Extraction robuste de primitives géométriques 3D dans un nuage de points et alignement basé sur les primitives

Thèse

Trung-Thien Tran

Doctorat en génie électrique
Philosophiæ doctor (Ph.D.)

Québec, Canada

Résumé

Dans ce projet, nous étudions les problèmes de rétro-ingénierie et de contrôle de la qualité qui jouent un rôle important dans la fabrication industrielle. La rétro-ingénierie tente de reconstruire un modèle 3D à partir de nuages de points, qui s'apparente au problème de la reconstruction de la surface 3D. Le contrôle de la qualité est un processus dans lequel la qualité de tous les facteurs impliqués dans la production est abordée. En fait, les systèmes ci-dessus nécessitent beaucoup d'intervention de la part d'un utilisateur expérimenté, résultat souhaité est encore loin soit une automatisation complète du processus. Par conséquent, de nombreux défis doivent encore être abordés pour atteindre ce résultat hautement souhaitable en production automatisée.

La première question abordée dans la thèse consiste à extraire les primitives géométriques 3D à partir de nuages de points. Un cadre complet pour extraire plusieurs types de primitives à partir de données 3D est proposé. En particulier, une nouvelle méthode de validation est proposée pour évaluer la qualité des primitives extraites. À la fin, toutes les primitives présentes dans le nuage de points sont extraites avec les points de données associés et leurs paramètres descriptifs. Ces résultats pourraient être utilisés dans diverses applications telles que la reconstruction de scènes on d'édifices, la géométrie constructive et etc.

La seconde question traiée dans ce travail porte sur l'alignement de deux ensembles de données 3D à l'aide de primitives géométriques, qui sont considérées comme un nouveau descripteur robuste. L'idée d'utiliser les primitives pour l'alignement arrive à surmonter plusieurs défis rencontrés par les méthodes d'alignement existantes. Ce problème d'alignement est une étape essentielle dans la modélisation 3D, la mise en registre, la récupération de modèles. Enfin, nous proposons également une méthode automatique pour extraire les discontinutés à partir de données 3D d'objets manufacturés. En intégrant ces discontinutés au problème d'alignement, il est possible d'établir automatiquement les correspondances entre primitives en utilisant l'appariement de graphes relationnels avec attributs.

Nous avons expérimenté tous les algorithmes proposés sur différents jeux de données synthétiques et réelles. Ces algorithmes ont non seulement réussi à accomplir leur tâches avec succès mais se sont aussi avérés supérieus aux méthodes proposées dans la literature. Les résultats présentés dans le thèse pourraient s'avérér utilises à plusieurs applications.

Abstract

In this research project, we address reverse engineering and quality control problems that play significant roles in industrial manufacturing. Reverse engineering attempts to rebuild a 3D model from the scanned data captured from a object, which is the problem similar to 3D surface reconstruction. Quality control is a process in which the quality of all factors involved in production is monitored and revised. In fact, the above systems currently require significant intervention from experienced users, and are thus still far from being fully automated. Therefore, many challenges still need to be addressed to achieve the desired performance for automated production.

The first proposition of this thesis is to extract 3D geometric primitives from point clouds for reverse engineering and surface reconstruction. A complete framework to extract multiple types of primitives from 3D data is proposed. In particular, a novel validation method is also proposed to assess the quality of the extracted primitives. At the end, all primitives present in the point cloud are extracted with their associated data points and descriptive parameters. These results could be used in various applications such as scene and building reconstruction, constructive solid geometry, etc.

The second proposition of the thesis is to align two 3D datasets using the extracted geometric primitives, which is introduced as a novel and robust descriptor. The idea of using primitives for alignment is addressed several challenges faced by existing registration methods. This alignment problem is an essential step in 3D modeling, registration and model retrieval. Finally, an automatic method to extract sharp features from 3D data of man-made objects is also proposed. By integrating the extracted sharp features into the alignment framework, it is possible implement automatic assignment of primitive correspondences using attribute relational graph matching. Each primitive is considered as a node of the graph and an attribute relational graph is created to provide a structural and relational description between primitives.

We have experimented all the proposed algorithms on different synthetic and real scanned datasets. Our algorithms not only are successful in completing their tasks with good results but also outperform other methods. We believe that the contribution of them could be useful in many applications.

Contents

List of Tables

List of Figures

List of Symbols

$\mathbf{P}$	the scanned data or the unorganized point cloud	17
$\mathbf{p}_i$	a data point i^{th} in the point cloud $\mathbf{P}$	17
$\mathbf{n}_i$ ($\vec{\mathbf{n}}_i$)	surface normal vector at a point $\mathbf{p}_i$	17
$\mathbf{C}$	covariance matrix of normal vector $\mathbf{n}_i$ and k-nearest neighbors	17
$\boldsymbol{\Psi}$	the set of inliers is updated each iteration	19
Δd	the orthogonal distance between a point $\mathbf{p}_i$ and the detected primitive	19
Δn	the angle between two unit vectors	19
MSE	mean square error	19
$\mathbf{k}_1, \mathbf{k}_2$	the maximum and minimum principal curvatures at each point.	21
$\mathbf{X}$	a list of primitive parameters.	21
$m(\mathbf{x})$	mean shift vector of mean shift clustering algorithm.	25
N_C	the number of cylinder in the point clouds.	31
x_{vir}	the parameters of virtual circle.	45
x_{est}	the parameters of estimated circle.	45
SC	total error between the virtual and estimated circles.	46
$\mathbf{CV}$	a 3x3 semi-definite covariance matrix.	48
σ	the average value of absolute signed distance of inliers.	49
h	the bandwidth value for mean shift clustering algorithm.	52
N_S	the number of spheres in the point clouds.	60
$\boldsymbol{\Phi}$	descriptive parameters of primitives.	69
T	The type of best-fit primitive at each data point.	72
$\mathbf{R}$	a rotation matrix of a a transformation	84

$\mathbf{t}$	a translation vector of a transformation	84
$\mathbf{T}$	a rigid transformation between two 3D datasets $\mathbf{T} = \{\mathbf{R}, \mathbf{t}\}$	84
$\mathbf{q}$	a quaternion vector	84
MAE	mean absolute error	89
pd_i	the projected distance calculated at a given point	114
Th	Threshold value to determine potential sharp feature at a given scale	115
$\mathbf{sf}$	a set of sharp features at a given scale	115
ns	the number of scales that are investigated for sharp feature extraction	116
$\mathbf{SF}$	a set of final sharp features	116
$\mathbf{G}$	graph of primitives extracted from a CAD model	123
$\mathbf{g}$	graph of primitives extracted from point clouds	123
$\mathbf{Q}$	the matrix indicates node correspondences between graph $\mathbf{g}$ and $\mathbf{G}$	124

To my family
To my wife

Acknowledgements

I have been interested in computer vision and graphics for a long time, especially geometry and shape processing. It was a great opportunity for me to work on this project. Although, the passion and interest I had for the project, I have been very lucky to have Denis Laurendeau as my supervisor to provide me with constant support and help during four years. For his excellent advice, encouragement and always patience, I am truly and greatly indebted. It was his supervision that helped me to reach the results that I have achieved.

I would also like to thank all research engineers and all the members of the Computer Vision and Systems Laboratory at Laval University for their help and numerous discussions on the project. In particular, I would like to thank Mrs Annette Schwerdtfeger for her proofreading and correcting all my documents. I am also grateful to NSERC/Creaform Industrial Research Chair on 3D Scanning for its financial support during this project. Finally, I would like to thank all Vietnamese students at Laval University for their encouragement and friendship that I have received during my study in Quebec city.

Last but not least, I would like to express my sincere gratitude to members of my family for their support and understanding, especially my wife Thanh-Binh during this project. She always gives me the inspiration and the courage that I need to pursue my goals. With my deepest sense of gratitude and love, this thesis is dedicated to her and to my family.

Chapter 1

Introduction

3D technologies have been developing rapidly during the last decade. **3D scanner** devices, in particular, are able to capture a large number of data points from the surface of objects in real-world scenes. These data points[1], called scanned data, may contain shape or color information of the object. A three-dimensional model is then reconstructed from the scanned data. There are many different ways to classify 3D scanners [Cur99] such as contact or non-contact, passive or active as shown in Fig. 1.1. Laser-based scanners (non-contact active) are widely used in industrial applications [May99, BR02, MS11]. They emit some types of radiation or light and detect the reflection of the rays from the surface of the object in order to capture 3D data.

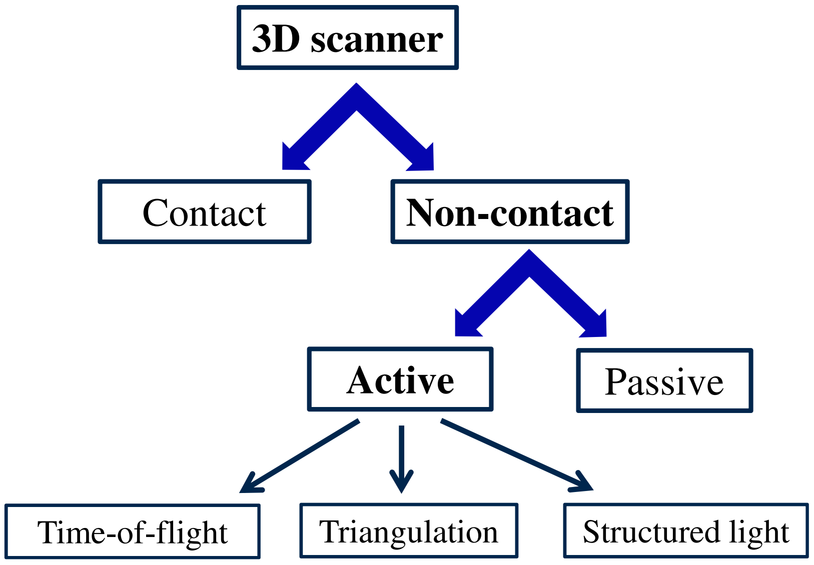
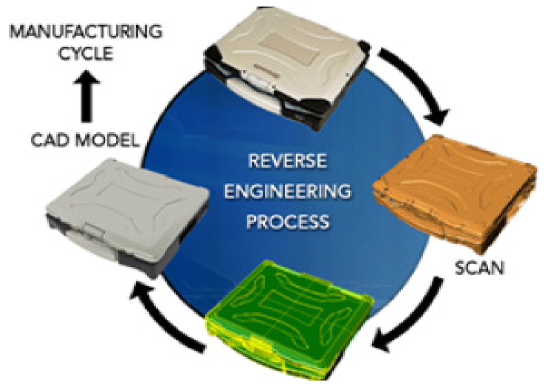


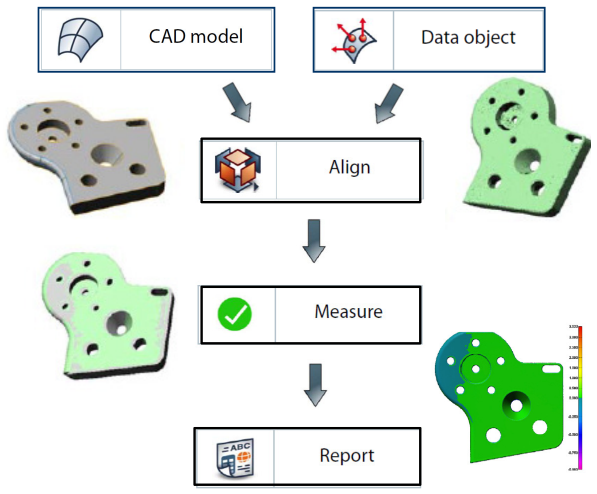
Figure 1.1. Classification of 3D scanner technologies and laser-based approaches (non-contact active).

1. "Scanned data", "point clouds", "scans" and "views" are considered as synonyms of 3D data, and these terms are used interchangeably throughout the thesis.

According to the development of scanning technologies (both hardware and software), millions of data points may be captured and processed per second by such sensors. Among common problems in computer vision, **surface reconstruction** has attracted much attention from the research community but still represents many challenges [MWA+13, BTS+14]. Building an accurate reconstructed model is the central goal in various applications such as reverse engineering [VMC97], urban area modeling [MWH+06, MWA+13], cultural heritage [KFH09, SCC+11], etc. For instance, reverse engineering of a manufactured object plays a fundamental role in the production process, especially when no original design drawings and specification documentations are available, but the real product is. While in a conventional engineering process the product is made from the designed model, the 3D model of the real object can be reconstructed from scanned data in the reverse engineering process. In essence, **reverse engineering** [VMC97] tries to build a digital model of an object which will resemble the object as closely as a computer-aided design would do. Then a modification can be applied to the reconstructed model in order to create a new design model for reproduction, as shown in Fig. 1.2(a).



(a) Reverse engineering process. Image taken from www.3dscanco.com.



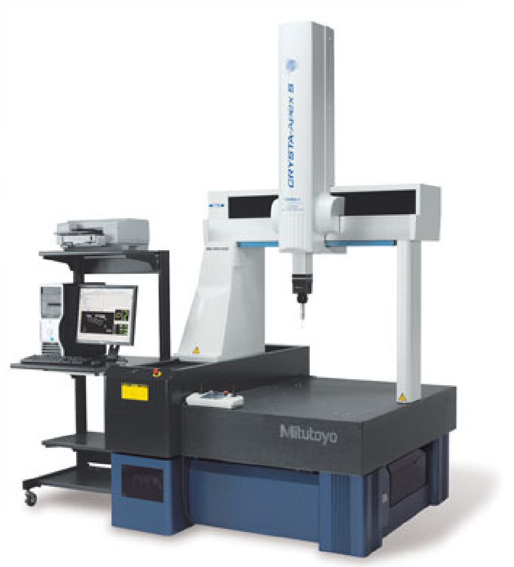
(b) Quality control and inspection. Image taken from PolyWorks®V10 Beginner's Guide.

Figure 1.2. Overall diagram: reverse engineering (a) and quality control process (b).

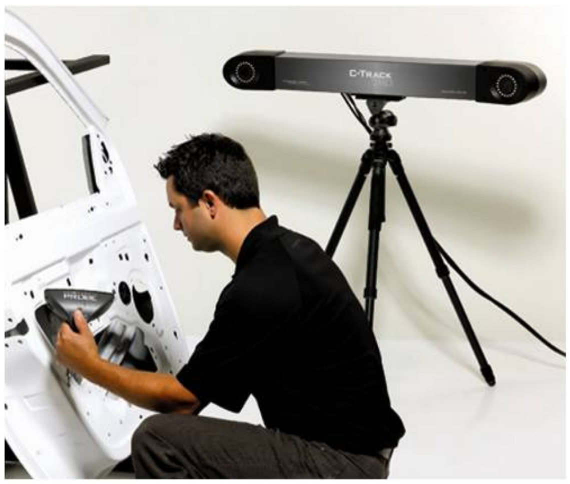
Quality control and inspection is the process of comparing the scanned data of a manufactured part against its CAD model. A human inspector will then be provided with a list and description of unacceptable product defects such as surface distortion or dimensional deviation (Fig. 1.2(b)). By performing quality control and inspection, companies are able to identify problems early in the fabrication process, the ultimate goal being to eliminate the causes of the presence of the defects on the manufactured parts.

Moreover, reconstructed models are also being widely used in other areas such as movies, entertainment and games [RA11]. Movie production often uses the latest technologies in 3D computer animation and visual effects. There are several blockbuster 3D films such as Final Fantasy (2001), Avatar (2009), Fast & Furious 7 (2015), etc. that used 3D model and rendering techniques. The use of 3D models has been increasing massively. Currently, it is not hard to make a live-action movie that is rendered from 3D models and animation software packages such as 3DS Max [3DS], Blender [Ble], etc.

Traditionally, the most common device used in reverse engineering and quality control is a coordinate measuring machine (CMM) [MHSRP08, Li11], which measures local points at the surface of an object. The principle of CMM is that measurements are made by a probe mounted on a robot or portable device (Fig. 1.3). Although the measurement is very accurate, CMMs require that the material be hard enough to resist the force applied by the probe. Moreover, one more problem of these systems is the limited number of data points collected at the surface of the object in a reasonable amount of time. CMM can capture several thousand points on an object, a density that may not be enough to describe the object completely. Recently, an increased variety of modern 3D hand-held sensors have been used in reverse engineering and quality control applications.



Crysta-Apex S 500/700/900/1200 from Mitutoyo



HandyProbe with C-Track system from Creaform

Figure 1.3. CMM systems with articulated mechanisms for moving the probe (a) and portable probe localized by a photogrammetric positioning system (b).

1.1 Problem Statement

As mentioned in [RvdH05], most objects are formed by the combination of basic geometric primitives such as planes, spheres, cylinders, etc. Therefore, geometric primitive extraction from 3D data, also known as 3D segmentation [APP$^+$07, Sha08, TPT15] and shape decomposition [CGF09, KHS10], plays a key role in 3D vision. No matter what data acquisition is used, an important problem in 3D vision consists of using the huge number of points collected by the sensors to build a geometric model of the objects of interest.

Segmentation is the process through which the data points are partitioned into connected regions or parts that can be approximated by standard CAD primitives (e.g. planes, cylinders, etc.) or volumetric primitives (e.g. super-ellipsoids), respectively. Existing segmentation methods can be grouped into two basic categories:

— *Patch-based approaches*: The 3D mesh is segmented into regions that represent distinct regions of the object and that can be described by various primitives such as planes, cylinders, spheres, polynomial patches, etc., as shown in Fig. 1.4.

— *Part-based approaches*: The 3D mesh is decomposed into volumetric parts which can be approximated by volumetric primitives (e.g. super-ellipsoids). Shape decomposition belongs to this category and often uses the minima rule [HR84] to extract meaningful parts of the object. An example of the part-based approach for mesh segmentation is shown in Fig. 1.5.

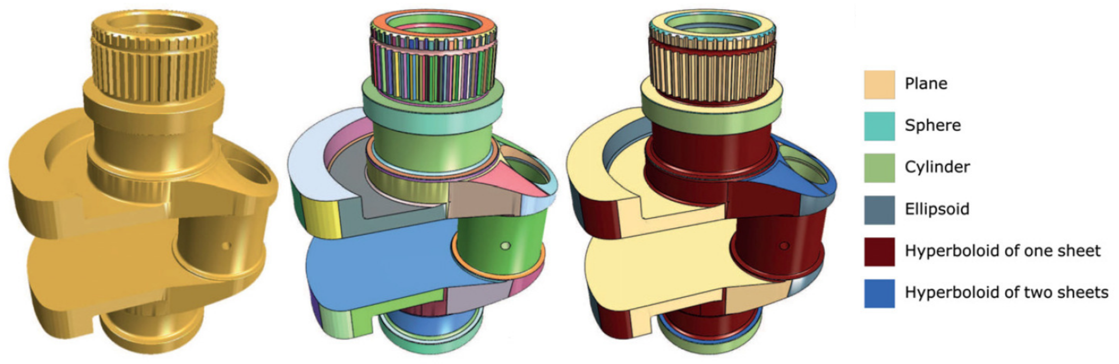


Figure 1.4. An example of patch-based approach for mesh segmentation. Image taken from [YWLY12].

