Sixth International Conference on Computer Vision, pp. 727–732, 1998.

[18] P.H.S. Torr and D.W. Murray, Guided-MLESAC: Faster Image Transform Estimation by Using Matching Priors, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pp. 1523–1535, 2005.

[19] O. Chum, J. Matas, Matching with PROSAC – Progressive Sample Consensus, In Proc. IEEE Computer Vision and Pattern Recognition, vol. 1, pp.220–226, 2005.

[20] D. Nister, Preemptive RANSAC for live structure and motion estimation, Ninth IEEE International Conference on Computer Vision, vol.1, pp. 199–206, 2003.

[21] O. Chum, J. Matas , J. Kittler, Locally Optimized RANSAC, Pattern Recognition, 25th DAGM Symposium, Magdeburg, Germany, vol. 2781, pp. 236–243, 2003.

[22] O. Chum, J. Matas, and S. Obdrzalek, Enhancing RANSAC by Generalized Model Optimization, In Proc. Asian Conf. Computer Vision, 2004.

[23] J. Chai and S.D. Ma, Robust epipolar geometry estimation using genetic algorithm, Pattern Recognition Letters, vol.19, no. 9, pp. 829–838, 1998.

[24] J. Chai and S.D. Ma, An evolutionary framework for stereo correspondence, In Proc. Int. Conf. on Pattern Recognition, vol. 1, pp. 841–844, 1998.

[25] V. Rodehorst, O. Hellwich, Genetic Algorithm SAmple Consensus (GASAC) - A Parallel Strategy for Robust Parameter Estimation, pp.103, Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06), 2006.

[26] H. Saito and M. Mori, Application of genetic algorithms to stereo matching of images, Pattern Recognition Letters, vol. 16, no. 8, pp. 815–821, 1995.

[27] P.H.S. Torr and D.W. Murray, The development and comparison of robust methods for estimating the fundamental matrix, Intl. J. Computer Vision, vol.24, pp. 271–300, 1997.

[28] D.A. Forsyth and J. Ponce, Computer Vision: A Modern Approach, Prentice Hall, 2002.

[29] J.V. Miller and C.V. Stewart, MUSE: Robust surface fitting using unbiased scale estimates, in Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, pp. 300–306, 1996.

[30] K.M. Lee, P. Meer, and R.H. Park, Robust adaptive segmentation of range images, IEEE Trans. Pattern Anal. Machine Intelligence, vol. 20, pp. 200–205, 1998.

[31] C. V. Stewart, MINPRAN: A new robust estimator for computer vision, IEEE Trans. Pattern Anal. Machine Intelligence, vol. 17, pp. 925–938, 1995.

[32] X. Yu, T.D. Bui, A. Krzyzak, Robust Estimation for Range Image Segmentation and Reconstruction, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 5, pp. 530–538, 1994.

[33] H. Chen and P. Meer, Robust regression with projection based M-estimators, 9th Intl. Conf. on Computer Vision, pp. 878–885, 2003.

[34] H. Wang and D. Suter, Robust Fitting by Adaptive-Scale Residual Consensus, In Proc. European Conference on Computer Vision, vol. 3023, pp. 107–118, 2004.

[35] H. Wang and D. Suter, Robust Adaptive-Scale Parametric Model Estimation for Computer Vision, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 26, no.11, pp. 1459–1474, 2004.

[36] H. Wang, D. Mirota, M. Ishii, G.D. Hager, Robust Motion Estimation and Structure Recovery from Endoscopic Image Sequences With an Adaptive Scale Kernel Consensus Estimator, IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–7, 2008.

[37] S. Rozenfeld, I. Shimshoni, The Modified pbM-Estimator Method and a Runtime Analysis Technique for the RANSAC Family, Conference on Computer Vision and Pattern Recognition, pp. 1113–1120, 2005.

[38] R. Subbarao, P. Meer, Beyond RANSAC: User Independent Robust Regression, p. 101–108, Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06), 2006.

[39] D. R. Myatt, Philip H. S. Torr, Slawomir J. Nasuto, J. Mark Bishop, R. Craddock, NAPSAC: High noise, high dimensional robust estimation - It's in the bag, In Proc. British Machine Vision Conference, 2002.

[40] A.B. Hadiashar, D. Suter, Robust Range Segmentation, 14th International Conference on Pattern Recognition (ICPR'98), vol. 2, pp. 969–971, 1998.

[41] R. Hoseinnezhad, A.B. Hadiashar, A Novel High Breakdown M-estimator for Visual Data Segmentation, 2007 IEEE 11th International Conference on Computer Vision, pp. 1–6, 2007.

