ROBUST METHOD IN PHOTOGRAMMETRIC RECONSTRUCTION OF GEOMETRIC PRIMITIVES IN SOLID MODELING

By

XIANG YANG

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Mechanical and Aerospace Engineering

Written under the direction of

Peter Meer & Hae Chang Gea

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2017

ABSTRACT OF THE DISSERTATION

Robust Method in Photogrammetric Reconstruction of Geometric Primitives in Solid Modeling

by Xiang Yang

Dissertation Directors: Peter Meer & Hae Chang Gea

The 3D point cloud is a widely used data format obtained from scanning a 3D model, either by using active 3D laser range scanners or passive photogrammetric methods. Since the topological information in a point cloud is captured on 3D point level, the inverse design cannot be carried out directly on the data. The point cloud is first segmented into various geometric primitives, such as planes, spheres and cylinders, then the modification and redesign of solid model can be more easily achieved.

A robust estimator is required to detect the multiple inlier structures while filtering out the outliers. In this dissertation, we present a new robust algorithm which processes each structure independently. The user gives only the number of elemental subsets for random sampling, which is also required in other robust algorithms. This method provides a general solution of robust estimation, and no tuning of other parameters are required for particular estimation tasks. The scales of the structures (tolerance of error) are estimated adaptively and no threshold is involved in spite of different objective functions. After classifying all the input data, the segmented structures are sorted by their strengths and the strongest inlier structures come out at the top. Like any robust estimators, this algorithm also has limitations which are described in detail.

To illustrate its efficiency and robustness, the algorithm is tested on various synthetic and

real examples in both 2D and 3D. We extend its applications through the entire process of the structure from motion method, to reconstruct the 3D point cloud from a sequence of 2D images. We automatically estimate and fit the 3D surfaces from the 3D point samples, without generating surface normals or mesh model. The designer can interact with the 3D points conveniently and direct modification of point cloud becomes applicable.

Acknowledgements

I would like to acknowledge with gratitude Professor Peter Meer and Professor Hae Chang Gea for their mentorship and foresight, who supported me both morally and technically during the past five years at Rutgers University. Also my special thanks to the lab colleagues and seniors for their advice and encouragement. Lastly, I thank my wife Lijia Li for all her love and help since the beginning of my PhD studies.

Dedication

To my darling wife Lijia,

whose ardent love and unending support

for me and our dearest son Felix,

granted me a chance to finish this work.

Table of Contents

List of Figures

Chapter 1

Introduction

Computer-aided design (CAD) system is used in many engineering fields to generate, modify and optimize digital designs. Over the last two decades [7], several software packages of 3D solid modeling became the most used tools in the mechanical design. With the digital models built by geometric primitives, the designers present ideas of innovation, illustrate their design efforts, and create virtual prototypes to validate engineering applications.

As more detailed models are now used for manufacturing, simulation and analysis, more accurate designs are also required. A digital model is often acquired "inversely" from an existing physical model [91]. This "reverse engineering" process converts the 3D geometries in real world into a digital data format which can be processed by computers. Then the obtained data will be imported into the 3D CAD/CAM (Computer-aided manufacturing) and CAE (Computer-aided engineering) software for 3D visualization, rapid prototyping and 3D printing [57].

A typical inverse design process consists of three major steps:

- Data acquisition of the 3D model.
- Segmentation of data into geometric primitives (2D curves and 3D surfaces).
- Redesign of solid model based on parametric features.

In this chapter we begin with a brief introduction of inverse design process for solid models in Section 1.1. The existing challenges in the current methods are discusses in Section 1.2. In Section 1.3 we state our research contributions in designing a new robust algorithm to recover the geometric primitives. The outline of the following chapters in the dissertation are described in Section 1.4.

(a) (b) (c)

Figure 1.1: Active 3D scanning devices. (a) A coordinate-measuring machine (CMM). (b) A LiDAR device. (c) A hand-held 3D scanner.

1.1 Inverse Design of Solid Models

Several 3D scanning techniques will be succinctly described below. Both terms "3D point cloud" and "3D reconstruction of the model" will be used when referring to the 3D scanning, as the former is just the outcome of the latter process.

1.1.1 3D Scanning Techniques

The physical models in real world are three dimensional in general. The measured data is represented by a 3D point cloud with the geometric information captured by the 3D coordinates of the points. The manual measurement of the coordinates is a complex and error-prone process, and it consumes huge amount of time. A self-incompatible measurement can also lead to an over-constrained design. Since the beginning of the inverse CAD, the engineers have been searching for automatic techniques to extract the 3D information from a physical model.

Two major approaches of the 3D reconstruction exist to generate 3D point cloud. The 3D scanning [34] *actively* interferes with the surfaces by using range finding devices, like the coordinate-measuring machine (CMM) or laser range sensors. A few active scanning devices are shown in Fig.1.1 [1]. The *passive* methods, such as the photogrammetric approach, are based on multiple view geometry [47] and rebuild the 3D point cloud using a sequence of 2D images from different views. The photogrammetric approach will be detailed in Chapter 2.

[1] Fig.1.1b is retrieved from LiDAR, Wikipedia, https://en.wikipedia.org/wiki/Lidar.
Fig.1.1c is retrieved from Go!SCAN 20, http://www.creaform3d.com/.

(a) (b)

Figure 1.2: Reference features to locate hand-held scanners. (a) References on the background surface. (b) Adhesive features on the model.

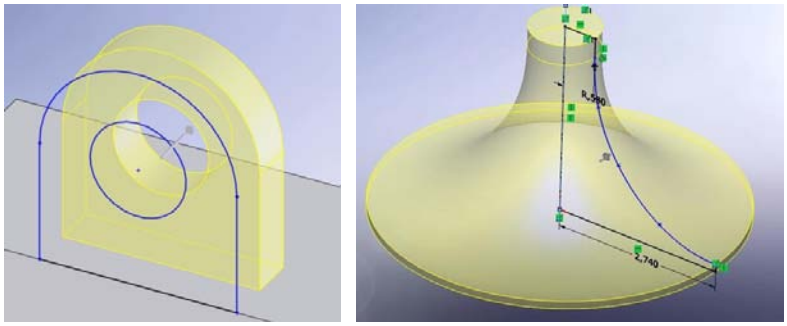
The active 3D laser scanning techniques are considered as the most accurate and efficient method to capture the 3D information of an object. These devices emit ultraviolet, or visible, or infrared light towards the object, and the reflected light is captured and translated into the coordinates of the 3D points [33]. Compared with the CMM system (Fig.1.1a) which measures only a single point at a time, the 3D laser scanners can survey a large region of 3D scene simultaneously.

The long-range 3D laser scanner measures the round-trip time of a pulse of light, such as the Light Detection And Ranging (LiDAR) technique [33] (Fig.1.1b). The LiDAR method is widely used in the digital terrain modeling, where a large area of ground surface can be scanned and reconstructed. The short-range hand-held scanner utilizes the triangulation mechanism to measure the 3D object from many different angles. Reference features [83] are used to recover the pose and location of the moving scanner (Fig.1.2) [2], then the 3D data scanned in different coordinate systems are registered together by the software.

1.1.2 Solid Modeling Techniques

Over the past decade, the development of the Product Lifecycle Management (PLM) changed the CAD software used in the mechanical design. Today the mechanical 3D CAD software is dominated by three vendors, Dassault, PTC and UGS [7]. Due to their similar functionally for 3D solid modeling, such as in CATIA [3], Creo [10] and NX [11], the basic rules of 3D

[2]Fig.1.2a is retrieved from Z-corp ZScanner 700, https://www.flickr.com/photos/creative_tools/.



(a) (b)

Figure 1.3: Parametric feature-based modeling in Solidworks [12]. (a) A linear extrusion. (b) A revole feature.

mechanical design evolves into an industrial standard followed by all the designers.

We now review some solid modeling techniques. This offers insight of the data format that can be used in the CAD software, into which the 3D point cloud has to be first converted. By understanding the basic design rules, we can retrieve the geometric and topological information to redesign a solid model, from the point cloud containing individual 3D points scattered in the scene.

The design of solid model generally starts with a group of sketch-based features, like the extrude or revolve shown in Fig.1.3. These features are applied, one at a time, until the model is complete. The corresponding parameters specify how a feature is created, including the geometric dimensions and the location of the reference plane. Since a solid model can be uniquely defined and rebuilt following the same design procedure, many 3D file formats only keep track of the exact process with the parameters in each step, along with the correct order to implement them. This modeling technique is called the parametric design approach, also named as the parametric feature based modeling [65]. In Fig.1.3a the model is created by sweeping a 2D profile along a linear axis, while in Fig.1.3b the model is produced by revolving a 2D curve around the central axis.

Many CAD programs support the freeform surface modeling, which is another method used in mechanical design. There are two basic methods in the freeform surface modeling. The first one begins with the construction of spline curves [23], from which the 3D surface is swept along the guide rail, or lofted through. Examples of freeform surfaces can be found from the design of car bodies, propelling nozzles and turbine blades. In most engineering applications, these

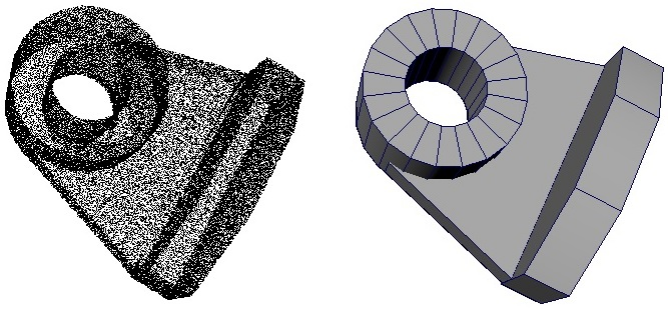


Figure 1.4: To rebuild the solid model, the geometric primitives should be first recovered from the point cloud.

models are relatively simple to construct and thus can be used in the manufacturing process. The second method directly establishes the surface with manipulation of the vertices or control points in 3D space, which is mostly used for art design and realistic modeling.

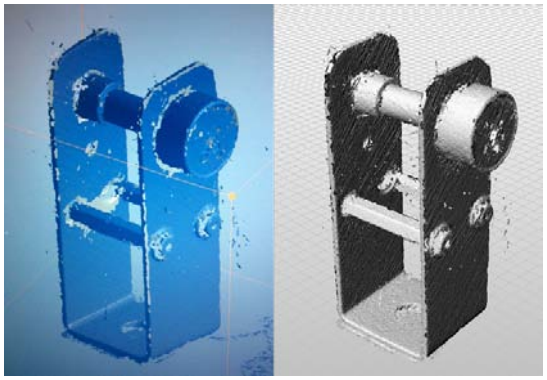
In summary, the feature-based modeling approach provides accurate design constrained by math equations. The feature-history makes it accessible to capture an engineer's original design intent, and the modifications of the model are also convenient. For example, a screw can be simply featured by a length and a radius, thus by varying these two parameters it can be fitted into different models. However, as shown in Fig.1.4, the functional changes in the model cannot be easily achieved directly from the 3D point cloud dataset. To rebuild these parametric features, the 3D point cloud has to be first simplified with the geometric primitives recovered.

1.1.3 Extraction of Geometric Primitives

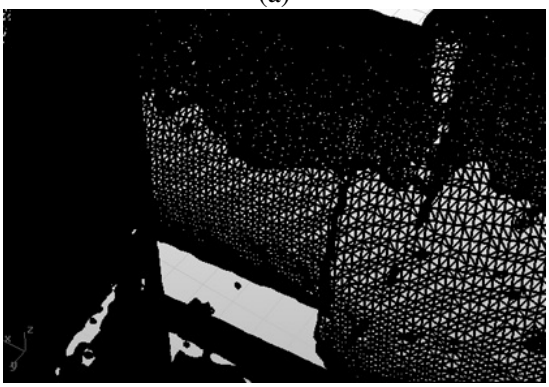
In Fig.1.5a [3], a model is scanned by a Z-corp ZScanner 700 (Fig.1.2a), giving the 3D point cloud and the mesh model (Fig.1.5b) in the STL (STereoLithography) format [1] generated by Geomagic [5]. In this example, the point cloud consists of 1,896,849 vertices capturing enough details to visually recognize the shapes and the surfaces, while the geometric features are not explicitly represented.

Instead of working with 3D point samples, the models simplified by geometric primitives are preferred by the designers of solid modeling. Some feature recognition systems, like Automatic Feature Recognition (AFR), evaluate the characteristics of a solid model only when the

[3] Images in Fig.1.5 are retrieved from 3D scanning and printing, http://fab.cba.mit.edu/classes/863.10/people/andy.-payne/Asst6.html.



(a)



(b)

Figure 1.5: Model scanned by a hand-held 3D scanner. (a) Point cloud obtained. (b) A close look at the mesh model.

geometric features are already available [18]. The high-level geometric or topological properties, such as the extrusion and the revolving features, are recognizable only if the geometric primitives are first recovered.

The surface extraction from the 3D point cloud can be done in two ways. In the first type of methods, the surface normals are recovered from the densely reconstructed point cloud, and a continuous surface [55] is generated to wrap the model [87] into, say, a triangular-faced mesh. Then 3D surfaces can be extracted from the mesh. For example, in [56] the geometric primitives were detected from the 3D range data containing thousands of points with the surface normals specified.

In Fig.1.5 a total of 632,283 triangles were generated, and a close look at the dense surfaces on the model is shown in Fig.1.5b. Another example illustrating the surface segmentation based

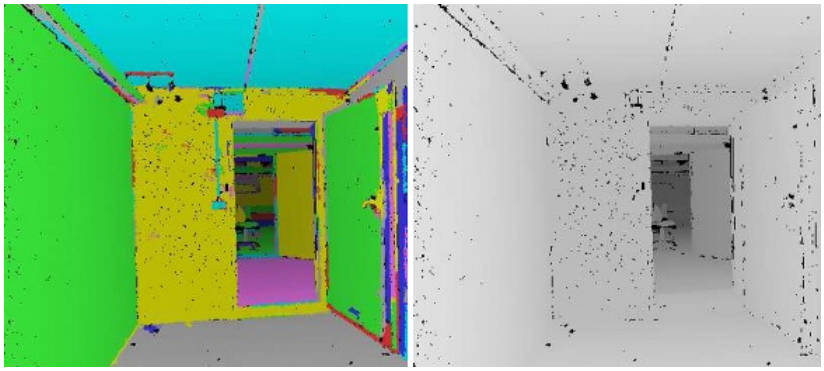


Figure 1.6: Normal based segmentation of a 3D point cloud acquired in a room with an open door.

on normals is shown in Fig.1.6 [4], where the region with a similar normal direction is classified as a plane.

The STL file [1] used in rapid prototyping and 3D printing, consists of the unit normals and the vertices of each triangulated planar surface, uniquely defines the solid model. This file format can be generated from many 3D scanners, and the interpolation problem introduced by the scattered data is solved by the dense sampling. However, since the evaluation of the normals requires smoothing over a small area, very dense point samples are required to capture accurate information of a surface. When millions of points exist in the point cloud, the computation becomes very expensive. In [20], the traditional least squares solution to find the surface normals was reformulated into an unconstrained and fast approximation of the problem. The same accuracy level was obtained, while the procedure was up to 17 times faster.

As explained in [16], the problem in recovering topological information in unstructured 3D data cannot be fully solved with surface normals. The second type of methods extract geometric primitives directly from a 3D point cloud, without going through the intermediate process to reconstruct the mesh model. In [26, 43] the boundary features of 3D models were extracted from the surface point cloud to rebuild wireframe models. In [49, 102], the surface extraction was combined with the object detection in the aerial LiDAR data, where the buildings and roofs were modeled by cuboids and inclined planes.

[4]Image is retrieved from Large-scale 3D point cloud processing tutorial 2013, http://kos.informatik.uni-osnabrueck.de/icar2013/.

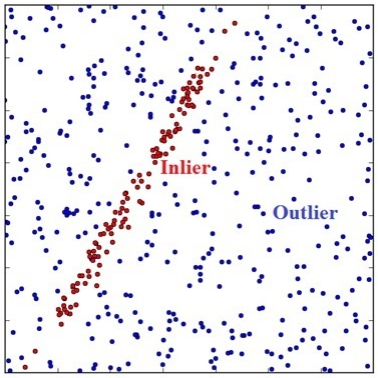


Figure 1.7: Robust estimation of a line in the presence of outliers.

All these surface extraction methods focused on the particular application and ad-hoc procedures were also applied. A general solution to extract geometric primitives from 3D point cloud has yet to be discovered.

1.1.4 Overview of the RANSAC Algorithm

All the surface extraction methods introduced in Subsection 1.1.3 involve the use of the RANdom SAmple Consensus (RANSAC) algorithm, which was developed by Fischler and Bolles in 1981 [39].

RANSAC is the most used robust regression algorithm in the past three decades, and is implemented in commercial CAD software to detect and fit 3D surfaces [4, 5]. For a particular surface, the estimation does not require the user to manually select a group of "good" points as *inliers*. Instead, RANSAC discards automatically the *outliers* in the input data once they are far away from the inlier structure. For example, given the input data in Fig.1.7, only the red points are chosen as inliers in estimating a line structure, while the blue points are rejected as outliers.

Elemental subsets are the building blocks of the robust regression, and each randomly chosen subset has the minimum number of points required to initialize a structure. For each elemental subset, RANSAC computes the total amount of points within a *user-specified threshold (scale)*. The elemental subset which covers the most points defines the inlier structure. RANSAC is based on the probabilistic relations to detect an inlier model, thus it can converge to the correct structure even without trying all possibilities. Similar types of methods also exist, like PROSAC, MLESAC, Lo-RANSAC, etc. These algorithms use different ways to generate the

random sampling and/or probabilistic relations for the elimination of the outliers. In paper [75] a review of these methods was given.

The use of RANSAC is easy, but requires a very careful adjustment of the scale in each problem. For the surface extraction from 3D point cloud, the RANSAC user should know how the specific 3D scanner performs, or if the same error level exists in different surfaces, etc. This is a major drawback in using RANSAC and more details are in Subsection 1.2.2.

1.2 Challenges of Inverse Solid Modeling

In this section we explain the challenges in the inverse solid modeling process, to which we will seek a solution in this dissertation.

1.2.1 Active vs. Passive Methods in 3D Reconstruction

By using a 3D laser scanner, the point cloud can be generated automatically. The designer doesn't have to worry about the details of the reconstruction process, as everything is handled by the equipment. The obtained 3D data can be checked in real-time to make sure that all the necessary information was collected. The investment of the scanning device strongly affects the degree of accuracy that can be achieved.

Some of the potential drawbacks in the use of active laser scanners are:

- Higher investment for the equipment.
- Requires the physical model and a well-trained surveyor.
- Model size must be compatible with the specific scanning device.

The cost issue should always be considered in the using of 3D laser scanning technology. Besides the initial investment for the device, it also involves the upgrading of the software and the training requirements for the system. The cost of a LiDAR device could vary from less than a few hundreds to a million dollars, depending on the specific requirements on accuracy and functionality.

The other two conditions establish hardware limitations on the active 3D scanners. A physical model has to be provided in order to carry out the measurements. If the 3D scanner is

located in a lab but not near to the model, this can be inconvenient. The physical model cannot be too large or too small, since the device may not work for either a tiny mechanical part or a big machine. Solutions of these problems may not be easily found for 3D laser scanners.

The photogrammetric approach using 2D images avoids these limitations. An example in [14] showed a massive 3D reconstruction of the major landmarks in the city of Rome. The data set consists of 150,000 images obtained from internet were matched after 13 hours processing, and then reconstructed as the 3D point cloud in another 8 hours, on a cluster with 496 compute cores. This work fully demonstrated the capacity of the passive 3D reconstruction method using 2D images. In Chapter 2 we will explain the algorithm. The processing time can be reduced to a few minutes when smaller amount of 2D images are used for a single model.

Why the 3D point clouds generated from passive techniques are not widely accepted in the engineering CAD? At first glance, the reason is simply that the photogrammetric approach based on 2D images is not yet a mature enough solution for many practical problems. Good results can be generated in controlled environments, but its performance may drop drastically in different noise and light conditions. For example, when the image sequence are taken by handheld cameras, and/or outdoor, the illumination of 2D images can be completely different.

The challenges in the passive 3D reconstruction method using multiple 2D images are:

- Improvement in robustness is required for image feature matching.
- Heavy noise and large area of missing data could exist in the 3D point cloud.
- Textureless and repetitive image area cannot be efficiently reconstructed.

The first problem listed above will be discussed in Subsection 1.2.2, while the other two are briefly addressed here.

A 3D point cloud does not necessarily capture all the details of the surfaces. The point samples are corrupted by noise, the model may contain sharp concave shapes, and a large area of surface is missing due to the scene occlusion. Then, the sharp edges and surface flatness may not be preserved without considering the missing data. These problems were mentioned in [41], where they concluded that the 3D point cloud generated from multi-view stereo reconstruction may not be suitable for inverse CAD.

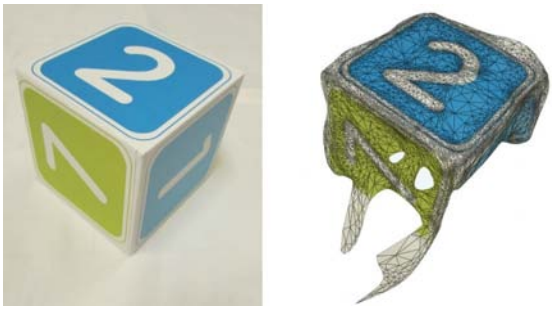


Figure 1.8: Inaccurate mesh gives no supportive information for surface extraction.

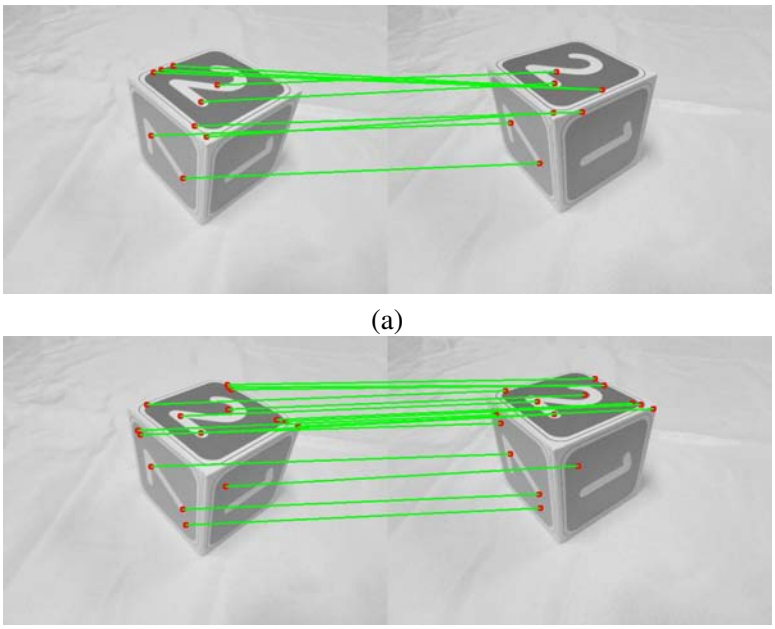
On the other hand, overfitting of a surface results in redundant meshes, like in Fig.1.8 generated by Autodesk ReMake [2]. The mesh model doesn't provide supportive information to the latter surface extraction process. As we will see in Fig.6.2, the planes can be directly estimated from the point cloud.

The textureless and/or repetitive surface patterns creates a problem for the matching algorithms. Several alternative solutions are proposed, as in [52, 58], to compute an optimal, smooth and consistent object over the image region without reliable point correspondences. These methods preserve the topology of the model during the reconstruction, but many details are lost due to the smoothing process. As a result, the obtained model may not show edges or corners. In other methods such as [101], a lot of details on the model were retained from the dense reconstruction, and can be used in 3D printing.

1.2.2 Using of RANSAC in the Robust Estimation

The robust estimation is involved in the entire process of 3D reconstruction and its performance greatly affects the overall result. In Subsection 1.1.4, we gave a simple introduction of the RANSAC algorithm which is widely used in the robust fitting of 3D surfaces (Subsection 1.1.3). Here we focus on the problems in the use of RANSAC.

When multiple inlier structures exist in the input data, each structure can be corrupted by noise independently, resulting in different scales. In the 3D reconstruction process, the noise level of the obtained point cloud depends on many factors, such as the accuracy of the 3D scanner, or the definition of images used in the photogrammetric method.



(a)

(b)

Figure 1.9: 2D image feature matching using RANSAC. (a) RANSAC scale = 0.1. (b) RANSAC scale = 1.

The major drawback of RANSAC is that the user has to assign an empirical scale *before* the robust fitting. This scale value identifies the maximal range of the inlier points and is also called as the tolerance of error. The selection of the scale relies on the experience of the RANSAC user. Providing a too small value for a scale could filter out many inlier points, while too large value bring in a lot of outliers. As shown in Fig.1.9, the relation "the smaller the scale, the better the result", is not valid in RANSAC. When the scale is 0.1 in Fig.1.9a, four incorrect matches can be found in the 14 pairs. In Fig.1.9b the scale is increased to 1 and 19 correct pairs are found out of 20. It is important to select a *suitable* scale in the RANSAC estimation.

When the user doesn't know the specifics of the 3D scanning devices, or the noise level of images used in a photogrammetric method is unknown, an adequate value of the scale is hardly predictable. The dimensions of the point cloud could be modified during the reconstruction process, which will also proportionally change the scale of the inlier noise. Since the RANSAC accepts only a single scale each time, when taking into account the different noise levels, the estimation has to be done step by step with every tolerance value adjusted separately.

The inlier noise is usually less than 2-3 pixels in real images, and the point cloud scanned by a 3D scanner has the measurement error within accuracy of 0.1 millimeters. Many times

RANSAC is successful and the threshold is not even mentioned. For example, in [56] RANSAC was combined with the projection based M-estimator [28] to detect 3D geometric primitives, but the method was tested on datasets with low noise obtained from 3D scanning. If the input data are scaled and/or cropped before estimation, or the inlier noise varies in a sequence of images, the RANSAC scale may no longer be valid for a correct result. This problem was explicitly mentioned in [35], where the plane detection was implemented in the presence of strong noise.

1.3 Research Contribution

A new robust algorithm to estimate multiple inlier structures from noisy dataset is presented in this dissertation. It provides a general solution to detect, segment and classify inlier points in the presence of outliers. The algorithm is applied to both 2D and 3D reconstruction of geometric primitives of objects. To acquire 3D point clouds, the structure from motion in the passive photogrammetric methods is used. Once the geometric primitives are extracted, they can be used to support the modifications of the 3D point cloud in the inverse design.

The contributions in this dissertation are summarized as follows:

• **General algorithm of robust estimation.**

The new robust algorithm avoids using the user-specified inlier scales (tolerance of error) before the estimation. Each structure is detected separately and the scale is estimated adaptively from the input data. Therefore, the method can segment inlier structures with different noise levels and work robustly without tuning any parameter. The confidence measure is strength based, and the inlier structures with larger strengths are returned first. The efficiency and robustness is demonstrated with datasets corrupted by a large amount of outliers.

• **Improvement in the Structure from Motion**

The structure from motion starts from a 2D image sequence. The new algorithm first finds the scales between every image pair, where the scale can vary in different pairs. With the robust feature matching, the image registration process results in more precise 3D points. Reference points/patterns are not required for the merge, since the 3D point correspondences are reliably found in the stitching of multiple patches with relatively large distortions.

• **Robust approach of 3D surface fitting in point cloud**

In the current computer-aided design (CAD) systems, before applying any modifications of the 3D point cloud, the surface normals or 3D mesh has to be generated to convert the point cloud data into a solid model. These procedures are avoided as we directly extract and fit the 3D surfaces from the point cloud. Instead of the manual selection of points for fitting a single surface, in the new method multiple surfaces are segmented in one estimation.

1.4 Dissertation Outline

The dissertation has seven chapters about the theoretical progress and practical application in both 2D and 3D experiments of the algorithm *Robust Estimation of Multiple Inlier Structures*.

In Chapter 2, we briefly introduce the major steps in the Structure from Motion (SfM) algorithm to reconstruct a 3D point cloud from a sequence of 2D images. We will give the background knowledge for the latter chapters.

The detailed robust algorithm for estimating multiple inlier structures is presented in Chapter 3, which is the major contribution in this dissertation. We will illustrate each step of the new robust estimator, and discuss also the limitations and condition of robustness.

The algorithm will be applied in the estimation of 2D planar structures (Chapter 4). We extend the applications to the structure from motion algorithm (Chapter 5), and present examples of the recovery of geometric primitives in 3D point cloud (Chapter 6).

Finally, in Chapter 7 we conclude the dissertation and discuss several open problems.

Chapter 2

Background: Photogrammetric Reconstruction of 3D Point Cloud

The increasing demand for 3D content from computer-aided design makes the 3D reconstruction a popular topic in recent years. It has applications range from the computer graphics and 3D data visualization, to the 3D printing and virtual reality. The photogrammetric techniques based on a series of 2D images, is one of the most popular method to obtain the 3D reconstruction from an ordinary camera. Starting from the essential matrix [60] introduced in 1981, the connection between the image pixels and the 3D spatial points can be established through numerous photogrammetric algorithms introduced over the past three decades in computer vision.