This thesis focuses on patch-based segmentation. Although significant research has recently been conducted on primitive shape extraction from a single model [VMC97, BV04, AFS06, SWK07], this problem still remains open, especially for complex data with noise and outliers. In this context, the primary goal of this thesis is to propose a robust framework for **geometric primitive extraction** from unorganized point clouds. The framework is proposed to work directly on unorganized point clouds, but , as discussed in the document, it could be extended easily to triangle meshes and range images. Moreover, although a few approaches have been proposed for primitive fitting and detection, validation methods for assessing the reliability of extracted primitives are still scarce. This important problem is also addressed in this thesis.

Since the field of view of 3D sensors is limited, it is necessary to scan an object from different

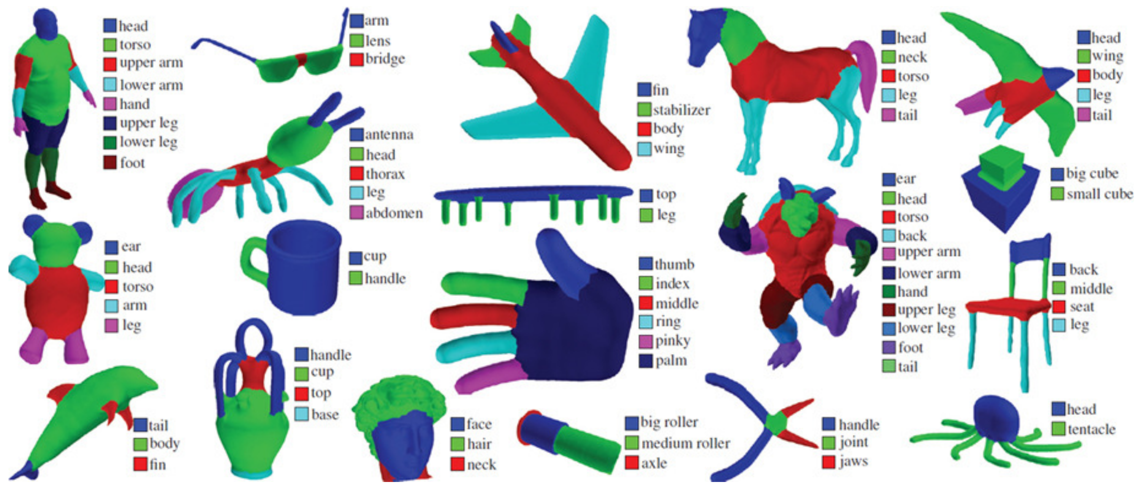


Figure 1.5. An example of part-based approach for mesh segmentation. Image taken from [KHS10].

viewpoints in order to cover its entire surface. Scans collected from the different viewpoints are used in the modeling process. Prior to building a model, a registration [2] step is required to transform the data in all of the scans into a common coordinate frame. Generally, the problem may be described as follows: Let $\{\mathcal{M}, \mathcal{S}\}$ be two finite size point sets in a finite-dimensional real vector space $\mathbb{R}^d$, which can contain a different number of points. The registration involves finding a transformation which moves "*model*" point set $\mathcal{M}$ to "*scene*" set $\mathcal{S}$ such that the difference between them is minimized. In other words, a mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$ is computed to find the best alignment between the transformed "*model*" set and the "*scene*" set.

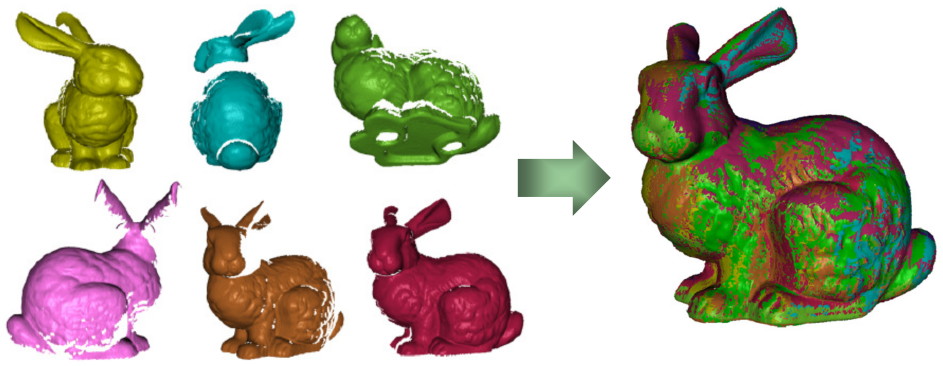
There are several ways to classify the 3D registration problem.

— According to the rigidity of the surface or object, registration may be classified as a *rigid* or a *non-rigid* approach as described in [TCL+13]. For rigid registration, only a unique transformation (rotation and translation) is needed to align all points in the scans, as shown in Fig. 1.6(a). However, for non-rigid registration, each data point requires its own transformation, as shown in Fig. 1.6(b).

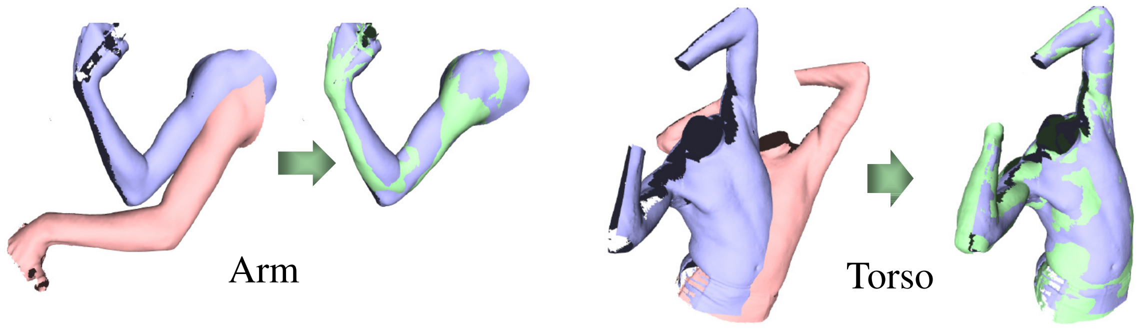
— Depending on the quality of the final result, registration methods can also be classified into *coarse registration* methods [PMW05, DRLS15] or *fine registration* methods [BM92, RL01, SMFF07]. While fine registration tries to align two datasets as close as possible, coarse registration brings the datasets closer to each other and is often considered as the initial step for fine registration. An illustrated example of coarse and fine registrations is shown in Fig 1.7.

As of today, the registration problem has been investigated intensively in an effort to align meshes or point clouds having similar densities of data points. Many 3D descriptors (ie.

2. Registration is also described by other terms such as "alignment" and "matching". These terms are used interchangeably in the thesis.



(a) Rigid registration [GMGP05]



(b) Non-rigid registration [CZ09].

Figure 1.6. Rigid and non-rigid registration of 3D data.

FHFP, Spin image, etc.) have been proposed [DRLS15] and keypoints (i.e. Harris3D, SUFT, etc.) have been extracted from the 3D data for correspondence matching [TSDS13]. However, not all of the keypoints and descriptors should be used for matching because many bad keypoints and descriptors are also detected by noise and outliers. Therefore, random sample consensus (RANSAC) [FB81] is often used to reduce the complexity of the correspondence search. RANSAC is known as a robust technique to deal with noise and outliers by using random sampling with a minimum number of data points selected from a set of keypoints. A minimum of three pairs of matching points is required to compute the transformation between two datasets [HZ03].

Although some descriptors [DRLS15, TSDS13] have been proposed to register two or more 3D datasets, very few descriptors [JH99, RBB09] can handle registration between scans of manufactured objects, which is often used in quality control application. The problem is even harder when the task is to align a point cloud with its CAD model, because CAD models contain a small number of vertices compared to the point cloud. In such cases, the variants of Iterative Closest Point (ICP) [RL01] as well as most 3D descriptors [DRLS15] (e.g. spin image [JH99], FPFH [RBB09], etc.) fail at the alignment task because of symmetrical and sim-

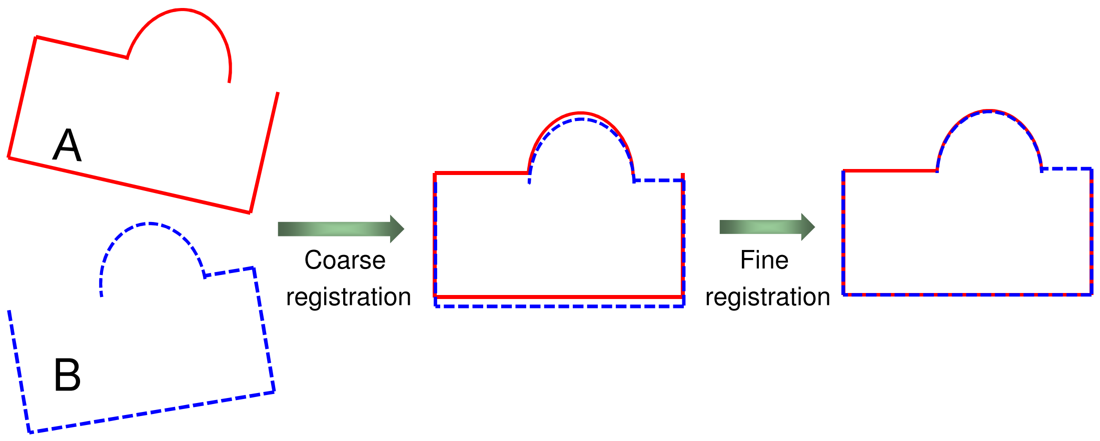


Figure 1.7. An illustrated example of coarse and fine registrations. While fine registration tries to align two data as close as possible, coarse registration just brings them closer and is considered as initial step for fine registration problem.

ilar geometry from the surface data of the primitives. Therefore, primitives extracted during the extraction step are used as a novel descriptor for alignment problem, and this referred to as **primitive-based registration**.

The secondary goal of this thesis is to introduce a novel framework for registering scanned 3D data with a CAD model using extracted primitives as a coarse registration step. This idea will not only be applied to the alignment of scanned 3D data with a CAD model but for the registration of multiple partial scans of an object as well.

1.2 Contributions of the research work

In this project, we propose a framework to extract geometric primitives from point clouds and to align point clouds with original CAD models using the extracted primitives. The overview of the framework is illustrated in Fig. 1.8. The framework is comprised of three modules: **geometric primitive extraction** from scanned data, **primitive extraction** from the CAD model and **alignment between scanned data and the CAD model** for quality control and inspection. The main contribution of the research consists of geometric primitive extraction and primitive-based alignment in which primitives are extracted robustly from point cloud data and are used for object reconstruction and alignment applications.

— First, the research focuses on processing the scanned 3D data from the surface of manufactured objects. A robust framework is proposed to extract geometric primitives such as cylinders, spheres and planes from unorganized point clouds [TCL15a, TCL16b, TCL15b]. Due to the novel validation approach proposed in the thesis, reliable primitives are extracted with high accuracy. In contrast, existing approaches that only extract approximate primitives.

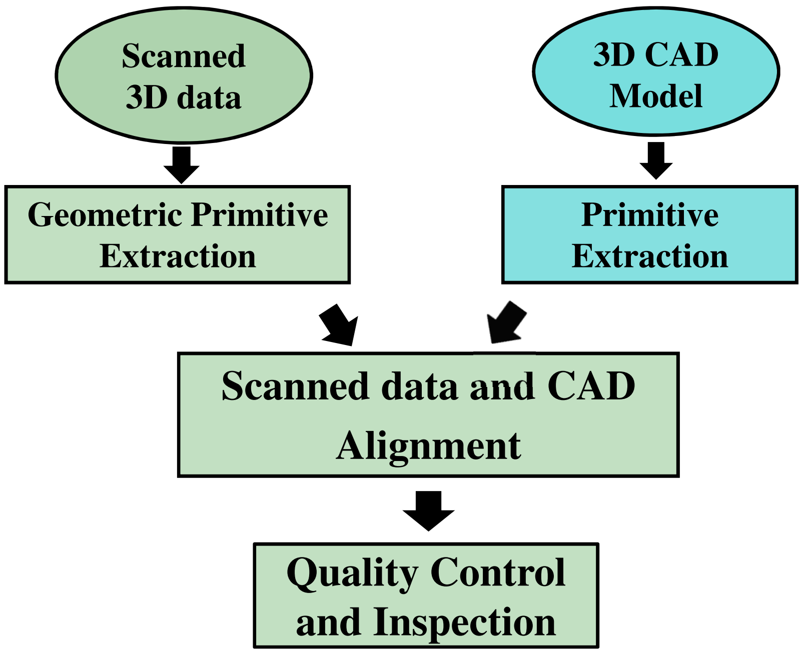

Scanned 3D data
3D CAD Model
Geometric Primitive Extraction
Primitive Extraction
Scanned data and CAD Alignment
Quality Control and Inspection

Figure 1.8. Overview of the framework for geometric primitive extraction and primitive-based alignment. The input is scanned data and a 3D CAD model. **Geometric primitives** are extracted from both the data and the model. A **primitive-based alignment** algorithm is then applied to register the scanned data and the CAD model. The alignment result can be used for quality control applications among other applications.

— Primitives in CAD model are segmented with a predefined number of clusters using hierarchical mesh segmentation [GWH01, AFS06]. This module is implemented offline as a preprocessing step. The basic idea of hierarchical face clustering is discussed in Section. 6.3.1.

— A novel approach is proposed to align the primitives extracted from the point cloud with its CAD model [TCL16a, TCL16c]. The approach combines a new optimization technique with a new objective function to guarantee convergence. An error map between the production and CAD model is then determined for quality control and inspection.

— The problem of **sharp feature extraction** from point cloud data is also investigated in the thesis [TAL13, TCN$^+$14]. As mentioned earlier, most manufactured objects consist of a combination of common geometric primitives. The intersection between these primitives forms a sharp feature. Sharp features help to understand the structure of the underlying geometry of a surface. Moreover, this could be used in other applications such as surface denoising [PLJ$^+$13], remeshing [VRKS01] and feature-aware reconstruction [HWG$^+$13].

— The extracted sharp features could be used to create a graph that describes the structure of scanned data. **Graph matching** is proposed and integrated into the proposed framework to create an automatic registration between the scanned data and its CAD model.

The algorithms presented in this thesis have already been published in or submitted to several well-known conferences and journals [TAL13, TCN$^+$14, TCL15a, TCL16b, TCL15b, TCL16a].

1.3 Thesis Outline

The content of this thesis is structured in two parts. **Part I** introduces a robust framework for extracting single primitive types separately and multiple primitives simultaneously from unorganized point clouds. **Part II** describes the complete framework for aligning scanned 3D data with the CAD model using the extracted geometric primitives. The remainder of the thesis is organized as follows.

Part I: 3D Geometric Primitive Extraction

— Chapter 2 introduces a novel framework for extracting multiple cylinders robustly from unorganized point clouds. An iterative cylinder fitting is developed from traditional algebraic fitting. In addition, a new technique for validating the detected cylinders is proposed. The validation problem is transformed into a simpler circle detection problem, a question that has been investigated thoroughly and for which reliable solution do exist. By exploiting a Mean Shift Clustering paradigm, multiple cylinders are extracted and estimated robustly. The output of this approach are the descriptive parameters of the extracted cylinders and their associated points.

— Chapter 3 extends the framework presented in Chapter 2 to the problem of sphere extraction from 3D data. The validation of the extracted spheres is also transformed into a simple problem of circle detection. Moreover, an efficient resampling technique is introduced to accelerate the extraction process.

— A general framework is proposed in Chapter 4 to extract multiple types of primitives from unorganized point clouds. The proposed framework extracts curved surfaces such as cylinders and spheres first. The points associated with curved surfaces are removed from the point cloud. Planar surfaces are then extracted from remaining points. The framework extracts primitives efficiently without model confusion in which cylindrical and spherical parts are often approximated by planes.

Part II: Primitive-based Alignment and Applications

— Chapter 5 is concerned with the problem of registering two sets of synthetic geometric primitives. The chapter develops the mathematical equations and optimization technique for each type of basic primitive (plane, cylinder and sphere). With the proposed minimization technique and objective function proposed in this chapter, good convergence results are achieved for the alignment.

— Chapter 6 presents a complete framework to align a point cloud and the CAD model of the corresponding object in which the proposed method in Chapter 5 is used. Since most 3D descriptors and variants of ICP fail for the problem, a solution is to use the primitives themselves for alignment. Then error map and tolerance measurement are estimated in the context of quality control and inspection applications.

— Chapter 7 presents an automatic technique for extracting sharp features from 3D data. First, a method is proposed to detect potential sharp features at a given scale. The process is iteratively applied for increasing scales. In the end, valid sharp features are determined by multi-scale analysis as a refinement. The extracted sharp features are integrated into the proposed framework in Chapter 6 to create an automatic registration.

— Finally, Chapter 8 concludes the thesis with a brief discussion of directions for future work.

Part I: 3D Geometric Primitive Extraction

Surface reconstruction [MWA+13, BTS+14] has been investigated intensively in the past decades. The general objective is to recover and reconstruct a digital representation of the object from the 3D data. The data acquisition process ranges from active methods such as laser-based sensors, structured-light sensors to passive methods such as multi-view stereo. This problem relates to many research fields such as image processing, computer vision and graphics. Berger et al. [BTS+14] is an excellent survey in which each research direction is clearly summarized and where advantages and drawbacks of each approach are listed. Previous works have often focused on reconstructing a piece-wise smooth surface of the original shape. However, as mentioned previously, man-made objects are generally composed of a combination of geometric primitives such as cylinders, spheres, etc., as shown in Fig 1.9.

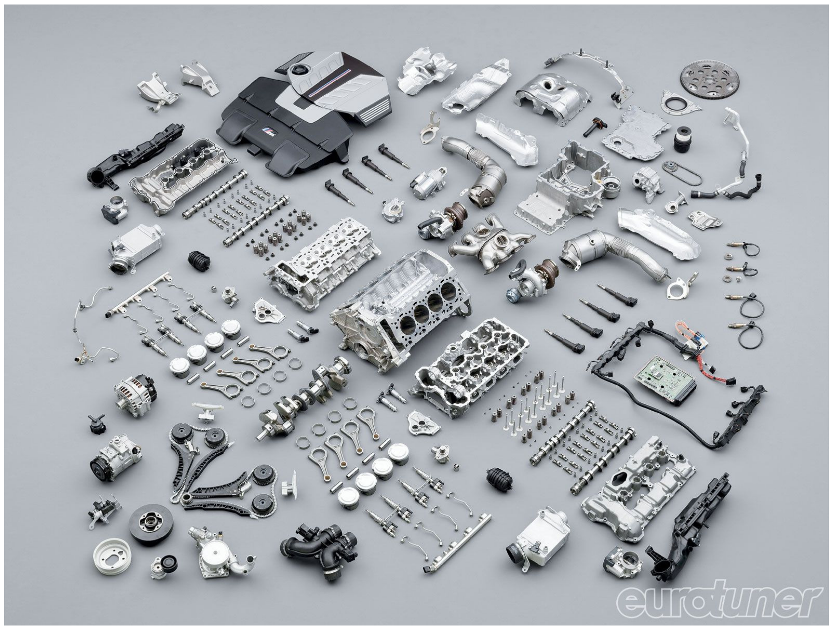


Figure 1.9. Most parts of the BMW X6M engine are composed of geometric primitives such as cylinders and planes. Image taken from Eurotuner.