[42] J. Choi; G. Medioni, StaRSaC: Stable Random Sample Consensus for Parameter Estimation, IEEE Conference on Computer Vision and Pattern Recognition, pp. 675–682, 2009.

[43] J. Illingworth and J. Kittler, A survey of the Hough transform, Computer Vision, Graphics, and Image Processing (CVGIP), vol. 44, pp. 87–116, 1988.

[44] L. Xu and E. Oja, Randomized Hough Transform (RHT): Basic mechanisms, algorithms, and computational complexities, CVGIP: Image Understanding, vol. 57, no. 2, pp. 131–154, 1993.

[45] D. Comaniciu and P. Meer, Mean shift: A robust approach toward feature space analysis, IEEE Trans. Pattern Anal. Machine Intell., vol. 24, pp. 603–619, 2002.

[46] D. Comaniciu, V. Ramesh, and A.D. Bue, Multivariate Saddle Point Detection for Statistical Clustering, Proc. Europe Conf. Computer Vision (ECCV), vol. 2352, pp. 561–576, 2002.

[47] C.V. Stewart, Robust Parameter Estimation in Computer Vision, SIAM Review, vol. 41, no. 3, pp. 513–537, 1999.

[48] S. Choi, T. Kim, W. Yu, Performance Evaluation of RANSAC Family, In Proc. British Machine Vision Conference, 2009.

[49] P. Meer, D. Mintz, A. Rosenfeld, D.Y. Kim, Robust Regression Methods for Computer Vision: A Review, International Journal of Computer Vision, vol. 6, no. 1, pp. 59–70, 1991.

[50] P. Meer, C.V. Stewart, D.E. Tyler, Robust Computer Vision: An Interdisciplinary Challenge, Computer Vision and Image Understanding, vol. 78, no. 1, pp. 1–7, 2000.

[51] G. Medioni, S.B. Kang, Emerging Topics in Computer Vision, Chapter 4, Prentice Hall, 2004.

[52] V.C. Raykar, R. Duraiswami, Fast optimal bandwidth selection for kernel density estimation, In Proc. 2006 SIAM Conference on Data Mining, pp. 524–528, 2006.

[53] G.R. Terrell, The Maximal Smoothing Principle in Density Estimation, Journal of the American Statistical Association, vol. 85, no. 410, pp. 470–477, 1990.

[54] G.R. Terrell, D.W. Scott, Oversmoothed Nonparametric Density Estimates, Journal of the American Statistical Association, vol. 80, no. 389, pp. 209–214, 1985.

[55] R.S Stephan , W.S David, On Locally Adaptive Density Estimation, Journal of the American Statistical Association, vol. 91, no. 436, pp. 1525–1534, 1996.

[56] J. Shi and C. Tomasi, Good Features to Track, Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600, 1994.

[57] C. Harris and M.J. Stephens, A combined corner and edge detector, In Proc. of the 4th Alvey Vision Conference, pp. 147–151, 1988.

[58] B. Horn and B. Schunck, Determining optical flow, In Artificial Intelligence, vol. 17, pp. 185–204, 1981.

[59] K. Prazdny, Egomotion and relative depth map from optical flow, Biol. Cybernatics 36, pp. 87–102, 1980.

[60] D.J. Heeger and A.D. Jepson, Subspace methods for recovering rigid motion I: Algorithm and Implementation, Intl. Journal of Computer Vision, vol.7, no. 2, pp. 95–117, 1992.

[61] H. C. Longuet-Higgins and K. Prazdny, The interpretation of a moving retinal image, Proc. R. Soc. London Ser. B 208, pp. 385–397, 1980.

[62] M. Brown, D. G. Lowe, Recognising Panoramas, Ninth IEEE International Conference on Computer Vision (ICCV'03), pp.1218–1225, vol. 2, 2003.

[63] N. Ohta, K. Kanatani, Optimal Estimation of Three-Dimensional Rotation and Reliability Evaluation, European Conference on Computer Vision, pp. 175–187, 1998.

[64] J. Lobo, J. Dias, Inertial Sensed Ego-motion for 3D Vision, Journal of Robotic Systems, vol. 21, no. 1, pp. 3–12, 2003.

[65] H. Rehbinder, B.K. Ghosh, Pose Estimation Using Line-Based Dynamic Vision and Inertial Sensors, IEEE Transactions on Automatic Control, vol. 48, no. 2, pp.186–199, 2003.

[66] R. Sagawa, N. Aoki, Y. Yagi, Mirror Localization for Catadioptric Imaging System by Observing Parallel Light Pairs, Proc. 8th Asian Conference on Computer Vision, LNCS 4843, pp.116–126, 2007.

[67] H.C.Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, Nature 293, pp. 133–135, 1981.