An early method using multiple views was proposed by Tomasi and Kanade [88]. They used an affine factorization approach to extract 3D information from image sequences, but the orthographic projection had significant limitations. Given a sequence of images for an object rotating about a single axis, the system in [40] generated a textured 3D model. Extensions for more sophisticated projection methods can be reviewed in [21]. All the methods however were limited by the available computing power and the problem formulations were simplified for practical use.

Today's computers are equipped with abundant computing power and the 3D reconstruction can be solved with advanced imaging devices in a more practical way. The software packages, such as VisualSFM [95], CMVS [42], Google's StreetView [6] and Autodesk's ReMake [2], can reconstruct good 3D models either for educational or commercial purposes. In Fig.2.1a a few sample images in the input sequence are shown, and by using 24 images the reconstructed 3D mesh model (Fig.2.1b) is generated with the commercial product Autodesk ReMake.

The photogrammetric reconstruction is a *passive* method starting from analyzing the 2D images. It is different from the 3D scanning which actively interferes with the 3D surfaces of

(a)

(b)

Figure 2.1: 3D reconstruction in Autodesk ReMake [2]. (a) Four image samples from 24 input images. (b) The reconstructed mesh model.

the physical parts. Examples of photogrammetric reconstruction can be found in [79, 82], and illustrated how the missing depth information in 2D images is recovered to obtain the 3D scene. The consistency among similar images provides a way to extract 3D geometric information. By retracing the imaging process, the pose and the location of each camera can also be estimated.

Compared with the active approaches introduced in Subsection 1.1.1, the photogrammetric methods give scalability in the acquisition of input data, regardless of the model sizes which may vary from a tiny figurine to a huge building. Hundreds and thousands of 2D images can be processed together to render a 3D scene, with each image containing several megapixels providing enough details for the accuracy in most engineering applications.

Algorithms are proposed based on the patterns in the 2D images, such as the consistency of the textures [13]. Others rely on the physical mechanisms involving the reflective properties and sources of light, trying to retrieve the concealed geometric information from the shading and contours [17, 27]. These require more complex problem formulations, and in some situations the optical phenomenon cannot be fully simulated without prior knowledge on the materials or imaging environment.

Based on the projective relations of the rigid bodies, many different Structure from Motion

(SfM) algorithms were proposed [47, 77]. In this dissertation we focus only on the photogrammetric method using SfM algorithm with the major steps:

- 2D feature extraction and matching
- 3D points triangulation and image registration
- Bundle adjustment and outlier rejection
- Local refinement and hierarchical merging

In the following sections we will briefly introduce these procedures as the background of the research in latter chapters. Most details can be found in Hartley and Zisserman's *Multiple View Geometry* [47].

2.1 Extraction and Matching of Image Features

All images in the 2D sequence should be first paired by matching pixels, where each pair corresponds to a same 3D point in real world. The SfM algorithm begins with the most challenging problem of the entire reconstruction pipeline, that to find the corresponding projections of a 3D point in separate 2D images. A typical image feature matching algorithm consists of three steps:

- Locate the interest points in every 2D image
- Describe each interest point by a descriptor
- Compute the similarity of the descriptors between points in different images

The optical flow is a method to find correspondences across images. An early version was proposed by Lucas and Kanade [63]. The corner points [45] were selected as the interest points, and for each point the algorithm detected the changes of a local image region. The point pair with maximal fit in the consecutive image frames provided the correspondence for the tracking. In Fig.2.2 the tracking trajectories are shown for two frames using the OpenCV [9] implementation. More recent work was proposed in [22] which extracted dense correspondences. These

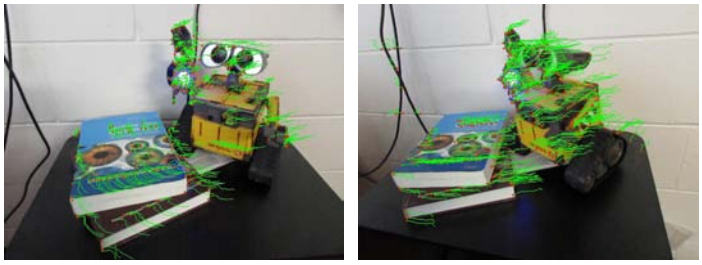
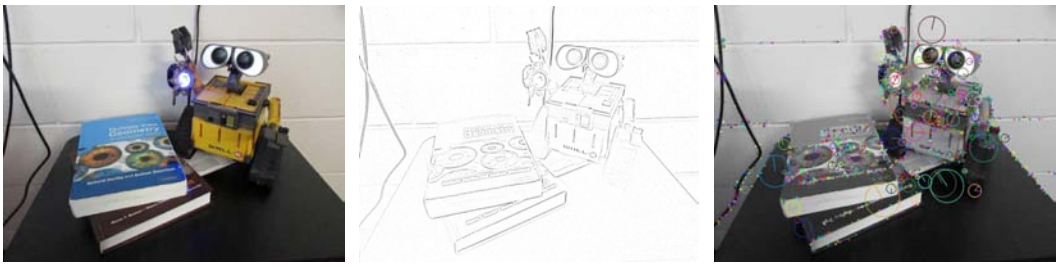


Figure 2.2: Optical flow tracking with Lucas-Kanade method [63].



(a) (b) (c)

Figure 2.3: SIFT feature extraction. (a) Original image. (b) Difference of Gaussians. (c) Extracted SIFT features.

approaches are mostly for 2D image interpolation instead of 3D reconstruction. They work reasonably well on continuous image sequence like video files, but when given a series of separate pictures, the point correspondences may not be found reliably.

The Scale-Invariant Feature Transform (SIFT) published in 2004 by Lowe [62], is one of the most successful image feature matching algorithms. SIFT first creates a series of smoothed and resampled images in an image pyramid to formulate the scale space, where the size variations of objects are represented by the changes in scales. The interest points are located as the extremas in the pyramid after applying the difference of Gaussians (DoG) functions. From Fig.2.3a, an example of DoG ($\sigma_1 = 1, \sigma_2 = 3$) is computed (Fig.2.3b), where the high-contrast region can be observed.

For each interest point, the SIFT descriptor consists of 128 components describing the intensity and direction information around a local image region. Fig.2.3c shows the SIFT features extracted with OpenCV, where the radius of the circle indicates the scale in which the interest

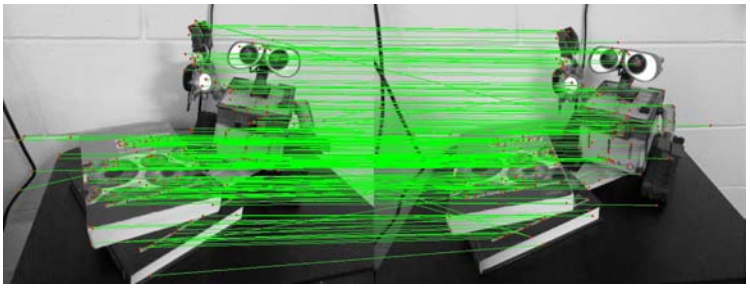


Figure 2.4: SIFT feature matching.

point was found. The larger the circle is, the interest point was detected over a broader region in the image. In the search of the most likely point pair, the major direction of the image gradient in a local region provides the rotationally invariant alignment of the descriptors. A pairing example is shown in Fig.2.4. In the first step toward the 3D reconstruction of a point cloud (Chapter 5), we will start from the SIFT matching in 2D images with OpenCV.

Several other methods to extract and describe image features were also proposed [68], such as Speeded-Up Robust Features (SURF), Binary Robust Independent Elementary Features (BRIEF), Features from Accelerated Segment Test (FAST) and Oriented FAST and Rotated BRIEF (ORB). All their implementations can be found in OpenCV. However, it should be noted that these methods (including SIFT) need to be adapted to the specific environment, and false point correspondences always exist due to the error-prone property. Without a robust method filtering out the majority of the outliers, the raw matches from SIFT cannot be used directly in the further steps.

2.2 Camera Model

The pinhole camera is a simplified but effective model to describes how a 3D point in real world is projected to a 2D image pixel. Fig.2.5 [1] illustrates the imaging process through a pinhole camera. A 3D point with homogeneous coordinate $\mathbf{X} = [X, Y, Z, 1]^\top$ is projected into the image coordinate system with $\mathbf{x} = [x, y, 1]$, and the depth (Z value) is absorbed by the projective mapping (2.1). Note that the image plane is presented in front of the camera center.

[1] Image retrieved from [47, Fig. 6.1].

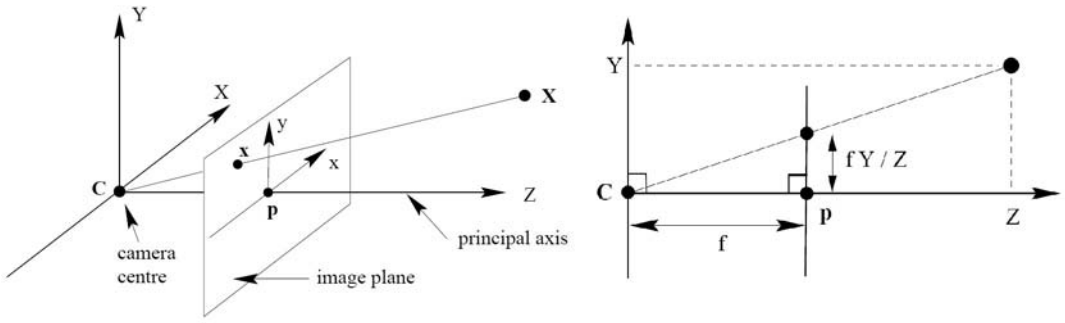


Figure 2.5: Pinhole camera geometry.

The same focal length f is in both x and y directions, later we will discuss the general case

$$x = \frac{fX}{Z}, \qquad y = \frac{fY}{Z}. \tag{2.1}$$

When the 3D object undergoes a rotation $\mathbf{R}_{3\times3}$ and/or translation $\mathbf{t}_{3\times1}$, its coordinate in 3D is then computed with a 4×4 matrix

$$\mathbf{X}' = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_3^\top & 1 \end{bmatrix} \mathbf{X}. \tag{2.2}$$

(2.2) can also be seen as moving the camera in the world coordinate system around a fixed 3D point. Combine equations (2.1) and (2.2) we obtain a 3×4 matrix $\mathbf{P}$ for the pinhole camera model, up to a scale

$$\mathbf{x} = \mathbf{P}_{3\times4}\,\mathbf{X} = \mathbf{K}\,[\,\mathbf{R} \mid \mathbf{t}\,]\,\mathbf{X} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix} \mathbf{X}. \tag{2.3}$$

The null vector of $\mathbf{P}$ is the camera center $\mathbf{C}$ in the world coordinate, and $\mathbf{P}\,\mathbf{C} = \mathbf{0}$.

The first 3×3 upper triangular matrix in (2.3) is denoted by the intrinsic matrix $\mathbf{K}$, which is specified by the particular hardware of the camera. In the general formulation, this matrix

contains five degrees of freedom

$$\mathbf{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

where the two focal lengths in the horizontal and vertical directions (f_x, f_y), principal point (p_x, p_y), and skew s can be independently parameterized. In practice, the two focal lengths are considered equal with zero skew, and the principal point sits right at the center of the image. These assumptions are usually valid when using an ordinary digital camera, and the images are not cropped before processing. For our 3D point cloud reconstruction, all the $\mathbf{K}$ matrices will contain only one parameter, that is the focal length f.

The extrinsic matrix $[\ \mathbf{R} \mid \mathbf{t}\ ]$ records the pose and location of the camera for taking an image. It contains 12 entries (2.3) with 11 unknowns. The 3×3 rotation can be described by the unitary rotation matrix $\mathbf{R}$ with only 3 degrees of freedom. Combined with the other 3 parameters related to translation, this 3×4 matrix has only 6 parameters to be estimated. Different parameterizations of extrinsic matrix exist, such as by using Euler angles and quaternions [64].

2.3 Epipolar Geometry

Inside a 2D image, the appearance of a 3D scene depends on two factors, the 3D shape of the object and the camera setup. They *uniquely* define the environment where the image was generated. Given enough images portraying the same 3D scene, the unknown camera parameters can be retrieved with an inverse approach.

This problem is ill-posed since the inverse estimation of camera requires the coordinates of the 3D points, while the Euclidean coordinates cannot be computed without knowing the calibrated cameras. To tackle this chicken and egg problem, a reasonable condition assumed in most SfM algorithms is that the image sequence portrays *rigid* objects. Then, the 3D points are constrained by geometric relations, which enforces constraints on the cameras and encapsulates the pose and location information.

The *epipolar geometry* depends only on the camera intrinsic parameters and relative pose,

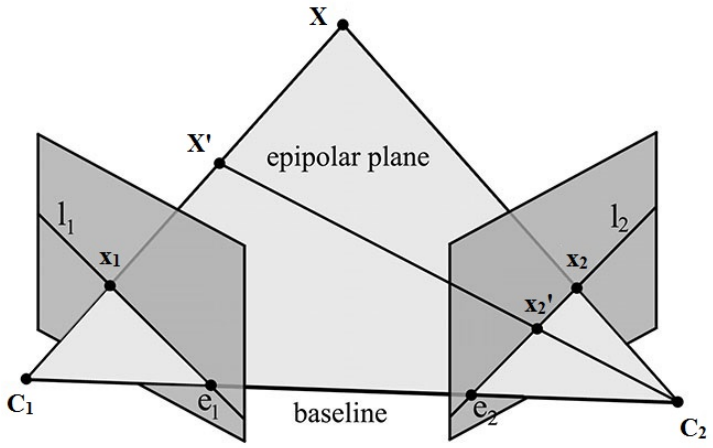


Figure 2.6: Epipolar geometry.

as shown in Fig.2.6. The two camera centers are denoted by points $\mathbf{C}_1$ and $\mathbf{C}_2$ and a 3D spatial point $\mathbf{X}$ is projected into the two image planes on point $\mathbf{x}_1$ and $\mathbf{x}_2$. By connecting these two camera centers (*the baseline*), the *epipoles* are found as the intersection of the baseline with the corresponding image plane, denoted $\mathbf{e}_1$ and $\mathbf{e}_2$. The 3D point $\mathbf{X}$ and two camera centers define the *epipolar plane*.

When the 3D point $\mathbf{X}$ moves along the ray $\overrightarrow{\mathbf{C}_1\mathbf{X}}$ to the point $\mathbf{X}'$, the image projection $\mathbf{x}_1$ remains the same, while $\mathbf{x}_2$ shifts to a new pixel $\mathbf{x}_2'$. The point $\mathbf{x}_2'$ always stays on the *epipolar line* $\mathbf{l}_2$, where $\mathbf{l}_2$ lies on the epipolar plane and passes through the epipole $\mathbf{e}_2$. Then the match for point correspondences between two views becomes a 1D search problem along the epipolar line.

We will follow the formulation in [98]. Let $\mathbf{P}_1$ and $\mathbf{P}_2$ denote the two camera matrices, thus $\mathbf{x}_1 = \mathbf{P}_1\mathbf{X}$, $\mathbf{x}_2 = \mathbf{P}_2\mathbf{X}$. The 3D point $\mathbf{X}$ can be solved from the one-parameter family of solutions of $\mathbf{P}_1\mathbf{X} = \mathbf{x}_1$ as

$$\mathbf{X}(\lambda) = \mathbf{P}_1^{+}\,\mathbf{x}_1 + \lambda\mathbf{C}_1. \tag{2.5}$$

where $\mathbf{P}_1^{+}$ is the pseudo-inverse of $\mathbf{P}_1$, and λ indicates that the 3D point $\mathbf{X}$ can shift along the direction $\overrightarrow{\mathbf{C}_1\mathbf{X}}$. The two particular points on this ray $\mathbf{X}(\lambda)$, namely $\mathbf{P}_1^{+}\,\mathbf{x}_1$ (when $\lambda = 0$) and $\mathbf{C}_1$ (when $\lambda = \infty$), will be observed by camera $\mathbf{P}_2$ and both of them are located on the epipolar line $\mathbf{l}_2$.

The epipolar line $\mathbf{l}_2$ can be computed from

$$\mathbf{l}_2 = (\mathbf{P}_2\ \mathbf{C}_1) \times (\mathbf{P}_2\ \mathbf{P}_1^+\ \mathbf{x}_1) = [\mathbf{e}_2]_\times \mathbf{P}_2\ \mathbf{P}_1^+\ \mathbf{x}_1. \tag{2.6}$$

Since the point correspondence $\mathbf{x}_2$ is also on $\mathbf{l}_2$, thus $\mathbf{x}_2^\top\ \mathbf{l}_2 = 0$ and finally we obtain

$$\mathbf{x}_2^\top\ \mathbf{F}\ \mathbf{x}_1 = 0, \qquad \mathbf{F} = [\mathbf{e}_2]_\times \mathbf{P}_2\ \mathbf{P}_1^+. \tag{2.7}$$

The 3×3 matrix $\mathbf{F}$ is called the *fundamental matrix*, which is rank two with 7 degrees of freedom. $\mathbf{x}_1$ and $\mathbf{x}_2$ are the homogeneous coordinates of the paired image points. More detailed properties of $\mathbf{F}$ can be found in [47, Chapter 9].

The correct point correspondences between two views satisfy the above relation (2.7) and gives a unique fundamental matrix $\mathbf{F}$. However, the camera matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ extracted from $\mathbf{F}$ are not unique since the relations are projective. Besides the robust 2D feature matching, the SfM algorithm also has to establish the calibration of the cameras. In Section 5.3 we will have several experiments on this topic.

2.4 Structure from Motion

The Structure from Motion algorithm uses a series of 2D images taken from different angles or distances as the input. The output is the camera parameters of each frame and the reconstructed 3D points viewed by the corresponding cameras. Fig.2.7 shows the basic processing pipeline of the SfM algorithms, also called as the incremental SfM. We first explain the major steps followed by the detailed subsections.

The process begins with a selected image pair. By auto-calibration the focal length f of the first two cameras is computed and through triangulation we obtain a group of 3D points visible in both cameras. This is a vital step since several key parameters are estimated, and a poor initialization could eventually lead to a bad reconstruction result. No perfect solution exists to evaluate the overall performance by selecting a particular pair, but the robustness can still be achieved from some ad-hoc techniques. For example, one can start from two images with the most 2D feature matches, or find the pair overlapping with the most other images.

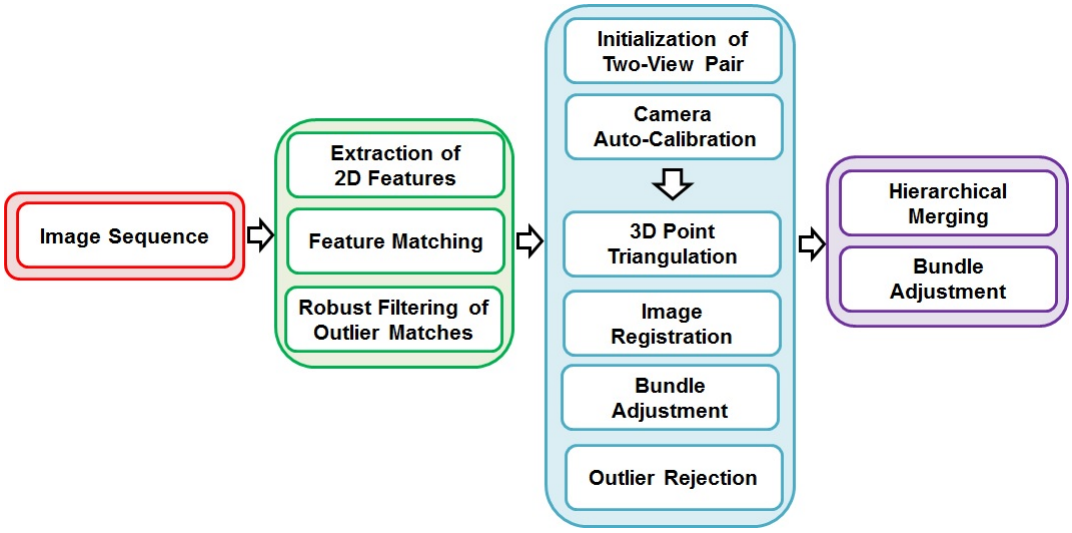


Figure 2.7: Structure from Motion algorithm pipeline.

After the initialization, the SfM algorithm undergoes a looping process where each iteration tries to register a new image in the track. Bundle adjustment is applied locally once some new 3D points are triangulated, see Subsection 2.4.3. A track represents both the cameras and 3D points collected in the reconstruction process sharing an overlapped 3D region (Fig.2.8) [2]. If no additional image can be added, the expansion of current track stopped. Then a new pair of images is choosen and a new track begins again.

After all the images were processed, an overall bundle adjustment is performed to merge different tracks hierarchically in the final step.

2.4.1 Extraction of Cameras

Once the two-view pair is initialized, a fundamental matrix $\mathbf{F}$ is computed based on the feature matches. The computation details are explained in Subsection 4.2.1. For each fundamental matrix $\mathbf{F}$, there exist infinite possible combinations of camera matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ satisfying the relation (2.7)

$$\mathbf{P}_1 = [\,\mathbf{I}\,|\,\mathbf{0}\,] \qquad \mathbf{P}_2 = [[\mathbf{e}_2]_\times \mathbf{F} + \mathbf{e}_2\mathbf{v}^\top | \lambda \mathbf{e}_2] \tag{2.8}$$

[2] Structure from Motion (SfM), Theia Vision Library, image retrieved from http://www.theia-sfm.org/sfm.html.

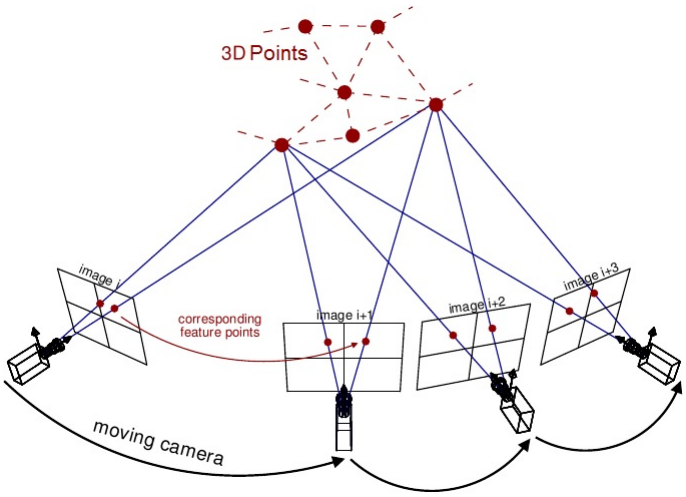


Figure 2.8: Track expansion.

where $\mathbf{I}$ is a 3×3 identity matrix, $\mathbf{v}$ is any 3-component vector, and λ is a non-zero scalar. The proof of this ambiguity can be found in [47, Chapter 9.5].

To resolve this projective ambiguity, the camera has to be first calibrated. Only the focal length f in (2.3) is unknown and the process will estimate it for the intrinsic matrix $\mathbf{K}$ of the camera. The f depends on the quality of the lens and the camera settings, and once obtained, it transforms the fundamental matrix into essential matrix $\mathbf{E}$ [60] from where a unique camera pair can be extracted.

The accuracy of f directly affects the metric precision in the final reconstruction. In general the camera used to take images is unknown to the SfM algorithm and is calibrated "automatically" based on the input images. The estimated focal length should be bounded in a reasonable range to avoid spurious values.

Several auto-calibration algorithms exist, as listed in [47, Chapter 19]. We focus on the method using the *Kruppa equation* [38], which reveals additional constraints on the fundamental matrix $\mathbf{F}$ of an image pair. The proof can be read in [93], here we directly go into the result as in [47, Chapter 19.4].

When the sequence of images is taken by the same camera (f is fixed), the problem is simplified. Let $\mathbf{F} = \mathbf{U}\ \mathrm{diag}(\sigma_1, \sigma_2, 0)\mathbf{V}^\top$ by SVD decomposition, $\mathbf{u}_i$ and $\mathbf{v}_i$ stand for the

columns in $\mathbf{U}$ and $\mathbf{V}$.

$$\frac{\mathbf{u}_2^\top \mathbf{K}\mathbf{K}^\top \mathbf{u}_2}{\sigma_1^2 \mathbf{v}_1^\top \mathbf{K}\mathbf{K}^\top \mathbf{v}_1} = -\frac{\mathbf{u}_1^\top \mathbf{K}\mathbf{K}^\top \mathbf{u}_2}{\sigma_1\sigma_2 \mathbf{v}_1^\top \mathbf{K}\mathbf{K}^\top \mathbf{v}_2} = \frac{\mathbf{u}_1^\top \mathbf{K}\mathbf{K}^\top \mathbf{u}_1}{\sigma_2^2 \mathbf{v}_2^\top \mathbf{K}\mathbf{K}^\top \mathbf{v}_2}. \tag{2.9}$$

If only focal length f remains unknown, $\mathbf{K}\mathbf{K}^\top$ is then a diagonal matrix $\text{diag}(f^2, f^2, 1)$. Each of the above relations (2.9) gives a quadratic equation in the unknown f^2. The Kruppa equation requires a large enough rotation between an image pair to be well constrained. Additional ambiguities were described in [84].

Once $\mathbf{K}$ is obtained, the essential matrix can be computed from

$$\mathbf{E} = \mathbf{K}^\top \mathbf{F} \mathbf{K}. \tag{2.10}$$

Generally, four pairs of camera matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ can be extracted from the essential matrix $\mathbf{E}$. Let the first camera $\mathbf{P}_1 = [\ \mathbf{I} \mid \mathbf{0}\ ]$, and $\mathbf{E} = \mathbf{U}\ \text{diag}(1,1,0)\mathbf{V}^\top$ by SVD decomposition, note that $\mathbf{U}$ and $\mathbf{V}$ are no longer the same as in (2.9) but computed from $\mathbf{E}$

$$\begin{aligned} \mathbf{P}_2 = [\mathbf{U}\mathbf{W}\mathbf{V}^\top \mid +\mathbf{u}_3] \quad \text{or} \quad [\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \mid +\mathbf{u}_3] \\ \text{or}\ [\mathbf{U}\mathbf{W}\mathbf{V}^\top \mid -\mathbf{u}_3] \quad \text{or} \quad [\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \mid -\mathbf{u}_3] \end{aligned} \tag{2.11}$$

$$\text{where} \quad \mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The mathematical derivations can be found in [47, Chapter 9.6]. Only one solution of the four can reconstruct all of the 3D points in front of both $\mathbf{P}_1$ and $\mathbf{P}_2$, and the correct $\mathbf{P}_2$ can be verified by checking the cheirality [72] [47, Chapter 21].

2.4.2 Triangulation of 3D Points and Image Registration

Given two camera matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, where the same 3D point $\mathbf{X}$ is observed at $\mathbf{x}_1$ and $\mathbf{x}_2$. Then the 3D coordinates of $\mathbf{X}$ can be computed from the linear triangulation method. The relations $\mathbf{x}_1 = \mathbf{P}_1\mathbf{X}$ and $\mathbf{x}_2 = \mathbf{P}_2\mathbf{X}$ are transformed by direct linear transform (DLT) into a form $\mathbf{A}\mathbf{X} = \mathbf{0}$.

Let $\mathbf{P}_i^{j\top}$ stand for the j-th row of camera $\mathbf{P}_i$, the scalar x_i and y_i are the horizontal and vertical pixel coordinates in $\mathbf{x}_i$, $\mathbf{A}_{4\times4}$ can be derived from $\mathbf{x}_i \times (\mathbf{P}_i\mathbf{X}) = \mathbf{0}$ as

$$\mathbf{A} = \begin{bmatrix} x_1\mathbf{P}_1^{3\top} - \mathbf{P}_1^{1\top} \\ y_1\mathbf{P}_1^{3\top} - \mathbf{P}_1^{2\top} \\ x_2\mathbf{P}_2^{3\top} - \mathbf{P}_2^{1\top} \\ y_2\mathbf{P}_2^{3\top} - \mathbf{P}_2^{2\top} \end{bmatrix} \tag{2.12}$$

and the 3D homogeneous coordinate $\mathbf{X}_h = [X_h, Y_h, Z_h, W_h]$ is computed as the null vector of $\mathbf{A}$. By dividing with the last term W_h, the true position is found as $\mathbf{X} = \mathbf{X}_h/W_h = [X, Y, Z, 1]$.