Recently, some research works have attempted to generate a higher level of representation of the data, so geometric primitives are used to abstract the 3D data [SWK07, AFS06]. In other words, primitives such as planes, cylinders and spheres are now used to represent the data points, as shown in Fig 1.10. In this part of the thesis, each type of primitive will be extracted and detected directly from point clouds and the reconstructed model consists of the combination of the extracted primitives.

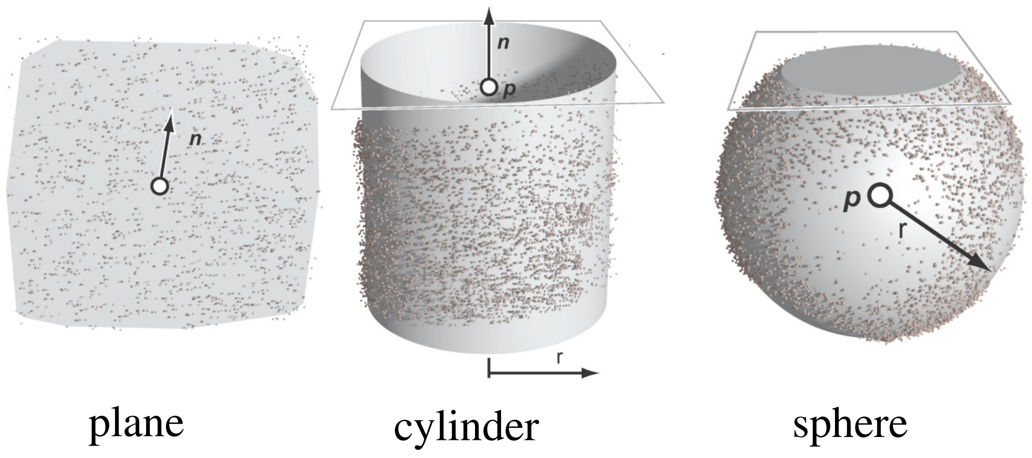


Figure 1.10. Geometric primitives are used to represent the data points. Image taken from Li et al. [LWC+11].

The data scanned from the surface of a manufactured object is considered as the input of the proposed framework. The output includes the type of primitives, their descriptive parameters and their associated points. As a preprocessing step, normal vectors and principal curvatures are first computed for each point in the point cloud. This differential geometry information is computed once and used throughout the procedure. The extraction of cylinders and spheres is described in Chapter 2-3, respectively. In these chapters, the extraction of a single type of primitive separately is described. Then a complete framework for extracting multiple types of primitives simultaneously is introduced in Chapter 4. The literature review, the proposed algorithm and experimental results related to each problem are described and discussed in Chapter 2-4, respectively.

Chapter 2

Cylinder Extraction

2.1 Introduction

Among basic primitives, cylinders may be the most frequently used type of primitive in industrial objects such as pipes, sleeves, connectors. Moreover, cylinder detection could be used in various applications: reverse engineering [VMC97], 3D registration [RDvdHV07], pipeline plant modeling [LZH$^+$13], etc. Currently, there are many methods [LZH$^+$13, CG01, RvdH05, LPGW87] that have investigated how to detect cylinders and estimate their descriptive parameters from point cloud data. A cylinder is frequently described by 3 parameters: axis orientation $\overrightarrow{\mathbf{a}}$, a point on the axis $\mathbf{p}^*$ and radius r [AP10]. However, few methods can determine these descriptive parameters accurately. Most methods [SWK07, CG01, RvdH05, LZH$^+$13] require the user to choose some threshold values- a difficult task for complex models. Moreover, the problem of detecting multiple cylinders is non-trivial, since it is easily affected by noise/outliers or depends on prior segmentation results. Cylinder extraction and cylinder fitting are problems that have been investigated recently, but which are still open.

Therefore, our goal is to propose a robust method for estimating cylinder parameters accurately and extracting multiple cylinders simultaneously from a given point cloud. Some examples of cylinder extraction detected by our method are shown in Fig. 2.1. The contributions of the proposed method are summarized as follows:

— Estimating the descriptive parameters of cylinders accurately.
— Extracting multiple cylinders at the same time.
— Proposing a novel method for validating the detected cylinders.
— Detection and estimation robust to acquisition noise and outliers.

The rest of this chapter is organized as follows. Previous work is summarized in Section 2.2. Single cylinder fitting is presented in Section 2.3. The procedure for multiple cylinder extraction is described in Section 2.4. Results and discussion are presented in Section 2.5, and Section 2.6 draws some conclusions on the proposed method.

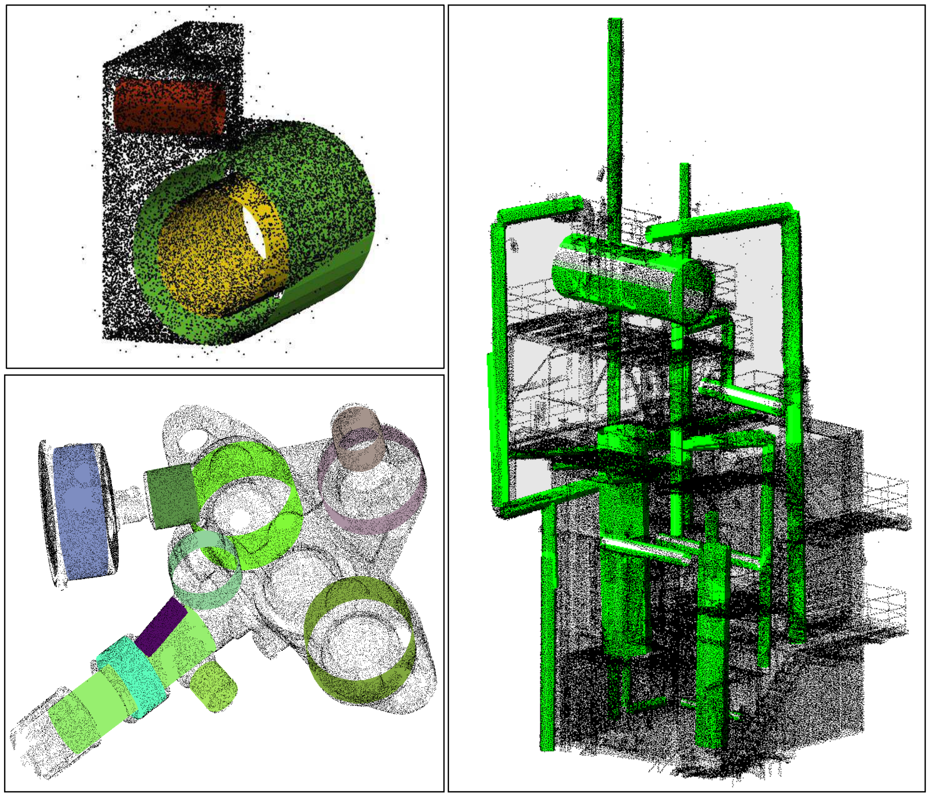


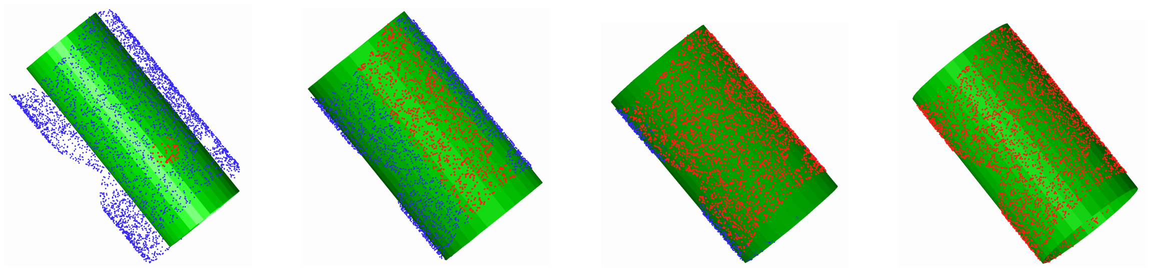
Figure 2.1. Detected cylinders from noisy data with outliers using the proposed method in this chapter.

2.2 Related work

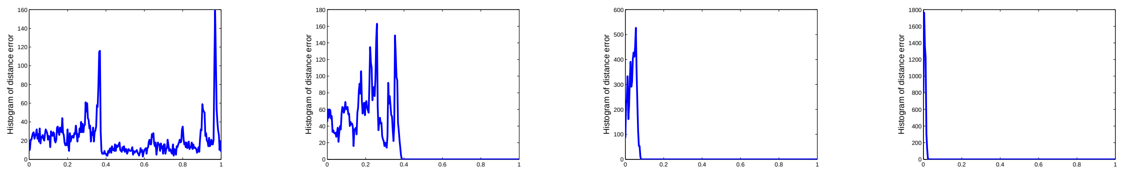
Many methods have investigated the problem of cylinder extraction. In this section, we briefly review some methods related to the one presented in this chapter. Existing methods are generally classified into two categories: those requiring prior segmentation and those working directly on point clouds.

The methods belonging to the first group fit a cylinder model to a set of segmented points. They often use non-linear optimization to minimize the sum of orthogonal distances to the cylinder surface [LMM98, BKV$^+$02]. These methods depend on the quality of the initial segmentation [Tau91] that needs to assign the correct type of primitive to each data point. However, this requirement is difficult to meet for real data with noise and outliers. Moreover, non-linear least-squares fitting is an iterative process that requires good initial parameters to avoid local minima.

Hierarchical face clustering [AFS06, AP10], which often requires that the number of segments be pre-selected by the user, is an interesting method for primitive extraction. However, this method is computationally intensive in both time complexity and memory requirement even if the model contains just a few thousand points, because the method investigates



(a) After 1st iteration. (b) After 2nd iteration. (c) After 3rd iteration. (d) After 4th iteration.



(e) Histogram of distance error after 1st iteration. (f) Histogram of distance error after 2nd iteration. (g) Histogram of distance error after 3rd iteration. (h) Histogram of distance error after 4th iteration.

Figure 2.2. Iterative fitting for cylinder extraction from iteration are shown in Fig. 2.2(a)-2.2(d). Red points are inliers propagated gradually on the surface of the cylinder. The distance error from a point to the surface of the detected cylinder is computed and shown in Fig. 2.2(e)-2.2(h). The error gradually drops close to zero after a few iterations as shown on the histograms in Fig. 2.2(e)-2.2(h).

all types of primitives at the same time to create a global list of priorities by using algebraic fitting [Pra87]. Moreover, primitive fitting is not accurate, especially when the number of points is small. This affects the priority list and leads to poor results. In addition, cylinder fitting in [AFS06, AP10] cannot identify cylinder parameters accurately and reliably in a single step. Fig. 2.2(a) shows an example of this method after one iteration step; the detected cylinder does not fit the data points very well. Our method improves this technique to estimate cylinder parameters accurately through an iterative fitting process.

The second group attempts to process data points directly using RANSAC [LWC$^+$11, SWK07] or Hough-based methods [RvdH05, CG01] which are popular techniques for parametric model extraction. RANSAC [FB81, BF81] fits a model by using random sampling with a minimum number of data points. For cylinders, two data points with their surface normals are enough for fitting [SWK07]. However, the user must set several thresholds that may vary for different models, especially with noisy data and outliers. The results of the algorithm depend on the initial selection of points. In the worst case, RANSAC does not converge to a valid solution even when cylinders are present.

The Hough transform [DH72] is popular for detecting simple shapes such as lines, circles and planes. But it is also known as a voting method with high computational requirements, and for which choosing the resolution of the accumulator cell is critical. In addition, 5 parameters are needed for cylinder detection [RvdH05]. As reported in [RvdH05, FO01], some authors have attempted to speed up the procedure by splitting and pruning the parametric space.

A significant research study focuses on the use of the Gaussian sphere [dC76] to extract primitives [RvdH05, CG01, LZH+13]. Liu et al. [LZH+13] detect cylinders in large-scale point clouds of a pipeline plant. However, the limitation of the method is that it only supports the extraction of pipes either perpendicular or parallel to the ground. Chaperon et al. [CG01] combine the Gaussian sphere with a random sampling method (RANSAC) to extract cylinders and estimate their parameters. However, based on our experiments, these methods do not achieve accurate results for complex models containing other types of primitives. For example, in Fig. 2.3(a), the joint model is composed of 3 cylinders and several planes leading to red and green circles, where two parallel cylinders correspond to the same red circle. However, the red circle results not only from points belonging to cylinders but also outliers and other points on a plane having coincident normals (Fig. 2.3(b)). The blue circle is not created by the points belonging to the cylinder surface but rather by points distributed all over the model (Fig. 2.3(c)).

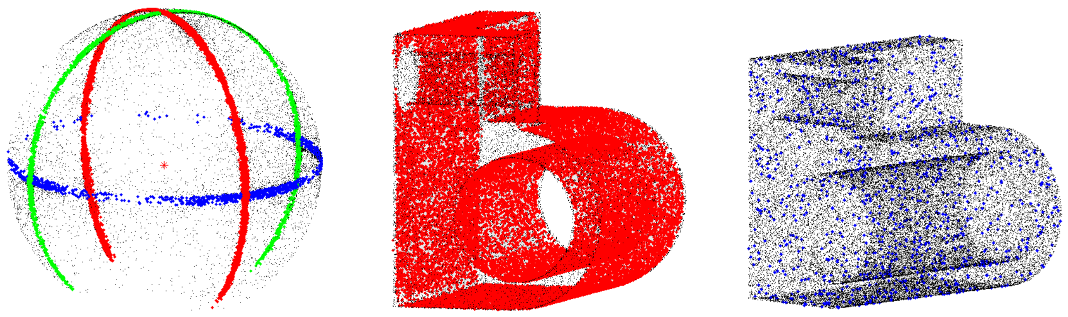

(a) Gaussian sphere of joint model contains three circles.
(b) Red circle is created by the red points belonging to cylinders and planes.
(c) Blue circle is created by the blue points distributed all over the model.

Figure 2.3. Gaussian sphere of the joint model in top left of Fig. 2.1 and related problems. The blue circle is caused by planes and outliers and leads to wrong results. The problem of detecting circles on the Gaussian sphere becomes harder when there are many cylinders or when there are small cylinders in the point clouds.

The proposed method belongs to the second group, which extracts cylinders directly from point cloud data. The problem addressed in this chapter is to estimate cylinder parameters accurately. The algorithm is an iterative process of distance-based and normal-based fitting starting at different initial points. Furthermore, we propose a novel two-step validation method exploiting geometric consistency to improve cylinder detection. By using mean shift clustering, multiple cylinders can be extracted simultaneously.

2.3 Iterative cylinder fitting

In this section, we assume that points are sampled on the surface of a cylinder, which includes noise and outliers. The case of multiple cylinder detection with the presence of other types of primitives is investigated in the next section. The pseudo-code for single cylinder fitting is shown in Algorithm 1. The steps of this method are described in more detail as follows.

The input of the algorithm is a set of points sampled on the surface of a cylinder $\mathbf{P} = \{\mathbf{p}_i \in \mathbb{R}^3, i = 1, ..., N\}$. An un-oriented normal vector is available at each point. Otherwise, normal vectors can be computed using the procedure described in [HDD+92]. For the sake of completeness, surface normal computations is explained in more details in Appendix .2. The output of the method is a set of parameters describing the detected cylinder $\{\mathbf{p}^*, \overrightarrow{\mathbf{a}}, r\}$.

A cylinder is described by a set of 3 parameters: axis orientation vector $\overrightarrow{\mathbf{a}}$, a point on the axis $\mathbf{p}^*$ and radius r. The first step is to compute the orientation of the axis of the cylinder, that is the vector being the most orthogonal to all of the normal vectors $\overrightarrow{\mathbf{n}_i}$ of $\mathbf{\Psi}$ (see Eq. 2.3). Therefore, a positive semi-definite matrix $\mathbf{C} = \sum_{i=1}^{|\mathbf{\Psi}|} \mathbf{n}_i^T \mathbf{n}_i$ is computed and analyzed, in which $\mathbf{n}_i$ is a row-vector (row-vector convention is used in this thesis). The eigenvector of $\mathbf{C}$ corresponding to the smallest eigenvalue is considered as the axis orientation $\overrightarrow{\mathbf{a}}$, and $\mathbf{C}_x$ and $\mathbf{C}_y$ are the two other eigenvectors that create a coordinate frame in the plane.