[68] Q.T. Luong and O.D.Faugeras, The fundamental matrix: Theory, algorithms, and stability analysis, Intl. Journal of Computer Vision, vol.17, no.1, pp. 43–75, 1996.

[69] R.I. Hartley, Projective reconstruction and invariants from multiple images, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 16, no. 10, pp. 1036–1041, 1994.

[70] B.D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision, In Proc. International Joint Conferences on Artificial Intelligence, pp. 674–679, 1981.

[71] D.G. Lowe, Object recognition from local scale-invariant features, Proc. of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157, 1999.

[72] Intel Corporation, Open Source Computer Vision Library.

[73] P.H.S. Torr, Outlier Detection and Motion Segmentation, PhD dissertation, Dept. of Eng. Science, Univ. of Oxford, 1995.

[74] M.P. Wand and M. Jones. Kernel Smoothing. Chapman & Hall. 1995.

[75] R. Subbarao, P. Meer, pbM-Estimator source code, http://www.caip.rutgers.edu/riul/research/robust.html

[76] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996.

Vita

• Education

Ngo Thanh Trung received his M.E. degree from Osaka University in 2005. He is currently a PhD candidate at Intelligent Robotics Laboratory, Department of Innovation Systems, Graduate School of Engineering Science, Osaka University. And he is also working as a research assistant at the Department of Intelligent Media, The Institute of Scientific and Industrial Research, Osaka University. He current research interests include robust estimation methods in computer vision, robot localization and map building.

• Publications

Journal Papers

1. **Ngo Trung Thanh**, Yuichiro Kojima, Hajime Nagahara, Ryusuke Sagawa, Yasuhiro Mukaigawa, Masahiko Yachida, Yasushi Yagi, " Real-time Estimation of Fast Egomotion with Feature Classification using Compound Omnidirectional Vision Sensor ", IEICE Transactions on Information and Systems, pp. 152–166, Vol.E93-D, No.01, Jan. 2010.

2. **Ngo Trung Thanh**, Hajime Nagahara, Ryusuke Sagawa, Yasuhiro Mukaigawa, Masahiko Yachida and Yasushi Yagi, "Highly Robust Estimator Using a Case-dependent Residual Distribution Model", IPSJ Transactions on Computer Vision and Applications, pp. 260–276, Vol.1, Nov. 2009.

International Conference Papers

1. **Ngo Trung Thanh**, Hajime Nagahara, Ryusuke Sagawa, Yasuhiro Mukaigawa, Masahiko Yachida, Yasushi Yagi, " An Adaptive-Scale Robust Estimator for Motion Estimation ", In Proc. IEEE International Conference on Robotics and Automation, 2009.

2. **Ngo Trung Thanh**, Hajime Nagahara, Ryusuke Sagawa, Yasuhiro Mukaigawa, Masahiko Yachida, Yasushi Yagi, " Adaptive-Scale Robust Estimator using Distribution Model Fitting ", In Proc. 9th Asian Conference on Computer Vision, 2009.

3. Hitoshi Yamada, Masayuki Kimura, Jun Ohmiya, Jun ichi Tagawa, **Ngo Trung Thanh**, Yasuhiro Mukaigawa, Yasushi Yagi, " Image stabilization algorithm for video with large image fluctuation ", In Proc. IEEE International Conference on Consumer Electronics, 2009

4. **Ngo Trung Thanh**, Hajime Nagahara, Ryusuke Sagawa, Yasuhiro Mukaigawa, Masahiko Yachida, Yasushi Yagi, " Robust and Real-Time Egomotion Estimation Using a Compound Omnidirectional Sensor ", In Proc. IEEE International Conference on Robotics and Automation, pp. 492–497, 2008.

5. **Ngo Trung Thanh**, Hajime Nagahara, Ryusuke Sagawa, Yasuhiro Mukaigawa, Masahiko Yachida, Yasushi Yagi, " Robust and Real-time Rotation Estimation of Compound Omnidirectional Sensor ", In Proc. IEEE International Conference on Robotics and Automation, pp. 4226–4231, 2007.

6. **Ngo Trung Thanh**, Yusuke Sakaguchi, Hajime Nagahara, Masahiko Yachida, " Stereo SLAM Using Two Estimators ", In Proc. IEEE International Conference on Robotics and Biomimetics, pp. 19–24, 2006.

Domestic Conference Papers

1. **Ngo Trung Thanh**, Hajime Nagahara, Ryusuke Sagawa, Yasuhiro Mukaigawa, Masahiko Yachida, Yasushi Yagi, " Robust and Real-time Estimation of Camera

Rotation with Translation-invariant Features ", In Proc. Computer Vision and Image Media CVIM 157-26, pp. 193-200, Jan. 2007.