Each time when a group of new 3D points are triangulated, potentially other images taken from the similar angle or/and position can be registered since the same 3D points should exist in the overlapped region. A corresponding camera matrix $\mathbf{P}$ will be estimated inversely from the relation $\mathbf{x}_i = \mathbf{P}\mathbf{X}_i$. The relation $\mathbf{x}_i \times (\mathbf{P}\mathbf{X}_i) = \mathbf{0}$ now is separated into the entries in camera matrix to form another $\mathbf{A}_{2\times12}$ for each $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$ correspondence

$$\mathbf{A}_i \operatorname{vec}\mathbf{P} = \begin{bmatrix} \mathbf{0}^\top & -\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ \mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \end{bmatrix} \begin{bmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{bmatrix} = \mathbf{0}. \tag{2.13}$$

If at least six $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$ correspondences are provided, the 12 entries of $\mathbf{P}$ are solved from the null vector. It is further decomposed into the form of (2.3), where the constraints on rotation matrix are applied.

2.4.3 Bundle Adjustment

All results from the 3D triangulation and image registration are computed from DLT method, which provides a total least squares (TLS) solution. Due to the noise and incorrect point correspondences, the 3D point $\mathbf{X}_i, i = 1, ..., n$ may not be projected exactly by the $\mathbf{P}_j, j = 1, ..., m$ camera into the image point $\mathbf{x}_{ij}$. The relation $\mathbf{x}_{ij} = \mathbf{P}_j\mathbf{X}_i$ should be further improved. As

more cameras are registered and more points triangulated, the error accumulates and eventually will lead to an incorrectly reconstructed geometry. The bundle adjustment [89] reduces the reprojection error in each iteration, which will significantly improve the track accuracy and register more images into the track.

The bundle adjustment is a sparse damped least-squares (DLS) method, which is also known as the Levenberg-Marquardt algorithm (LMA). It interpolates between the Gauss-Newton and gradient descent methods with a damping factor. With the objective function

$$\underset{\mathbf{P}_j,\mathbf{X}_i}{\text{minimize}} \sum_{j=1}^{m}\sum_{i=1}^{n} \|\mathbf{P}_j\mathbf{X}_i - \mathbf{x}_{ij}\| \tag{2.14}$$

only the visible 3D points in each camera are considered in the error measurement. The camera parameters over all m views and a total of n 3D points will be optimized. If the camera is parameterized by 11 unknowns up to scale, the above process involves $11m + 3n$ variables.

The detailed procedures of bundle adjustment are listed in [47, Appendix 6], and a comprehensive description of bundle adjustment can be found in [89]. Large problems are implemented with multithreading to save processing time [97]. Several improvements were introduced in [37] which achieved better convergence behavior. The implementation details in our research will be explained in Chapter 5.

2.4.4 Hierarchical Merging

Multiple tracks result from the reconstruction process when the images cannot be continuously paired. For example, the gaps existed in the sequence can prevent image registrations. Since the bundle adjustment may not fully eliminate the drift errors, a track expansion will stop once the error accumulates beyond a certain threshold. A very long track is also not preferred if it captures only little changes of camera poses.

An overall bundle adjustment is to be done for the hierarchical merging process, even though each track has already been locally optimized. The different coordinate systems of the tracks have to be fused together based on 3D point correspondences. The merged model is initialized following the concept of divide and conquer, with its errors distributed more evenly inside each track. In Section 5.4 we will address this topic with experiments.

Chapter 3

Algorithm: Robust Estimation of Multiple Inlier Structures

In this chapter we present the new robust algorithm [99] to detect, estimate and segment inlier structures from the noisy input dataset. The new method processes each structure independently. The scales of the structures are estimated adaptively and no threshold is involved in spite of different objective functions. The user has to specify only the number of elemental subsets for random sampling. After classifying all the input data, the segmented structures are sorted by their strengths and the strongest inlier structures come out at the top. Like any robust estimators, this algorithm also has limitations which will be described in detail.

The robust estimation algorithm of multiple inlier structures is the main contribution in this dissertation. It will be applied in the estimation of 2D planar structures (Chapter 4), robust matching of SIFT features (Section 5.2), hierarchical merging of multiple tracks (Section 5.4), and recovery of geometric primitives in 3D point cloud (Chapter 6). No tuning of any parameters is involved for different tasks, which is a clear advance compared with other robust methods.

3.1 Algorithms without the User-Specified Inlier Scale

In Subsection 1.2.2 we explicitly explained the drawbacks in RANSAC [39] and its other variations. A scale externally specified by the user brings in the major disadvantage of using RANSAC, that the scale (tolerance of errorr) has to be tuned for particular problems. For more general applications, new algorithms were also proposed to avoid specifying an inlier scale before the estimation.

In [94], several techniques were described which did not require a scale value from the user. The proposed method derived the scale based on the k-th ordered absolute residual, where k was 10% of the size of the input data. Similar as in [86], $p + 2$ points were randomly selected, where p was the size of the elemental subset. The k-th order statistics was iteratively improved

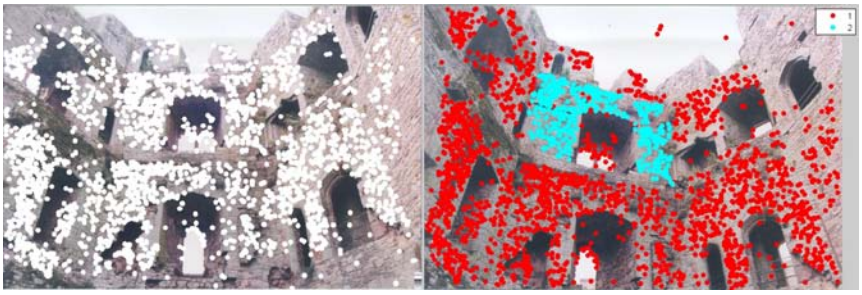
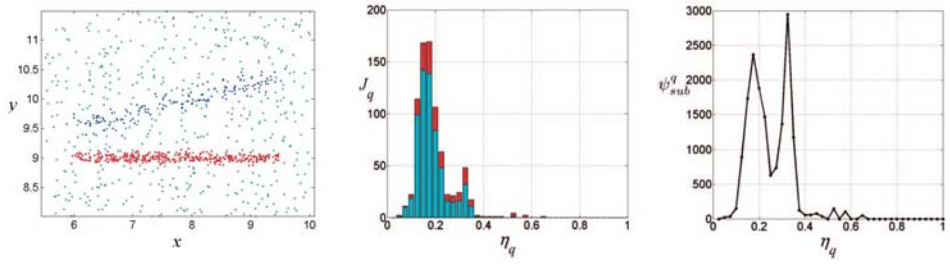


Figure 3.1: Incorrect homography estimation from RCMSA [73] when the model complexity parameter is not well-tuned.

to get the final estimate. The significance of taking two additional points in a sample was not justified and the parameter k varied largely in the experiments. The number of inlier structures was given before the estimation, in order to get comparable results with the sparse subspace clustering method [36].

Energy-based minimization approach can also be used in robust estimation, as it optimizes the quality of the entire solution in the Propose Expand and Re-estimate Labels (PEARL) algorithm [51]. The method started with RANSAC, then followed with alternative steps of expansion (inlier classification) and re-estimation to minimize the energy of the errors. PEARL converged to a local optimum, generally with a small number of inlier structures. A synthetic example in Figure 11 of [51] showed that PEARL can handle the estimation of multiple 2D lines with different Gaussian noises, but it was not tried for real images. The amount of outliers was relatively small in all the experiments.

The Random Cluster Model SAmpler (RCMSA) in [73] was similar to [51], but simulated annealing was used to minimize the overall energy function. Small clusters were discarded based on a function of the average fitting error. In the comparison of different algorithms, RCMSA performed better than PEARL. However, their scalar segmentation error may not fully justify the correctness of the conclusion. The data containing a larger number of inliers can tolerate more outliers for the same segmentation error. The model complexity was used as one of the parameters and had to be changed according to the specific estimation problem. It was 100 for the fundamental matrix and 10 for homography. Applying the values vice-versa, the estimator will no longer work all the time (Fig.3.1). This kind of adjustment cannot be done without prior knowledge.



(a) (b) (c)

Figure 3.2: Estimation of 2D lines in gpbM [69]. (a) Input image. (b) Histogram of peaks J_q (number of times a peak occurs at η_q fraction). (c) Cumulative distribution function weighted by its size η_q.

Most of the methods introduced above either used automatically a 2-3 pixels scale assumption for RANSAC, or tried to avoid the scale threshold but introduced additional parameter(s) for a particular task. If no empirical values of the parameters are known before the estimation, many experiments are required in order to tune them correctly. In general, there is no systematic way to predict the values of these parameters. The scale estimate problem can be solved only if for each structure separately, the inlier scale is estimated adaptively from the input data.

The generalized projection-based M-estimator (gpbM) [69] solved the robust estimation problem for each iteration in three independent steps: scale estimation, mean shift based recovery of the structure and inlier/outlier dichotomy. The only parameter to be specified by the user was the number of trials for random sampling, which is required in any robust estimator using elemental subsets. In the first step, the scale was estimated with all remaining data involved, by locating the highest value of the cumulative distribution function weighted by its size. This implementation made a structure containing more points easier to be detected first. The weights were critical to obtain the correct scale estimate, as Fig.3.2 [1] shows. The correct mode in Fig.3.2c ($\eta_q = 0.35$) cannot be identified from the other one ($\eta_q = 0.15$) without applying the weights. This strategy may not work all the time due to the interaction between inliers and outliers.

In the following sections, we propose a new robust estimator for multiple inlier structures which also uses three independent steps: scale estimation, structure recovery, strength based

[1] Image retrieved from [69, Figure 3].

structure classification, but each step has a completely different implementation from that in gpbM. The major innovations of the algorithm are summarized below.

- The scale estimation is carried out on a small set consisting of points in a single structure.
- The simplest mean shift algorithm is used.
- All the input data are classified into different structures first, without using any threshold.
- Structures are characterized by their strength (average density) and in general an inlier structure has a stronger strength.
- The limitations of the new robust estimator are discussed in detail.

3.2 Robust Regression with Elemental Subsets

In the following subsections we will first explain how a structure is initialized from an elemental subset (Subsection 3.2.1), and then define the error measurement from an input point to a structure (Subsection 3.2.2).

3.2.1 Linearized Space of Carriers

The objective functions in estimation problems can be either linear or nonlinear. For example, the plane estimation in 3D surface fitting is a linear problem, while the detection of cylinders or spheres, and the computation of fundamental matrix $\mathbf{F}$ between two images, are examples of nonlinear estimations.

The nonlinear objective functions in most estimation problems can be transformed into linear relations in higher dimensions. These linear relations, containing terms formed by the input measurements and their pairwise products, are called *carriers*. Each relation gives a carrier vector. For linear objective functions, such as plane fitting, the input variables and the carrier vector are identical.

In the estimation of fundamental matrix, the input data $\mathbf{y}$ are the point correspondences from two images $[x \;\; y \;\; x' \;\; y']^\top \in \mathbb{R}^4$, with $l = 4$ dimensions. The objective function with noisy image coordinates is

$$[x' \;\; y' \;\; 1]\, \mathbf{F}\, [x \;\; y \;\; 1]^\top \simeq 0 \tag{3.1}$$

which gives a carrier vector $\mathbf{x} \in \mathbb{R}^8$ containing $m = 8$ carriers

$$\mathbf{x} = \begin{bmatrix} x & y & x' & y' & xx' & xy' & yx' & yy' \end{bmatrix}^\top. \tag{3.2}$$

The linearized function of the carriers is

$$\mathbf{x}_i^\top \boldsymbol{\theta} - \alpha \simeq 0 \quad i = 1, \ldots, n_{in} \tag{3.3}$$

where n_{in} denotes the number of inliers. Vector $\boldsymbol{\theta} \in \mathbb{R}^8$ and scalar intercept α derived from the 3×3 matrix $\mathbf{F}$ are to be estimated. The constraint $\boldsymbol{\theta}^\top \boldsymbol{\theta} = 1$ eliminates the multiplicative ambiguity in (3.3).

In the general case, *multiple* linear equations can be derived from a single input $\mathbf{y}_i$

$$\mathbf{x}_i^{[c]\top} \boldsymbol{\theta} - \alpha \simeq 0 \quad c = 1, \ldots, \zeta \quad i = 1, \ldots, n_{in} \tag{3.4}$$

corresponding to ζ different carrier vectors $\mathbf{x}^{[c]}$. For example, the estimation of homography (Subsection 4.2.2) has $\zeta = 2$ carrier vectors derived from x and y image coordinates.

The Jacobian matrix is required for the first order approximation of the covariance of carrier vector. From each carrier vector $\mathbf{x}^{[c]}$, an $m \times l$ Jacobian matrix $\mathbf{J}_{\mathbf{x}^{[c]}|\mathbf{y}}$ is derived. Each column of the Jacobian matrix contains the derivatives of the m carriers in $\mathbf{x}^{[c]}$ with respect to one measurement from $\mathbf{y}$. The Jacobian matrices derived from linear objective functions are not input dependent, while those derived from nonlinear objective functions rely on the specific input point. The carrier vectors are *heteroscedastic* for nonlinear objective functions. For example, the transpose of the 8×4 Jacobian matrix of the fundamental matrix

$$\mathbf{J}_{\mathbf{x}_i|\mathbf{y}_i}^\top = \begin{bmatrix} 1 & 0 & 0 & 0 & x_i' & y_i' & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & x_i' & y_i' \\ 0 & 0 & 1 & 0 & x_i & 0 & y_i & 0 \\ 0 & 0 & 0 & 1 & 0 & x_i & 0 & y_i \end{bmatrix} \tag{3.5}$$

depends on $\mathbf{y}_i$.

The $l \times l$ covariance matrix of the measurements $\sigma^2 \mathbf{C}_\mathbf{y}$ with $\det \mathbf{C}_\mathbf{y} = 1$, has to be provided

before estimation. This is a chicken and egg problem, since the input points have not yet been classified into inliers and outliers. A reasonable assumption is to set $\mathbf{C_y}$ as the identity matrix $\mathbf{I_y}$, if no additional information is given. The input data are considered as independent and identically distributed, and contain *homoscedastic* measurements with the same covariance.

The covariance of the carrier vector $\sigma^2\mathbf{C}_i^{[c]}$ is computed from

$$\sigma^2\mathbf{C}_i^{[c]} = \sigma^2\mathbf{J}_{\mathbf{x}_i^{[c]}|\mathbf{y}_i}\ \mathbf{C_y}\ \mathbf{J}^{\top}_{\mathbf{x}_i^{[c]}|\mathbf{y}_i} \tag{3.6}$$

where the dimensions of $\mathbf{C}_i^{[c]}$ is $m \times m$. The scale σ of the structure is unknown and to be estimated.

3.2.2 Computation of the Mahalanobis Distance

In the general case, an elemental subset needs $m_e = \lceil \frac{m}{\zeta} \rceil$ input points to uniquely define $\boldsymbol{\theta}$ and α in the linear space. For example, three non-collinear points define a plane $(m = 3,\ \zeta = 1)$. The homography $(m = 8,\ \zeta = 2)$ requires four point pairs, and eight pairs are necessary for the fundamental matrix $(m = 8,\ \zeta = 1)$ if the 8-point algorithm is used. The projective transformation $(m = 15,\ \zeta = 3)$ in 3D has 15 degrees of freedom, while it can be solved with only five point correspondences.

When using elemental subset to initialize a structure, the m points inside should be first normalized [47, Section 4.4], and give correspondingly m equations in the linear space. Then the linear system can be solved up to a scale factor, and mapped back onto the original space.

For each $\boldsymbol{\theta}$ initialized from an elemental subset, every carrier vector is projected to a scalar value $z_i^{[c]} = \mathbf{x}_i^{[c]\top}\boldsymbol{\theta},\ c = 1, \ldots, \zeta$. The average projection of the m vectors from an elemental subset is α. The variance of $z_i^{[c]}$ is $\sigma^2 H_i^{[c]} = \sigma^2\boldsymbol{\theta}^{\top}\mathbf{C}_i^{[c]}\boldsymbol{\theta}$.

The Mahalanobis distance, scaled by an unknown σ, indicating how far is a projection $z_i^{[c]}$ from α, is computed from

$$\begin{aligned} d_i^{[c]} &= \sqrt{\left(\mathbf{x}_i^{[c]\top}\boldsymbol{\theta} - \alpha\right)^{\top}\left(H_i^{[c]}\right)^{-1}\left(\mathbf{x}_i^{[c]\top}\boldsymbol{\theta} - \alpha\right)} \\ &= \frac{|\mathbf{x}_i^{[c]\top}\boldsymbol{\theta} - \alpha|}{\sqrt{\boldsymbol{\theta}^{\top}\mathbf{C}_i^{[c]}\boldsymbol{\theta}}} \qquad c = 1, \ldots, \zeta. \end{aligned} \tag{3.7}$$

Each input point $\mathbf{y}_i$ gives a ζ-dimensional Mahalanobis distance vector

$$\mathbf{d}_i = \left[\begin{array}{ccc} d_i^{[1]} & \ldots & d_i^{[\zeta]} \end{array} \right]^\top \qquad i = 1, \ldots, n. \tag{3.8}$$

The worst-case scenario is taken to retain the largest Mahalanobis distance $d_i^{[\tilde{c}_i]}$ from all the ζ values

$$\tilde{c}_i = \underset{c=1,\ldots,\zeta}{\arg\max}\ d_i^{[c]}. \tag{3.9}$$

For different $\boldsymbol{\theta}$-s, the same input point may have its largest distance computed from different carriers. When the structure to be estimated has a linear objective function, the Mahalanobis distance and Euclidean distance are equivalent since the Jacobian matrix does not depend on the specific input point, and $H_i = 1$ for all the points.

The symbols related to the largest Mahalanobis distance are: $\tilde{d}_i$, the largest Mahalanobis distance for input $\mathbf{y}_i$; $\tilde{\mathbf{x}}_i$, the corresponding $m \times 1$ carrier vector; $\tilde{z}_i$, the scalar projection of $\tilde{\mathbf{x}}_i$; $\widetilde{\mathbf{C}}_i$, the $m \times m$ covariance matrix of $\tilde{\mathbf{x}}_i$; $\widetilde{H}_i$, the variance of $\tilde{z}_i$; $\hat{\sigma}$, the scale multiplying $\widetilde{\mathbf{C}}_i$ and $\widetilde{H}_i$, which has to be estimated.

3.3 Estimation of Multiple Inlier Structures

The new algorithm is detailed in this section. The scale $\hat{\sigma}$ for a structure is estimated in Subsection 3.3.1 by an expansion criteria. The estimated scale is used in the mean shift to re-estimate the structure in Subsection 3.3.2. The iterative process continues until not enough input points remain for a further estimation. In Subsection 3.3.3, all the estimated structures are ordered by strengths with the strongest inlier structures returned first.

3.3.1 Scale Estimation

Assume that n input points remain in the current iteration. The estimation process starts with M randomly generated elemental subsets, each giving a $\boldsymbol{\theta}$ and an α.

For every point $\mathbf{y}_i$, compute the largest Mahalanobis distance $\tilde{d}_i$

$$\tilde{d}_i = \frac{|\tilde{\mathbf{x}}_i^\top \boldsymbol{\theta} - \alpha|}{\sqrt{\boldsymbol{\theta}^\top \widetilde{\mathbf{C}}_i \boldsymbol{\theta}}} \geq 0 \qquad i = 1, \ldots, n. \tag{3.10}$$

Sort the n-distances in ascending order, denoted $\tilde{d}_{[i]}$. In total $j = 1, \ldots, M$ sorted sequences $\tilde{d}_{[i,j]}$ are found from all the trials.

Let $n_\epsilon \ll n$ represent a small amount of points

$$n_\epsilon = \frac{\epsilon\, n}{100} \qquad 0 < \epsilon \ll 100 \tag{3.11}$$

where ϵ defines the size of n_ϵ in percentage of the input amount.

Among all M trials, find the sequence that gives the *minimum sum* of Mahalanobis distances from the first n_ϵ points

$$\min_M \sum_{i=1}^{n_\epsilon} \tilde{d}_{[i,j]}. \tag{3.12}$$

This sequence is denoted as $\tilde{d}_{[i]_M}$ and contains n points in total. The first $\tilde{n}_\epsilon$ points are collected as the initial set.

If inlier structures still exist and M is sufficiently large, these $\tilde{n}_\epsilon$ points have a high probability to be selected from a single inlier structure, since it is more dense than the outliers. Neither information on the number of structures, nor the inlier amounts for each structure is known beforehand to establish $\tilde{n}_\epsilon$ deterministically.

Two rules should be considered for the ratio $\epsilon\%$. First, $\tilde{n}_\epsilon$ should be smaller than the size of any inlier structure to be estimated. Therefore, a small ratio is preferred to detect potential structures. In the following sections, all our experiments start with $\epsilon\% = 5\%$. The second rule is to have the size of $\tilde{n}_\epsilon$ at least five times the number of points in the elemental subset, as suggested in [47, page 182]. This condition reduces unstable results when relatively few input points are provided.

To recover the scale $\hat{\sigma}$, points belonging to the same structure have to be classified together as many as possible. The following example justifies the use of expansion criteria for scale estimation.

In Fig.3.3a two ellipses are shown; each of them has $n_{in} = 200$ inlier points. They are corrupted by different Gaussian noise with standard deviation $\sigma_g = 5$ and 10 respectively. Another $n_{out} = 200$ outliers are randomly placed in the 700×700 image. With $M = 2000$ and $\epsilon\% = 5\%$ ($\tilde{n}_\epsilon = 30$) we obtain the initial set in Fig.3.3b. The distances in $\tilde{d}_{[i]_M}$ for the first

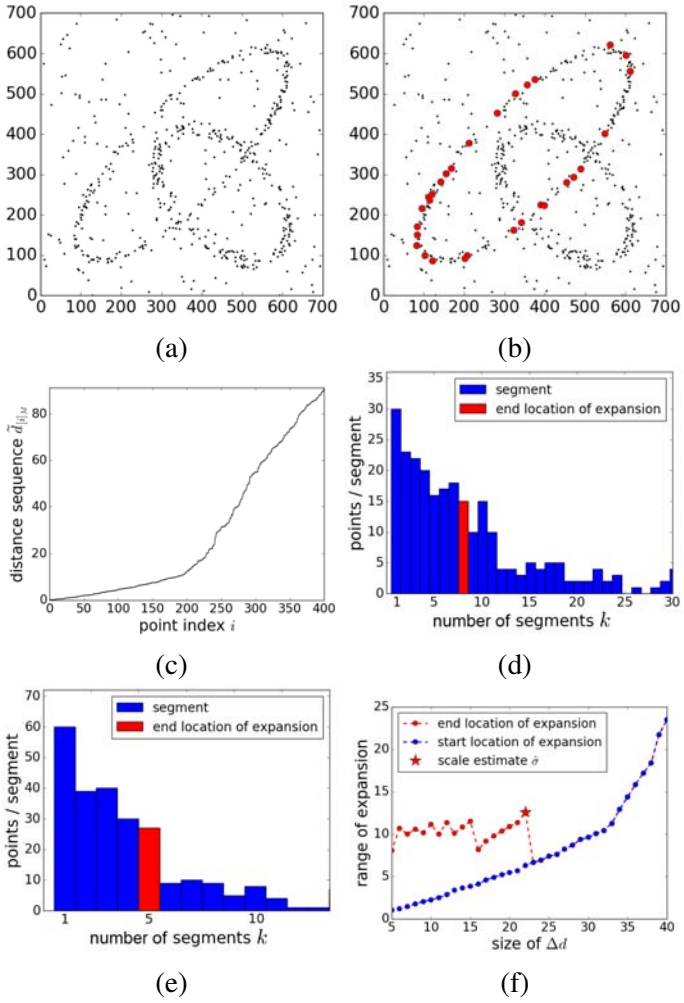


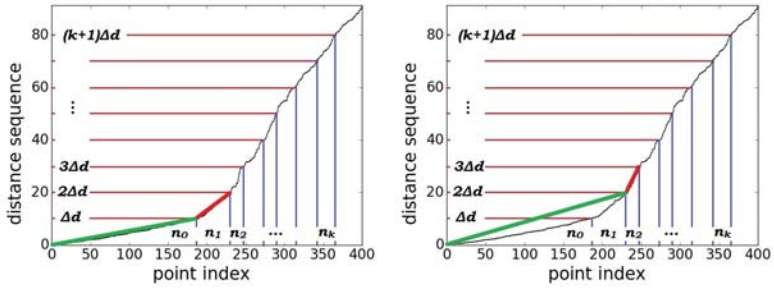
Figure 3.3: Scale estimation. (a) Input data. (b) Initial set for an iteration. (c) The first 400 points in the sequence $\tilde{d}_{[i]_M}$. (d) Histogram of point amounts with segment size $\Delta d_0 = \tilde{d}_{[5\%]_M}$. (e) Histogram of point amounts with segment size $\Delta d_5 = \tilde{d}_{[10\%]_M}$. (f) Expansion criteria applied to increasing sets.

400 points from 600 in total, are shown in Fig.3.3c.

Divide the sequence $\tilde{d}_{[i]_M}$ into multiple segments (Fig.3.4a), and each segment has an equal range of Mahalanobis distance, Δd. Let n_k denote the number of points within the k-th segment, $k = 0, 1, 2, \ldots$. Then $k = 0$ represents the first segment where $\tilde{n}_\epsilon = n_0$, and the average density is also n_0. The expansion process verifies the following condition for each k

$$\frac{n_{k+1}}{\frac{1}{k}\sum_{i=1}^{k} n_i} \leq 0.5 \tag{3.13}$$

where the numerator is the number of points in the $(k+1)$-th segment and the denominator is



(a) (b)

Figure 3.4: Comparison of segment densities, condition (3.13). (a) $k = 0$. (b) $k = 1$.

the average point numbers inside all the k segments.

This can also be illustrated by the slope of the sorted distance sequence. When $k = 0$ in Fig.3.4a, we check if the slope of the green segment becomes smaller than half of that in the red one. If not, we extend the green segment to cover both n_0 and n_1 as in Fig.3.4b, then verify the condition (3.13) again. Once the point density drops below half of the average in the previous segments, the boundary to separate this structure from the outliers is found, $k = k_t$. The value of the scale estimate is $k_t \Delta d$.

Due to the randomness of the input data, a single estimation of the scale is not enough. If the size of initial set is too small compared with the true structure, the scale estimate can also be too small to fully recover the complete structure in the mean shift.

In Fig.3.3d the expansion starts with $\Delta d_0 = \tilde{d}_{[5\%]_M}$ (the first segment is the initial set), and stops at $k_{t_0} = 8$, giving $\hat{\sigma} = 8.06$ (red bar). This is a relatively small estimate since a scale larger than 10 is expected when $\sigma_g = 5$. In Fig.3.3e the sampling with a larger $\Delta d_5 = \tilde{d}_{[10\%]_M}$ has its expansion stopped at $k_{t_5} = 5$ giving $\hat{\sigma} = 11.10$. This shows that various estimates can be generated from different segment layouts, and it is similar to the discretization effect over the scale space in SIFT [62].

In Fig.3.3f the expansion process is applied to an increasing sequence of sets. The Δd starts from $\tilde{d}_{[\epsilon\%]_M}$ and increases by 1% for each new sampling. The blue points in the figure indicate the length of Δd in percentage of points used as the segment size. Every expansion process is performed separately, and stops at the corresponding red point when condition (3.13) is met. The length of Δd continues to increase until it reaches the bound, $j = T + 1$, where the sets can no longer expand, as it is 23% in Fig.3.3f.

The scale estimate is found from this *region of interest* where the sets of points can expand. In Fig.3.3f it ranges from 5% to 22%. The expansion process may not always be able to start from $\Delta d = \tilde{d}_{[\epsilon\%]_M}$, but stops immediately at $k = 1$. Then as Δd increases, the starting point of region of interest is at the place where the expansion process begins.

The largest estimate from the region of interest gives the scale $\hat{\sigma}$

$$\hat{\sigma} = \max_{j=0,\ldots,T} k_{t_j} \Delta d_j \qquad \text{in region of interest} \tag{3.14}$$

which is the farthest expansion inside the region of interest. In Fig.3.3f the scale estimate is $\hat{\sigma} = 12.54$. From the sequence $\tilde{d}_{[i]_M}$, collect all the points within the scale estimate for the next step.

If the scale estimator locates an outlier structure, $\hat{\sigma}$ in general is much larger and the structure has weaker strength than inlier structures, as will be discussed in Section 3.3.3. The condition (3.13) is a heuristic criteria since the true distribution of the inliers is unknown. However, it does not play a sensitive role in the estimation process, as will be demonstrated by different experiments in Chapter 4 and Chapter 6. Some methods mentioned in Section 3.1 assumed a Gaussian distribution, or proposed a sophisticated theoretical model to classify inliers. These approximations may only be valid in specific problems.

3.3.2 Mean Shift Based Structure Recovery

From the points collected in the first step, another $N \ll M$ elemental subsets are generated. Most points in this set come from the same structure thus $N = M/10$ is enough.