In the second step, $\tilde{\mathbf{p}}_i = [\langle \mathbf{p}_i, \mathbf{C}_x \rangle_2, \langle \mathbf{p}_i, \mathbf{C}_y \rangle_2]$ is the 2D projection of $\mathbf{p}_i \in \mathbf{\Psi}$ on the plane with normal $\overrightarrow{\mathbf{a}}$ that passes through the origin O. These projected points distribute on the plane as a circle. Then the problem of determining the radius r and a point p^* on the axis reduces to determining the radius r and center $\tilde{\mathbf{c}}$ of the circle. Algebraic fitting [Pra87] is used to compute the radius r and center $\tilde{\mathbf{c}} = [\tilde{\mathbf{c}}_x, \tilde{\mathbf{c}}_y]$ because it is faster than implicit fitting [Tau91]

Algorithm 1: Algorithm for iterative single cylinder fitting.

```
Data: Point cloud and normal vectors at each point.
Result: Parameters {$\mathbf{p}^*$, $\overrightarrow{\mathbf{a}}$, $r$} of fitted cylinder.
index:=0; $k$:=10 (maximum number of iterations);
$\mathbf{\Psi}$:=a given point $\in \mathbf{P}$ and its neighborhood whose size is initially set to 50;
while index$\leq k$ do
    1. Find the cylinder axis orientation $\overrightarrow{\mathbf{a}}$ by finding the eigenvector corresponding to the
       smallest eigenvalue of the covariance matrix $\mathbf{C}$ of normal vectors of inliers $\mathbf{\Psi}$;
    2. Project $\mathbf{\Psi}$ on a plane with normal $\overrightarrow{\mathbf{a}}$ and going through the origin O;
    3. Fit a circle to projected points to find radius $r$ and a point $\mathbf{p}^*$ belonging to the axis (Eq.
       2.1-2.2);
    4. Use normal and distance filtering to update $\mathbf{\Psi}$ for the next iteration (Eq. 2.3);
    5. index:=index+1;
end
```

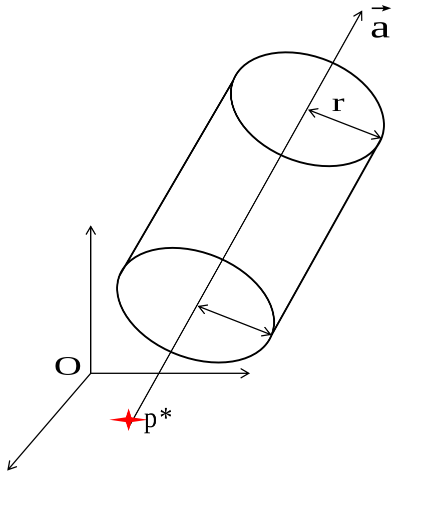
and efficient for the circle fitting problem. Therefore, algebraic distances are minimized in the following equation:

$$[\tilde{\mathbf{c}}_x, \tilde{\mathbf{c}}_y, r^2 - \tilde{\mathbf{c}}_x^2 - \tilde{\mathbf{c}}_y^2]^T = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{2.1}$$

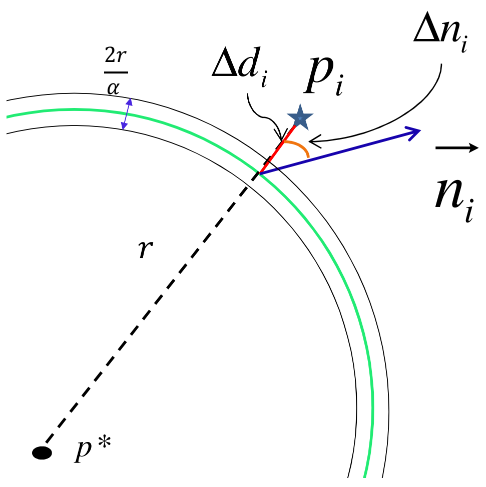
where

$$\mathbf{A} = \begin{pmatrix} 2\tilde{\mathbf{p}}_{1x} & 2\tilde{\mathbf{p}}_{1y} & 1 \\ 2\tilde{\mathbf{p}}_{2x} & 2\tilde{\mathbf{p}}_{2y} & 1 \\ \vdots & \vdots & \ddots \\ 2\tilde{\mathbf{p}}_{|\mathbf{\Psi}|x} & 2\tilde{\mathbf{p}}_{|\mathbf{\Psi}|y} & 1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} ||\tilde{\mathbf{p}}_1||_2^2 \\ ||\tilde{\mathbf{p}}_2||_2^2 \\ \vdots \\ ||\tilde{\mathbf{p}}_{|\mathbf{\Psi}|}||_2^2 \end{pmatrix} \tag{2.2}$$

The center $\tilde{\mathbf{c}}$ of the circle is transformed back in 3D coordinates and considered as the point $\mathbf{p}^*$ belonging to the axis of the cylinder, i.e., $\mathbf{p}^* = \tilde{\mathbf{c}}_x\mathbf{C}_x + \tilde{\mathbf{c}}_y\mathbf{C}_y$.



(a) Cylinder model and its descriptive parameters.



(b) Top view of cylinder.

Figure 2.4. Computation of the orthogonal distance Δd_i (red line) between a point $\mathbf{p}_i$ and the detected cylinder (green curve). Angle Δn_i (orange) between its normal vector and a vector on the detected cylinder surface. $\mathbf{p}_i$ is the point of interest. $\mathbf{p}^*$ is a point on the axis of the detected cylinder. $\vec{\mathbf{n}}_i$ is the normal vector at a given point $\mathbf{p}_i$. $\frac{2r}{\alpha_c}$ is the margin for inlier points.

We iteratively apply these three steps to extract $\mathbf{\Psi}$ points until the maximum number of iterations is reached. At each iteration, the set of inliers (line **4** in Algorithm 1) is updated with the following normal and distance criterion:

$$\mathbf{\Psi} = \left\{ i \middle| (|\Delta d_i| < \frac{r}{\alpha_c})\&(|\Delta n_i| > \beta_c) \right\} \tag{2.3}$$

where Δd_i is the distance of point $\mathbf{p}_i \in \mathbf{P}$ to the detected cylinder and Δn_i is the angle between its normal $\overrightarrow{\mathbf{n}_i}$ and the vector from point $\mathbf{p}_i$ to the axis, as shown in Fig. 2.4(b). And r is the radius of the detected cylinder. The terms Δd_i and Δn_i are computed as follows:

$$\Delta d_i = \| \overrightarrow{\mathbf{p}_i - \mathbf{p}^*} - (\overrightarrow{\mathbf{p}_i - \mathbf{p}^*} \cdot \overrightarrow{\mathbf{a}})\overrightarrow{\mathbf{a}} \| - r \tag{2.4}$$

and

$$\Delta n_i = \frac{\cos\left(\left(\overrightarrow{\mathbf{p}_i - \mathbf{p}^*} - \left(\overrightarrow{\mathbf{p}_i - \mathbf{p}^*} \cdot \overrightarrow{\mathbf{a}} \right) \overrightarrow{\mathbf{a}} \right) \cdot \overrightarrow{\mathbf{n}_i} \right)}{\| \overrightarrow{\mathbf{p}_i - \mathbf{p}^*} - \left(\overrightarrow{\mathbf{p}_i - \mathbf{p}^*} \cdot \overrightarrow{\mathbf{a}} \right) \overrightarrow{\mathbf{a}} \|} \tag{2.5}$$

where $(\cdot)$ is the dot product of two vectors.

Parameters α_c and β_c in Eq. 2.3 are adaptive threshold values for distance and orientation differences respectively. Large values for $\frac{r}{\alpha_c}$ and β_c make the process greedy for merging data points into the fitting process, but small ones make the process converge more slowly. Therefore, $\alpha_c = 50$ and $\beta_c = 0.95$ are good values found empirically and were used for all experiments reported in this chapter. These threshold values are used to update the set of inliers (see Eq. 2.3) automatically at each iteration.

For each iteration, only inlier points are considered as input for cylinder fitting. This process propagates rapidly to points belonging to the cylinder surface, as shown in Fig. 2.2. The major advantage of this method consists of choosing the inliers iteratively based on distance and normal information, so outliers and noise are sequentially removed by using the distance and normal filter in Eq. 2.3. Therefore, the descriptive parameters of a cylinder are estimated by inliers only, which makes the estimates accurate.

Some practical experiments were conducted to choose the number of iterations k used in Algorithm 1. We applied the above process to different models of cylinders. The distance Δd_i between the points and the surface of the detected cylinder was computed after each iteration. The mean square error (MSE) of the distance error was computed as follows:

$$\text{MSE} = \frac{1}{N} \sum_{i \in N} (\Delta d_i)^2 \tag{2.6}$$

The MSE of different models is normalized and plotted in Fig. 2.5. For all of the models with α_c and β_c mentioned above, MSE converges to a small value after a number of iteration smaller than 10. Therefore, k was set to 10 in all of the experiments.

The proposed method is based on hierarchical face clustering (HFC) [AP10], but it outperforms HFC's performance on many aspects:

- Hierarchical clustering is similar to region growing and merges just one point to the cluster at each iteration. This makes the process slow because of high computational cost. While the proposed method rather detects and merges multiple points at each iteration, which speeds up the overall process.
- Hierarchical clustering is dependent on a priority list which is affected easily by noisy data because the error fitting for a sphere or a plane is often smaller than that of a cylinder when the number of points is small. The proposed method avoids these problems by focusing on local data continuously and by using surface normal and distance criteria to remove noisy data and outliers.

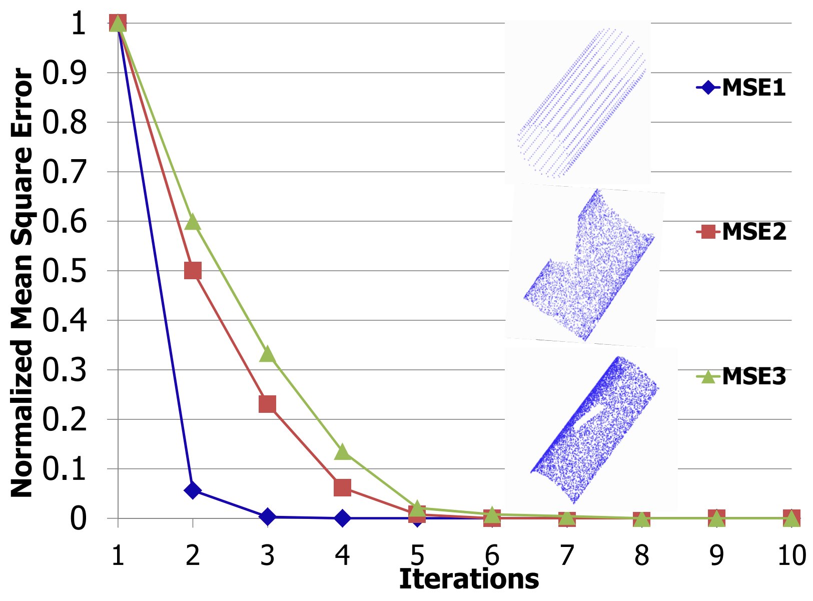


Figure 2.5. Iterations for different models: complete synthetic cylinder (top), cylinder with hole (center) and partial cylinder (bottom). The Mean Square Error (MSE) of the distance error converges to zero after a few iterations smaller than 10.

— As shown in Section 2.5, the proposed method is faster and more reliable than HFC and other methods. The method is able to estimate cylinder parameters accurately, even in the presence of noise and outliers. Fig. 2.2 shows the results of cylinder fitting. After 4 iterations, all points belonging to the cylinder surface are detected with accuracy.

2.4 Multiple cylinders extraction

As mentioned before, manufactured objects are generally composed of multiple cylinders combined with other primitives. Therefore, choosing only points belonging to cylinder surfaces for fitting is not a trivial problem. In this section, a robust method for the simultaneous detection of multiple cylinders is proposed. The diagram of the method is given in Fig. 2.6 and Algorithm 2. Each block is described in the following.

Preprocessing

Normal vector and principal curvatures at each data point are used in the proposed method. Therefore, this information is computed at a preprocessing step. Hoppe et al. [HDD+92] have proposed to use classical principal component analysis (PCA) [Jol86], for normal vector estimation (Appendix .2). The same approach is selected in this chapter because of its simplicity and efficiency.

Algorithm 2: Multiple cylinders extraction from a point cloud.

```
Data: Point cloud; Normal vectors and Potential points (Section 2.4)
Result: Parameters of cylinders
index:=0; $m$:=# Potential points; $\mathbf{X}$:=$\varnothing$;
while $index \leq m$ do
    1. Find neighorhood for a given initial seed point and then fit a cylinder using Section
    2.3, Algorithm 1;
    2. Validation by first phase Eq. 2.7, Section 2.4.1;
    3. if true then
        4. Validation by second phase by computing $SC$ Eq. 2.9 and Algorithm 3;
        5. if $SC < c_{sc}$ then
            $\mathbf{X} = \mathbf{X} \cup newcylinder$ ;
        end
    end
    6. index:=index+1;
end
Apply mean-shift clustering to $\mathbf{X}$;
```

As shown in Fig. 2.6, the proposed method applies an iterative fitting procedure (Algorithm 1) to each potential point that could belong to a cylinder surface. These potential points are extracted by using principal curvature information. A point is considered as belonging to a cylinder if the ratio between the maximum principal curvature $\mathbf{k}_2$ and the minimum principal curvature $\mathbf{k}_1$ is larger than 100 ($\mathbf{k}_2 > 100\mathbf{k}_1$). To the best of our knowledge, Taubin's method [Tau95] is referred to as one of the most popular methods for principal curvature estimation [MSR07, HS03] ((Appendix .3))

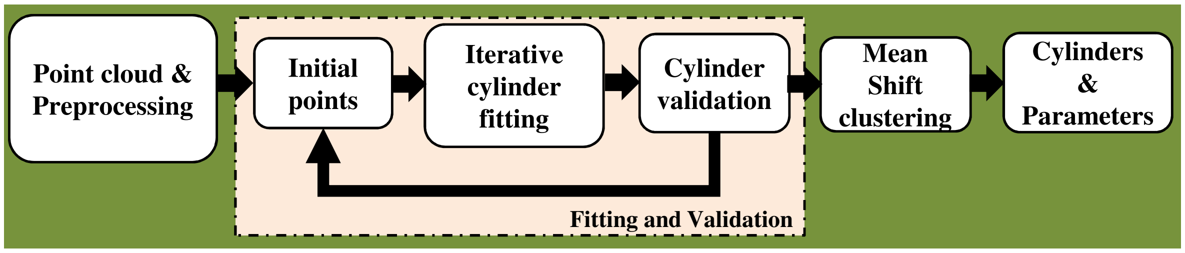


Figure 2.6. Diagram for multiple cylinder extraction.

2.4.1 Fitting and Validation

Following the selection of potential points, the next step consists oin fitting a cylinder at each potential point and checking the fitting validity. For each potential point, a neighborhood is chosen and the cylinder fitting procedure, which is described in Section 2.3, Algorithm 1, is applied to this neighborhood to estimate the cylinder parameters corresponding to this potential point. As mentioned earlier, the fitting process is able to propagate the primitive to all points that belong to the cylinder.

The result of the fitting process depends on the initial seed point and the propagation process. Because of acquisition noise or point cloud structure, some detected cylinders may not converge to a good cylinder and the detection process could be stuck in a geometry trap as shown in Fig. 2.7(e). To eliminate such cases, a novel method for validating the detected cylinder is proposed. The validation algorithm consists of two phases: (i) a first phase checks the geometric consistency of the detected cylinder; (ii) a second phase checks the local structure around it.

First phase of the validation step

The *first phase* of the validation step verifies the geometric consistency of the detected cylinder. Based on the experiment reported in Fig. 2.5, the estimation of the parameters of good cylinders converge after a few iterations only. Therefore, the convergence property is used to verify the geometric consistency of a cylinder. A cylinder primitive is considered as reliable if it satisfies the following conditions:

$$\begin{cases} \|r_k - r_{k-1}\| < \gamma \\ \dfrac{(\overrightarrow{\mathbf{a}_k} \cdot \overrightarrow{\mathbf{a}_{k-1}})}{(\overrightarrow{\mathbf{a}_{k-1}} \cdot \overrightarrow{\mathbf{a}_{k-2}})} > \theta \end{cases} \tag{2.7}$$

where the number of iterations k is set to 10 mentioned in Algorithm 1.

The first inequality evaluates the convergence of the radius r, and the second one assesses the robustness of axis orientation $\overrightarrow{\mathbf{a}}$. In our experiments $\gamma = \frac{1}{100} \sum_{i=0}^{k} \frac{r_i}{10}$ is set as 1% of the average radius of the detected cylinders at all iterations and $\theta = 0.99$. After the first phase, only the detected cylinders satisfying Eq. 2.7 are fed to the second phase for further investigation.

Second phase of the validation step

As the first phase may still lead to unreliable cylinders because the fitting process is trapped in a local minimum in geometry, the *second phase* is executed. The main idea of this second phase is to explore the structure around the detected cylinder to assess its quality.

— The *Virtual circle* $\{\tilde{\mathbf{c}}_{vir}; r_{vir}\}$ is generated by projecting the detected cylinder on a plane normal to the axis direction $\overrightarrow{\mathbf{a}}$ and through origin O.

— All points near the surface of the detected cylinder are also projected on the same plane and generate a circular shape which is detected and called the *estimated circle* $\{\tilde{\mathbf{c}}_{est}; r_{est}\}$.

— The problem is thus transformed into one of detecting a circle in the plane. The detected cylinder is reliable if these two circles coincide closely (Fig. 2.7(d)).

The second phase of the validation step is summarized in Algorithm 3 and explained in more detail as follows. First, the detected cylinder is projected on a plane, which has normal $\overrightarrow{\mathbf{a}}$ and passes through the origin O, to generate a circle called the *virtual circle* (green in Fig. 2.7(b)-Fig. 2.7(f)). Then the data points within the $\pm\frac{5*r_{10}}{100}$ margin (Fig. 2.7(b)-Fig. 2.7(f)) from the surface of the detected cylinder are also projected on the same plane and distributed as a circle. These projected points are stored into a $S \times S$ accumulator-grid that is then normalized to produce a grey-level image (Fig. 2.7(b)-Fig. 2.7(f)). The size of the accumulator cell is automatically set to $\frac{r_{10}}{100}$, which is chosen to balance between accuracy, complexity and computational efficiency. The resolution of the grey-level image is thus 210×210 and is computed as follows (Fig. 2.7(f)):

$$S = 2 * \frac{r_{10} + 5 * \frac{r_{10}}{100}}{\frac{r_{10}}{100}} = \frac{2 * 100 * \frac{r_{10}}{100} + 2 * 5 * \frac{r_{10}}{100}}{\frac{r_{10}}{100}} = 210 \tag{2.8}$$

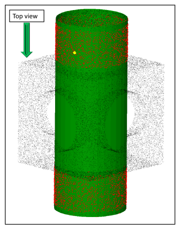
A binary image is then generated using Otsu's method [Ots79]. The problem is reduced to detecting a circle in this binary image, which is called the *estimated circle* (Fig. 2.7(c)-Fig. 2.7(g)). Finally, the virtual circle of the detected cylinder and the estimated circle extracted from the binary image are compared. Based on our experiments, RANSAC is a good method for detecting circles in a binary image [LGF07, LZH+13]. The difference between the estimated circle and the virtual circle is assessed by a parameter SC that is defined as the sum of the center deviation and radius deviation in pixels.

$$SC = \frac{\|\tilde{\mathbf{c}}_{vir} - \tilde{\mathbf{c}}_{est}\| + |r_{vir} - r_{est}|}{r_{vir}} * 100 \tag{2.9}$$

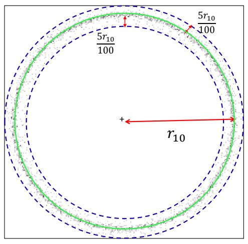
where $\tilde{\mathbf{c}}_{vir}$ and r_{vir} are the center and radius of the virtual circle, and $\tilde{\mathbf{c}}_{est}$ and r_{est} are the center and radius of the estimated circle in the binary image (red in Fig. 2.7(c)-Fig. 2.7(g)). Only the cylinders that satisfy the condition ($SC < c_{sc}$) are considered as reliable primitives (Fig.

Algorithm 3: Second phase of the validation step.

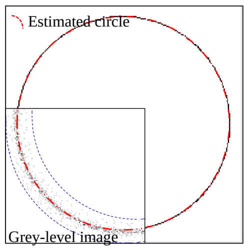
```
Data: Point cloud; the descriptive parameters of the detected cylinder that passed the first
      phase.
Result: $SC$ assessing the difference between the estimated circle and the virtual circle.

1. Project the detected cylinder on a plane with normal $\vec{\mathbf{a}}$ and passing through origin O and
generate the virtual circle $\{\tilde{\mathbf{c}}_{vir}; r_{vir}\}$;
2. Project the data points within $\pm\frac{5*r_{10}}{100}$ margin from the surface of the detected cylinder on
the same plane;
3. Store and count the projected points by using an accumulator grid with cell size $\frac{r_{10}}{100}$;
4. Normalize accumulator to create a grey-level image;
5. Generate a binary image from the grey-level image [Ots79];
6. Detect an estimated circle $\{\tilde{\mathbf{c}}_{est}; r_{est}\}$ using RANSAC;
7. Compute $SC$ using Eq. 2.9;
```



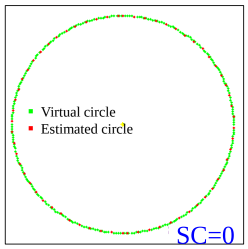
(a) Valid detected cylinder. The yellow dot is the initial point.