For each trial all the input points are projected by $\boldsymbol{\theta}$ to a one-dimensional space $\tilde{z}_i = \tilde{\mathbf{x}}_i^\top \boldsymbol{\theta},\ i = 1, \ldots, n$. The mean shift [31] moves the z from $z = \alpha$ to the *closest mode*

$$\begin{aligned} \left[\widehat{\boldsymbol{\theta}}, \widehat{\alpha}\right] &= \arg\max_{\boldsymbol{\theta},\alpha} \frac{1}{n\hat{\sigma}} \sum_{i=1}^{n} \kappa\left((z - \widetilde{z}_i)^\top \widetilde{B}_i^{-1} (z - \widetilde{z}_i) \right) \\ &= \frac{1}{n\hat{\sigma}} \arg\max_{\boldsymbol{\theta}} \left(\arg\max_{z} f_{\boldsymbol{\theta}}(z) \right). \end{aligned} \tag{3.15}$$

The variance $\tilde{B}_i$ is computed from

$$\begin{aligned} \widetilde{B}_i &= \hat{\sigma}^2 \widetilde{H}_i = \hat{\sigma}^2 \boldsymbol{\theta}^\top \tilde{\mathbf{C}}_i \boldsymbol{\theta} \\ &= \hat{\sigma}^2 \boldsymbol{\theta}^\top \mathbf{J}_{\tilde{\mathbf{x}}_i|\mathbf{y}_i} \mathbf{J}_{\tilde{\mathbf{x}}_i|\mathbf{y}_i}^\top \boldsymbol{\theta} \end{aligned} \tag{3.16}$$

with $\mathbf{C_y} = \mathbf{I_y}$.

The function $\kappa(u)$ is the profile of a radial-symmetric kernel $K(u^2)$ defined only for $u \geq 0$. For the Epanechnikov kernel

$$\kappa(u) = \begin{cases} 1-u & (z-\widetilde{z}_i)^\top \widetilde{B}_i^{-1} (z-\widetilde{z}_i) \leq 1 \\ 0 & (z-\widetilde{z}_i)^\top \widetilde{B}_i^{-1} (z-\widetilde{z}_i) > 1. \end{cases} \tag{3.17}$$

Let $g(u) = -\kappa'(u)$ and for the Epanechnikov kernel, $g(u) = 1$ when $0 \leq u \leq 1$ and 0 if $u > 1$. All the points inside the window contribute equally in the mean shift. The convergence to the closest mode is obtained by assigning zero to the gradient of (3.15) in each iteration. The z_{new} is updated from the current value $z = z_{old}$ by

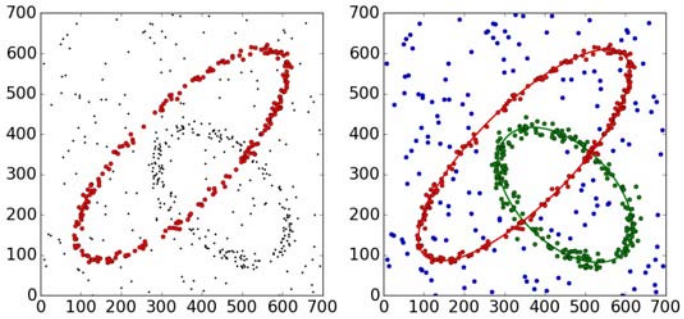
$$z_{new} = \left[\sum_{i=1}^{n} g(u)\right]^{-1} \left[\sum_{i=1}^{n} g(u) \widetilde{z}_i\right]. \tag{3.18}$$

Many of the n input points have their projections more distant from z_{old} than $\pm\widetilde{B}_i$ and their weights are zeros.

The highest mode among all N trials gives the estimate $\hat{z} = \hat{\alpha}$. The vector $\hat{\boldsymbol{\theta}}$ is obtained from the same elemental subset which gives the highest mode. All the input points that can converge into the $\pm\hat{\sigma}$ region around $\hat{\alpha}$ are classified as inliers, resulting in n_{in} points (Fig.3.5a). The total least squares (TLS) estimate for the structure is computed and $\hat{\boldsymbol{\theta}}^{tls}$, $\hat{\alpha}^{tls}$ and $\hat{\sigma}^{tls}$ are obtained.

3.3.3 Strength Based Classification

After the mean shift step, the n_{in} points are removed from the current input data before the next iteration. If the remaining data are not enough for another initial set, the algorithm terminates and all the recovered structures are sorted by their strengths in descending order (Fig.3.5b).



(a) (b)

Figure 3.5: Structure recovery. (a) Structure recovered after mean shift. (b) Structure sorted by strengths ($s_R > s_G > s_B$).

The *strength of a structure* is defined as

$$s = \frac{n_{in}}{\hat{\sigma}^{tls}}. \tag{3.19}$$

which can also be seen as the density in the linear space of that structure. The value n_{in} represents the point amount removed at each iteration and it could be either a structure of inliers or outliers.

Structures with stronger strengths are detected first, and in general are inlier structures with more dense points and smaller scales. The new method does not rely on any threshold to separate inliers from the outliers. After all the input data are segmented, the difference between the structures declared as inliers with stronger strengths and the first outlier structure is clear. With the results sorted by strength, the user has an easy task to retain the inlier structures, as the examples in Chapter 4 will show. If an ambiguous inlier/outlier threshold appears, like in Fig.4.3d, the strongest inlier structures are still detected correctly.

3.4 Limitations

The major limitation of every robust estimator comes from the interactions between inliers and outliers. As the outlier amount increases, eventually the inliers and outliers become less separable in the input space. In our algorithm most of the processing is done in a linear space, but the limitation introduced by outliers still exists. We will illustrate it in the following example.

In a 700×700 image, a circle consists of $n_{in} = 200$ inliers is corrupted by Gaussian noise

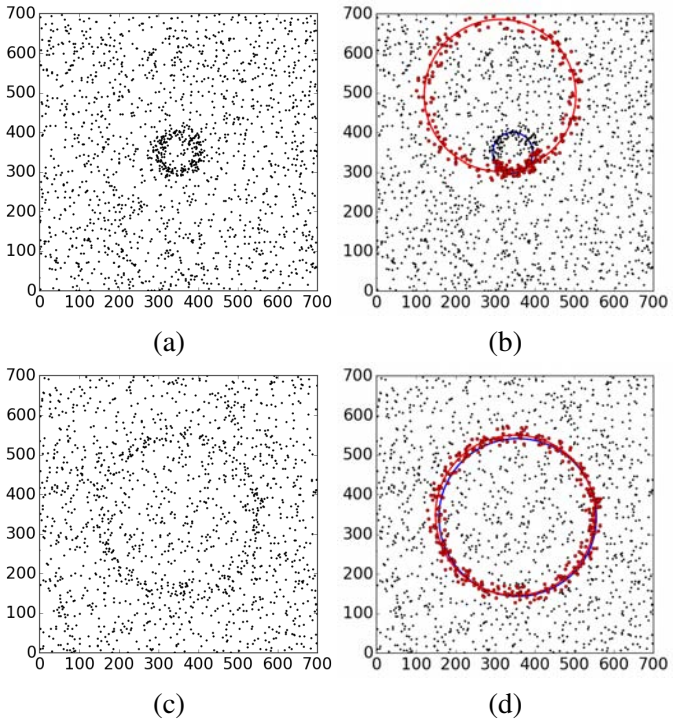


Figure 3.6: The inlier/outlier interaction. (a) Input data of a circle with radius 50. (b) Incorrect final result obtained from a correct scale estimate. (c) Input data of a circle with radius 200. (d) Good final result obtained from a correct scale estimate.

with $\sigma_g = 10$, together with $n_{out} = 1500$ outliers. The first circle has a radius of 50 (Fig.3.6a), and the other one has a radius of 200 (Fig.3.6c). In both these figures, the estimator finds the correct scale estimates from the structure (blue circles) corresponding to the initial sets, where $\hat{\sigma}_{50} = 23.65$ and $\hat{\sigma}_{200} = 23.58$.

In the true inlier structure, about 196 points should exist inside the scale $\hat{\sigma}_{50} = 23.65$, based on the Gaussian distribution. The number of outliers in the same location can be roughly estimated as

$$(2\pi\, 50)(2 * 23.65)\frac{1500}{700 \times 700} = 45 \text{ points.}$$

thus about 241 points can be found in the true inlier structure. However, after the mean shift step an incorrect final result (red circle) containing 261 points is obtained in Fig.3.6b, where 84 points are true inliers and 177 points from the outliers. Although the true structure appears more dense in the input space, the mean shift converges to an incorrect mode due to the heavy noise from outliers.

The circle in Fig.3.6c appears much weaker, however after 100 tests with randomly generated data (inlier/outlier), it returns more stable estimations than the smaller circle in Fig.3.6a. In a result shown in Fig.3.6d, 346 points are classified as inliers, where 190 points are from true inliers and 156 points from the outliers. The mean shift has a much lower probability to converge to another, incorrect mode, and this inlier structure resists more outliers.

Similar limitation exists in RANSAC when many outliers are present. Even a correct scale given by the user can still lead to an incorrect estimation. The methods proposed in [73] and [86] returned incorrect results if too many outliers existed. The failure of RANSAC also occurs due to not explicitly considering the underlying task [48].

In [92], a robust estimator was proposed to track objects in an image sequence, by combining an extended Kalman filter with a structure from motion algorithm. The Figure 5 in that paper showed that the correct estimate was returned with the data containing more than 60% outliers. For the homography estimation in Figure 6 of [70], more than 90% of the points were outliers, and correct result were obtained after 10000 iterations of a contrario outlier elimination process. Since these results did not provide repetitive tests, the stability of the methods cannot be verified.

The strength of inlier points is another factor with a strong influence on the inlier/outlier interaction. Firstly, the level of the inlier noise affects the number of outliers that can be tolerated. With the same number of inliers, structures with lesser inlier noise can be estimated more robustly since a smaller scale estimate results in a stronger strength. The inlier structures with weaker strengths generally have larger noises and the scale estimates are also larger. The outliers will have a stronger interaction with these weak inlier structures and can lead to spurious results, see Fig.4.3f.

Secondly, when the inlier amount is too small, the initial set may not closely align with a true structure. The scale estimation becomes unstable since the region of interest can sometimes cover a very narrow range of Δd in the expansion process. The scale estimate could be much smaller than the true value and only the minority of the points will converge to the inlier structure. Instead of a single structure estimate, two or more split structures could be obtained.

In Fig.3.7a, the expansion process is applied to the same example as in Fig.3.3, but with $n_{out} = 400$. The expansion stops soon and the algorithm locates the region of interest between

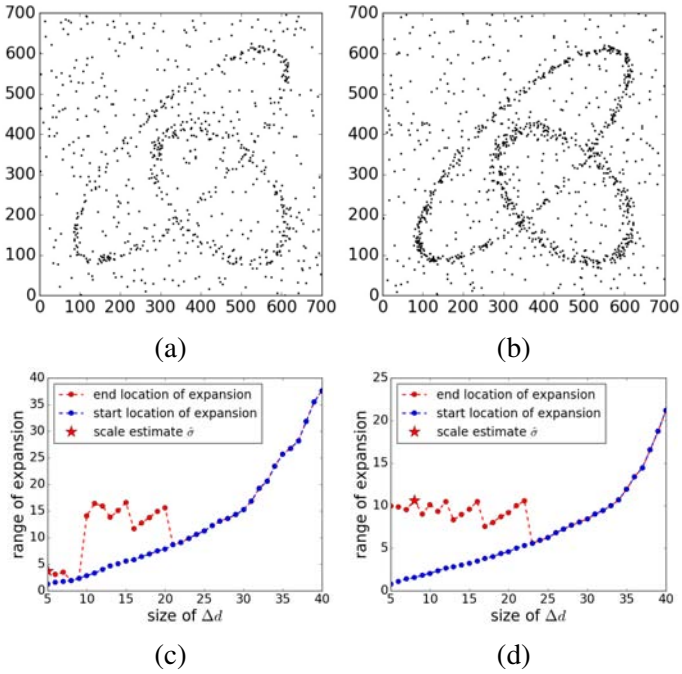


Figure 3.7: Limitation of scale estimation. (a) Case 1: a small number of inliers ($n_{in} = 200, n_{out} = 400$). (b) Case 2: a larger number of inliers ($n_{in} = 400, n_{out} = 400$). (c) Unstable estimate obtained from case 1. (d) More robust estimate obtained from case 2.

$5\% - 7\%$ giving a small scale estimate. After applying the expansion criteria many times, the range of expansion from different testing data are not stable (Fig.3.7c). In Fig.3.7b the number of inliers is raised to $n_{in} = 400$. The scale estimate becomes a more stable value with the region of interest located between $5\% - 22\%$ (Fig.3.7d).

When the inlier/outlier interaction is strong, preprocessing on the input data is required to obtain more inlier points, and/or reduce the outlier amount for a better performance. In Fig.4.7a of Section 4.2.2, an example is given where homography estimation in 2D is used to segment objects in 3D scene. Under the small translational motions, the two planes on the bus though orthogonal in 3D, are not separable in 2D due to the relatively small amount of inlier points. In Fig.4.8 we show that by using more inlier points, the estimator will recover more inlier structures.

If an inlier structure appears split in several structures with fewer points, post-processing is needed to merge them. The user can easily locate them by their strengths since most of these split structures are still stronger compared with the outliers. The similarity of two structures should be compared in the input space where measurements are obtained, as the derived carriers

in the linear space do not represent the nonlinearities of the inputs explicitly.

For two inlier structures with linear objective function, the merge can be implemented based on the orientation of each structure and the distance between them. For two ellipses, the geometric tools to determine the overlap area can be used [50]. The measurements of fundamental matrices and the homographies are in the projective space instead of euclidean. If the reconstructed 3D scene can be provided from auto-calibration [47, Chapter 19], the 3D information should be applied for the segmentation or merge of two structures.

3.5 Conditions of Robustness

We have discussed several limitations for the algorithm without considering the number of trials M for random sampling, which is the only parameter given by the user.

The M trials in this algorithm are used to estimate the scale. The required amount of M depends strongly on the data to be processed. The complexity of the objective function, the size of the input data, the number of inlier structures, the inlier noise levels, and the amount of outliers, all are factors which can affect the required number of trials. If no information on the size of M is known, the user can run several tests with different M-s until the results become stable. Once M is large enough, the estimation will not improve by using larger sampling size.

With all the other robustness conditions satisfied, M has to be large enough to detect the weakest inlier structure. This process gives a quasi-correct scale estimate and then the mean shift recovers a desired inlier structure. Once the interaction between inliers and outliers is apparent, the quality of the estimation cannot really be compensated by a larger M since the initial set has become less reliable. Only through preprocessing of the input data will the number of inlier points increase and a better result be obtained.

Three key conditions to improve the robustness of the proposed algorithm are summarized:

- Preprocessing to reduce the amount of outliers, while bring in more inliers.
- The sampling size M should be large enough to stably find the inlier estimates.
- Post-processing should be done in the input space when an inlier structure comes out split or has to be separated.

3.6 Review of the Algorithm

The new robust algorithm is summarized below.

Robust Estimation of Multiple Inlier Structures

Input: $\mathbf{y}_i$, $i = 1, \ldots, n$ data points that contain an unknown number of inlier structures with their scales unspecified, along with outliers. The covariance matrices for $\mathbf{y}_i$ are $\mathbf{C}_\mathbf{y} = \mathbf{I}_\mathbf{y}$ if not provided explicitly.

Output: The sorted structures with inliers came out first.

- Compute the carriers $\mathbf{x}_i^{[c]}$, $c = 1, \ldots, \zeta$, and the Jacobians $\mathbf{J}_{\mathbf{x}_i^{[c]}|\mathbf{y}_i}$, for each input $\mathbf{y}_i$, $i = 1, \ldots, n$.

$\odot$ Generate M random trials based on elemental subsets.

  - For each elemental subset find $\boldsymbol{\theta}$ and α.
  - Compute the Mahalanobis distances from α for all carrier vectors $\mathbf{x}_i^{[c]}$, $c = 1, \ldots, \zeta$. Keep the largest distance $\tilde{d}_i$ for each point.
  - Sort the Mahalanobis distances in ascending order.
  - Among all M trials, find the sequence $\tilde{d}_{[i]_M}$ with the minimum sum of distances for $\epsilon\%$ of the input points remained for processing.

- Apply the expansion criteria to an increasing sequence of sets and determine the region of interest for a structure.

- In the region of interest find the largest estimate as $\hat{\sigma}$ and collect all points inside this scale.

- Generate $N \ll M$ random trials from these points.

  - Apply the mean shift to all still existing points, to find the closest mode from α.
  - Find $\hat{\alpha}$ at the maximum mode among all N trials, and $\hat{\boldsymbol{\theta}}$ from the same elemental subset.

  - The recovered structure contains n_{in} points which converged to $\pm\hat{\sigma}$ from $\hat{\alpha}$.

- Compute the TLS solution for the structure and remove the n_{in} points from the inputs.
- Go back to $\odot$ and start another iteration.
- If not enough input points remain, sort all structures by their strengths and return the result.

Chapter 4

Robust Method in the Recovery of 2D Structures

Several synthetic and real examples are presented in this section. In most cases a single carrier vector exists, $\zeta = 1$, except for the homography estimation which has two and $\zeta = 2$. The Epanechnikov kernel is used in the mean shift.

The input data for synthetic problems are generated randomly and Gaussian noise is added to each inlier structure. The standard deviation σ_g is specified only to verify the results, while not used in the estimation process. The values of the scales and point amounts for each structure are returned as the output of the algorithm.

All the following experiments were tested on an i7-2617M 1.5GHz PC, and the processing time for each example will be given.

4.1 Estimation of 2D Geometric Primitives

4.1.1 2D Line

In the first example *multiple 2D lines* are estimated. The noisy objective function is

$$\theta_1 x_i + \theta_2 y_i - \alpha \simeq 0 \qquad i = 1, \ldots, n_{in}. \tag{4.1}$$

The input variable $\mathbf{y} = [x\ y]^\top$ is identical with the carrier vector $\mathbf{x}$.

Five lines are placed in a 700×700 plane (Fig.4.1a) and corrupted with different two-dimensional Gaussian noise. They have $n_{in} = 300, 250, 200, 150, 100$ inlier points, and $\sigma_g = 3, 6, 9, 12, 15$, respectively. Another 350 unstructured outliers are uniformly distributed in the image. The amount of points inside each inlier structure is small compared to the entire data.

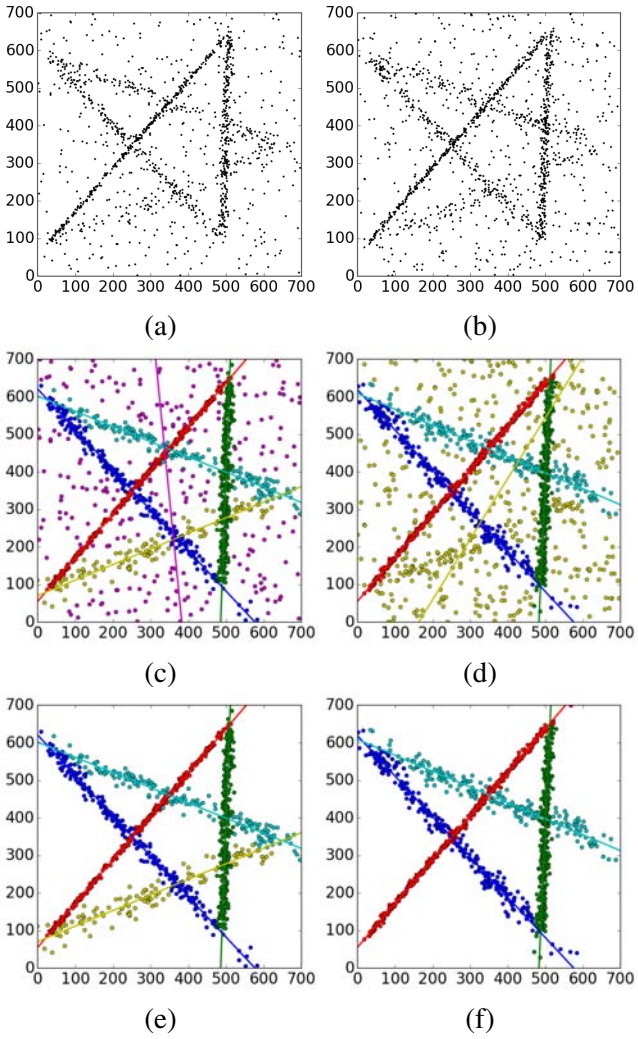


Figure 4.1: Synthetic 2D line estimations. (a) Case 1: five lines with 350 outliers. (b) Case 2: five lines with 500 outliers. (c) Recovered six structures, case 1. (d) Recovered five structures, case 2. (e) Five strongest structures, case 1. (f) Four strongest structures, case 2.

With $M = 1000$, a test result is shown in Fig.4.1c. The algorithm recovers six structures

	red	$green$	$blue$	$cyan$	$yellow$	$purple$
$scale$:	9.6	18.7	28.1	37.1	44.2	370.8
$inliers$:	321	282	240	161	106	240
$strength$:	33.4	15.1	8.5	4.3	2.4	0.6.

The first five structures are inliers with stronger strengths as Fig.4.1e shows. The sixth structure, is formed by outliers distributed over the whole image.

When the randomly generated inputs are tested independently for 100 times, the first four

lines are correctly segmented in all the tests. In the other six tests the weakest line ($n_{in} = 100,\ \sigma_g = 15$) is not correctly located. Of the 94 correct estimations, the average result of the scale estimates and the classified inlier amounts as well as their respective standard deviations are

$$\begin{array}{rccccc} scale: & 10.48 & 19.94 & 29.36 & 36.86 & 38.17 \\ & (1.17) & (2.44) & (5.30) & (10.40) & (18.29) \\ inliers: & 335.9 & 285.8 & 240.8 & 155.6 & 93.4 \\ & (8.2) & (9.5) & (21.3) & (27.0) & (28.4). \end{array}$$

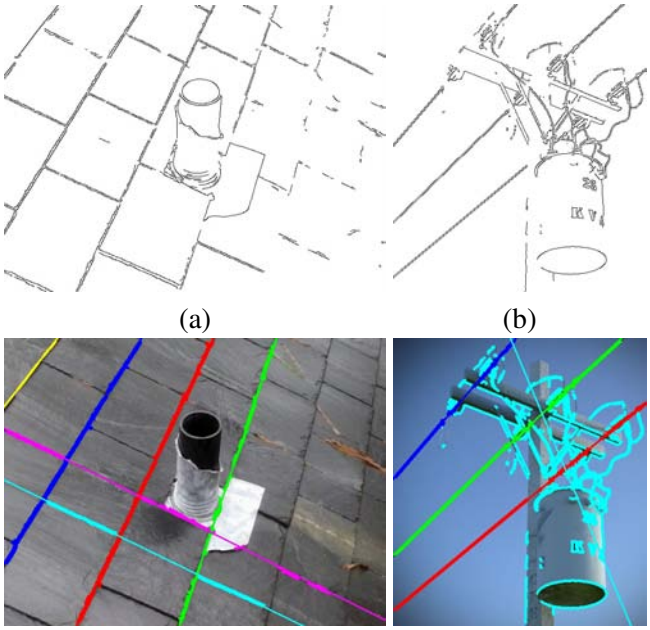
The average processing time is 0.58 seconds. The estimated scale covers about $3\sigma_g$ area of an inlier structure. In general, the number of classified inliers is larger than the true amount due to the presence of outliers in the same area.

As the outlier amount increases, the weakest structure will gradually blend into the background, and the expansion criteria can hardly separate inliers and outliers for this structure. In Fig.4.1b a case is shown where the outlier amount is raised to 500 instead of 350, and the number of inliers remain the same. In Fig.4.1d five structures are returned

$$\begin{array}{rccccc} & red & green & blue & cyan & yellow \\ scale: & 12.0 & 18.6 & 33.8 & 41.5 & 483.6 \\ inliers: & 351 & 304 & 256 & 182 & 407 \\ strength: & 29.3 & 16.4 & 7.6 & 4.4 & 0.8. \end{array}$$

The last structure is mixed with the outliers, and thus not estimated correctly. The first four inlier structures are still retained based on the strength (Fig.4.1f). In 100 tests the weakest line is detected 64 times, the fourth structure ($n_{in} = 150,\ \sigma_g = 12$) 98 times, and the three strongest structures are estimated correctly in all the trials.

In Fig.4.2a and Fig.4.2b, the Canny edge detection extracts similar sizes of input data (8310 and 8072 points) from two real images. Again with $M = 1000$, the six strongest line structures are superimposed over the original image in Fig.4.2c. In Fig.4.2d the three line structures together with the first outlier structure are shown. The processing time depends on the number of structures that detected by the estimator, these two estimations take 7.44 and 4.35 seconds, respectively.



(a) (b)

(c) (d)

Figure 4.2: 2D lines in real images. (a) *Roof*: Canny edges, 8310 points. (b) *Pole*: Canny edges, 8072 points. (c) *Roof*: Six strongest inlier structures. (d) *Pole*: Three strongest inlier structures and one outlier structure.

4.1.2 2D Ellipse

In the next experiment *multiple 2D ellipses* are estimated. The noisy objective function is

$$(\mathbf{y}_i - \mathbf{y}_c)^\top \mathbf{Q} (\mathbf{y}_i - \mathbf{y}_c) - 1 \simeq 0 \qquad i = 1, \ldots, n_{in} \tag{4.2}$$

where $\mathbf{Q}$ is a symmetric 2×2 positive definite matrix and $\mathbf{y}_c$ is the position of the ellipse center. Given the input variable $\mathbf{y} = [x\ y]^\top$, the carrier is derived as $\mathbf{x} = [x\ y\ x^2\ xy\ y^2]$. The condition $4\theta_3\theta_5 - \theta_4^2 > 0$ also has to be satisfied in order to represent an ellipse. We also enforce the constraint that the major axis cannot be more than 10 times longer than the minor axis, to avoid classifying line segment as a part of a very flat ellipse.

The transpose of the 5×2 Jacobian matrix is

$$\mathbf{J}_{\mathbf{x}|\mathbf{y}}^\top = \begin{bmatrix} 1 & 0 & 2x & y & 0 \\ 0 & 1 & 0 & x & 2y \end{bmatrix}. \tag{4.3}$$

The ellipse fitting is a nonlinear estimation and biased, especially for the part with large curvature. When the inputs are perturbed with zero mean Gaussian noise with σ_g, the standard deviation of carrier vector $\mathbf{x}$ relative to the true value $\mathbf{x}_o$ is not zero mean

$$\mathbf{E}(\mathbf{x} - \mathbf{x}_o) = [0 \quad 0 \quad \sigma_g^2 \quad 0 \quad \sigma_g^2]^\top \tag{4.4}$$

since the carrier contains $x^2,\ y^2$ terms. A bias in the estimate can be clearly seen when only a small segment of the noisy ellipse is given in the input. Taking into account also the second order statistics in estimation still does not eliminate the bias. See papers [54], [85] and their references for additional methods.

In Fig.4.3a three ellipses are placed with 350 outliers in the background. The inlier structures have $n_{in} = 300, 250, 200$ and $\sigma_g = 3, 6, 9$. The smallest ellipse with $n_{in} = 200$ is corrupted with the largest noise $\sigma_g = 9$. We use $M = 5000$ in the ellipse fitting experiments. When tested (Fig.4.3c), four ellipses are recovered

	red	$green$	$blue$	$cyan$
$scale:$	12.1	28.9	48.0	1321.2
$inliers:$	337	292	222	248
$strength:$	28.0	10.1	4.6	0.2.

Based on results sorted by strength, the first three structures are inliers and are shown in Fig.4.3e.

When the estimation is repeated 100 times, the three inlier structures are correctly located 97 times, while in the other three tests the smallest ellipse is not estimated correctly. From the 97 correct estimations, the average scales, the classified inlier amounts, along with their standard deviations are

$scale:$	11.60	21.59	32.87
	(1.54)	(3.64)	(14.71)
$inliers:$	336.2	272.9	196.4
	(8.2)	(27.5)	(53.2).

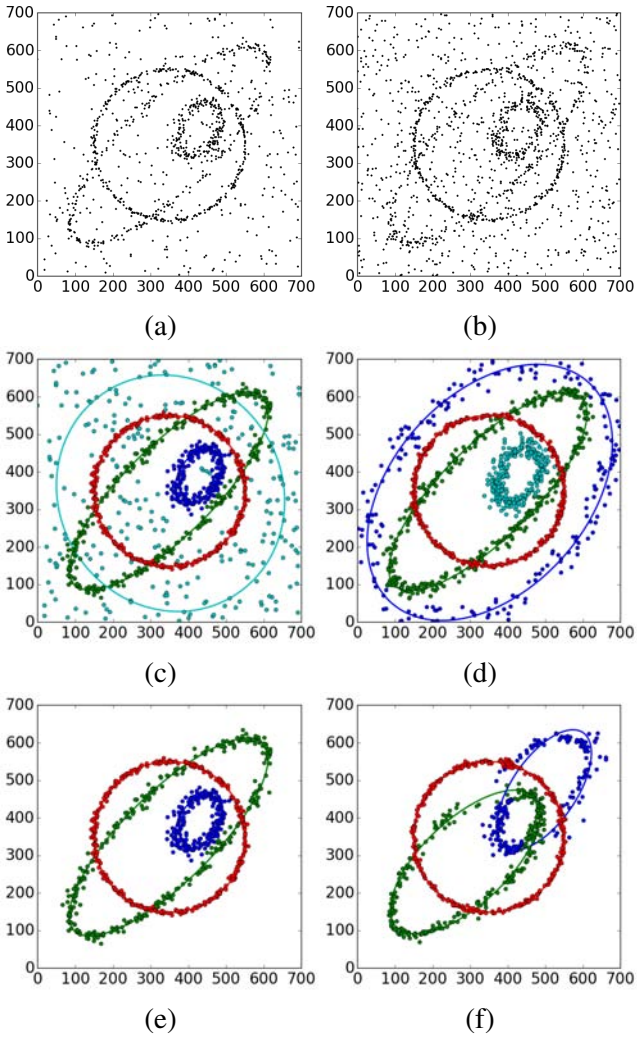
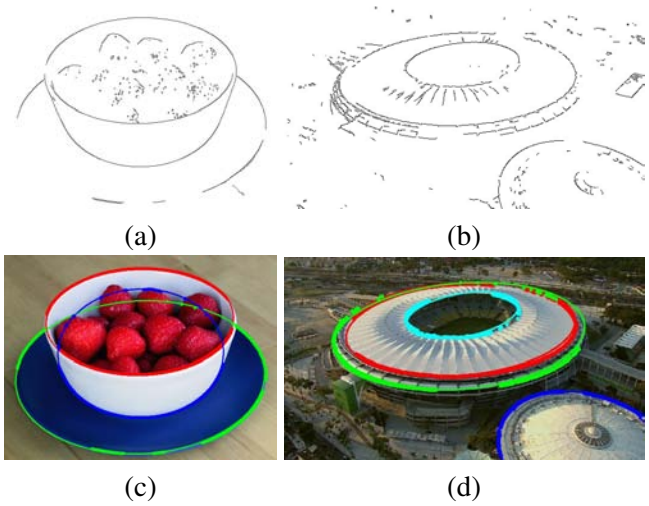


Figure 4.3: Synthetic 2D ellipse estimations. (a) Case 1: three ellipses and 350 outliers. (b) Case 2: three ellipses and 800 outliers. (c) Recovered four structures, case 1. (d) Recovered first four structures, case 2. (e) Three strongest structures, case 1. (f) Interaction between two ellipses, case 2.

The average processing time is 3.28 seconds.

When the outlier amount reaches the limit, the inlier structure with weakest strength may no longer be sorted before the outliers. The scale estimate becomes inaccurate due to the heavy outlier noise, and the outliers can form more dense structure with comparable strength. When 800 outliers (Fig.4.3b) are placed in the image, a test gives the result in Fig.4.3d. The outlier structure (blue) has a strength of 4.8, while the value of the inliers (cyan) is 3.9. However, the first two inlier structures are still recovered due to their stronger strengths.

Fig.4.3b also gives an example to show one of the limitation explained in Section 3.4, when



(a) (b)

(c) (d)

Figure 4.4: 2D ellipses in real images. (a) *Strawberries*: Canny edges, 4343 points. (b) *Stadium*: Canny edges, 4579 points. (c) *Strawberries*: Three strongest inlier structures. (d) *Stadium*: Four strongest inlier structures (see also text).

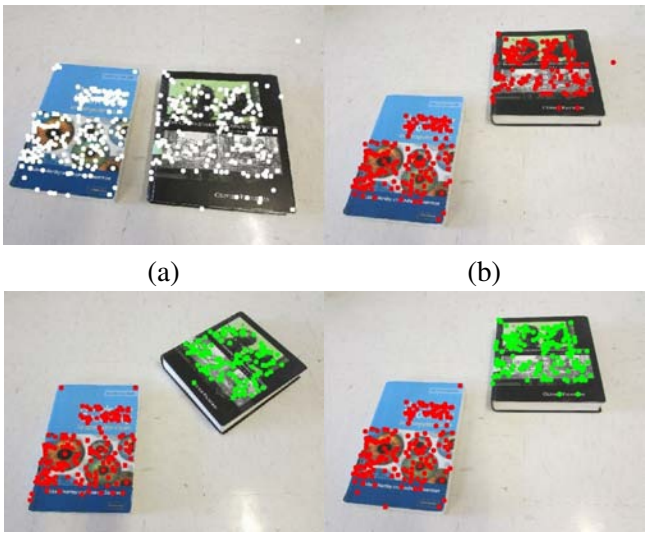
the inlier strength is too weak to tolerate more outliers. In Fig.4.3f two inlier structures interact, and the mean shifts converge to incorrect modes.

From Canny edge detection, 4343 and 4579 points are obtained from two real images containing several objects with elliptic shapes, as shown in Fig.4.4a and Fig.4.4b. With $M = 5000$, the three strongest ellipses are drawn in Fig.4.4c, superimposed over the original images. The processing time is 18.90 seconds in this case. In Fig.4.4d the estimation takes 23.14 seconds to detect four strongest ellipses, which are inlier structures. After 100 repetitive tests using the data shown in Fig.4.4b, only the first two ellipses (red and green) are detected reliably in 98 times. The other two ellipses (blue and cyan) have smaller amounts of inliers and therefore are less stable. The data acquired from Canny edge detection do not necessarily render the overall inlier structures in a more dense state. Preprocessing on the edge data is generally required for better performance.

4.2 Estimation of Projective Geometric Relations from 2D Images

4.2.1 Fundamental Matrix

The next experiment shows the estimation of the *fundamental matrices*. The corresponding linear space was introduced in Subsection 3.2.1. The 3×3 matrix $\mathbf{F}$ is rank-2 and a recent paper



(a) (b)

(c) (d)

Figure 4.5: Motion segmentation with different objective functions. In the following figures, the input points (white) are shown in the first view, the processed structures (colored) in the second view. (a) The input points. Fundamental matrix: (b) Translation only. (c) Translation and rotation. Homography: (d) Translation only.

[29] solved the non-convex problem iteratively by a convex moments based polynomial optimization. It compared the results with a few RANSAC type algorithms. The method required several parameters and in each example only one fundamental matrix was recovered.

The fundamental matrix cannot be used to segment objects with only translational motions, as proved in [24]. The input shown in Fig.4.5a comes out in Fig.4.5b as a single structure instead of two. Only when one of the books has a large enough rotation, the correct output is obtained (Fig.4.5c). Applying the homography estimation (Subsection 4.2.2) to the translational case, the two books can be easily separated (Fig.4.5d).

Each example of Fig.4.6 shows the movement of multiple rigid objects. The point correspondences are extracted by OpenCV with a distance ratio of 0.8 for SIFT [62], giving 608, 614 and 457 matches, respectively. With $M = 5000$, the structures are retained as

Fig.4.6a	red	$green$	$blue$
$scale$:	0.56	0.73	11.78
$inliers$:	407	101	51
$strength$:	727.3	139.3	4.33.

(a)

(b)

(c)

Figure 4.6: Fundamental matrix estimation. (a) Image pair from *Hopkins 155 dataset* with two inlier and one outlier structures. (b) The *books* with three inlier and one outlier structures. (c) The *dinabooks* from [73] with four inlier and one outlier structures.

Fig.4.6b	red	$green$	$blue$	$cyan$
$scale$:	0.46	0.40	1.12	10.42
$inliers$:	192	96	221	47
$strength$:	413.4	242.8	196.8	4.5.

Fig.4.6c	red	$green$	$blue$	$cyan$	$yellow$
$scale$:	0.22	0.72	0.65	0.70	23.9
$inliers$:	135	117	84	48	43
$strength$:	623.0	161.5	129.0	68.2	1.8.

The estimations take 1.75, 2.30 and 2.10 seconds for these three cases. In real images, the outlier structures can be easily filtered out since they have much larger scales than the inliers. It can be observed that the scales of the inlier structures are very close, therefore the methods

with fixed thresholds may be used here. However, if the images are scaled before estimation, the error in the inliers will change proportionally. Correct scale estimate can only be found adaptively from the input data.

As discussed in Section 3.4, the first (red) and the fourth (cyan) structures obtained from Fig.4.6c can be fused as a single structure. This merge has to be done by post-processing in the input space but also requires a threshold from the user.

The SIFT matches are error-prone and false correspondences always exist. If the images contain repetitive features, such as the exterior of buildings, parametrization of the repetitions can reduce the uncertainty [76]. Preprocessing of the images is not described in this dissertation, therefore we will not explain it further.

4.2.2 Homography

The last example in this section is for 2D *homography* estimation. In [74], a universal framework for RANSAC, the Universal RANSAC (USAC), was introduced. This method failed in homography estimation when a wide angle existed between two images [59]. In the latter paper, a very dense sampling of the grid is used and combined with probabilistic reasoning to obtain the inlier rate estimate. Only one inlier structure can be recovered each time.

Each inlier structure of 2D homography is represented by a 3×3 matrix $\mathbf{H}$, which connects two planes inside the image pair

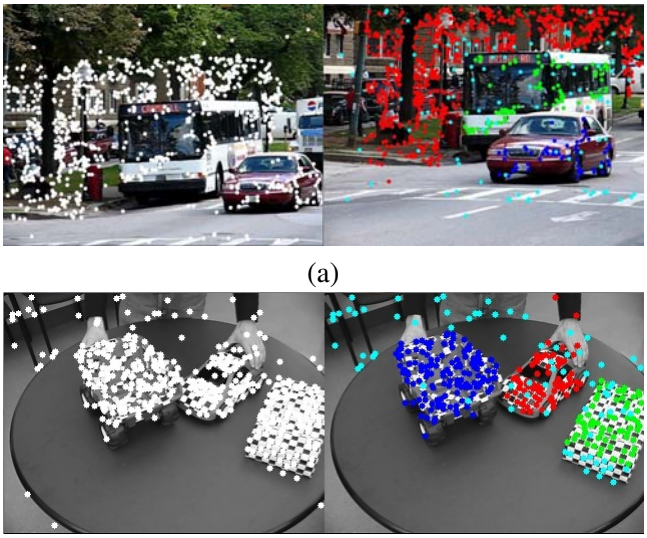
$$\mathbf{y}'_i \simeq \mathbf{H}\mathbf{y}_i, \quad i = 1, \ldots, n_{in} \tag{4.5}$$

where $\mathbf{y} = [x\ y\ 1]^\top$ and $\mathbf{y}' = [x'\ y'\ 1]^\top$ are the homogeneous coordinates in these two images.

The homography estimation has $\zeta = 2$. The input variables are $[x \quad y \quad x' \quad y']^\top$. Two linearized relations can be derived from the constraint (4.5) by the direct linear transformation (DLT)

$$\mathbf{A}_i\mathbf{h} = \begin{bmatrix} -\mathbf{y}_i^\top & \mathbf{0}_3^\top & x'_i\mathbf{y}_i^\top \\ \mathbf{0}_3^\top & -\mathbf{y}_i^\top & y'_i\mathbf{y}_i^\top \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} \simeq \mathbf{0}_2 \ . \tag{4.6}$$

The matrix $\mathbf{A}_i$ is 2×9 and both rows satisfy the relations with the vector derived from the



(a)

(b)

Figure 4.7: Homography estimation with image pairs from *Hopkins 155* dataset [8]. (a) Three inlier and one outlier structures. (b) Three inlier and one outlier structures.

matrix $\text{vec}(\mathbf{H}^\top) = \mathbf{h} = \boldsymbol{\theta}$.

The carriers are obtained from the two rows of $\mathbf{A}_i$

$$\begin{aligned} \mathbf{x}^{[1]} &= [-x \ -y \ -1 \ 0 \ 0 \ 0 \ x'x \ x'y \ x']^\top \\ \mathbf{x}^{[2]} &= [0 \ 0 \ 0 \ -x \ -y \ -1 \ y'x \ y'y \ y']^\top. \end{aligned} \tag{4.7}$$

The transpose of the two 9×4 Jacobians matrices are

$$\begin{aligned} \mathbf{J}^\top_{\mathbf{x}_i^{[1]}|\mathbf{y}} &= \begin{bmatrix} -\mathbf{I}_{2\times 2} & & x_i'\mathbf{I}_{2\times 2} & \mathbf{0}_2 \\ \mathbf{0}_2^\top & \mathbf{0}_{4\times 4} & \mathbf{y}_i^\top & \\ \mathbf{0}_2^\top & & \mathbf{0}_2^\top & 0 \end{bmatrix} \\ \mathbf{J}^\top_{\mathbf{x}_i^{[2]}|\mathbf{y}} &= \begin{bmatrix} & -\mathbf{I}_{2\times 2} & & y_i'\mathbf{I}_{2\times 2} & \mathbf{0}_2 \\ \mathbf{0}_{4\times 3} & \mathbf{0}_2^\top & \mathbf{0}_4 & \mathbf{0}_2^\top & 0 \\ & \mathbf{0}_2^\top & & & \mathbf{y}_i^\top \end{bmatrix}. \end{aligned} \tag{4.8}$$

Based on the discussion in Subsection 3.2.2, for every $\boldsymbol{\theta}$ only the larger Mahalanobis distance is used for each input $\mathbf{y}_i,\ i = 1, \ldots, n$.

The motion segmentation involves only translation in 3D in Fig.4.7a and Fig.4.7b, both

images are taken from the *Hopkins 155* dataset [8]. With $M = 2000$, the processing time is 1.12 and 1.09 seconds for the inputs containing 990 and 482 SIFT point pairs, respectively. As mentioned in Section 3.4, in Fig.4.7a the estimator cannot separate these two 3D planes on the bus because the 2D homographies corresponding to them are very similar. The condition (3.13) does not stop where it should since the points on either 2D planes are not dense enough. The example shows that the desired results can only be obtained by increasing the amount of inliers from preprocessing. In Fig.4.7b, the three objects can be correctly separated in spite of the very small motions, due to the stronger strengths in all the inlier structures.

Fig.4.7a	red	$green$	$blue$	$cyan$
$scale:$	1.62	1.12	4.49	203.11
$inliers:$	713	101	67	105
$strength:$	440.2	89.5	14.9	0.5

Fig.4.7b	red	$green$	$blue$	$cyan$
$scale:$	0.20	0.17	0.56	5.16
$inliers:$	107	88	165	89
$strength:$	529.6	517.7	293.4	17.3.

The importance of preprocessing can also be seen in [80], where the geometric and appearance priors were used to increase the amount of consistent matches before estimation, when the PROSAC [30] failed in the presence of many incorrect matches.

The significance of inlier amounts is demonstrated in Fig.4.8. The point pairs from OpenCV SIFT are used in Fig.4.8a and Fig.4.8c, while the more dense points from datasets [73] are tested in Fig.4.8b and Fig.4.8d for comparison. After the estimations, the structures are sorted by their strengths until the first outlier structure appears, where a large increase in the scale estimate can be observed.

(a) (b)

(c) (d)

Figure 4.8: Homography estimation. (a) *Merton College* with 536 point pairs. Four inlier and one outlier structures. (b) *Merton College* with 1982 point pairs. Five inlier and one outlier structures. (c) *Unionhouse* with 619 point pairs. Three inliers and one outlier structures. (d) *Unionhouse* with 2084 point pairs. Five inliers and one outlier structures.

Fig.4.8a	*red*	*green*	*blue*	*cyan*	*yellow*
scale :	1.41	0.89	2.01	1.37	14.26
inliers :	157	87	158	51	44
strength :	110.8	97.2	78.5	37.0	3.1

Fig.4.8b	*red*	*green*	*blue*	*cyan*	*yellow*	*purple*
scale :	0.97	0.92	2.55	0.41	0.97	798.0
inliers :	507	478	529	55	127	286
strength :	521.5	518.9	207.6	133.3	130.1	0.4

Fig.4.8c	*red*	*green*	*blue*	*cyan*
scale :	0.59	0.48	0.74	60.34
inliers :	176	87	91	204
strength :	295.9	181.0	123.3	3.4

Fig.4.8d	red	$green$	$blue$	$cyan$	$yellow$	$purple$
$scale:$	0.67	1.35	0.57	1.83	1.53	82.18
$inliers:$	495	508	162	513	69	210
$strength:$	738.7	376.7	282.3	280.1	44.9	2.6.

The processing time for these four examples is 1.29, 3.07, 1.36 and 3.78 seconds, respectively. This is a faster implementation than RCMSA [73] which cost 24.58 and 25.40 sec for the same image pairs in Fig.4.8b and Fig.4.8d on the same computer. With denser inlier points, more inlier structures can be detected.

Chapter 5

Robust Method in the Reconstruction of 3D Point Cloud with Photogrammetry

In this chapter we demonstrate the use of the estimator for multiple inlier structures in the structure from motion (SfM) algorithm. The presentation follows the processing pipeline introduced in Fig.2.7, but now we will detail the implementation of each procedure. We follow an example to illustrate the entire process, from the capture of 2D images to the final output as 3D point cloud, and this example will also be used for 3D surface extraction in Chapter 6.

5.1 Capture of the 2D Image Sequence

All the image sequences used in this dissertation are extracted from videos taken with one of the three digital cameras (Fig.5.1a):

- Canon PowerShot SX500 IS, 16-megapixel.
- GoPro Hero 4, 12-megapixel.
- iphone SE camera, 12-megapixel.

The cameras capture short videos with different sizes, resolutions and qualities. For the Canon and GoPro cameras we resize the 2D images to 640×480, while the iphone videos are scaled to 1024×576 resolution. In Section 2.2 we have shown that only the focal lengths f have to be estimated from the cameras to obtained the intrinsic matrices $\mathbf{K}$.

The hand-held camera records a video circling around a 3D physical model, where shaking and defocusing can happen due to the movement. For the SfM algorithm a total of 50-100 images are extracted. In Fig.5.1b five samples are shown from the video. The 3D reconstruction example from this 2D sequence will be followed in Section 5.2 to Section 5.4.

(a)

(b)

Figure 5.1: Capture of a 2D image sequence. (a) The three cameras used to capture different videos. (b) Five 2D frames extracted from the video used in the example.

5.2 Robust Matching of Image Features

The video was captured by the iphone SE camera and 70 images were extracted. To obtain the SIFT features, see Section 2.1, for every frame in the 2D image sequence a built-in functions of OpenCV [9] [1] was applied. The same value 0.8 is used in the SIFT distance ratio test. The amount of SIFT features relies on the size of the image and the scene complexity. From every 1024×576 image, around $7,000 \sim 10,000$ SIFT points were extracted (Fig. 5.2).

After the SIFT feature extraction, the raw matching with FLANN [71] has to be applied to every two-view image pair. If the 2D images are not arranged according to their corresponding camera locations, the matching algorithm has to exhaust all the possible image pairs. This is an expensive computation if many images are to be processed, e.g., half the entire processing time for building Rome [14], and multithreading solution were proposed [97]. In our example, we take advantage that the image sequence is extracted from a continuous video and the good image pairs will be detected among adjacent frames.

[1] For more details, check OpenCV documentation on SIFT feature matching at http://docs.opencv.org/2.4/-modules/nonfree/doc/feature_detection.html.

```
Detecting SIFT features ...
  (0)9587 (1)10255 (2)10513 (3)10526 (4)8969 (5)8949 (6)10487 (7)8554 (8)10431 (
9)10815 (10)10011 (11)9976 (12)8691 (13)9761 (14)3144 (15)10030 (16)6322 (17)106
74 (18)7947 (19)9952 (20)9898 (21)9281 (22)10357 (23)10700 (24)9475 (25)6152 (26
)8837 (27)10317 (28)10906 (29)10112 (30)10246 (31)10233 (32)9165 (33)9351 (34)92
65 (35)9332 (36)8905 (37)9507 (38)9905 (39)8684 (40)9938 (41)9154 (42)10098 (43)
8281 (44)9249 (45)9279 (46)8657 (47)8830 (48)9285 (49)8819 (50)9538 (51)10067 (5
2)8438 (53)10031 (54)9836 (55)8846 (56)10200 (57)10192 (58)9741 (59)9977 (60)101
81 (61)9642 (62)9537 (63)9258 (64)9318 (65)9705 (66)8303 (67)9485 (68)10566 (69)
9235
```

Figure 5.2: Extraction of SIFT features in 2D image sequence.

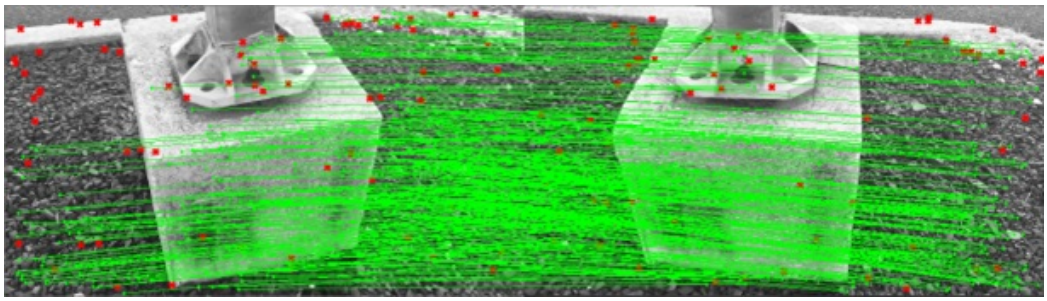


Figure 5.3: Robust filtering of outlier SIFT matches.

5.2.1 Robust Filtering of Outlier Matches

The raw matching compares only the similarity between two SIFT descriptors. The obtained point correspondences will not satisfy the geometric verification using the fundamental matrix $\mathbf{F}$ and neither the rigid scene assumption between two images. Therefore, outlier pairs always exist and robust filtering of outliers is required.

The fundamental matrix estimation (Subsection 4.2.1) is applied to each pair of images. Since the images are ordered, each image will be paired with the neighbouring frames to find the *inlier* SIFT matches, until the epipolar geometry can no longer be estimated. We retain, like in Fig.5.3, only the strongest structure as the inlier because generally the target 3D rigid object occupies most of the scene. The raw matching gives 379 matches between these two sample frames, and after the robust filtering, 314 pairs (green) are retained and the red points are discarded as outliers. Thus, 83% pairs in this example are classified as the inlier structure.

In Subsection 3.3.3, we have shown that the new robust algorithm segments all the input data into different structures, and does not rely on any threshold to identify the inlier structures from outliers. A special case appears when the fundamental matrix cannot be found with a decent scale estimate, from a pair of images taken in two distant locations. Once happened, even the strongest structure should not be claimed as inlier.

```
Matching features ...
  <0> : < 1 2 3 4 5 6 7 8 69 68 67 66 65 64 >
  <1> : < 2 3 4 5 6 7 8 9 69 68 67 66 65 64 63 >
  <2> : < 3 4 5 6 7 8 9 10 69 68 67 66 65 64 63 >
  <3> : < 4 5 6 7 8 9 10 69 68 67 66 65 64 >
  <4> : < 5 6 7 8 9 10 11 69 68 67 66 >
  <5> : < 6 7 8 9 10 11 12 69 68 67 >
  <6> : < 7 8 9 10 11 12 13 68 >
  <7> : < 8 9 10 11 12 13 >
  <8> : < 9 10 11 12 13 >
  <9> : < 10 11 12 13 15 >
  <10> : < 11 12 13 15 16 >
  <11> : < 12 13 14 15 16 17 >
  <12> : < 13 14 15 16 17 18 >
  <13> : < 14 15 16 17 18 19 20 >
  <14> : < 15 16 17 18 >
  <15> : < 16 17 18 19 20 21 >
  <16> : < 17 18 19 20 21 >
  <17> : < 18 19 20 21 22 23 >
  <18> : < 19 20 21 22 23 >
  <19> : < 20 21 22 23 24 >
  <20> : < 21 22 23 24 >
  <21> : < 22 23 24 25 26 28 >
  <22> : < 23 24 25 26 27 29 >
  <23> : < 24 25 26 27 28 29 >
  <24> : < 25 26 27 28 29 30 >
  <25> : < 26 27 28 29 31 >
  <26> : < 27 28 29 30 31 >
  <27> : < 28 29 30 31 32 33 34 >
  <28> : < 29 30 31 32 33 >
  <29> : < 30 31 32 33 34 35 36 >
  <30> : < 31 32 33 34 35 36 37 38 >
  <31> : < 32 33 34 35 36 37 38 >
  <32> : < 33 34 35 36 37 38 39 >
  <33> : < 34 35 36 37 38 39 40 41 >
  <34> : < 35 36 37 38 39 40 41 >
  <35> : < 36 37 38 39 40 41 42 >
  <36> : < 37 38 39 40 41 42 43 >
  <37> : < 38 39 40 41 42 43 44 >
  <38> : < 39 40 41 42 43 44 45 >
  <39> : < 40 41 42 43 44 45 46 >
  <40> : < 41 42 43 44 45 46 47 >
  <41> : < 42 43 44 45 46 47 >
  <42> : < 43 44 45 46 47 49 >
  <43> : < 44 45 46 47 48 49 >
  <44> : < 45 46 47 48 49 50 >
  <45> : < 46 47 48 49 50 51 >
  <46> : < 47 48 49 50 51 >
  <47> : < 48 49 50 51 52 53 >
  <48> : < 49 50 51 52 53 >
  <49> : < 50 51 52 53 54 55 >
  <50> : < 51 52 53 54 55 56 >
  <51> : < 52 53 54 55 56 >
  <52> : < 53 54 55 56 57 >
  <53> : < 54 55 56 57 58 >
  <54> : < 55 56 57 58 59 60 >
  <55> : < 56 57 58 59 60 61 >
  <56> : < 57 58 59 60 61 62 >
  <57> : < 58 59 60 61 62 >
  <58> : < 59 60 61 62 >
  <59> : < 60 61 62 63 64 65 >
  <60> : < 61 62 63 64 65 66 >
  <61> : < 62 63 64 65 >
  <62> : < 63 64 65 66 67 >
  <63> : < 64 65 66 67 68 69 >
  <64> : < 65 66 67 68 69 >
  <65> : < 66 67 68 69 >
  <66> : < 67 68 69 >
  <67> : < 68 69 >
  <68> : < 69 >
```

Figure 5.4: Paired image indices after robust filtering of outlier features.

From the experiments in Subsection 4.2.1, the scales of correct fundamental matrices in real images are around $\hat{\sigma} = 1.0$. This value can be used as the threshold to eliminate unreliable image pairs. This is a rough elimination but can reduce a lot of the possible errors. Although it appears similar to the user-given scale in RANSAC, this value has a completely different role:

- The $\hat{\sigma} = 1.0$ is *not used* as a threshold to classify inliers.
- It is only an upper bound of inlier noise to avoid false result.
- The new robust algorithm works as before for the strongest structure.

In our example, the paired image indices are shown in Fig.5.4, where each frame is matched with around 10 other images. The good pairs are recorded only in the entry of the smaller image index, e.g., the two particular frames ($< 4 >$ and $< 11 >$) in Fig.5.3 are listed after frame 4, while not shown repetitively in frame 11's. Thus when counting the total pairs of frame 4, those in entries of frames $0, 1, 2$ and 3 should also be checked.

5.3 Track Initialization and Expansion

Let concentrate on a single track in this section. For the auto-calibration, we require the rotation between the pair selected as initial two-view pair to be wider than 5°, as discussed in Subsection 2.4.1. The approximate focal length f is estimated from the Kruppa equation (2.9). The value of f, though imprecise, provides the starting point for the 3D point triangulation, and will be optimized through bundle adjustment.

The drift error accumulates as more images are registered in the track, in [32] a method was proposed to detect and remove drift, by enforcing the geometric constraints on the start frame and end frame. While this error cannot be fully eliminated since in general the bundle adjustment converges to a local minimum, the metric accuracy of the reconstructed 3D point cloud heavily relies on the 2D image sequence. A rule of thumb for the track expansion, is to register as many images as possible, and at the same time maintain the reprojection error in a reasonable range. In the next subsection, we stop the track expansion once the error increases beyond a threshold, and then reduce the drift distortion through the overall bundle adjustment in Section 5.4.

5.3.1 Bundle Adjustment in Track Expansion

The bundle adjustment reduces the reprojection error as was explained in Subsection 2.4.3. In this subsection we list the Jacobian matrices of the parameters, for both cameras and the 3D points.