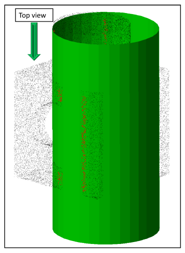
(b) Grey level image of the projection (top view). Dashed blue circles are $\frac{5r_{10}}{100}$ margins.



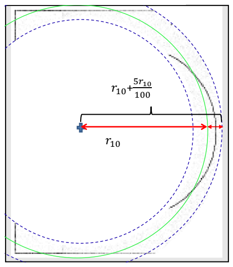
(c) A part of the grey-level image and its binary image of the projection.



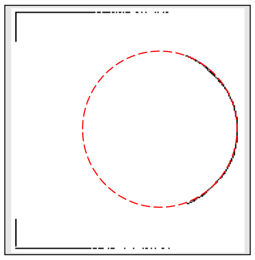
(d) Green dots are re-sampled from the virtual circle, red dots are re-sampled from the estimated circle.



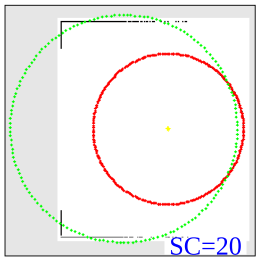
(e) Non-valid detected cylinder.



(f) The detected cylinder creates a virtual circle (green).



(g) The dashed red circle is an estimated circle in the binary image.



(h) The virtual circle and estimated circle do not coincide which results in a large value of the score.

Figure 2.7. Validation of cylinder detection. With a good cylinder, both the virtual circle and estimated circle coincide. On the other hand, this condition is not satisfied even if the detected cylinder passes the first validation step. Therefore, it is considered as an unreliable cylinder and rejected.

2.7(d)). Otherwise, they are rejected (Fig. 2.7(h)). c_{sc} is chosen according to the level of noise in the 3D point cloud.

After the two validation phases, reliable cylinders are identified and their descriptive parameters are stored into **X** list.

2.4.2 Mean shift clustering on the list of parameters of reliable cylinders

Once the above steps are executed, descriptive parameters for multiple cylinders are stored into **X** list. Potential points belonging to the same cylinder should have the same set of descriptive parameters. Fig. 2.8(a) shows the histogram of radius values of detected cylinders. The next step consists of clustering points sharing the same set of parameters and leading to the detection of multiple cylinders. A non-parametric clustering approach, mean-shift clustering [CM02], is used for this purpose. Contrarily to k-means clustering methods, mean-shift clustering does not require the number of clusters to be chosen a *priori*.

Mean-Shift clustering [CM02] involves grouping data into a set of clusters: Given n_c data points $\mathbf{X} = \{\mathbf{x}_i | i = 1, ...n_c\} \subset \mathbb{R}^d$ sampled from a density $f(\mathbf{x})$. The multivariate kernel density estimator with a symmetric kernel, $K(\mathbf{x})$, is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{n_c h^d} \sum_{i=1}^{n_c} K(\frac{\mathbf{x} - \mathbf{x}_i}{h}) \tag{2.10}$$

where h is called the bandwidth parameter. The symmetric kernel is defined as

$$K(\mathbf{x}) = qk(||\mathbf{x}^2||) \tag{2.11}$$

where $k(\mathbf{x})$ is called the profile function and q represents a normalizing constant:

$$\int_{\mathbb{R}^d} k(||\mathbf{x}^2||) d\mathbf{x} = \frac{1}{q} \tag{2.12}$$

$\hat{f}(\mathbf{x})$ has local maxima at the cluster centers. The simplest method to find the local maxima is to use a gradient-ascent technique. Taking the gradient of the density estimator in Eq. 2.10 and performing further algebraic manipulations yields:

$$\nabla \hat{f}(\mathbf{x}) = \frac{2q}{n_c h^{d+2}} \sum_{i=1}^{n_c} g\left(\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\|^2\right) \left[\frac{\sum_{i=1}^{n_c} \mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{x}\right] \tag{2.13}$$

where $g(\mathbf{x}) = -k'(\mathbf{x})$ denotes the derivative of the selected kernel profile. The first term $g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)$ is proportional to the density estimate at $\mathbf{x}$. The second term is called the mean shift vector $\mathbf{m}$,

$$m(\mathbf{x}) = \frac{\sum_{i=1}^{n_c} \mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{x} \tag{2.14}$$

The mean-shift vector points toward the direction of the maximum increase in density and is proportional to the density gradient estimate at point X. Therefore, the mean-shift procedure for a given point $\mathbf{x}_i$ is as follows:

1. Compute the mean shift vector $m(\mathbf{x}_i^t)$.
2. Translate the density estimation window:
$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + m(\mathbf{x}_i^t).$$
3. Iterate steps 1 and 2 until convergence.

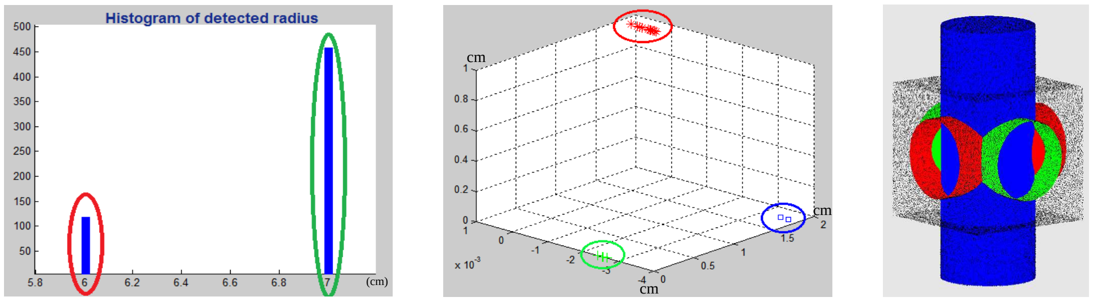
In this chapter, we apply mean shift clustering in the cylinder parameters space $\mathbf{X} \in \mathbb{R}^7$. Each detected cylinder is considered as a data point $\mathbf{X} = \{\mathbf{x}_i = (\overrightarrow{\mathbf{a}_i}, \mathbf{p}_i^*, r_i)\}$. In this problem, mean shift clustering with separable Gaussian kernels is used:

$$\hat{f}(\mathbf{x}) = \frac{1}{n_c h_1 h_2 h_3} \sum_{i=1}^{n_c} K_1(\frac{\overrightarrow{\mathbf{a}} - \overrightarrow{\mathbf{a}_i}}{h_1}) K_2(\frac{\mathbf{p}^* - \mathbf{p}_i{}^*}{h_2}) K_3(\frac{r - r_i}{h_3}) \tag{2.15}$$

The mean shift vector m is currently defined as:

$$m(\mathbf{x}) = \frac{\sum_{i=1}^{n_c} \mathbf{x}_i g\left(\|\frac{\overrightarrow{\mathbf{a}} - \overrightarrow{\mathbf{a}_i}}{h_1}\|^2\right) g\left(\|\frac{\mathbf{p}^* - \mathbf{p}_i{}^*}{h_2}\|^2\right) g\left(\|\frac{r - r_i}{h_3}\|^2\right)}{g\left(\|\frac{\overrightarrow{\mathbf{a}} - \overrightarrow{\mathbf{a}_i}}{h_1}\|^2\right) g\left(\|\frac{\mathbf{p}^* - \mathbf{p}_i{}^*}{h_2}\|^2\right) g\left(\|\frac{r - r_i}{h_3}\|^2\right)} - \mathbf{x} \tag{2.16}$$

The only parameters that must be defined are bandwidth values $\mathbf{h}_c = \{h_1, h_2, h_3\}$. h_1 is the bandwidth for the angle difference of axis orientation $\overrightarrow{\mathbf{a}}$, h_2 is the bandwidth for the points $\mathbf{p}^*$ on the axis, and h_3 is the bandwidth for the radius. In all experiments reported in this chapter, the bandwidth is set automatically $\mathbf{h}_c = \{0.01, \sigma, \frac{1}{100} \sum_{i=1}^{n_c} \frac{r_i}{n_c}\}$ where σ is the average distance between point cloud data. The number of clusters is the number of cylinders and the cluster center is the descriptive parameter of cylinders in the point cloud. Fig. 2.8 shows the result of using mean shift clustering for cylinder extraction for the block model in Fig. 2.8(c).



(a) Histogram of detected cylinders. Two clusters are visible.

(b) Three clusters of points on axis space $\mathbf{p}^*$ correspond to three cylinders.

(c) Final detected cylinders.

Figure 2.8. Mean shift clustering and final results. (a) In radius space, there are two clusters since the block model has three cylinders, but two having the same radius. (b) There are three clusters in the $\mathbf{p}^*$ space. (c) The final results after mean shift clustering: three cylinders are detected correctly.

2.5 Experimental results

2.5.1 Performance of the single cylinder fitting approach compared to other methods

To illustrate the robustness of the proposed method for fitting a single cylinder (Section 2.3) and estimating its descriptive parameters, the results were compared to those of three other methods: Gaussian sphere-based [CG01], Hierarchical face clustering (HFC) [AP10] and RANSAC-based methods [SWK07]. Three different cylinders with synthetic, partial and missing data shown in Fig. 2.5 are used as datasets. The results are compared based on the following criterion:

— The difference between the estimated radius r and the ground truth radius $\hat{r}$.

$$\mathrm{MSE}_r = \frac{1}{3}\sum_{i=1}^{3}(\hat{r}_i - r_i)^2 \tag{2.17}$$

— The angle difference ϑ (°) between the estimated axis orientation and ground truth axis orientation.

$$\mathrm{MSE}_\vartheta = \frac{1}{3}\sum_{i=1}^{3}(\vartheta_i)^2 \tag{2.18}$$

— The distance is computed from the points to the detected cylinder.

$$\mathrm{MSE}_e = \frac{1}{3}\sum_{i=1}^{3}\mathrm{MSE}_i \tag{2.19}$$

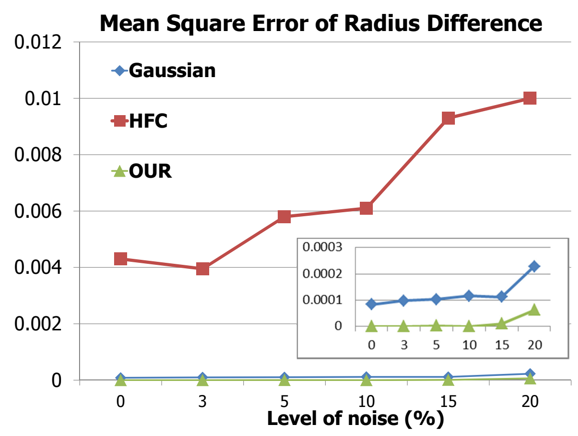
where MSE_i is computed with Eq.2.6.

Table 2.1 lists the value of the criterion for these methods. The Gaussian sphere and Hierarchical face clustering perform well but achieve less accurate results compared to the proposed method. As anticipated, the results obtained by RANSAC-based methods do not perform very well in the current example as indicated by the large MSE values. Our iterative approach helps to detect every point belonging to a given cylinder, and achieves a performance very close to ground truth. It illustrates that an iterative method based on distance and normal filtering works better than the greedy single fitting procedure used by both the Gaussian sphere and HFC methods. It is worth mentioning that the results reported in Table 2.1 are averaged values over 10 runs for all of the methods.

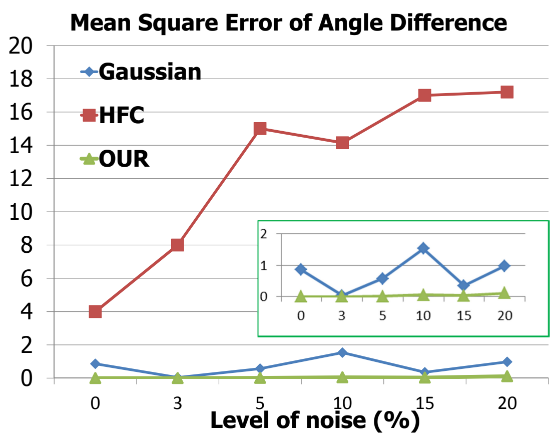
2.5.2 Experiments on multiple cylinder extraction

In this section, our algorithm (OUR) [TCL15a] for multiple cylinder extraction is tested. The result is compared to the two other methods: Gaussian sphere-based (GAUSS) [CG01], Hierarchical face clustering (HFC) [AP10]. The RANSAC-based method [SWK07] is not considered because it is hard to tune a set of parameters for the complex models used in the experiments.

In a first experiment, a synthetic block model resampled to 12k points with three cylinders is used for testing. The reason for resampling the data is to compare our method with the HFC method which requires high computation time and memory. Then Gaussian noise with zero mean and standard deviations of 0-20 % of the average distance between points was added to the point cloud data. Three methods were used to extract cylinders and estimate their parameters. The results are compared with ground-truth values on the basis of the mean square error of radius difference MSE_r (Eq. 2.17) and angle difference MSE_ϑ (Eq. 2.18). Observing Fig. 2.9(a) and Fig. 2.9(b), the HFC method shows larger errors on detected radius and axis direction. The Gaussian sphere method achieves better results than HFC but still results in an averaged MSE 20 times larger than the proposed method.



(a) MSE of detected radius with different levels of noise.



(b) MSE of detected axis direction with different levels of noise.

Figure 2.9. Mean square error of radius and angle difference.

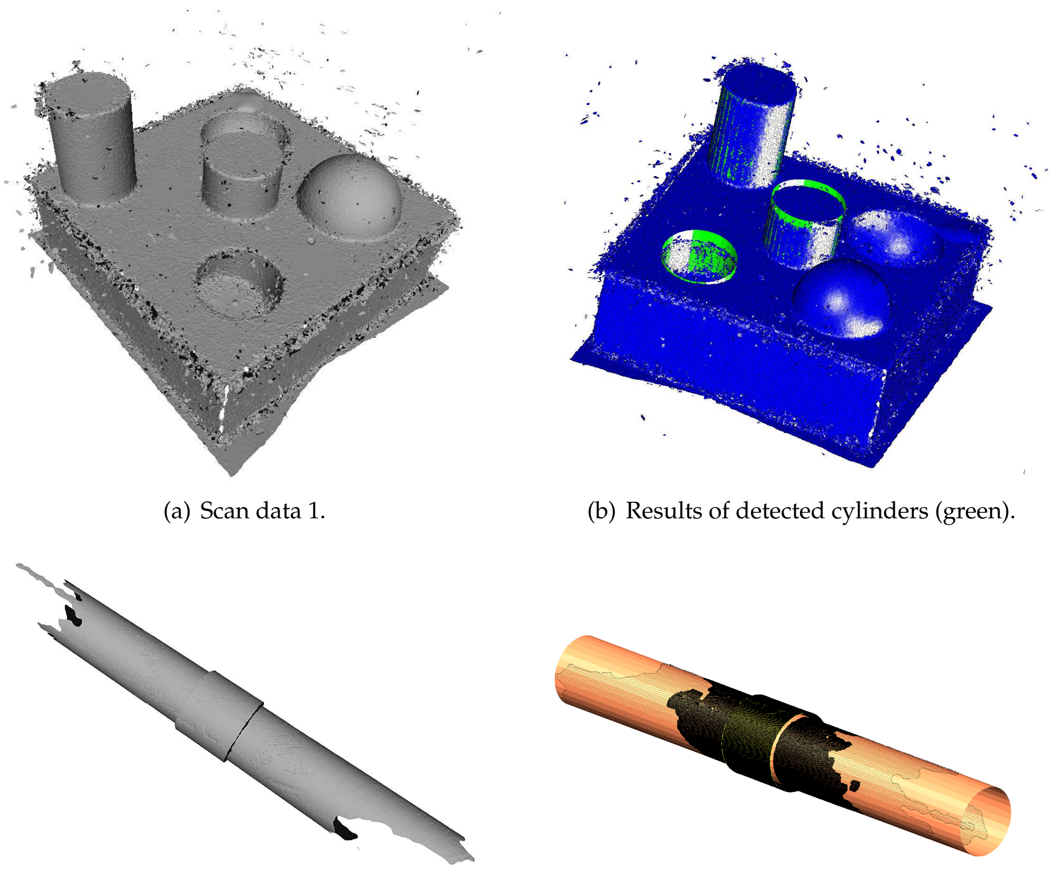
We tested the proposed method, GAUSS method and HFC method for different levels of noise and outliers for the resampled block model. First, Gaussian noise with zero mean and standard deviation between 0-20 % of the average distance between points was added to the point cloud. In addition, 2% of the points were added as outliers with a magnitude 0-10 % of the average distance between points. Fig. 2.11 shows the detected cylinders for the block model. Table 2.2 lists the set of threshold values used for the dataset and averaged MSE of the estimated radii and angle differences of the three cylinders. These results were averaged over 10 runs. Despite noise and the presence of outliers, our method is robust for the detection of cylinders and the estimation of the radius and axis direction compared to the two other methods. It demonstrates that the proposed method using two validation steps helps to reduce the effect of noise and outliers. The proposed method was tested on datasets captured in the lab by the Creaform GO!SCAN hand-held 3D scanner. Fig. 2.10 shows the real scan data, which contains many outliers and holes. The method still achieves good results without using any preprocessing step (e.g. denoising or hole-filling).

Table 2.1. Comparison of the proposed method with Gaussian sphere, Hierarchical clustering (HFC) and RANSAC based methods.

Method	MSE_r	MSE_ϑ	MSE_e
RANSAC [SWK07]	1.02E-02	0.01	1.08E-02
HFC [AP10]	4.56E-05	2.5E-04	7.6E-04
Gaussian [CG01]	5.87E-05	5.37E-03	6.5E-03
Our method [TCL15a]	4.06E-05	2.3E-04	5.56E-04

Table 2.2. Threshold values used for this experiment and MSE for the three methods.

Noise-Outlier	α_c	β_c (rad)	c_{sc}	OUR		GAUSS		HFC	
				MSE_r	MSE_a	MSE_r	MSE_a	MSE_r	MSE_a
0%-0%	50	0.95	0	1.01E-08	1.03E-04	3.68E-05	0.0622	4.43E-03	5.9203
5%-5%	50	0.9	1	5.22E-07	0.0280	5.68E-05	0.0959	5.10E-03	10.271
10%-5%	50	0.9	1	4.90E-06	0.0301	5.05E-04	0.2431	9.31E-03	13.069
15%-10%	50	0.85	2	1.62E-05	0.0304	7.94E-04	0.5857	6.94E-03	15.033
20%-10%	50	0.8	3	2.64E-05	0.0314	8.11E-03	0.8882	2.11E-02	18.062



(a) Scan data 1.

(b) Results of detected cylinders (green).

(c) Scan data 2 of a pipeline.

(d) Results of detected cylinders.