Each camera matrix is parameterized by 8 parameters, namely the focal length f, the 3-component vector $\mathbf{r} = [r_1, r_2, r_3]$ for the rotation axis, the angle of rotation ϕ, and another 3-component vector $\mathbf{t} = [t_1, t_2, t_3]$ for the translation. Let $c\phi = \cos\phi$ and $s\phi = \sin\phi$, from the Rodrigues' rotation formula, the 3×3 rotation matrix $\mathbf{R}$ can be expressed by the axis-angle

formulation [67, page 49]

$$\mathbf{R}(\mathbf{r},\phi) = c\phi \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + (1-c\phi) \begin{bmatrix} r_1^2 & r_1r_2 & r_1r_3 \\ r_1r_2 & r_2^2 & r_2r_3 \\ r_1r_3 & r_2r_3 & r_3^2 \end{bmatrix} + s\phi \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} c\phi + r_1^2(1-c\phi) & r_1r_2(1-c\phi) - r_3s\phi & r_1r_3(1-c\phi) + r_2s\phi \\ r_1r_2(1-c\phi) + r_3s\phi & c\phi + r_2^2(1-c\phi) & r_2r_3(1-c\phi) - r_1s\phi \\ r_1r_3(1-c\phi) - r_2s\phi & r_2r_3(1-c\phi) + r_1s\phi & c\phi + r_3^2(1-c\phi) \end{bmatrix}. \tag{5.1}$$

The Jacobian of $\mathbf{R}$ with respect to the angle ϕ and axes $\mathbf{r}$ are

$$\frac{\partial \mathbf{R}}{\partial \phi} = \begin{bmatrix} -s\phi + r_1^2 s\phi & r_1r_2s\phi - r_3c\phi & r_1r_3s\phi + r_2c\phi \\ r_1r_2s\phi + r_3c\phi & -s\phi + r_2^2 s\phi & r_2r_3s\phi - r_1c\phi \\ r_1r_3s\phi - r_2c\phi & r_2r_3s\phi + r_1c\phi & -s\phi + r_3^2 s\phi \end{bmatrix} \tag{5.2}$$

$$\frac{\partial \mathbf{R}}{\partial r_1} = \begin{bmatrix} 2r_1(1-c\phi) & r_2(1-c\phi) & r_3(1-c\phi) \\ r_2(1-c\phi) & 0 & -s\phi \\ r_3(1-c\phi) & s\phi & 0 \end{bmatrix} \tag{5.3}$$

$$\frac{\partial \mathbf{R}}{\partial r_2} = \begin{bmatrix} 0 & r_1(1-c\phi) & s\phi \\ r_1(1-c\phi) & 2r_2(1-c\phi) & r_3(1-c\phi) \\ -s\phi & r_3(1-c\phi) & 0 \end{bmatrix} \tag{5.4}$$

$$\frac{\partial \mathbf{R}}{\partial r_3} = \begin{bmatrix} 0 & -s\phi & r_1(1-c\phi) \\ s\phi & 0 & r_2(1-c\phi) \\ r_1(1-c\phi) & r_2(1-c\phi) & 2r_3(1-c\phi) \end{bmatrix}. \tag{5.5}$$

We concluded in Section 2.2 that a 3×4 camera matrix $\mathbf{P}$ without knowing the focal length, has only seven degrees of freedom, the one additional constraint applied here is $||\mathbf{r}|| = 1$.

The implementation of bundle adjustment in [47, Appendix 6] requires the Jacobians of the reprojected image coordinate $\hat{\mathbf{x}}_{ij} = [\hat{x}_{ij}, \hat{y}_{ij}]$, with respect to the camera parameters in $\mathbf{P}_j$ and the 3D points $\mathbf{X}_i$, denoted by $\mathbf{a}_j = [f_j, \phi_j, r_{1j}, r_{2j}, r_{3j}, t_{1j}, t_{2j}, t_{3j}]$ and $\mathbf{b}_i = [X_i, Y_i, Z_i]$

respectively, where

$$\hat{x}_{ij} = \frac{\mathbf{P}_{1_j}^{\top}\,\mathbf{X}_i}{\mathbf{P}_{3_j}^{\top}\,\mathbf{X}_i}, \qquad \hat{y}_{ij} = \frac{\mathbf{P}_{2_j}^{\top}\,\mathbf{X}_i}{\mathbf{P}_{3_j}^{\top}\,\mathbf{X}_i}. \tag{5.6}$$

Plug (5.1) into $\mathbf{P} = \mathbf{K}\ [\ \mathbf{R}\ |\ \mathbf{t}\ ]$, and let $\mathbf{P'} = [\ \mathbf{R}\ |\ \mathbf{t}\ ]$ represent the extrinsic matrix, the following expressions of Jacobians can be derived.

For focal length f

$$\frac{\partial \hat{x}_{ij}}{\partial f_j} = \frac{\hat{x}_{ij}}{f_j}, \qquad \frac{\partial \hat{y}_{ij}}{\partial f_j} = \frac{\hat{y}_{ij}}{f_j}. \tag{5.7}$$

For ϕ, r_1, r_2 and r_3 in $\mathbf{a}_j$

$$\begin{aligned}
\frac{\partial \hat{x}_{ij}}{\partial \mathbf{a}_j} &= \frac{1}{\mathbf{P'}_{3j}^{\top}\mathbf{X}_i}\left[f_j\left(\frac{\partial \mathbf{R}_j}{\partial \mathbf{a}_j}\right)_1^{\top} - \hat{x}_{ij}\left(\frac{\partial \mathbf{R}_j}{\partial \mathbf{a}_j}\right)_3^{\top}\right]\mathbf{X}_i \\
\frac{\partial \hat{y}_{ij}}{\partial \mathbf{a}_j} &= \frac{1}{\mathbf{P'}_{3j}^{\top}\mathbf{X}_i}\left[f_j\left(\frac{\partial \mathbf{R}_j}{\partial \mathbf{a}_j}\right)_2^{\top} - \hat{y}_{ij}\left(\frac{\partial \mathbf{R}_j}{\partial \mathbf{a}_j}\right)_3^{\top}\right]\mathbf{X}_i.
\end{aligned} \tag{5.8}$$

For t_1, t_2 and t_3 in $\mathbf{a}_j$

$$\begin{aligned}
\frac{\partial \hat{x}_{ij}}{\partial t_{1j}} = \frac{\partial \hat{y}_{ij}}{\partial t_{2j}} &= \frac{f_j}{\mathbf{P'}_{3j}^{\top}\mathbf{X}_i}, \qquad \frac{\partial \hat{x}_{ij}}{\partial t_{2j}} = \frac{\partial \hat{y}_{ij}}{\partial t_{1j}} = 0, \\
\frac{\partial \hat{x}_{ij}}{\partial t_{3j}} &= \frac{-\hat{x}_{ij}}{\mathbf{P'}_{3j}^{\top}\mathbf{X}_i}, \qquad \frac{\partial \hat{y}_{ij}}{\partial t_{3j}} = \frac{-\hat{y}_{ij}}{\mathbf{P'}_{3j}^{\top}\mathbf{X}_i}.
\end{aligned} \tag{5.9}$$

For X, Y and Z in $\mathbf{b}_i$

$$\begin{aligned}
\frac{\partial \hat{x}_{ij}}{\partial \mathbf{b}_j} &= \frac{1}{\mathbf{P'}_{3j}^{\top}\mathbf{X}_i}\left[f_j\mathbf{P'}_{1j}^{\top} - \hat{x}_{ij}\mathbf{P'}_{3j}^{\top}\right] \\
\frac{\partial \hat{y}_{ij}}{\partial \mathbf{b}_j} &= \frac{1}{\mathbf{P'}_{3j}^{\top}\mathbf{X}_i}\left[f_j\mathbf{P'}_{2j}^{\top} - \hat{y}_{ij}\mathbf{P'}_{3j}^{\top}\right].
\end{aligned} \tag{5.10}$$

We take 100 iterations for convergence in each bundle adjustment. If the average reprojection error within any 2D frame registered in the track, after the optimization, is *larger than one image pixel*, the track expansion process stops. In Fig.5.5 the detailed information is listed for generating a track number 4 from Fig.5.6. Each line records a specific image registered in the track, the initial average reprojection error, the value after optimization and the focal length adjusted after each iteration.

```
initializing with pair <50, 51>
+ < 49 > : 3.21427 -> 0.105992 (1055.65)
+ < 48 > : 1.21574 -> 0.12495 (1019.8)
+ < 47 > : 0.870254 -> 0.139864 (999.026)
+ < 52 > : 0.209652 -> 0.144858 (999.918)
+ < 53 > : 0.185728 -> 0.153301 (1000.94)
+ < 54 > : 0.173068 -> 0.158419 (1001.49)
+ < 55 > : 0.18783 -> 0.159166 (1000.89)
+ < 56 > : 0.173147 -> 0.165993 (1000.84)
+ < 46 > : 0.323949 -> 0.167588 (997.107)
+ < 45 > : 0.190633 -> 0.168425 (997.332)
+ < 44 > : 0.247725 -> 0.167071 (997.552)
+ < 43 > : 0.257581 -> 0.169677 (998.226)
+ < 42 > : 0.190622 -> 0.171228 (998.591)
+ < 41 > : 0.200209 -> 0.174551 (998.662)
+ < 40 > : 0.202806 -> 0.184198 (998.703)
+ < 39 > : 0.218537 -> 0.205383 (998.853)
+ < 38 > : 0.267636 -> 0.209299 (999.159)
+ < 37 > : 0.224448 -> 0.186279 (999.181)
+ < 36 > : 0.20092 -> 0.190483 (999.196)
+ < 35 > : 0.224997 -> 0.197846 (999.274)
+ < 34 > : 0.211131 -> 0.20191 (999.284)
+ < 33 > : 0.217573 -> 0.210541 (999.28)
+ < 32 > : 0.390894 -> 0.229067 (999.746)
+ < 31 > : 0.709651 -> 0.374258 (T)
(Track 4) f = 999.746
25 frames collected
```

Figure 5.5: A track expansion and bundle adjustment.

5.4 Robust Merge of Tracks

Every track contains about 10,000 spatial points. No significant change in the reconstruction can be observed since we discarded all the 3D points which had their reprojection error larger than one pixel.

As shown in Fig.5.6, a total of 10 tracks are obtained from the 70 image frames. Each track is represented by a curve illustrating the range being covered. A geometric verification strategy was introduced in [78] to give more 2D correspondences for image registration, and obtain a larger 3D point cloud. In our example, the tracks will be paired together by collecting the 3D points triangulated from the same 2D SIFT features in the overlapped frames. These 3D-3D point correspondences provide a stronger relation (point to point) to verify geometry, compared to the epipolar geometry which matches a point to a line. When enough frames are fused, the image gaps among tracks are reduced and the merged structure registers as many images as possible. In the example all the 70 images in the 2D sequence were collected by the merge.

5.4.1 Robust Pairing of Tracks

The robust merge of tracks involves the estimation of multiple 4×4 transformations $\mathbf{H}$-s (3D homographies), to fuse the 3D points into the *merged* coordinate system. In spite of the outlier

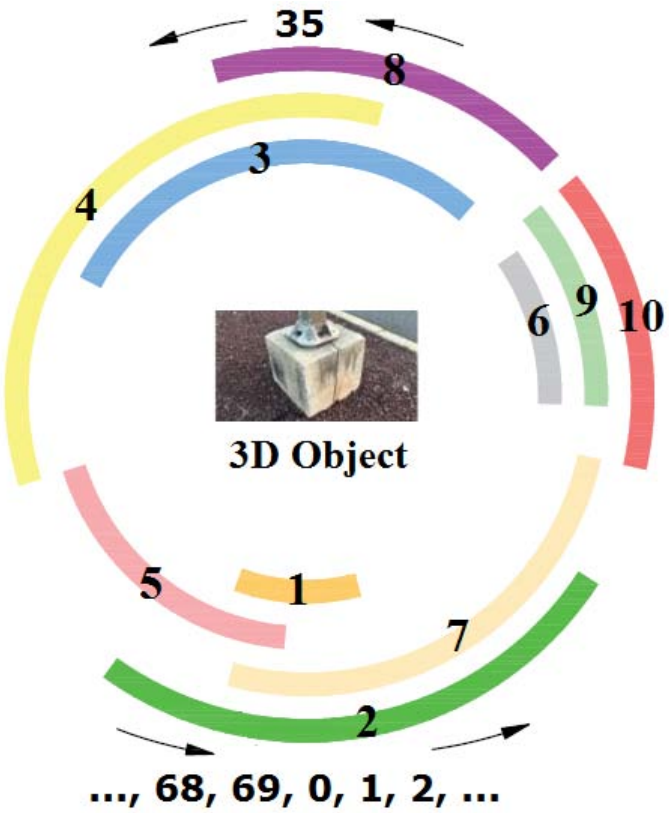


Figure 5.6: Ten tracks obtained from 70 image frames.

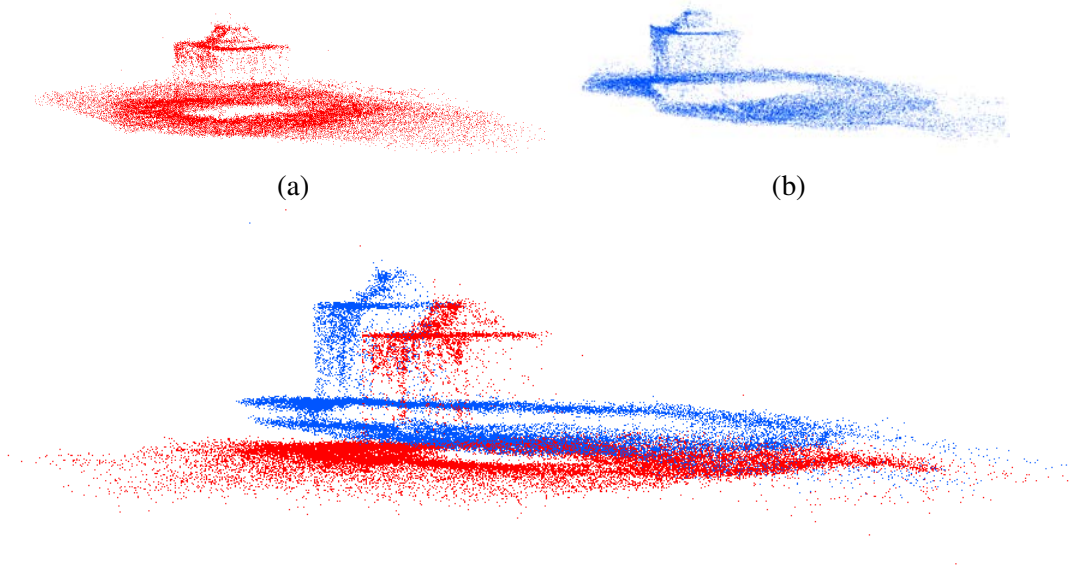
elimination process in each track, incorrect point correspondences could still exist, especially at both ends of the track.

In Fig.5.7c, the Track 5 (blue) is plotted in the 3D coordinate system of Track 4 (red). Besides the differences in the coordinate systems, the two point groups have different scalings and distortions due to the inaccuracy of focal length estimates. In order to obtain a metric reconstruction, the k-th track should be multiplied by the respective 4×4 transformation $\mathbf{H}_k$. This topic was further discussed in [47, Chapter 19].

The estimation of $\mathbf{H}$ is a nonlinear problem, where the objective function is similar to (4.5),

$$\mathbf{y}'_i \simeq \mathbf{H}\mathbf{y}_i, \quad i = 1, \ldots, n_{in} \tag{5.11}$$

while in this case $\mathbf{y} = [X\ Y\ Z\ 1]^\top$ and $\mathbf{y}' = [X'\ Y'\ Z'\ 1]^\top$ are the homogeneous forms of 3D coordinates $\mathbf{X}$ and $\mathbf{X}'$ in these two tracks.



(a) (b)

(c)

Figure 5.7: Point clouds reconstructed in the tracks. (a) Track 4. (b) Track 5. (c) Track 5 (blue) plotted in the 3D coordinate system of Track 4 (red).

From (5.11), for every 3D point correspondence three linear relations can be derived

$$\begin{aligned} \mathbf{h}_1^\top \, \mathbf{y}_i - X_i' \, \mathbf{h}_4^\top \, \mathbf{y}_i = 0 \\ \mathbf{h}_2^\top \, \mathbf{y}_i - Y_i' \, \mathbf{h}_4^\top \, \mathbf{y}_i = 0 \\ \mathbf{h}_3^\top \, \mathbf{y}_i - Z_i' \, \mathbf{h}_4^\top \, \mathbf{y}_i = 0. \end{aligned} \tag{5.12}$$

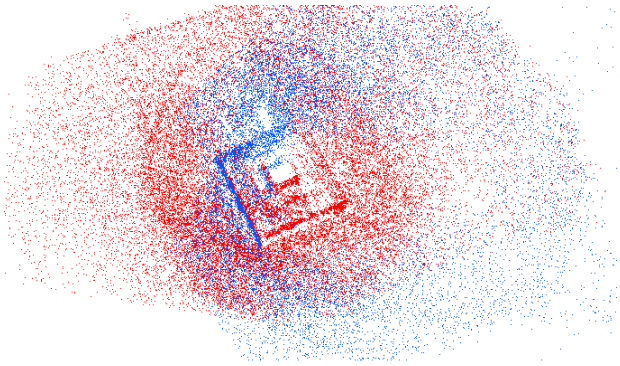
Let $\text{vec}(\mathbf{H}^\top) = \mathbf{h}$, and $\mathbf{h}_i^\top$ stand for the i-th row of $\mathbf{H}$, the entries of $\mathbf{H}$ are solved by the direct linear transformation (DLT)

$$\mathbf{A}_i \mathbf{h} = \begin{bmatrix} -\mathbf{y}_i^\top & \mathbf{0}_3^\top & \mathbf{0}_3^\top & X_i' \mathbf{y}_i^\top \\ \mathbf{0}_3^\top & -\mathbf{y}_i^\top & \mathbf{0}_3^\top & Y_i' \mathbf{y}_i^\top \\ \mathbf{0}_3^\top & \mathbf{0}_3^\top & -\mathbf{y}_i^\top & Z_i' \mathbf{y}_i^\top \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \\ \mathbf{h}_4 \end{bmatrix} \simeq \mathbf{0}_3 \, . \tag{5.13}$$

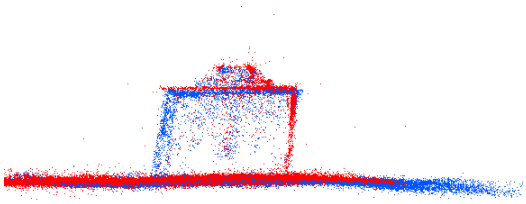
The matrix $\mathbf{A}_i$ is 3×16 and $\zeta = 3$. The elemental subset of $\mathbf{H}$ contains five 3D point correspondences to initialize the 15 unknowns in the 4×4 matrix up to scale. The three carrier vectors can be extracted from $\mathbf{A}_i$, and three 16×3 Jacobian matrices are also obtained. These derivations are done in a similar way as in (5.7) and (5.10), here we will not detail the steps.

```
Pairing tracks ...
  Track <1> : < 2-<5266/12543> 5-<2776/4156> 7-<4148/8314> >
  Track <2> : < 5-<5605/8871> 7-<13359/18353> >
  Track <3> : < 4-<12284/20874> 8-<10012/17513> >
  Track <4> : < 5-<1096/1247> 8-<3984/6720> >
  Track <5> : < 7-<1383/2336> >
  Track <6> : < 9-<4133/8829> 10-<6234/10665> >
  Track <7> : < 10-<663/897> >
  Track <8> : < >
  Track <9> : < 10-<4002/9138> >
```

Figure 5.8: Pairing of tracks (see explanation in text).



(a)



(b)

Figure 5.9: Merge of Track 4 (red) and Track 5 (blue). (a) A top view. (b) A side view.

We take $M = 500$ valid elemental subsets, where each randomly initialized $\mathbf{H}$ has to be validated first by checking its cheirality constraint (Subsection 2.4.1). This is a critical step to guarantee a quasi-affine transformation and preserve the convex hull. The detailed algorithm to verify the cheirality was given in [72] and [47, Chapter 21]. After the robust pairing process of every track pair, we find the inlier structures shown in Fig.5.8. For example, the Track 4 is connected with Track 5 and 8. From Track 5 out of the inital 1247 3D point correspondences, a total of 1096 pairs are retained as inliers, giving 12% of the input discarded as outliers.

5.4.2 Bundle Adjustment in Hierarchical Merging

In Fig.5.9, the merged point cloud combining both Track 4 and Track 5 is shown. All the four side surfaces can be observed in Fig.5.9a, as the two tracks registered more than half of the images according to Fig.5.6. The merged structure, however, can still have large projective distortion as shown in the side view in Fig.5.9b. This error can be reduced if the bundle adjustment on $\mathbf{H}_k$-s is applied.

Let $\mathbf{X}_i$ denote the i-th 3D point in the merged coordinate system, and $\mathbf{X}_{ik}$ stand for the corresponding point found in the k-th track, the bundle adjustment tries to reduce the error

$$\underset{\mathbf{H}_k,\mathbf{X}_i}{\text{minimize}} \sum_{k=1}^{t}\sum_{i=1}^{n} \|\mathbf{H}_k\mathbf{X}_i - \mathbf{X}_{ik}\| \tag{5.14}$$

by optimizing $\mathbf{H}_k$-s in all the t tracks, and the 3D coordinates of points in the merged system $\mathbf{X}_i$. This gives $15t + 3n$ unknowns in the problem.

Similar as in Subsection 5.3.1, the Jacobians of measurements $\mathbf{X}_{ik}$ with respect to the parameters are derived. Let $\mathbf{a}_k$ denote the entries in $\mathbf{H}_k$, and $\mathbf{b}_i = [X_i, Y_i, Z_i]$, with

$$\hat{X}_{ij} = \frac{\mathbf{H}_{1_k}^\top \mathbf{X}_i}{\mathbf{H}_{4_k}^\top \mathbf{X}_i}, \qquad \hat{Y}_{ij} = \frac{\mathbf{H}_{2_k}^\top \mathbf{X}_i}{\mathbf{H}_{4_k}^\top \mathbf{X}_i}, \qquad \hat{Z}_{ij} = \frac{\mathbf{H}_{3_k}^\top \mathbf{X}_i}{\mathbf{H}_{4_k}^\top \mathbf{X}_i}, \tag{5.15}$$

the Jacobian matrix for $\mathbf{a}_k$ is

$$\mathbf{A}_{ik} = \frac{\partial \hat{\mathbf{X}}_{ij}}{\partial \mathbf{a}_k} = \frac{1}{\mathbf{H}_{4_k}^\top \mathbf{X}_i} \begin{bmatrix} \mathbf{X}_i^\top & \mathbf{0} & \mathbf{0} & -\hat{X}_{ij}\mathbf{X}_i^\top \\ \mathbf{0} & \mathbf{X}_i^\top & \mathbf{0} & -\hat{Y}_{ij}\mathbf{X}_i^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{X}_i^\top & -\hat{Z}_{ij}\mathbf{X}_i^\top \end{bmatrix}_{3\times 16} \tag{5.16}$$

and for $\mathbf{b}_i$

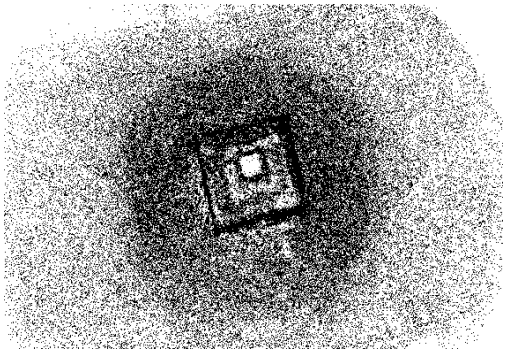
$$\mathbf{B}_{ik} = \frac{\partial \hat{\mathbf{X}}_{ij}}{\partial \mathbf{b}_i} = \frac{1}{\mathbf{H}_{4_k}^\top \mathbf{X}_i} \begin{bmatrix} \mathbf{H}_{1_k}^\top - \hat{X}_{ij}\mathbf{H}_{4_k}^\top \\ \mathbf{H}_{2_k}^\top - \hat{Y}_{ij}\mathbf{H}_{4_k}^\top \\ \mathbf{H}_{3_k}^\top - \hat{Z}_{ij}\mathbf{H}_{4_k}^\top \end{bmatrix}_{3\times 4}. \tag{5.17}$$

The 3D error reduced in each iteration is listed in Fig.5.10. Once all the possible tracks were merged, the bundle adjustment (Subsection 5.3.1) enforcing constraints on cameras is applied again to the merged point cloud. The final result is shown in Fig.5.11.

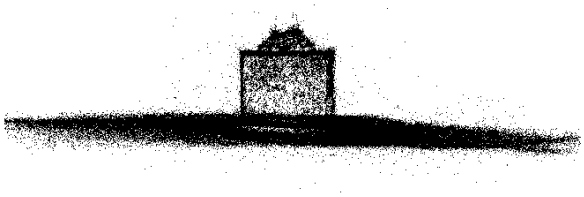
```
Hierarchical merging ...

  merging track <2>
  + < 7 > : 0.14828 -> 0.146314
  + < 5 > : 0.114535 -> 0.113624
  + < 1 > : 0.122335 -> 0.122218
  + < 10 > : 0.119456 -> 0.119433
  + < 4 > : 0.115433 -> 0.115433
  + < 6 > : 0.0968498 -> 0.0968498
  + < 9 > : 0.0927546 -> 0.0927546
  + < 3 > : 0.0801341 -> 0.0800993
  + < 8 > : 0.0887474 -> 0.0886219
```

Figure 5.10: Track merging with bundle adjustment.



(a)



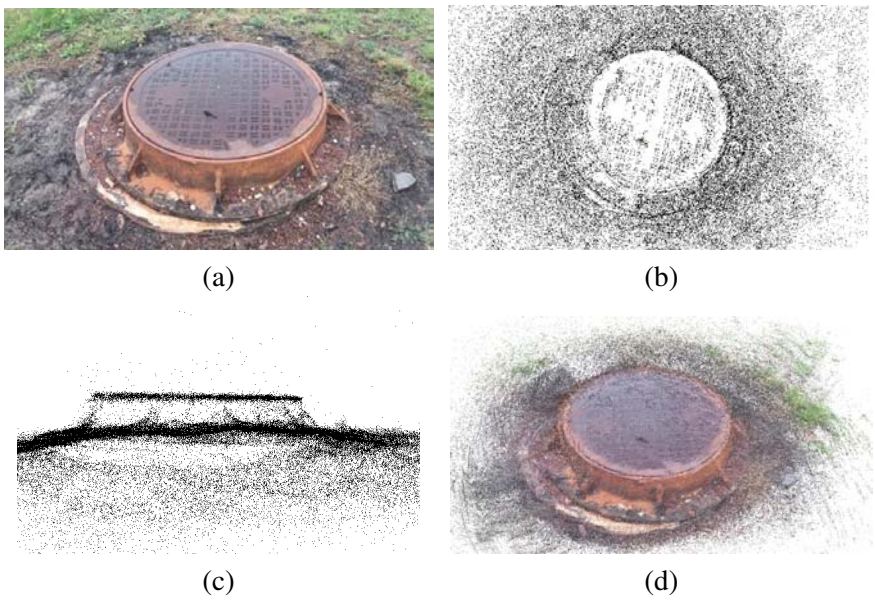
(b)

Figure 5.11: Point cloud obtained after hierarchical merging. (a) A top view. (b) A side view.

5.5 3D Points Triangulated from Stereo Views

From the SIFT features, the sparse 3D poins and the camera parameters can be estimated through the SfM algorithm. Since most of these 2D features were extracted only from high-contrast regions in the images, like the boundary between textures, the sparse points may not be dense enough to capture the detailed shape information. For example, the plane representing the sewer cover reconstructed in Fig.5.12b and Fig.5.12c, cannot be detected without removing the very dense outliers scattered around.

In order to triangulate more 3D points, more 2D matches among images should be found



(a) (b)

(c) (d)

Figure 5.12: More 3D Points obtained from stereo views. (a) One frame in the 2D image sequence. (b) & (c) 3D point cloud reconstructed from SIFT features. (d) Colored 3D Points reconstructed from stereo views.

first. Several image rectification methods were proposed which generate a stereo pair from two images. In [61] each 2D pairing becomes a 1D search along the horizontal direction. The accurate camera parameters are obtained after bundle adjustment, and then required by the stereo matching to give valid 3D points. OpenCV function based on [46] is used in our method. After the stereo matching, these additional 3D points fill the space among the SIFT points, as shown in Fig.5.12d with colors.

Chapter 6

Robust Method in the Recovery of 3D Geometric Primitives

In this section the robust algorithm will be applied to 3D point clouds. The majority of the examples were also presented in [100]. For each type of surface we first derive the carrier vector and the Jacobian matrix. The inlier structures for synthetic cases are generated by CloudCompare software [4] and corrupted by Gaussian noise.

The real 3D datasets are constructed either from the 3D mesh models in Autodesk ReMake [2] or the photogrammetric methods using 2D images (Chapter 5). The outliers come from the incorrect point correspondences in 2D and/or 3D matches, and introduce much higher noise than the inliers. Like in Chapter 4, the processing time to generate the geometric primitives from 3D point cloud is measured on an i7-2617M 1.5GHz computer.

6.1 Estimation of 3D Geometric Primitives

In this section we present the recovery of different categories of 3D primitives. The limitations will be explained in Section 7.2.

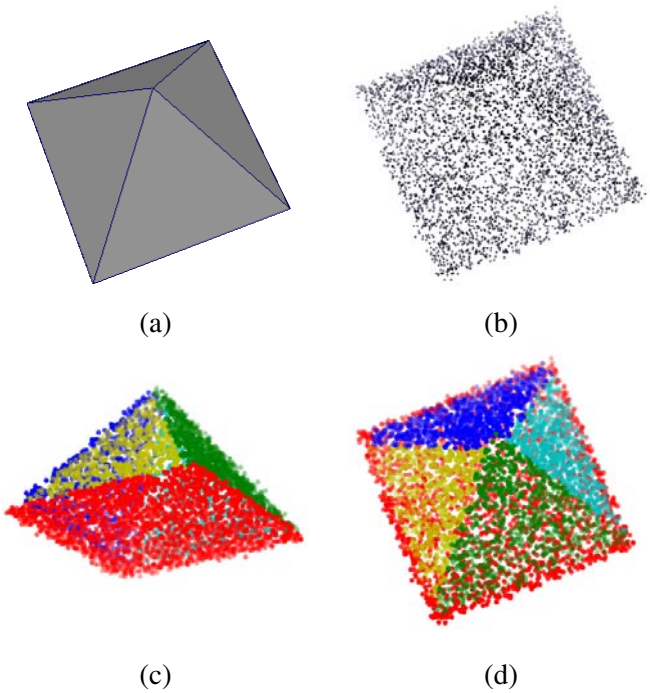
6.1.1 3D Plane

Multiple planes are estimated in the first example. The planes are estimated in 3D space with the input data $\mathbf{Y} = [X\ Y\ Z]^\top$. The objective function for each 3D plane satisfied by its points $n_{in} \ll n$ is

$$\theta_1 X_i + \theta_2 Y_i + \theta_3 Z_i - \alpha \simeq 0 \qquad i = 1, \ldots, n_{in}. \tag{6.1}$$

This is a linear case and the carrier vector $\mathbf{X}$ is the same as $\mathbf{Y}$.

In Fig.6.1 a synthetic case is shown. A pyramid shaped solid model (Fig.6.1a) is given and the point cloud is extracted by evenly sampling points on its surfaces. With side length of the



(a) (b)

(c) (d)

Figure 6.1: Synthetic plane estimation. (a) A pyramid model with side length $a = 1$. (b) Point cloud extracted with 5000 points, $\sigma_g = 0.01$. (c) & (d) Five planes estimated (viewed from different angles).

pyramid $a = 1$, this dataset consists of 5000 points uniformly distributed, and are corrupted by three-dimensional Gaussian noise with standard deviation $\sigma_g = 0.01$ (Fig.6.1b). After $M = 1000$ trials, the input point cloud is segmented into five plane structures, as shown in different colors in Fig.6.1c and Fig.6.1d. The entire processing time for the estimation is 1.70 seconds.

From the total of 5000 points, the scale estimates and the amount of points in each structure are shown below

	red	$green$	$blue$	$cyan$	$yellow$
$scale:$	0.038	0.032	0.037	0.034	0.033
$inliers:$	2205	827	817	581	570

where the scale estimates are slightly different in these planes, and no outliers are classified. The average value of the scale estimate is 0.348 which is around $3\sigma_g$. The point clouds obtained from 3D scanners usually have their measurement error within accuracy of 0.01-0.1 mm, thus

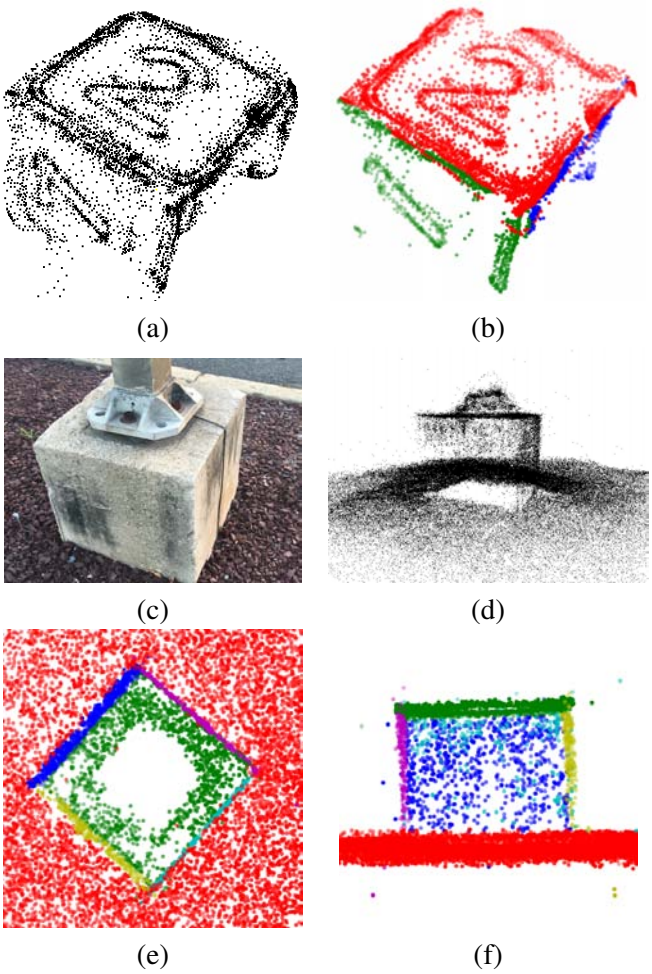


Figure 6.2: Plane detection in point cloud. (a) From Fig.1.8, a total of 5463 points selected. (b) Three planes recovered. (c) An image from the sequence in Fig.5.11. (d) A total of 23077 points selected. (e) & (f) Six planes recovered (viewed from different angles).

the Gaussian noise $\sigma_g = 0.01$ is relatively larger compared to the scale of the model (side length $a = 1$). In Subsection 7.2.2 we will discuss how the estimation becomes unstable when σ_g further increases.

The model reconstructed from 8 images in Autodesk ReMake (Fig.1.8) is used in Fig.6.2a, where the 5463 vertices of the mesh are kept for the plane estimation. After 2.48 seconds with $M = 1000$, five structures are returned. The first three planes with the strongest strengths are kept, as shown in Fig.6.2b. The surface normals can be obtained directly from the segmented planes, instead of the mesh in Fig.1.8.

The plane estimation is also applied to the point cloud in Fig.6.2d generated in Fig.5.11 with the photogrammetric method. Outliers also exist in the retained 23077 points. Since only

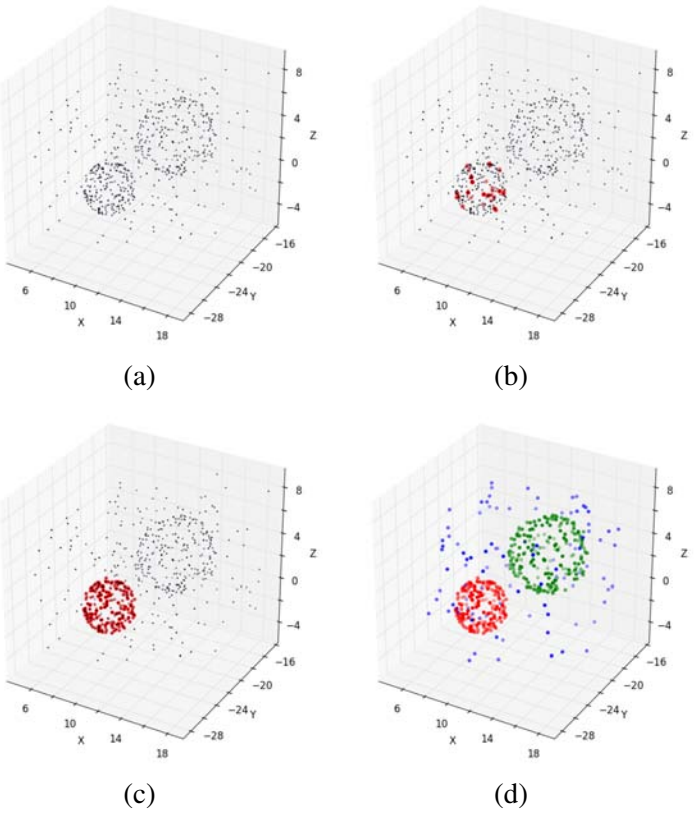


Figure 6.3: Synthetic sphere estimation. (a) Input data. (b) The initial set. (c) First structure obtained after mean shift. (d) Three structures sorted by strengths ($s_R > s_G > s_B$).

the main structure is portrayed, the majority of the outliers are located on top of the cubic base. After 7.04 seconds, with $M = 1000$, the estimator locates six planes, as shown in Fig.6.2e and Fig.6.2f with 21758 inlier points in total. Sharp edges of the model are well preserved after the estimation.

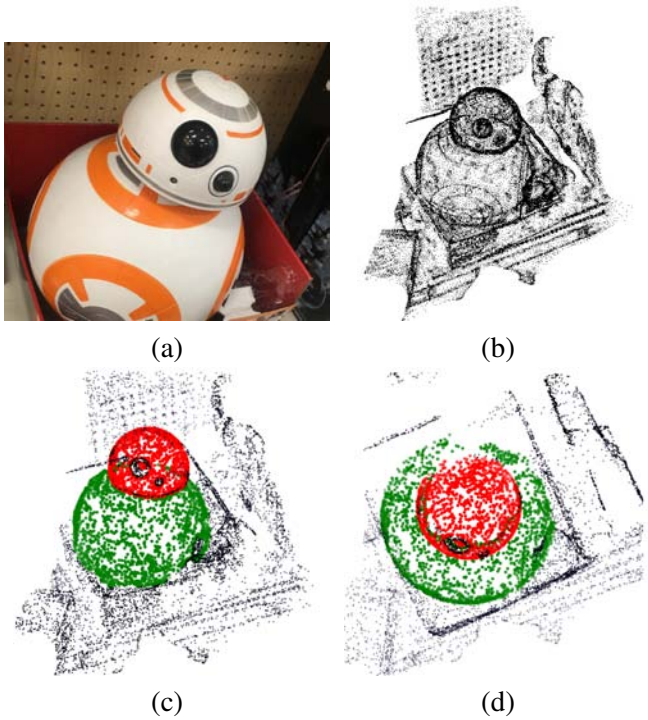
6.1.2 3D Sphere

The spherical surface fitting is a nonlinear estimation problem, where the objective function of a sphere is

$$(X-a)^2 + (Y-b)^2 + (Z-c)^2 - r^2 \simeq 0 \tag{6.2}$$

with $\mathbf{Y} = [X\ Y\ Z] \in \mathbb{R}^3$ and the carrier vector $\mathbf{X} \in \mathbb{R}^4$ given by

$$\mathbf{X} = \left[X^2 + Y^2 + Z^2 \;\; X \;\; Y \;\; Z\right]^\top. \tag{6.3}$$



(a) (b)

(c) (d)

Figure 6.4: Sphere detection in point cloud. (a) An image from the sequence. (b) A total of 10854 points selected. (c) & (d) Two spheres recovered (viewed from different angles).

A sphere can be found from four points ($\zeta = 1$), with any three non-collinear and all four non-coplanar.

The 4×3 Jacobian matrix of a spherical surface is derived as

$$\mathbf{J}_{\mathbf{X}_i|\mathbf{Y}_i} = \begin{bmatrix} 2X_i & 2Y_i & 2Z_i \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

and relies on the specific 3D coordinates of the input point $\mathbf{Y}_i$.

A synthetic example is shown in Fig.6.3. Two spheres are placed in the 3D space and each of them consists of $n_{in} = 200$ points, along with $n_{out} = 200$ outliers (Fig.6.3a). The smaller sphere has a radius of $r_1 = 2$, and the other one $r_2 = 3$. These inlier structures are corrupted by Gaussian noise with standard deviation $\sigma_g = 0.05$ and 0.1, respectively. The initial set ($n_\epsilon = 30$), obtained after $M = 1000$ trials, is shown in Fig.6.3b. With the scale estimate $\hat{\sigma} = 0.079$, the first estimated structure is obtained as in Fig.6.3c. After all 3D points were

processed, the strongest three structures are shown in Fig.6.3d, where the third one (blue) is an outlier structure. where the third one (blue) is an outlier structure. The estimation stably segmented the two inlier structures in all of the 100 trails.

A sample image of a toy with spherical surfaces is shown in Fig.6.4a. In Fig.6.4b the 3D point cloud is generated from 36 images with [2], containing 10854 points. A large number of points in the planes should be rejected as outliers since only spheres will be detected. Two different types of surfaces cannot be estimated simultaneously and this problem will be further discussed in Subsection 7.2.2. With $M = 1000$, we locate two inlier structures, as shown in red and green colors in Fig.6.4c and Fig.6.4d. A total of 3504 points are inliers and the estimation took 7.24 seconds. More than 7000 points were rejected as outliers.

The processing time depends on the total amount of inlier structures. When more structures exist, more iterations are needed to segment all points into different surfaces. If a larger input data size is given, more Mahalanobis distances are also calculated since the computation involves all the data points. This issue will be discussed in Chapter 7.

6.1.3 3D Cylinder

A 3D *cylinder* aligned with the Z-axis is defined by the equation

$$(X-a)^2 + (Y-b)^2 - r^2 = 0 \tag{6.5}$$

$a,\ b$ stand for the 2D coordinates where Z-axis passes through the XY-plane, and r is the radius. With the input variable $\mathbf{Y} = [X\ Y\ Z]^\top$, this relation can be reformulated by a quadric matrix

$$[\mathbf{Y}_i\ 1]\,\mathbf{Q}'\,[\mathbf{Y}_i\ 1]^\top \simeq 0 \qquad i = 1, \ldots, n_{in} \tag{6.6}$$

where $\mathbf{Q}'$ is a 4×4 symmetric matrix

$$\mathbf{Q}' = \lambda \begin{bmatrix} \mathbf{D}' & \mathbf{d}' \\ \mathbf{d}'^T & a^2+b^2-r^2 \end{bmatrix} \quad \mathbf{D}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{d}' = \begin{bmatrix} -a \\ -b \\ 0 \end{bmatrix}. \tag{6.7}$$

When an euclidean transformation is applied $\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$, a general cylinder under rotation and translation is found with

$$\mathbf{Q} = \mathbf{M}^{-T}\mathbf{Q}'\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{D} & \mathbf{d} \\ \mathbf{d}^T & d \end{bmatrix}. \tag{6.8}$$

A sphere, ellipsoid, paraboloid and hyperboloid of two sheets can all be represented by this quadric matrix, and are projectively equivalent. The 4×4 quadric matrix $\mathbf{Q}$ has nine unknown parameters up to a scale. To define a general cylinder, only five degrees of freedom are required, four for its axis of rotation and one for radius.

Several solutions of the cylinders were described in [25], computed at elemental subsets with various numbers of points from 5 to 9. The nine-point-solution is used in this experiment, where the carrier vector is derived as $\mathbf{X} = [X^2 \;\; XY \;\; XZ \;\; Y^2 \;\; YZ \;\; Z^2 \;\; X \;\; Y \;\; Z]$. A general quadric solution from each randomly initialized elemental subset is found from $\mathbf{A}\,\text{vech}\mathbf{Q} = \mathbf{0}$, where vech is the vectorization of the lower part of a symmetric matrix $\mathbf{Q}$, and $\mathbf{A}$ a 9×10 matrix formed by stacking carrier vectors.

An elemental subset consisting of nine points gives an over-determined solution, and the parameters in $\boldsymbol{\theta}$ should be constrained for a cylinder. From equations (6.7) and (6.8), it is easy to prove that two of the three singular values of matrix $\mathbf{D}$ are identical and the third one is zero, and $\mathbf{d}$ is an eigenvector of $\mathbf{D}$. These constraints should be verified for each elemental subset.

The transpose of the 9×3 Jacobian matrix is

$$\mathbf{J}^{\top}_{\mathbf{x}_i|\mathbf{y}_i} = \begin{bmatrix} 2X_i & Y_i & Z_i & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & X_i & 0 & 2Y_i & Z_i & 0 & 0 & 1 & 0 \\ 0 & 0 & X_i & 0 & Y_i & 2Z_i & 0 & 0 & 1 \end{bmatrix}. \tag{6.9}$$

In Fig.6.5a two cylinders are placed with 500 outliers. Each cylinder has its rotation axis in a randomly generated direction. The inlier structures have radius $r = 2, 3$, the number of inliers $n_{in} = 400, 300$ and $\sigma_g = 0.06, 0.1$. The i.i.d. inlier noise is applied in 3D dimension, which is roughly $\pm 10\%$ of the size of radius.

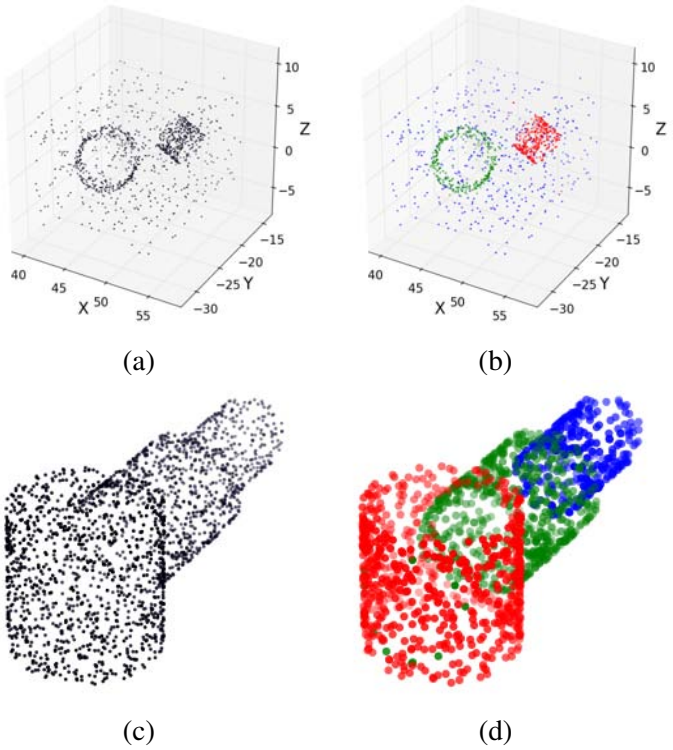


Figure 6.5: Synthetic cylinder estimations. (a) Two synthetic cylinders and 500 outliers. (b) Two inlier and one outlier structures are recovered. (c) Point cloud with 2000 points. (d) Three cylinders estimated.

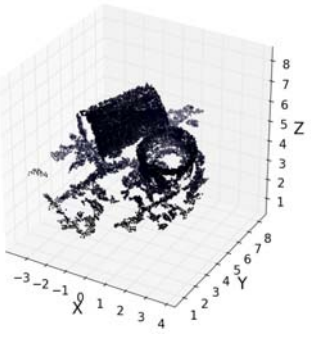
With $M = 5000$ the result becomes stable and three strongest structures are returned

	red	$green$	$blue$
$scale:$	0.28	0.48	5.56
$inliers:$	413	337	449
$strength:$	1487.1	705.0	80.8.

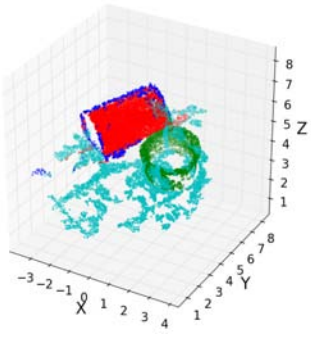
in 25.02 seconds. The first two structures are inliers and the first outlier structure has a much weaker strength, as shown in Fig.6.5b. After testing the randomly generated data for 100 trials, in 96 trials the stronger cylinder is correctly segmented, while the weaker one in 94 trials.

In Fig.6.5c a synthetic point cloud containing three cylinders is extracted by sampling points on a solid model. This dataset consists of 2000 points and the radius of each cylinder is $r = 1, 2$ and 3, respectively. Gaussian noise with standard deviation $\sigma_g = 0.01$ is applied independently on all three dimensions, and no outliers are introduced at the input. With $M = 2000$, three

(a)



(b)



(c)

Figure 6.6: 3D cylinder estimations with VisualSFM [96, 42]. (a) Sample real images used for 3D reconstruction. (b) The 3D cloud of input points. (c) Three inlier and one outlier structures are recovered.

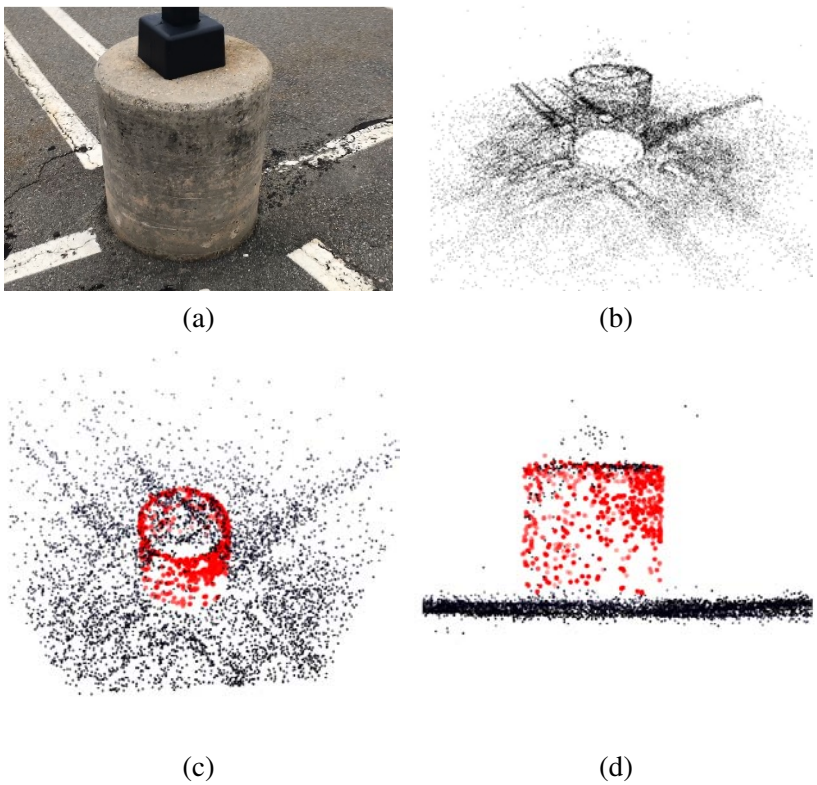
correct cylinders are retained after 15.83 seconds (Fig.6.5d).

The amount of points in each structure along with the scale estimates are shown in the table below

	red	$green$	$blue$
$scale:$	0.075	0.079	0.058
$inliers:$	670	568	318.

The remaining 444 points from the 2000 are not estimated correctly, and are returned in the end as an outlier structure with a large scale. The 3D cylinder estimation is based on a non-linear objective function and the scale estimate $\hat{\sigma}$ is not equal to the distance measured in the Euclidean space.

The *VisualSFM* software [96, 42] is used to generate the point cloud of a 3D scene captured by a 2D image sequence containing 25 pictures. Five sample images are shown in Fig.6.6a. The point cloud consists of 12103 points (Fig.6.6b). Compared with the synthetic data, the inliers in the point cloud are more dense and have much smaller noise. A smaller sampling size $M = 2000$ gives a stable result. The processing takes 44.75 seconds and as Fig.6.6c shows, the three inlier structures (red, green and blue) and the first outlier structure (cyan) are recovered.



(a) (b)

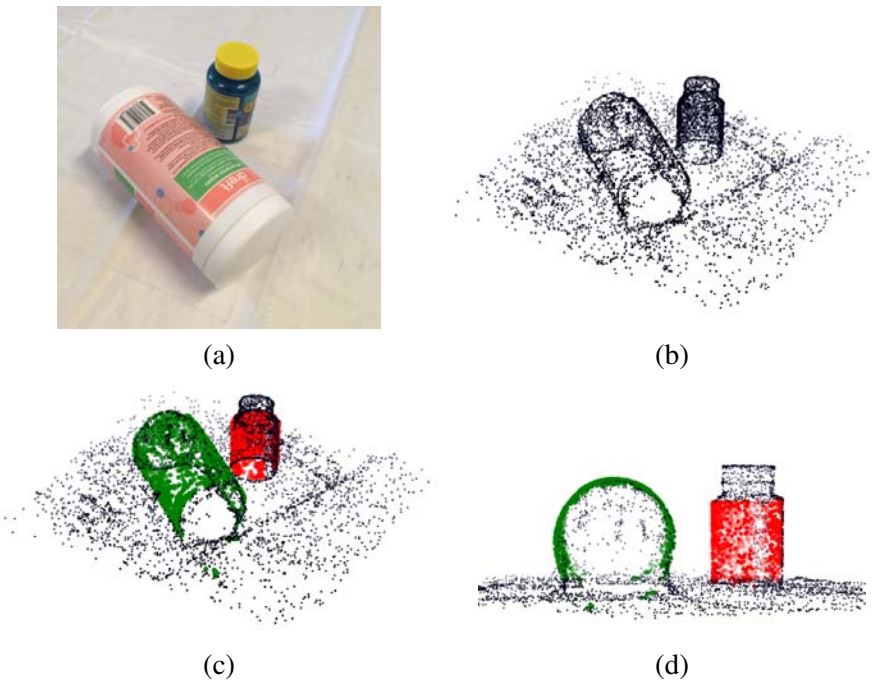
(c) (d)

Figure 6.7: Detection of a cylindrical pole in 3D point cloud. (a) A sample image portraying one cylinder. (b) A total of 7241 points selected. (c) & (d) One cylinder recovered (viewed from different angles).

A cylindrical pole is shown in Fig.6.7a. From 54 images we generate the point cloud (Fig.6.7b) through the structure from motion (SfM) algorithm. The dataset contains 7241 points with most on the ground being outliers for cylinder detection. With $M = 2000$, after 18.55 seconds the single cylinder is located in the noisy dataset containing 568 points, as the red points shown in Fig.6.7c and Fig.6.7d. More than 6500 points are discarded as outliers.

In Fig.6.8a another sample image is shown. A total of 6500 vertices are obtained in Fig.6.8b from the mesh rebuit in [2], by using 22 images. The two bottles with cylindrical shapes are detected in 12.65 seconds. The two inlier structures are shown in Fig.6.8c and Fig.6.8d containing 2262 points.

The two examples show the advantage of using robust surface fitting. In the modification of the point cloud, the major problem comes from the lack of surface continuity information. Designers prefer to apply changes directly to a surface, while in the 2D user interface of CAD software, the scattered points on the surface may not be easily selected. With the presented



(a) (b)

(c) (d)

Figure 6.8: Detection of cylindrical objects in 3D point cloud. (a) A sample image containing two cylinders. (b) A total of 6500 points selected. (c) & (d) Two cylinders recovered (viewed from different angles).

robust algorithm, the points only on a 3D surface are picked as a group, without collecting unwanted points from the background.

Other categories of quadrics can be estimated in a similar way like the cylinders. For example, a paraboloid is detected by removing the constraint that $\mathbf{d}$ is an eigenvector of matrix $\mathbf{D}$. More details can be found in [47, Chapter 3].

6.2 Recovery of Parametric Features

The goal of feature recognition is to regenerate the mechanical design features, e.g., an extruded or a revolved feature from geometric primitives. The Automatic Feature Recognition (AFR) to handle this problem is considered as the ideal solution for the design automation, which could potentially achieve the simultaneous design and development of a product in the concurrent engineering [81]. The details of existing AFR algorithms were reviewed in [44] and can be categorized into two types.

- Graph-based feature recognition [53].

(a) (b)

Figure 6.9: Recovery of parametric information in cylinders. (a) The augmented cylindrical surface recovered from Fig.6.7. (b) Two cylindrical surfaces recovered from Fig.6.8.

- Hint-based feature recognition [90].

Methods involving the use of artificial neural networks were introduced in [19].

The major concern on all these approaches is about efficiency. The total amount of surfaces in a solid model can be very large, which gives too many possible combinations of features to be extracted. In [90], the "*hint*" was used to reduce the computation, e.g., two parallel planes could be a hint for an extrusion. However, the process to identify a hint cannot always be reliable when the tolerance and noise is considered.

A completely automatic feature recognition system is not possible at this moment. Some guidance from the designer will always be required to extract the accurate topological and geometric relations. In Fig.6.9 several augmented cylindrical surfaces in the 3D point clouds are shown. Each cylinder is defined by its parameters where the height is also recovered. Compared with the 3D point cloud and the mesh model, the modification of a structure becomes much easier and the texture mapping can also be applied. However, when the 3D point cloud is not dense enough, the height of a cylinder may not be accurately evaluated, and the adjustment should be done manually by the designer. The post-processing such as the stitching and surface interpolation is also required, to refine a solid model for reuse in the inverse design.

Chapter 7

Conclusion and Future Directions

7.1 Conclusion

A new robust algorithm was presented which does not require the inlier scales to be specified by the user prior to the estimation. It estimates the scale for each structure adaptively from the input data, and can handle inlier structures with different noise levels. In a strength based classification, the inlier structures with larger strengths are returned first, while a large quantity of outliers are removed.

In the structure from motion resulting in a 3D point cloud as the output, the new algorithm estimates first the scale between every image pair. The robust matching of 2D image features improves the image registration process. As more images are processed, more detailed 3D information is captured for the point cloud. In the hierarchical merging, the 3D point correspondences are more reliably found among the different patches with relatively large projective distortions. No 3D reference points are used when these segments of point clouds are stitched together.