Figure 2.10. Results on real scanned data captured by Creaform GO!SCAN 3D sensor (http://www.goscan3d.com) ($\alpha_c = 50; \beta_c = 0.9; c_{sc} = 2$).

Fig. 2.12 shows multiple cylinders detected from CAD models; all cylinders in the models are detected successfully. We have tested our method on real point cloud datasets used in the literature [SWK07, LWC+11]. Fig. 2.13 shows the results of these experiments. These results show that our method is successful in extracting most of complete and partial cylinders in point clouds of complex models. Some small and thin cylinders have not been detected because there is not enough data for fitting.

2.5.3 Computation time

The performance of the method was evaluated in terms of computation time. It is compared to the Gaussian sphere-based method (GAUSS) [CG01] and Hierarchical clustering (HFC) [AP10]. These methods were implemented in MATLAB on a 3.2 GHz Intel Core i7 platform. As mentioned earlier, the HFC method has high complexity with respect to time and memory, so the test models (block and joint) were downsampled before running the experiments. Fig. 2.14 shows the computational time for three methods. The proposed method is

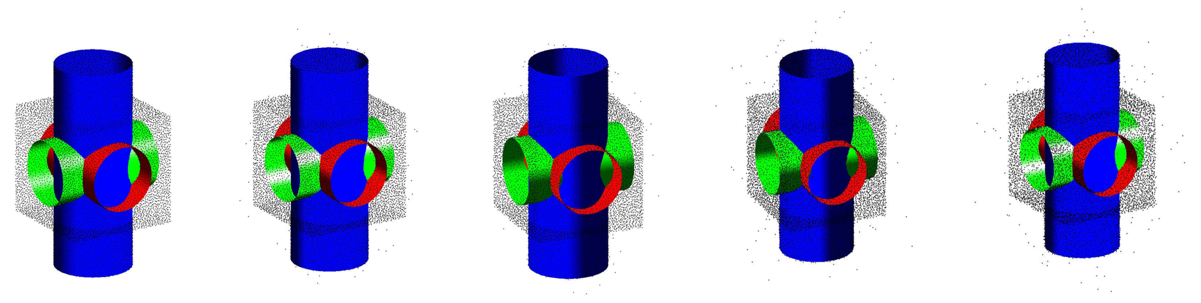

(a) Data without noise and outliers (b) 5 % noise and 5 % outliers (c) 10 % noise and 5 % outliers (d) 15 % noise and 10 % outliers (e) 20 % noise and 10 % outliers

Figure 2.11. Cylinders detected with different levels of noise and outliers for the block model. Table 2.2 lists the threshold values used in the experiment and MSE compared to other methods.

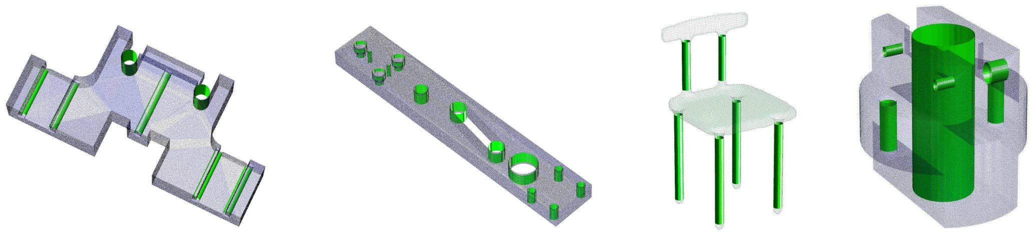

(a) Bracket (b) Linkage arm (c) Chair (d) Socket

Figure 2.12. Cylinder extraction from point cloud of CAD models ($\alpha_c = 50; \beta_c = 0.95; c_{sc} = 0$).

slightly slower than the Gaussian sphere method, but much faster than the HFC method, because the proposed method merges multiple points at each iteration, but HFC method only merges one point at a time to the cluster.

Table 2.3 shows the computation time of the method averaged over 5 runs. The bottleneck of the algorithm is the fitting and validation steps. Computation time depends on how many initial points were extracted at the preprocessing step and the number of data points in the point cloud (see Table 2.3). Therefore, the processing time should drop if a resampling method is applied on the initial points. Furthermore, the whole process is currently implemented without parallel processing. The proposed framework is eligible for a parallel implementation that could improve the performance significantly.

2.5.4 Applications

The proposed algorithm could be used for various applications such as inspection, reverse engineering, segmentation, registration, etc. The first application is defect detection in Fig. 2.15. The surface of pipes often contains defects originating from many sources such as the manufacturing process or chemical oxidation. After extracting the cylinders and their parameters, some defect areas are detected easily by computing the distance from data points

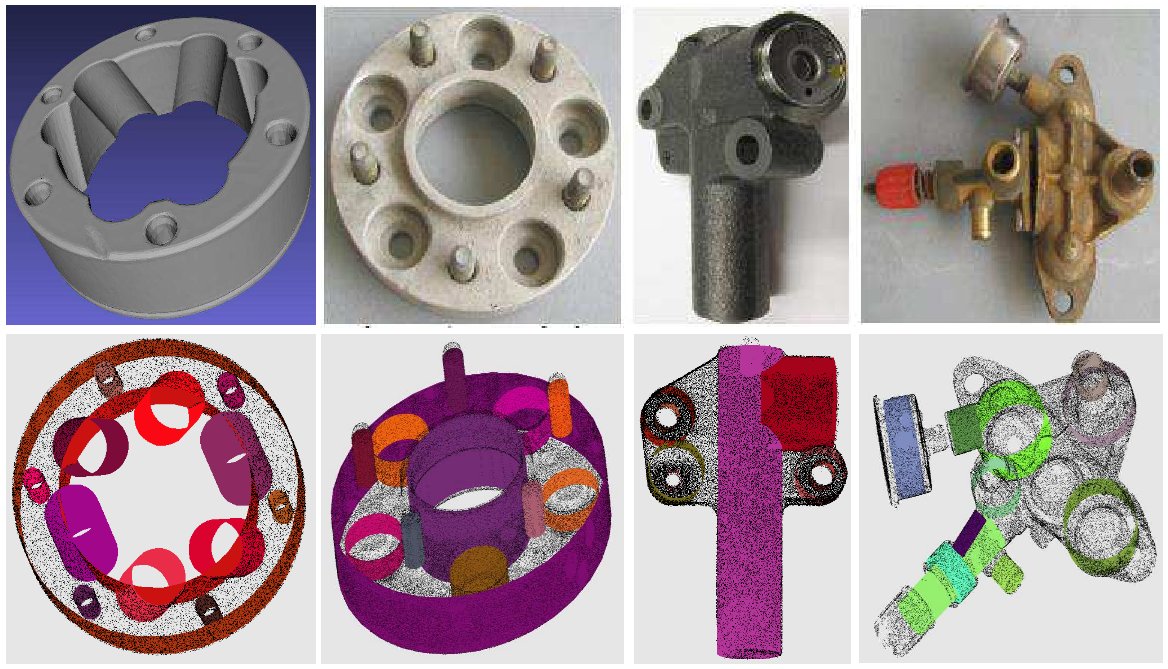


Figure 2.13. Results on real point cloud data: rolling stage [SWK07], Data 1, Data 2 and Data 3 [LWC+11]. The first column shows the original models. Point clouds are shown in the second column. The third column shows the detected cylinders. Table 2.3 shows the number of detected cylinders. The thin and small cylinders are not detected ($\alpha_c = 50; \beta_c = 0.85; c_{sc} = 2$).

Table 2.3. Computation time of our method for different models. FV means for fitting and validation step. N_c is the number of detected cylinders.

Model Name	#Points	#Initial	N_C	Normal (s)	Curvature (s)	FV (mins)
Joint Fig. 2.1	50 000	2172	3	3.52	12.12	2.79
Block Fig. 2.8	50 000	1971	3	4.25	14.28	2.85
Rolling stage Fig. 2.13	50 000	3914	14	2.36	10.16	3.41
Data 1 Fig. 2.13	105 802	7274	12	10.38	24.89	7.21
Data 2 Fig. 2.13	84 130	2612	8	6.01	21.88	3.57
Data 3 Fig. 2.13	141 267	3767	10	7.62	27.45	4.32
Scan data 1 Fig. 2.10(a)	289 393	7339	3	12.95	30.74	15.71
Scan data 2 Fig. 2.10(c)	181 504	7930	2	22.97	50.52	10.4

to the surface of the detected cylinder, as shown in Fig. 2.15.

One more promising result of the proposed method is to extract pipelines from large datasets of industrial plants, as shown in Fig. 2.1. This kind of data contains significant noise and many outliers, as well as cylinders making adjoints. In the future we plan to implement preprocessing steps to split the point cloud into smaller data sets which would improve the performance of cylinder extraction that could then be merged into complete cylinders. These detected cylinders could be the input for pipeline reconstruction problems.

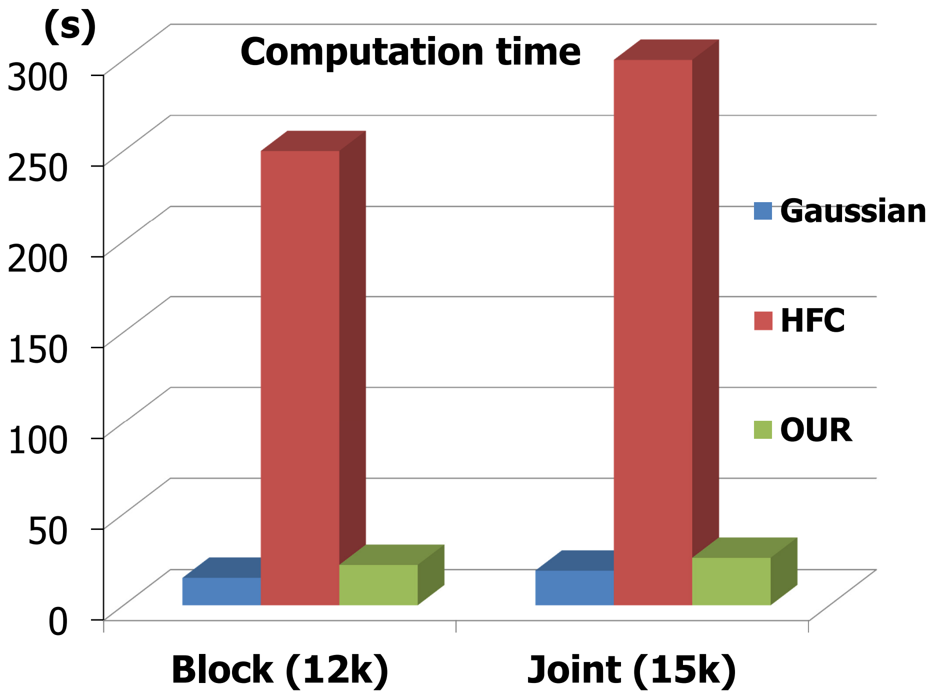


Figure 2.14. Time comparison with other methods. Block and Joint model are resampled to 12k and 15k points respectively.

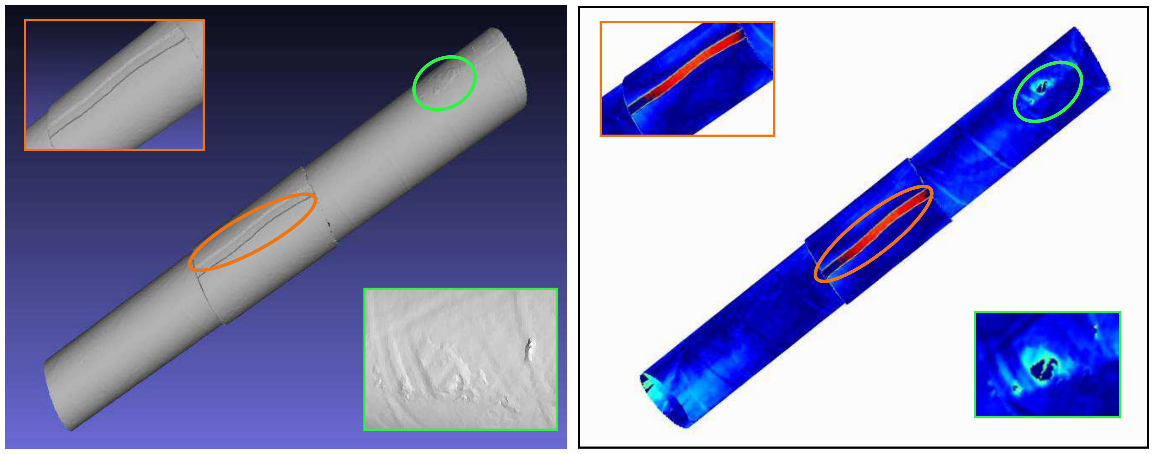


Figure 2.15. Defects are detected from the pipe surface after extracting cylinders and their parameters.

2.6 Conclusion

We have proposed a novel algorithm for extracting cylinders and estimating their parameters from scanned 3D point cloud data. A novel validation method is also proposed to assign a confidence score to the detected cylinders. The algorithm is tested on various complex models and different levels of noise and outliers. The quality and robustness of the results suggests that the algorithm can be used in several applications such as reverse engineering and quality control, modeling, etc.

The future work will be to apply our algorithm to handle large data from industrial plants or urban area. Moreover, this framework could also be extended to other types of primitives such as spheres, cones and toruses. We also plan to implement the algorithm in C++. The framework proposed in this chapter is extended to sphere extraction in the next chapter.

Chapter 3

Sphere Extraction

3.1 Introduction

Spheres are popular primitives found in manufactured objects and even in biochemistry models, as shown in Fig. 3.1. Moreover, spheres have many applications such as RoboCup games [DSCJRN14], reverse engineering [VMC97, BV04], medical imaging [vdGVBV02], etc. Especially, spherical markers are used extensively for monocular or RGB-D camera and laser scanner calibration [AD03, WZC11, WSZZ14, RH14, SBMM15] in which robust sphere estimation is necessary to achieve good results. Many metrology and engineering applications (reverse engineering, part-to-CAD analysis) also require that spheres be extracted automatically and reliably from meshes or point clouds. However sphere extraction from 3D data has not yet been investigated as thoroughly as plane [HHRB12, DG10, BELN11] and cylinder [LZH$^+$13, TCL15a] extraction. There is thus a need for robust and computationally efficient approaches for sphere extraction. More specifically, methods have been proposed for sphere fitting and extraction from 3D point clouds [WSZZ14, AFS06, SWK07, CVC14, ANC13]. However, validation methods for assessing the validity and reliability of extracted spheres remain scarce.

Among the above approaches, hierarchical clustering [AFS06, AP10] cannot be applied for sphere detection only, because all primitives (plane, cylinder and sphere) must be processed simultaneously to create a global clustering priority. Other methods such as [WSZZ14, SWK07] propose to detect spheres using random sample consensus (RANSAC) with a minimal number of sample points [FB81]. However this extraction is a greedy and sequential process in which two points and their normal vectors are used for sphere fitting and need some global predefined values. The result depends on the probability of picking optimal samples, especially with noisy data and outliers.

In addition to RANSAC-based techniques, some approaches [CVC14, ANC13] are based on the Hough transform [DH72]. The Hough transform applies a clustering process to the sam-

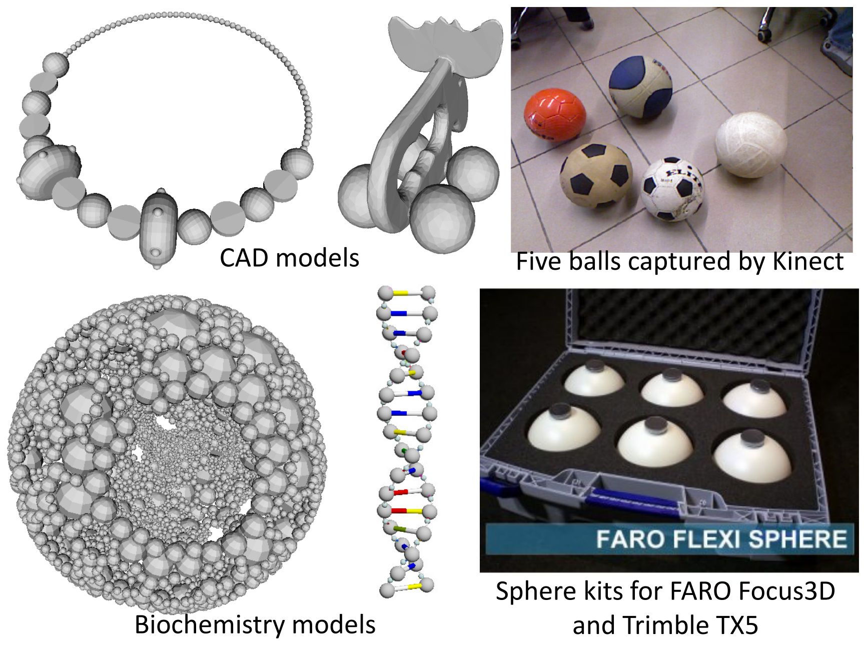

CAD models
Five balls captured by Kinect
FARO FLEXI SPHERE
Biochemistry models
Sphere kits for FARO Focus3D and Trimble TX5

Figure 3.1. Spheres exist in different types of models such as CAD, games, biochemistry and commercial sphere kits for FARO and TRIMBLE scanners.

ples in the parameter space and then finds cluster centers. The quantization of the parameter space, memory requirement and computation time are the main challenges of Hough transform, especially for high dimensional models. Hence several priors such as the interval of radius values or the total number of spheres in the model must be known in advance or set by the user for search space reduction- a nontrivial task for complex and large models.

In this chapter, we propose an efficient method for sphere extraction from unorganized point clouds. The method is comprised of two passes: one for iterative sphere fitting with robust sphere validation and another for sphere parameters computation by implementing an efficient sample strategy combined to a clustering step. Four main contributions of the approach are summarized as follows:

— Sphere parameters are estimated accurately by two passes of the proposed method despite the effect of noise and outliers.

— Currently, previous methods only extract spheres based on error fitting criteria, while in this chapter, a novel shape-based validation method is proposed to evaluate the quality of detected spheres, which helps to eliminate spherical shapes such as ellipsoids.

— An efficient sampling strategy is also proposed so that sphere extraction related to the number of spheres rather than to the number of points in the point cloud.

— Combining the above steps, the proposed framework outperforms previous approaches with less complexity and computational load.

The chapter is organized as follows. Previous work is summarized in Section 3.2. The proposed algorithm is presented in Section 3.3-3.4. Results and discussion are covered in Section 3.5, and Section 3.6 draws some conclusions on the proposed method.