The geometric primitives are segmented directly from the 3D point cloud without obtaining the surface normals or building a 3D mesh. The efficiency and robustness of the method was also demonstrated with datasets corrupted by a large amount of outliers. In many current 3D surface fitting approaches, the user has to manually select first most of the inlier points from a surface, and estimated only one surface each time.

In the new robust algorithm multiple surfaces are detected in one process, where the designer has a much more convenient way to select the 3D points as a group from one inlier surface, without an error-prone process to avoid the outliers. In the future, new CAD design tools can be created to modify the 3D point clouds, and make the inverse design process even

more effective.

The Python/C++ program for the robust estimation of multiple inlier structures is posted on our website at

`rci.rutgers.edu/riul/research/code/MULINL/index.html`.

7.2 Open Problems

In this section we will summarize several open problems which have not yet been solved completely, and may appear when the new robust algorithm is used. Some of these issues only affect the quality of the robustness in the method and can be improved by pre-processing the input and/or post-processing the output. See Section 3.4 and 3.5. For the other problems we discuss below the possible solutions, but further research will still be needed.

7.2.1 Scale Estimation in DataSet without i.i.d. Noise

In Chapter 6 we assumed that the noise in the reconstructed 3D points are independent and identically distributed. However, when the input data do have different covariance in each dimension, the inlier covariances have the form

$$\mathbf{C_Y} = \begin{bmatrix} \sigma_X^2 & 0 & 0 \\ o & \sigma_Y^2 & 0 \\ 0 & 0 & \sigma_Z^2 \end{bmatrix} \tag{7.1}$$

with three scales unknown. The covariance propagation of the carrier vector (3.6) places the σ-s into the product of the Jacobian matrices, and cannot be estimated directly by our method.

A possible solution is to start with a uniform σ, that is, $\sigma^2\mathbf{I}_{3\times 3}$. Normally, when the errors in various dimensions do not differ by more than one order of magnitude, the algorithm still work. It finds a result close to the true inlier structure, but not with the correct amount of points. Focused on a local region in the input space where the structure was estimated, apply the scale estimation in this region but *for each dimension* separately. More accurate $\hat{\sigma}$-s will be found and the inlier estimate is updated for each dimension.

(a) (b)

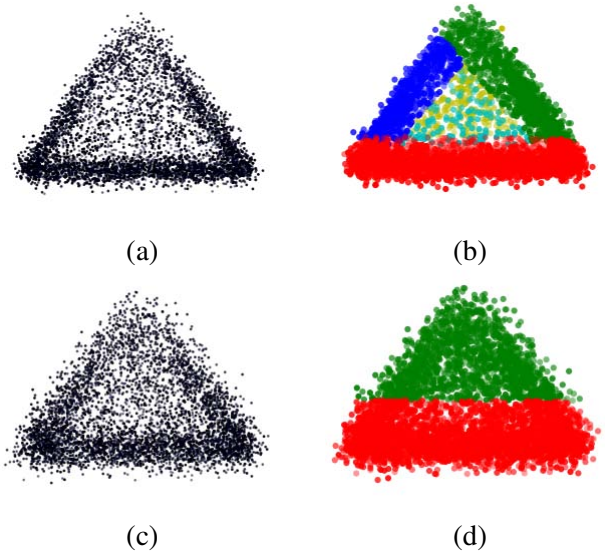
Figure 7.1: Heteroscedastic noise in the 3D triangulation. (a) Error is low in the front view. (b) Stronger error is observed along the depth direction.

For example, the measured points obtained from 3D triangulation have different errors as the depth varies. The more distant a point is away from the camera, the larger error in Z (depth) is introduced. As shown in Fig.7.1a, the noise viewed in the front is relatively small, while from a side view in Fig.7.1b, the error along the depth direction is much larger. A correct estimation in a 3D point cloud should consider that the error along the Z direction is not identical to those in X and Y.

When the heteroscedasticity of the input points is taken into account, the computation of the input covariance $\mathbf{C_Y}$ requires the specific location of each 3D point. This problem was addressed in [15] by using the method proposed in [66]. For our examples in Chapter 5, the heteroscedastic error in depth was reduced by using a circular 2D image sequence. Since the 3D scene were viewed from several wide angles and different depths, the distortion along Z direction lessened and was considered the same in all three dimensions. By using the technique in [66], and starting with an equal 2D inlier noise variance in two dimensions of the image, the different 3D variances can also be computed.

7.2.2 Problems in the Recovery of 3D Geometric Primitives

When the input point cloud is heavily corrupted with both inlier noise and large amount of outliers, the scale estimate is no longer accurate due to the interaction between the inliers and outliers. The structure detected first could inaccurately take away some points from other inlier structures. As a result, the following structures become less detectable and give unreliable segmentations.



(a) (b)

(c) (d)

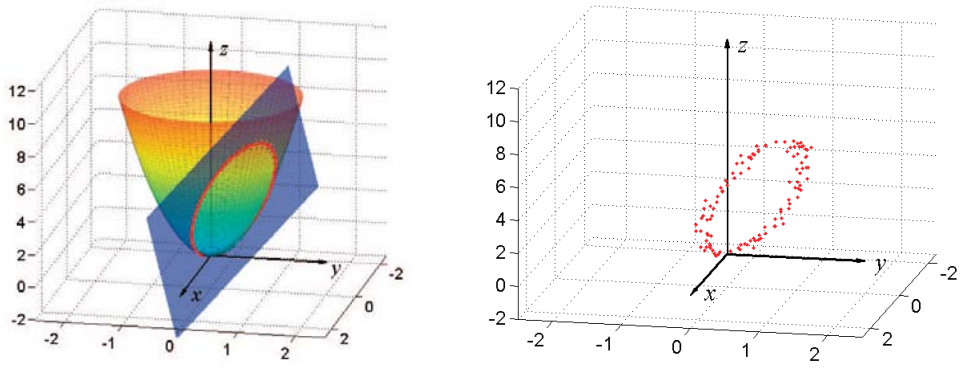
Figure 7.2: Inaccurate segmentations from increased inlier noise with the synthetic point cloud in Fig.6.1. (a) $\sigma_g = 0.03$. (b) Six planes are detected. (c) $\sigma_g = 0.05$. (d) Incorrect segmentation of the planes.

In Fig.7.2a and Fig.7.2c, the same point cloud as in Fig.6.1 are used, but applied with stronger Gaussian noises, $\sigma_g = 0.03$ and 0.05. Two side views are used here to show the noise levels. With the same trial size of $M = 1000$, the six planes are detected from Fig.7.2b, while in Fig.7.2d the estimator segments the data into only two planes. Once the noise level reaches the limitation, a larger size of M will still not improve the quality of the result.

Every geometric primitive estimated in the previous chapters were governed by a single objective function. But in general a 3D shape can also be defined by multiple constraints, like a ring shaped object in Fig.7.3, where a 3D ellipse is to be estimated by cutting a paraboloid with a plane. The inlier points thus should satisfy both the equations of a paraboloid and a plane, the estimate is an $m \times k$ matrix $\boldsymbol{\Theta}$ and a k-dimensional vector $\boldsymbol{\alpha}$, in this case $k = 2$.

The covariance of $\mathbf{Z}_i$ is $\sigma^2\mathbf{H}_i = \sigma^2\boldsymbol{\Theta}^\top\mathbf{C}_i\boldsymbol{\Theta}$, with σ^2 unknown. This gives a $k \times k$ symmetric Mahalanobis distance matrix for $i = 1, \ldots, n$, and since ζ is one dimensional

$$\mathbf{D}_i = \sqrt{\left(\mathbf{X}_i^\top\boldsymbol{\Theta} - \boldsymbol{\alpha}\right)^\top \mathbf{H}_i^{-1} \left(\mathbf{X}_i^\top\boldsymbol{\Theta} - \boldsymbol{\alpha}\right)} \tag{7.2}$$



(a) (b)

Figure 7.3: Estimation of a 3D ellipse. (a) An ellipse obtained by cutting a paraboloid with a plane. (b) Inlier points of the 3D ellipse.

which could be expressed as the union of k vectors

$$\mathbf{D}_i = [\mathbf{d}_{i:1} \ \ldots \ \mathbf{d}_{i:k}]. \tag{7.3}$$

A possible solution is to order the Mahalanobis distances $\mathbf{d}_{[i:*]}$ for each column separately, and collect the inputs corresponding to the minimum sum of distances for $\epsilon\%$ of the data. The $k \times k$ matrices are reduced to k initial sets, one for each dimension.

Apply independently k times the expansion process described in Subsection 3.3.1 and define the $k \times k$ diagonal scale matrix

$$\mathbf{S}_k = \begin{bmatrix} \hat{\sigma}_1 & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \hat{\sigma}_k \end{bmatrix}. \tag{7.4}$$

The $k \times k$ covariance matrix $\mathbf{B}_i$ is computed as

$$\mathbf{B}_i = \mathbf{S}_k^\top \, \mathbf{\Theta}^\top \mathbf{C}_i \mathbf{\Theta} \, \mathbf{S}_k. \tag{7.5}$$

The second step in the algorithm, the mean shift, is now also multidimensional and further experiments will be needed to verify the feasibility of this solution. Once the paraboloid and the plane were estimated robustly, their intersection gives the 3D ellipse.

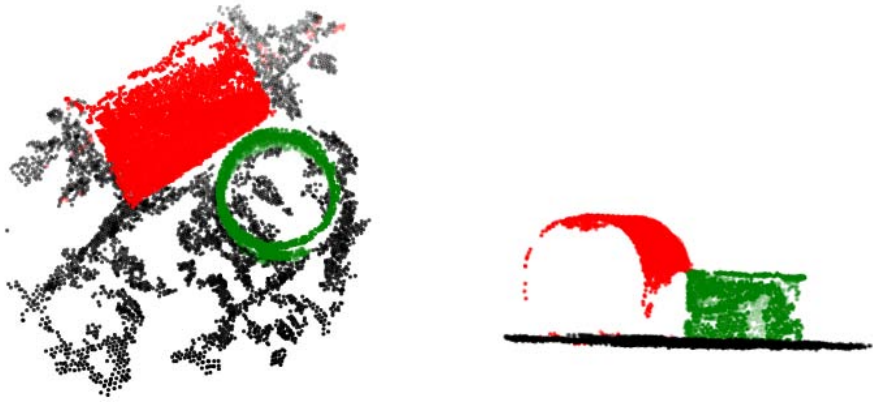


Figure 7.4: An open problem to segment both 3D planes and cylinders.

Another limitation appears when a point cloud contains different types of surfaces. There exists no easy way to estimate both of them in a same process. For example, in Fig.7.4 both planes and cylinders are portrayed, the estimation starting with the planes may also fit some points from the cylinders into the planes. These points attracted to the planes should be rejected and returned into the dataset for latter cylinder estimation. Moreover, the surface details may not be captured by the dense samples, when only sparse points are available. Without enough inlier points, incorrect surfaces could be generated. In photogrammetric approaches starting from 2D sequences, these problems appear more frequently and manual adjustments on the output point clouds are required. With a user-guided system/interface, the correctness of the segmentations is easily verified by the designer.

Finally, a large point cloud may contain too many surfaces to be segmented, as shown in Fig.7.5 [1]. Due to the low inlier ratio of each single structure, the required size of M increases rapidly and eventually becomes impractical for a fast implementation.

A possible solution of this problem is as follows. First, instead of applying the surface fitting algorithm on the entire input, the dataset is separated into smaller clouds by the designer, and the 3D points can also be downsampled if necessary. Then the estimation can be carried out independently for each segment. The respective robust outputs are merged together in post-processing to fuse surfaces in two adjacent segments, and eliminate possible contradictions.

[1]Fig.7.5 is retrieved from SPARC DESIGN, http://www.sparc-design.com/what-are-point-clouds/.

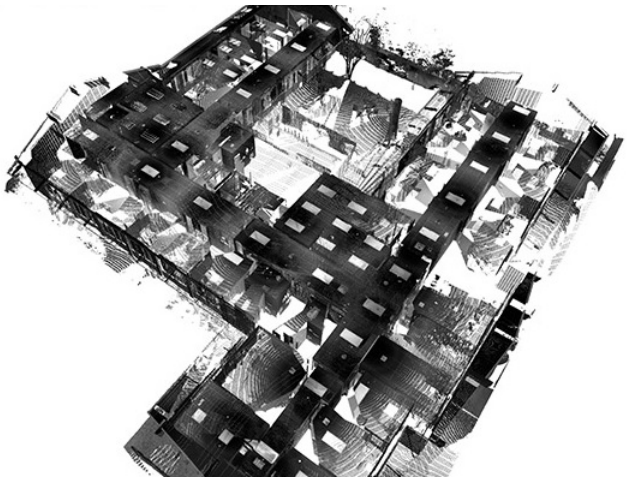


Figure 7.5: An open problem to extract surfaces from a large point cloud.

7.3 Future Work

The future work contains all the issues described as open problems in the previous section. Besides that, it also covers the further improvements in the computer-aided design (CAD) systems. A challenging problem in the inverse CAD is to reconstruct an editable 3D solid model, and a popular topic in the computer graphics community since 1990s. As discussed in Section 6.2, even with state-of-the-art dense 3D point clouds and robust multi-view stereo algorithms, this is not an easy task.

New design tools are still expected in today's mechanical CAD software to accelerate the design process, and make it more convenient and less error-prone. By using our robust algorithm of multiple inlier structures, the 3D points on one surface can be easily grouped, as illustrated in Fig.7.6. Similar to the pixel editing in Photoshop, modification of the cloud dataset can be directly done on the point level, without generating a compact solid model.

Two problems have to be solved first to realize the above concept:

- Real time processing for the surface estimation in 3D point cloud.
- A user interface to support the entire design pipeline.

To reach real time processing, multi-threading and parallel computation are necessary. Most of the processing time in our robust algorithm currently is spent on the initialization of elemental subsets, calculation of the Mahalanobis distances and the sorting. All of these procedures

Figure 7.6: Surface selection tool based on robust estimation algorithm.

can be adapted into the parallel process on multi-core CPU and Graphics Processing Unit. When an interactive user interface is developed, which could also gain access to all the levels of the structure-from-motion pipeline, direct modification of the 3D point cloud can be easily achieved in the CAD systems.

Finally, a compatible user interface to edit point cloud, is probably the greatest challenge beyond the algorithms. The virtual reality provides a solid platform to visualize the 3D data, while it is still an open question how to best display the points, and provide feedback to the designers. Different from the current feature-based modeling, the new design tools should adapt into the work on point level.

"*Just as propeller powered aircraft could not compete with jets, neither will today's CAD software be able to compete with new products on the imminent horizon.*" [7]

References

[1] "3D Systems, Inc. Stereolithography Interface Specification, July 1988."

[2] "Autodesk ReMake." https://remake.autodesk.com/.

[3] "CATIA, Dassault Systèmes." www.3ds.com/catia/official-site.

[4] "CloudCompare release history." http://www.cloudcompare.org/release/history.txt.

[5] "Geomagic." http://www.geomagic.com/.

[6] "Google maps." https://www.google.com/streetview/.

[7] "History of CAD/CAM, CADAZZ, 2004." http://www.cadazz.com/cad-software-history.htm.

[8] "Hopkins 155 dataset." http://www.vision.jhu.edu/data/hopkins155/.

[9] "OpenCV." http://opencv.org/.

[10] "PTC Creo." http://www.ptc.com/cad/creo.

[11] "Siemens NX." https://www.plm.automation.siemens.com/en_us/products/nx/.

[12] "Solidworks, Dassault Systèmes." http://www.solidworks.com/.

[13] Y. Adato, Y. Vasilyev, O. Ben-Shahar, and T. Zickler, "Toward a theory of shape from specular flow," in *ICCV2007*, 2007.

[14] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building Rome in a day," *Communications of the ACM*, vol. 54(10), pp. 105–112, 2011.

[15] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," in *ICPR2006*, volume 3, 2006, pp. 1063–1068.

[16] M. E. Algorri and F. Schmitt, "Surface reconstruction from unstructured 3D data," in *Computer graphics forum, Blackwell Science Ltd*, volume 15(1), 1996, pp. 47–60.

[17] G. A. Atkinson and E. R. Hancock, "Recovery of surface orientation from diffuse polarization," *IEEE Trans. Image Process*, vol. 15, pp. 1653–1664, 2006.

[18] B. Babic, N. Nesic, and Z. Miljkovic, "A review of automated feature recognition with rule-based pattern recognition," *Computers in Industry*, vol. 59, no. 4, pp. 321–337, 2008.

[19] B. Babic, N. Nesic, and Z. Miljkovic, "A review of automated feature recognition with rule-based pattern recognition," *Computers in Industry*, vol. 59(4), pp. 321–337, 2008.

[20] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *ICRA1996, IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 3084–3091.

[21] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *Intl. J. of Computer Vision*, vol. 56(3), pp. 221–255, 2004.

[22] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Intl. J. of Computer Vision*, vol. 92(1), pp. 1–31, 2011.

[23] R. Bartels, J. Beatty, and B. Barsky, *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1995.

[24] S. Basah, A. Bab-Hadiashar, and R. Hoseinnezhad, "Conditions for motion-background segmentation using fundamental matrix," *IET Comput. Vis.*, vol. 3, pp. 189–200, 2009.

[25] C. Beder and W. Förstner, "Direct solutions for computing cylinders from minimal sets of 3D points," in *ECCV2010*, volume 3952, Springer, 2010, pp. 135–146.

[26] P. Benkő, R. R. Martin, and T. Várady, "Algorithms for reverse engineering boundary representation models," *Computer-Aided Design*, vol. 33(11), pp. 839–851, 2001.

[27] E. Boyer and M. Berger, "3D surface reconstruction using occluding contours," *Intl. J. of Computer Vision*, vol. 22, pp. 219–233, 1997.

[28] H. Chen and P. Meer, "Robust regression with projection based M-estimators," in *ICCV2003*, 2003, pp. 878–885.

[29] Y. Cheng, J. A. Lopez, O. Camps, and M. Sznaier, "A convex optimization approach to robust fundamental matrix estimation," in *CVPR2015*, 2015, pp. 2170–2178.

[30] O. Chum and J. Matas, "Matching with PROSAC - Progressive sample consensus," in *CVPR2005*, volume I, 2005, pp. 220–226.

[31] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 24, pp. 603–619, 2002.

[32] K. Cornelis, F. Verbiest, and L. V. Gool, "Drift detection and removal for sequential structure from motion algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26(10), pp. 1249–1259, 2004.

[33] A. P. Cracknell and L. Hayes, *Introduction to Remote Sensing*. CRC Press, second edition, 2007.

[34] B. Curless, "From range scans to 3D models," *Computer Graphics*, vol. 33, no. 4, pp. 38–41, 1999.

[35] J. E. Deschaud and F. Goulette, "A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing," in *3D Processing, Visualization and Transmission Conference*, 2010.

[36] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 35, pp. 2765–2781, 2013.

[37] C. Engels, H. Stewénius, and D. Nistér, "Bundle adjustment rules," *Photogrammetric computer vision*, vol. 2, pp. 124–131, 2006.

[38] O. D. Faugeras, Q. Luong, and S. Maybank, "Camera self-calibration: Theory and experiments," in *ECCV1992*, 1992, pp. 321–334.

[39] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. Assoc. Comp. Mach*, vol. 24, pp. 381–395, 1981.

[40] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D model construction for turntable sequences," in *Workshop 3D Structure from Multiple Images of Large-scale Environments*, SMILE, 1998, pp. 154–169.

[41] Y. Furukawa and C. Hernández, "Multi-view stereo: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 9(1-2), pp. 1–148, 2015.

[42] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 32, pp. 1362–1376, 2010.

[43] S. Gumhold, X. Wang, and R. S. MacLeod, "Feature extraction from point clouds," in *IMR*, 2001.

[44] J. Han, M. Pratt, and W. C. Regli, "Manufacturing feature recognition from solid models: a status report," *IEEE Transactions on Robotics and Automation*, vol. 16(6), pp. 782–796, 2000.

[45] C. Harris and M. Stephens, "A combined corner and edge detector," in *4th Alvey Vision Conference*, 1988, pp. 147–151.

[46] R. I. Hartley, "Theory and Practice of Projective Rectification," *IJCV*, vol. 35(2), pp. 115–127, 1999.

[47] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[48] T. Hassner, L. Assif, and L. Wolf, "When standard RANSAC is not enough: Cross-media visual matching with hypothesis relevancy," *Machine Vision and Applications*, vol. 25, pp. 971–983, 2014.

[49] A. Henn, G. Gröger, V. Stroh, and L. Plümer, "Model driven reconstruction of roofs from sparse LIDAR point clouds," *ISPRS Journal of photogrammetry and remote sensing*, vol. 76, pp. 17–29, 2013.

[50] G. B. Hughes and M. Chraibi, "Calculating ellipse overlap areas," *Comput. Visual Sci.*, vol. 15, pp. 291–301, 2012.

[51] H. Isack and Y. Boykov, "Energy-based geometric multi-model fitting," *International J. of Computer Vision*, vol. 97, pp. 123–147, 2012.

[52] H. Jin, S. Soatto, and A. Yezzi, "Multi-view stereo reconstruction of dense shape and complex appearance," *Intl. J. of Computer Vision*, vol. 63(3), pp. 175–189, 2005.

[53] S. Joshi and T. C. Chang, "Graph-based heuristics for recognition of machined features from a 3D solid model," *Computer-Aided Design*, vol. 20(2), pp. 58–66, 1988.

[54] K. Kanatani, "Ellipse fitting with hyperaccuracy," *IEICE Trans. Inf. & Syst.*, vol. E89-D, pp. 2653–2660, 2006.

[55] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32(3), p. 29, 2013.

[56] I. Lavva, E. Hameiri, and I. Shimshoni, "Robust methods for geometric primitive recovery and estimation from range images," *IEEE Trans. Sys. Man and Cybernetics B*, vol. 38, pp. 826–845, 2008.

[57] K. H. Lee and H. Woo, "Direct integration of reverse engineering and rapid prototyping," *Computers & Industrial Engineering*, vol. 38, no. 1, pp. 21–38, 2000.

[58] M. Lhuillier and L. Quan, "A quasi-dense approach to surface reconstruction from uncalibrated images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27(3), pp. 418–433, 2005.

[59] R. Litman, S. Korman, A. Bronstein, and S. Avidan, "Inverting RANSAC: Global model detection via inlier rate estimation," in *CVPR2015*, 2015, pp. 5243–5251.

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

[61] C. Loop and Z. Zhang, "Computing rectifying homographies for stereo vision," in *CVPR1999*, volume 1, 1999, pp. 125–131.

[62] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International J. of Computer Vision*, vol. 60, pp. 91–110, 2004.

[63] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Int. Joint Conf. on Artificial Intelligence*, 1981, pp. 674–679.

[64] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2004.

[65] M. Mantyla, D. Nau, and J. Shah, "Challenges in feature based manufacturing research," *Commun. ACM*, vol. 39(2), pp. 77–85, 1996.

[66] B. Matei and P. Meer, "Optimal rigid motion estimation and performance evaluation with bootstrap," in *CVPR1999*, volume 1, 1999, pp. 339–345.

[67] C. McGlone, E. Mikhail, and J. Bethel, *Manual of Photogrammetry*. ASPRS, fifth edition, 2004.

[68] O. Miksik and K. Mikolajczyk, "Evaluation of local detectors and descriptors for fast feature matching," in *21st International Conference on Pattern Recognition*, 2012.

[69] S. Mittal, S. Anand, and P. Meer, "Generalized projection-based M-estimator," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 34, pp. 2351–2364, 2012.

[70] L. Moisan, P. Moulon, and P. Monasse, "Automatic homographic registration of a pair of images, with a contrario elimination of outliers," *Image Proc. Online*, vol. 2, pp. 56–73, 2012.

[71] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *VISAPP*, vol. 1, pp. 331–340, 2009.

[72] D. Nistér, "Untwisting a projective reconstruction," *Intl. J. of Computer Vision*, vol. 60(2), pp. 165–183, 2004.

[73] T. T. Pham, T. J. Chin, J. Yu, and D. Suter, "The random cluster model for robust geometric fitting," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 36, pp. 1658–1671, 2014.

[74] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J. Frahm, "USAC: A universal framework for random sample consensus," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 35, pp. 2022–2038, 2013.

[75] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus," in *ECCV2008*, volume 5303, Springer, 2008, pp. 500–513.

[76] F. Schaffalitzky and A. Zisserman, "Geometric grouping of repeated elements within images," in *Shape, Contour and Grouping in Computer Vision*, Springer, 1999, pp. 165–181.

[77] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Intl. J. of Computer Vision*, vol. 47, pp. 7–42, 2002.

[78] J. L. Schönberger and J. M. Frahm, "Structure-from-motion revisited," in *CVPR2016*, 2016, pp. 4104–4113.

[79] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *CVPR2006*, 2006, pp. 519–528.

[80] E. Serradell, M. Özuysal, V. Lepetit, P. Fua, and F. Moreno-Noguer, "Combining geometric and appearance priors for robust homography estimation," in *ECCV2010*, volume 6313, Springer, 2010, pp. 58–72.

[81] G. Sohlenius, "Concurrent engineering," *CIRP Annals-Manufacturing Technology*, vol. 41(2), pp. 645–655, 1992.

[82] C. Strecha, W. V. Hansen, L. V. Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery," in *CVPR2008*, 2008, pp. 1–8.

[83] K. H. Strobl, E. Mair, T. Bodenmüller, S. Kielhöfer, W. Sepp, M. Suppa, D. Burschka, and G. Hirzinger, "The self-referenced DLR 3D-modeler," in *IEEE/RSJ Intelligent Robots and Systems, IROS 2009*, 2009, pp. 21–28.

[84] P. Sturm, "Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length," in *10th British Machine Vision Conference*, 1999, pp. 63–72.

[85] Z. L. Szpak, W. Chojnacki, and A. van den Hengel, "Guaranteed ellipse fitting with a confidence region and an uncertainty measure for centre, axes, and orientation," *J. Math. Imaging Vision*, vol. 52, pp. 173–199, 2015.

[86] R. B. Tennakoon, A. Bab-Hadiashar, Z. Cao, R. Hoseinnezhad, and D. Suter, "Robust model fitting using higher than minimal subset sampling," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 38, pp. 350–362, 2016.

[87] R. B. Tilove and A. A. G. Requicha, "Closure of boolean operations on geometric entities," *Computer-Aided Design*, vol. 12, no. 5, pp. 219–220, 1980.

[88] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: A factorization approach," *Intl. J. of Computer Vision*, vol. 9(2), pp. 137–154, 1992.

[89] B. Triggs, P. McLauchlan, R. I. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," *Vision Algorithms*, pp. 298–372, 1999.

[90] J. H. Vandenbrande and A. A. G. Requicha, "Spatial reasoning for the automatic recognition of machinable features in solid models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1–17, 1993.

[91] T. Varady, R. R. Martin, and J. Cox, "Reverse engineering of geometric models - An introduction," *Computer-Aided Design*, vol. 29(4), pp. 255–268, 1997.

[92] A. Vedaldi, H. Jin, P. Favaro, and S. Soatto, "KALMANSAC: Robust filtering by consensus," in *ICCV2005*, volume 1, 2005, pp. 633–640.

[93] T. Viéville and D. Lingrand, "Using singular displacements for uncalibrated monocular vision systems," in *ECCV1996*, 1996, pp. 207–216.

[94] H. Wang, T.-J. Chin, and D. Suter, "Simultaneously fitting and segmenting multiple-structure data with outliers," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 34, pp. 1177–1192, 2012.

[95] C. Wu, "VisualSFM: A visual structure from motion system," *http://ccwu.me/vsfm/*, 2011.

[96] C. Wu, "Towards linear-time incremental structure from motion," in *Inter. Conf. of 3D Vision*, 2013, pp. 127–134.

[97] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *CVPR2011*, 2011, pp. 3057–3064.

[98] G. Xu and Z. Zhang, "Epipolar geometry in stereo, motion and object recognition," *Kluwer Academic Publishers*, 1996.

[99] X. Yang and P. Meer, "Robust estimation of multiple inlier structures." Submitted to IEEE Trans. Pattern Anal. Mach. Intell., 2017. For a previous version, see also URL `https://arxiv.org/abs/1609.06371`.

[100] X. Yang, P. Meer, and H. C. Gea, "Robust recovery of 3D geometric primitives from point cloud." Submitted to ASME IDETC/CIE 2017.

[101] G. Zeng, S. Paris, L. Quan, and F. Sillion, "Accurate and scalable surface representation and reconstruction from images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29(1), pp. 141–158, 2007.

[102] W. Zhang, H. Wang, Y. Chen, K. Yan, and M. Chen, "3D building roof modeling by optimizing primitive's parameters using constraints from LiDAR data and aerial imagery," *Remote Sensing*, vol. 6(9), pp. 8107–8133, 2014.