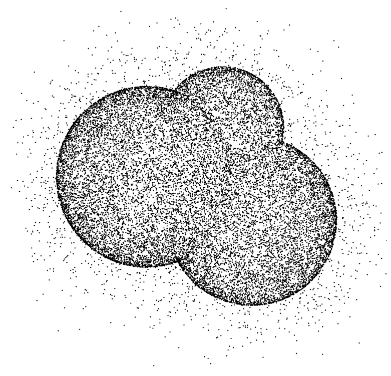
3.2 Related Work

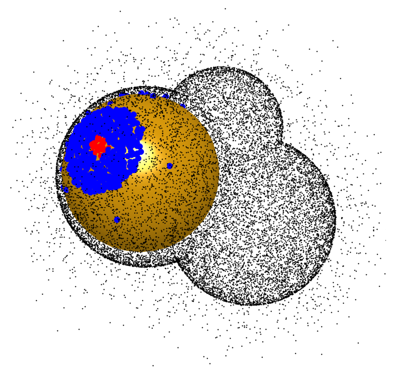
A sphere is simply represented by two parameters: a center **c** and a radius r. Many studies have been conducted on sphere fitting to estimate these parameters and on sphere detection in point clouds. Methods related to the one proposed in this chapter are reviewed in this section. Their advantages and drawbacks are discussed briefly.

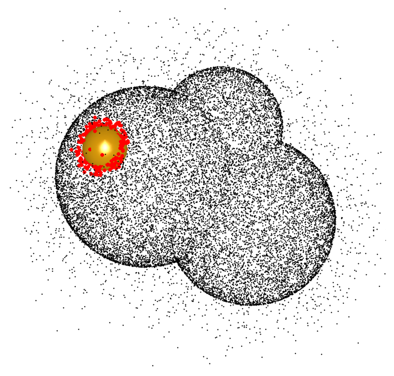
In general, there are two common methods for sphere fitting: algebraic [Pra87] and geometric techniques [For89, GGS94]. These methods are applied after a prior segmentation, a problem that is often nontrivial. The methods are different in the manner in which they compute and minimize the distances of data points to the surface of the fitted primitive. An algebraic representation is expressed by an implicit function $f(x) = 0$ for which $f(x)$ is a polynomial function. Algebraic fitting [Pra87] attempts to minimize the sum of algebraic distance errors and the solution is usually the result of solving an overdetermined system of equations. Attene et al. [AFS06] proposed hierarchical clustering for plane, cylinder and sphere segmentation from a triangle mesh. However, this approach requires the desired number of clusters and a mesh without outliers as an input. The authors extended the algorithm to point clouds [AP10], but the global clustering priority is affected significantly by noise and outliers (Fig. 3.2(b)-(c)) leading to invalid label assignment. Therefore, a post-processing step is required to reassign data points to the correct primitive.

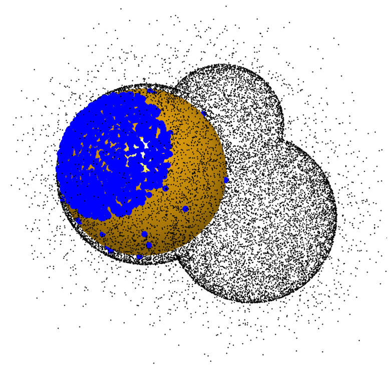
Geometric fitting minimizes the sum of orthogonal distance errors from points to the fitted primitive. Geometric fitting [For89, GGS94, LMM98, FCSW09] is an iterative method for which good initial parameters could be possibly obtained from algebraic fitting. Recently, moving least-squares (MLS) techniques have been used for feature preserving surface reconstruction [Lev98, OGG09]. However, the algebraic approach is widely used for primitive fitting because of its convenience and efficiency [Pra87].

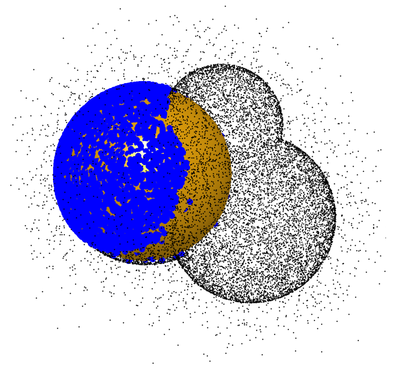
Sphere extraction has been recently investigated for unstructured point clouds using RANSAC or the Hough transform. RANSAC [FB81, BF81] fits a primitive with the minimum number of random samples. For example, two data points with their surface normals are needed for sphere fitting [SWK07]. It is thus fast and efficient. However, several global threshold values are not easy to set for different models, especially for noisy data and outliers. The results of the algorithm depend on the initial selection of points, which must thus be done carefully. One more problem is that primitive extraction in [SWK07] is a sequential process in which detected primitives and their associated points are removed immediately from the


(a) Three-sphere model with noise and outliers.


(b) Result of traditional algebraic fitting after 1st iteration with small neighborhood (red).


(c) Result of traditional algebraic fitting after 1st iteration with large neighborhood (red).


(d) Result after 3rd iteration of the proposed method from Fig. 3.2(b).


(e) Result after 5th iteration of the proposed method from Fig. 3.2(b).

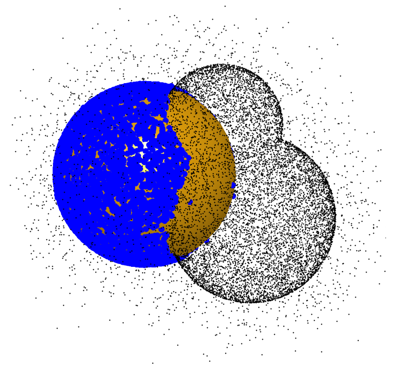

(f) Result after 10th iteration of the proposed method from Fig. 3.2(b).

Figure 3.2. A toy example for sphere fitting and extraction. Traditional algebraic fitting [Pra87] fails with both small and large neighborhoods because of local minimums found in the optimization process. The proposed iterative sphere fitting process extracts the points belonging to the sphere after a few iterations. The initial point and its neighborhood used for fitting are in red. Blue points are inliers which are extracted at each iteration. The data includes 5% noise and outliers with 50 times the average space between points.

point clouds before searching for the next instance of the primitive. This is thus a *greedy* technique. In contrast, the proposed method does not discard any point and can detect multiple spheres simultaneously.

Another common method for model extraction is the Hough transform [DH72] that uses a voting scheme applied on an accumulator array. Although the Hough transform achieves satisfactory results for simple models (e.g. lines, circles), computational cost and memory requirement are major limitations for high dimensional models such as cylinders or spheres. Variants of the Hough transform (e.g. Standard, Randomized and Probabilistic HT) have recently been used for sphere extraction [vdGVBV02, ANC13, CYDL06, OCBH07]. Camurri et al. [CVC14] is a good reference on Hough-based sphere extraction. A family of Hough transforms is discussed and the results of several implementations for sphere extraction are summarized in the paper. However, Hough-based methods for sphere extraction still suffer from the drawbacks inherent to this transform. In general, prior information such as the space of radius values being searched for or the number of spheres in point clouds are required to reduce the memory and computational challenges. Moreover, it is also a sequential method that cannot detect multiple spheres simultaneously.

The proposed method extracts multiple spheres from point clouds simultaneously and estimates their descriptive parameters accurately. In this chapter, a framework for iterative sphere fitting uses geometric information to improve traditional sphere fitting. Furthermore, a novel validation step is also proposed to evaluate the quality of the detected spheres. The advantage of the approach is that the challenge of sphere validation is reduced to the simpler problem of half-circle detection, a problem that has been investigated thoroughly [TCL15a, LGF07]. Moreover, an efficient 3-sample strategy is proposed to guarantee that every sphere is extracted with a reduced computational burden. The approach for sphere extraction proposed in this chapter shares many common points with the cylinder extraction approach presented in Chapter 2.

3.3 Single sphere fitting and extraction

In this section, a robust fitting method for a single sphere is proposed. Generally, starting at a given point on the sphere and using its nearest neighbors, the remaining points associated with the sphere are determined after a few iterations, and the descriptive parameters of the sphere are also computed. We assume that $\mathbf{P} = \{\mathbf{p}_i | i = 1, ..., N\}$ are the 3D data points, which may include noisy data and outliers. First, a traditional algebraic fitting approach is described in Section 3.3.1. The iterative fitting and extraction algorithm proposed in this thesis are then covered in Section 3.3.2. Finally, a novel approach for sphere validation is introduced in Section 3.3.3.

3.3.1 Algebraic sphere fitting

As mentioned earlier, a sphere is represented by two parameters: a center $\mathbf{c}^* = [c_x, c_y, c_z]$ and a radius r^*. Algebraic fitting [Pra87], which minimizes the algebraic distance, is an efficient method to compute these parameters $\{\mathbf{c}^*, r^*\}$ for noise-free data including spherical points only. Attene et al. [AFS06, AP10] used this technique for hierarchical primitive clustering. The center and radius of a sphere are computed by minimizing the algebraic distance as follows:

$$[c_x, c_y, c_z, r^{*2} - c_x^2 - c_y^2 - c_z^2]^T = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{b} \tag{3.1}$$

where

$$\mathbf{H} = \begin{pmatrix} 2p_{1x} & 2p_{1y} & 2p_{1z} & 1 \\ 2p_{2x} & 2p_{2y} & 2p_{2z} & 1 \\ \vdots & \vdots & \ddots & \\ 2p_{Nx} & 2p_{Ny} & 2p_{Nz} & 1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} ||\mathbf{p}_1||_2^2 \\ ||\mathbf{p}_2||_2^2 \\ \vdots \\ ||\mathbf{p}_N||_2^2 \end{pmatrix} \tag{3.2}$$

With noisy data or outliers, traditional algebraic fitting is not able to achieve accurate results for sphere parameters but rather becomes trapped in a local minimum as shown in Fig. 3.2(b)-(c).

3.3.2 Iterative single sphere fitting algorithm

An iterative method for sphere fitting is proposed to improve the results achieved by classical algebraic fitting. Basically, starting at a given point on a sphere, the remaining points

Algorithm 4: Algorithm for iterative sphere fitting.

```
Data: Point cloud $\mathbf{P}$, normal vectors $\mathbf{n}$.
Result: $\{\mathbf{c}^*, r^*\}$ of fitted sphere and its associated points.
Inlier set $\mathbf{\Psi}$:=a given point $\mathbf{p_i}$ and its neighborhood;
1. Use algebraic fitting to find initial parameters of the sphere: center and radius (Section 3.3.1);
$k := 1; A = B = false$; Maximum number of iterations:=30;
while $k \leq 30$ do
    2. Extraction: Use normal and distance criteria to update $\mathbf{\Psi}$ for next iteration by Eq. 3.3;
    3. Fitting: $\mathbf{\Psi}$ are used for fitting a sphere $\{\mathbf{c}_k, r_k\}$ by minimizing the algebraic distance (Section 3.3.1);
    4. if $\mathbf{\Psi} = \varnothing$ then
        $A := true$;                                    ▷ No inliers found
    end
    5. if $\mathrm{CR}_k = 0$ then
        $B := true$;                                    ▷ Parameter convergence
    end
    6. if $A \vee B$ then
        break;
    7. else
        $k$++;
    end
end
8. if $A \vee \mathrm{CR}_k > 10^{-6}$ then
    $\mathbf{p_i}$- Poor initial point; No sphere detected;
else if $B \vee \mathrm{CR}_k <= 10^{-6}$ then
    $\{\mathbf{c}^*, r^*\} := \{\mathbf{c}_k, r_k\}$;
    $\mathbf{\Psi}$:=Points associated to the sphere;
end
```

associated with the sphere are determined just after a few iterations. This iterative method includes two interlaced steps: fitting and extraction. Algorithm 4 summarizes the main steps of the proposed method for single sphere fitting and extraction.

The first step (**1**. in Algorithm 4) is to apply traditional algebraic fitting (Section 3.3.1) to inliers $\mathbf{\Psi}$ that are currently comprised of a randomly selected point from $\mathbf{P}$ and its neighborhood. This first step provides initial sphere parameters used by the iterative fitting and extraction process, which is shown in Fig. 3.2(d)-(f).

The **extraction** step (**2**. in Algorithm 4) determines the points in $\mathbf{\Psi}$ that satisfy the distance and angle conditions for the detected sphere in Eq. 3.3. At each iteration, the set of inliers is updated as follows:

$$\mathbf{\Psi} = \left\{ i \middle| \Big(|\Delta d_i| < \alpha_s * r_k \Big) \&\& \Big(|\Delta n_i| > \beta_s \Big) \right\} \tag{3.3}$$

where Δd_i is the distance from point $\mathbf{p}_i \in \mathbf{P}$ to the detected sphere and Δn_i is the angle between its normal $\mathbf{n}_i$ and the vector between point $\mathbf{p}_i$ to the detected center $\mathbf{c}_k$, as shown in Fig. 3.3. The index of the iteration is k.

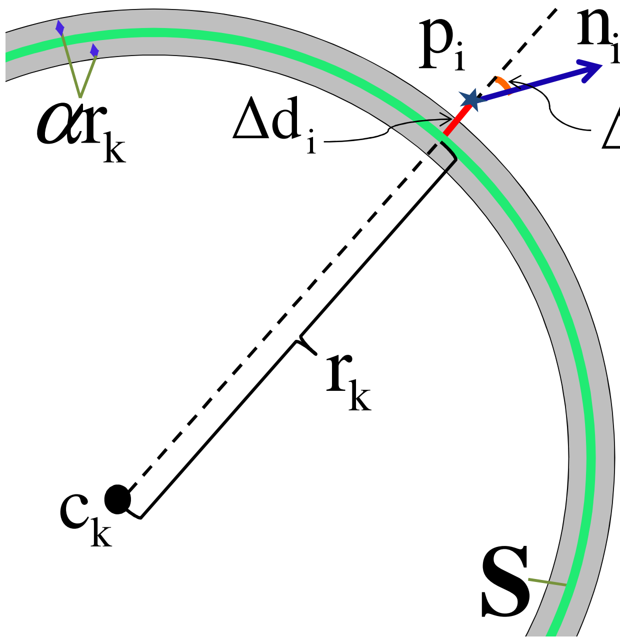


Figure 3.3. Computation of the orthogonal distance Δd_i (red line) between a point $\mathbf{p}_i$ and the detected sphere $\mathbf{S}$ (green circle). Angle Δn_i (orange) between its normal vector and the vector between $\mathbf{p}_i$ and the sphere center. $\mathbf{p}_i$ is the point of interest. $\mathbf{c_k}$ is the center of the detected sphere. $\mathbf{n}_i$ is the normal vector at a given point $\mathbf{p}_i$. $2\alpha_s r_k$ is the margin for inlier points (grey).

Quantities Δd_i and Δn_i are computed by using the detected sphere at the k^{th} iteration $\{\mathbf{c}_k, r_k\}$:

$$\Delta d_i = d(\mathbf{p}_i, \mathbf{S}) = \|\mathbf{p}_i - \mathbf{c}_k\| - r_k \tag{3.4}$$

and

$$\Delta n_i = \frac{\cos\left(\mathbf{p}_i - \mathbf{c}_k, \mathbf{n}_i\right)}{\|\mathbf{p}_i - \mathbf{c}_k\|} \tag{3.5}$$

The **fitting** step (**3**. in Algorithm 4) fits a sphere on the set of updated inliers by the algebraic method described in Section 3.3.1. These two consecutive steps (**2**. − **3**.) execute until all remaining points on the sphere are found or the stopping condition (**4**. or **5**. in Algorithm 4) is met. Generally, the algorithm terminates when there are no inliers found (A: $\mathbf{\Psi} = \varnothing$), parameter convergence (B: $\text{CR}_k = 0$ in Eq. 3.6) has occurred or the maximum number of iterations has been reached.

Parameter $\alpha_s r_k$ is a radius-based threshold value for $\mathbf{\Psi}$ update in Eq. 3.3. β_s is less sensitive to noisy data and outliers than α_s. Therefore, only variants of α_s are investigated, while $\beta_s = 0.95$ is set for all experiments. Large values of α_s may cause the process to fit either noisy points or points from other primitives, while small values slow down the extraction process. Hence an experiment was conducted to choose balanced values for α_s and the maximum number of iterations k in Algorithm 4. First, the convergence rate (CR) at each iteration is computed as:

$$\text{CR}_k = \left\| \{\mathbf{c}_k, r_k\} - \{\mathbf{c}_{k-1}, r_{k-1}\} \right\| \tag{3.6}$$

Selection of α and β values: Different models (noise in Fig. 3.2, partial sphere in Fig. 3.7) are taken as examples. Algorithm 4 is applied to the initial points that are spherical points extracted in Section 3.4.2. With the variant of α_s, the average number of iterations is computed when the condition B in Algorithm 4 is met. Fig. 3.4 shows the relation between the variants of α_s (1-15 % of r_k) and the average number of iterations for different models. Noisy data with outliers (Fig. 3.2) requires more iterations to converge than the noise-free model (Fig. 3.7). Therefore, $\alpha_s = 0.05$ and $k = 30$ are chosen to balance between computational speed and robustness to noise. $\alpha_s = 0.05$ means that the points whose distances are less than 5% of the detected radius r_k from the surface of the detected sphere will be considered as inliers at the next iteration (grey region in Fig. 3.3). For all experiments in this chapter, $\{\alpha_s = 0.05; \beta_s = 0.95; k = 30\}$ were used.

Using CR information plotted in Fig. 3.5, inconsistent cases ($\text{CR}_k > 10^{-6}$) associated with poor initial point selection are rejected (red line in Fig. 3.5), and it is decided that no sphere is detected. Otherwise, detected spheres are stored and evaluated further (**8**. in Algorithm 4).

In summary, the advantages of iterative sphere fitting are summarized as follows:

— Iterative sphere fitting extracts the points belonging to the spheres after a few iterations and allows the descriptive parameters of the spheres to be estimated accurately as demonstrated in Section 3.5.

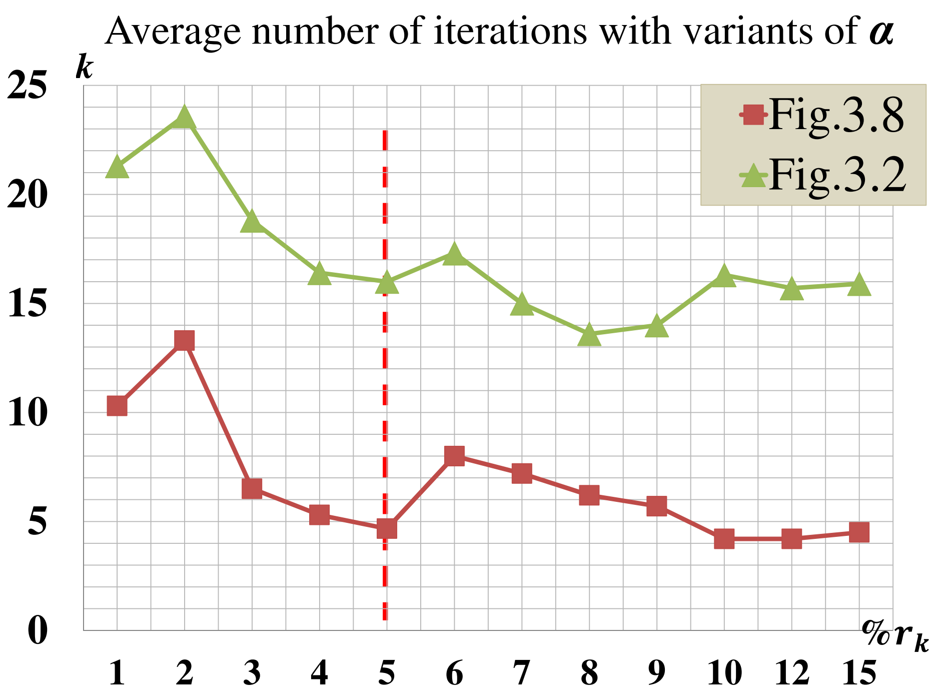


Figure 3.4. Experiment on the average number of iterations with the variants of α_s for different models in Fig. 3.2 and Fig. 3.7. $\alpha_s = 0.05$ is chosen to balance between computation speed and sensitivity to noise.

— Eq. 3.3 is considered as a filter with an adaptive threshold value $\alpha_s r_k$, which is able to cope with different values of radii and helps to reduce the effect of noisy data and outliers since the method uses only inliers to fit the sphere.

3.3.3 Sphere validation

The iterative sphere fitting process described in Section 3.3.2 reduces the effect of noise and outliers, but poor initial point selection or large amounts of noisy data and outliers can produce unpredictable results. Even when the detected spheres satisfy the convergence condition ($\mathrm{CR}_k <= 10^{-6}$), the process can still become trapped in a local minimum (blue line in Fig. 3.5), and the detected spheres are not valid (Fig. 3.7(e)). Therefore, a robust validation step is necessary to evaluate the quality and reliability of the detected sphere $\{\mathbf{c}^*, r^*\}$ by verifying the compatibility of the local structure around the detected spheres, as shown in Fig. 3.6.

The validation step is based on the observation that if a sphere is cut along a longitude and collapsed with respect to the z axis, which is illustrated in Fig. 3.8, the result is a half-circle. This corresponds to the transformation of a point in Cartesian coordinates $\{x, y, z\}$ into cylindrical coordinates $\{\phi, \rho, z\}$. Therefore, all points on the sphere surface are mapped on a half-circle with $\{\rho, z\}$ coordinates and for which ϕ, a component of cylindrical coordinates, is ignored in the collapse step (Fig. 3.8). The points on the surface of a reliable sphere

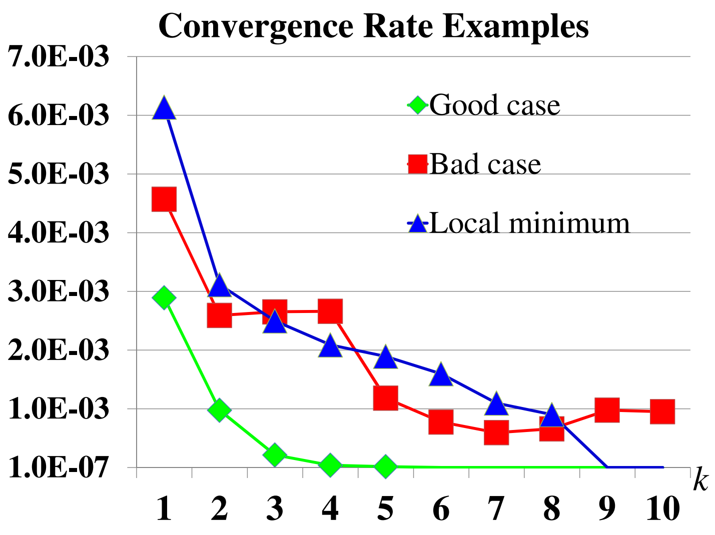


Figure 3.5. Convergence plots of CR for three cases: good case (Fig. 3.7(a)), local minimum (Fig. 3.7(d)) and bad case. For good cases, CR converges after a few iterations, while these conditions are not met for bad cases ($\mathrm{CR}_k > 10^{-6}$).

will distribute on a half-circle whose radius is the same as the one of the detected sphere Fig. 3.7(c). Local structures of unreliable spheres do not map to a half-circle, but different shapes as shown in Fig. 3.7(f).

The transformation from Cartesian coordinates $\{x, y, z\}$ to cylindrical coordinates $\{\phi, \rho, z\}$ is well known and is expressed as follows:

$$\begin{cases} \phi = tan^{-1}(\frac{y}{x}) \\ \rho = \sqrt{x^2 + y^2} \\ z = z \end{cases} \tag{3.7}$$

where $\{x; y; z\} = \{p_x - c_x^*; p_y - c_y^*; p_z - c_z^*\}$ and $\mathbf{p} = \{p_x; p_y; p_z\}$ belongs to the sphere surface. $\mathbf{c}^*$ is obtained with Algorithm 4 and becomes the origin of the cylindrical coordinate system.

The problem of sphere validation is now reduced to one of half-circle detection in 2D space $\{\rho, z\}$ as shown in Fig. 3.7. The RANSAC algorithm [LZH+13, FB81, LGF07] is a robust method for model extraction in noisy data and outliers, so it is used to detect a half-circle in the 2D space (red half-circle Fig. 3.7(c)). A score SC is computed to measure the difference between the estimated circle detected by RANSAC and a virtual circle corresponding to the detected sphere.

— The *virtual circle* $\{\mathbf{c}_{vir}, r_{vir}\}$, has the same radius as the one of the detected sphere ($r_{vir} = r^*$) and its center $\mathbf{c}_{vir}$ is located at the origin of the cylindrical coordinate system in Fig. 3.8.

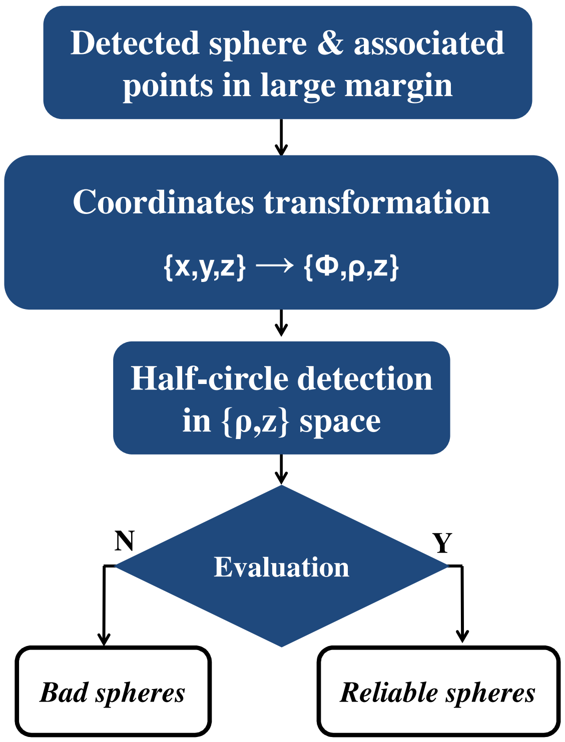

Detected sphere & associated points in large margin
Coordinates transformation
{x,y,z} ⟶ {Φ,ρ,z}
Half-circle detection in {ρ,z} space
Evaluation
N
Y
Bad spheres
Reliable spheres

Figure 3.6. Diagram for novel validation step of the detected sphere. $\{x, y, z\}$ is Cartesian coordinates and $\{\phi, \rho, z\}$ is cylindrical coordinates.

— The *estimated circle* $\{\mathbf{c}_{est}, r_{est}\}$ is generated by the collapse of the points on the $\{\rho, z\}$ plane. These points, blue in Fig. 3.7(b)-3.7(e), are located within a large margin from the surface of the detected sphere.

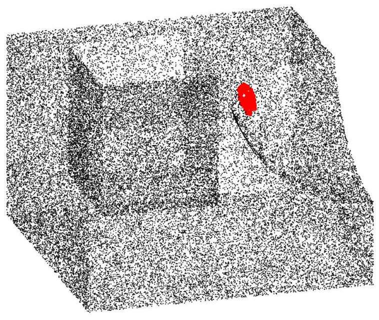
— SC is computed as the total error between the virtual and estimated circles in terms of radius values and center locations (see Eq. 3.8). This parameter evaluates how close the two circles are.

The following provides more details on the above validation procedure. The local geometric structure around the detected sphere is explored to evaluate its validity. The points that are inside a larger margin $\{3\alpha_s r_k; \beta_s = 0.9\}$ than the one in Eq. 3.3 (blue points in Fig. 3.7(b)-3.7(e)), are mapped onto the 2D space with Eq. 3.7. The value of SC is computed as the sum of the center deviation and radius deviation:

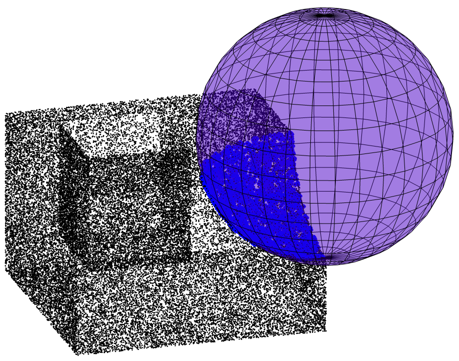
$$SC = \frac{\|\mathbf{c}_{vir} - \mathbf{c}_{est}\| + |r_{vir} - r_{est}|}{r_{vir}} * 100 \tag{3.8}$$

where $\mathbf{c}_{vir} = \{0, 0\}$ and $r_{vir} = r^*$ are the center and radius of the virtual circle (green in Fig. 3.7(c)-(e)), and $\mathbf{c}_{est}$ and r_{est} are the center and radius of the estimated circle by RANSAC in 2D space (red circle in Fig. 3.7(c)).

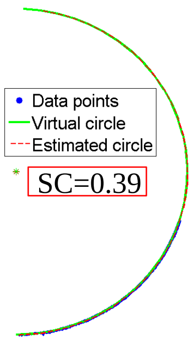
SC verifies the coincidence between the virtual circle and the estimated circle. This score depends on the level of noise in point clouds. Therefore, only the detected spheres that



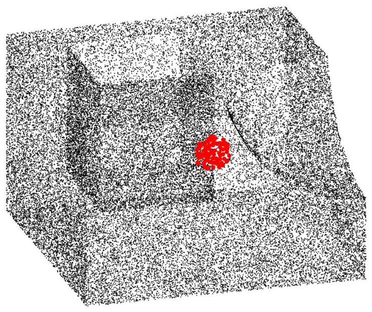
(a) Initial point and its neighborhood on a sphere.



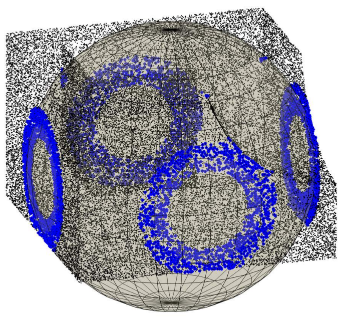
(b) Detected partial sphere and inlier points from the sphere surface.



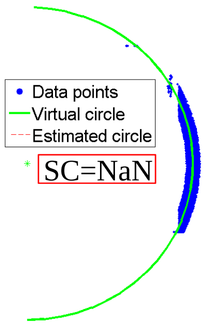
(c) Reliable detected sphere with small score. Data points, virtual and estimated circles are superimposed.



(d) Initial point and its neighborhood on a plane.



(e) Invalid sphere that has passed Algorithm 4 because fitting was trapped in a local geometric structure.



(f) The estimated circle was not detected for the invalid sphere.

Figure 3.7. Robust validation step for detected spheres. A reliable sphere leads to a half-circle after coordinate transformation, while an unreliable sphere transforms to undetermined shapes.

satisfy the condition ($SC < s_{sc}$) are considered as being valid. Otherwise, they are rejected (Fig. 3.7(e)). s_{sc} is defined in Table 3.2 (Section 3.5.3).

The novel validation step is robust and efficient to evaluate sphere-like shapes such as ellipsoids, paraboloids, etc. Fig. 3.9 illustrates that invalid spheres are detected by previously proposed methods [AFS06, SWK07], but are evaluated and rejected by the proposed validation method. Valid spheres that pass the validation procedure are stored for further processing.

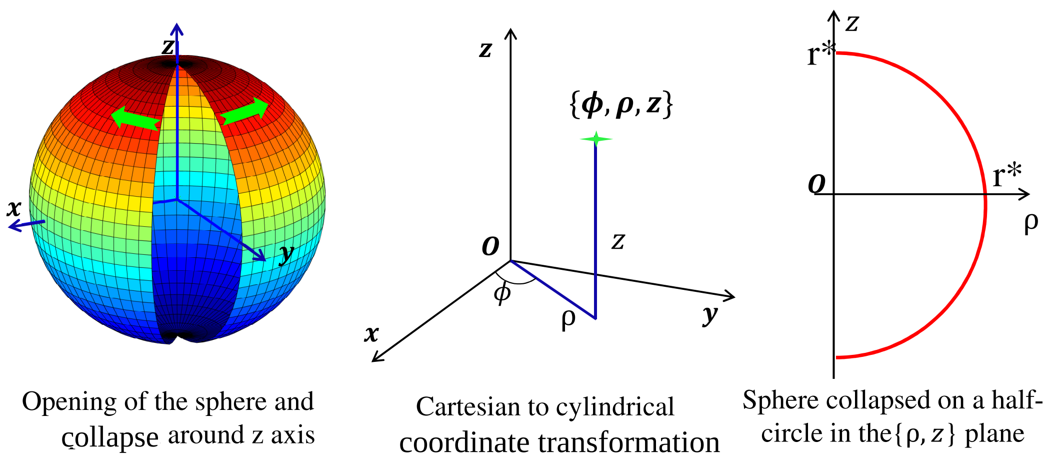
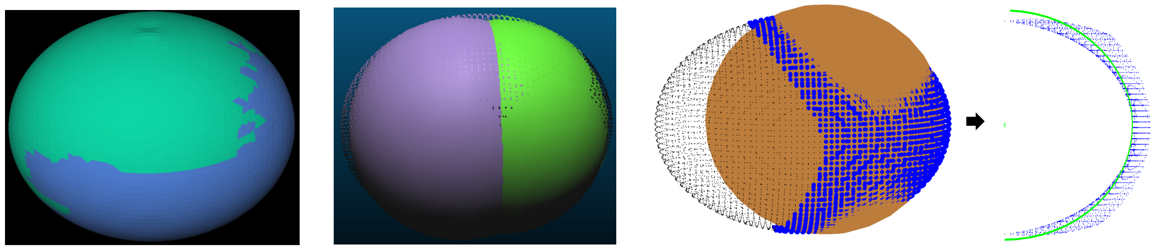

z
x
y
Opening of the sphere and collapse around z axis
z
$\{\phi, \rho, z\}$
O
z
ϕ
ρ
x
y
Cartesian to cylindrical coordinate transformation
z
r*
O
r*
ρ
Sphere collapsed on a half-circle in the $\{\rho, z\}$ plane

Figure 3.8. The transformation from Cartesian coordinates $\{x, y, z\}$ to cylindrical coordinates $\{\phi, \rho, z\}$. r^* is the radius of the detected sphere and half-circle.



(a) Two spheres detected by hierarchical clustering [AFS06]. (b) Two spheres detected by RANSAC [SWK07]. (c) An unreliable sphere passes Algorithm 4 of the proposed method, but is declared invalid by the validation step.

Figure 3.9. Validation test for an ellipsoid. Hierarchical clustering [AFS06] and RANSAC [SWK07] extract two spheres. The validation method proposed in this thesis removes these unreliable spheres because inliers generate shapes different from a half-circle.

3.4 Multiple sphere extraction

3.4.1 Overview of the method

In addition to noise and outliers, the problem of multiple spheres extraction faces many challenging issues: (i) manufactured objects often contain many spheres, partial or complete, with different radii; (ii) these objects are the combination of several primitives other than spheres, such as planes and cylinders, that may also cause the sphere fitting process to become trapped in a local minimum. Therefore, a novel framework called eSphere for extracting multiple spheres accurately and simultaneously addressing the above challenges is presented in this section. The diagram of eSphere is outlined in Fig. 3.10 and Algorithm 5 and is described in the following.

Multiple sphere detection begins with the idea that the algorithm for iterative sphere fitting and extraction described in Section 3.3 is applied to extract spheres and their parameters

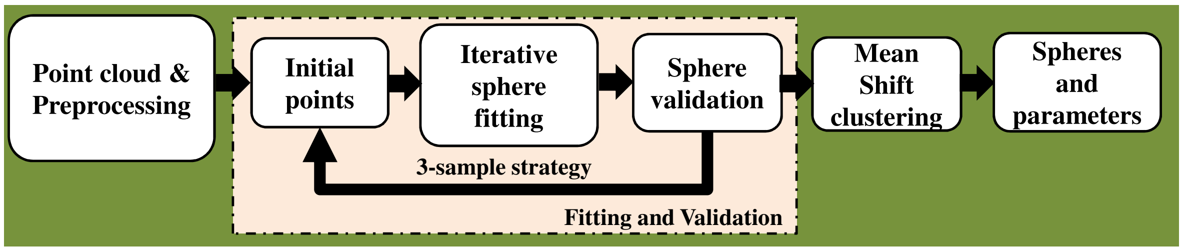


Figure 3.10. The diagram of the eSphere approach. The framework is comprised of preprocessing, iterative sphere fitting and validation, efficient 3-sample strategy and Mean Shift clustering. An efficient 3-sample strategy is proposed to guarantee that all spheres are detected from the point cloud.

starting at initial points and their nearest neighbors. Initial points belonging to the same sphere result in the same set of descriptive parameters. The set of initial points is extracted from point clouds using principal curvature information (Section 3.4.2). In order to guarantee that all spheres in the point cloud will be detected while maintaining reasonable execution time, an efficient initial point selection strategy is proposed (Section 3.4.3). Finally, Mean Shift clustering is applied to group data in the parameter space of the detected spheres (radii r^* and centers $\mathbf{c}^*$) (Section 3.4.4). Each cluster corresponds to the descriptive parameters of a valid sphere in the point cloud.

3.4.2 Preprocessing

At the preprocessing step, differential geometry information on the surface such as normal vectors $\mathbf{n}$ and principal curvatures $(\mathbf{k}_1, \mathbf{k}_2)$ are computed at each point in the point cloud. Then a set of initial points, which could belong to the surface of a sphere, is determined by using principal curvatures [BSG+13]. These computations are carried out only once and are used for the whole process.

Normal vectors can be computed by a PCA-based method [HDD+92] because of its simplicity (Appendix .2). A given point $\mathbf{p}_i$ and its k-nearest neighbors are used to create a 3x3 semi-definite covariance matrix ($\mathbf{CV}$).

$$\mathbf{CV} = \sum_{j=1}^{|N_{\mathbf{p}_i}|} (\mathbf{p}_j - \bar{\mathbf{p}}_i)^T (\mathbf{p}_j - \bar{\mathbf{p}}_i) \tag{3.9}$$

where $\bar{\mathbf{p}}_i$ is the centroid of points $\mathbf{p}_j$ in the neighborhood $N_{\mathbf{p}_i}$ of $\mathbf{p}_i$. Then the eigenvector of ($\mathbf{CV}$) associated with the smallest eigenvalue is considered as the normal vector $\mathbf{n}_i$ at $\mathbf{p}_i$.

Principal curvatures $(\mathbf{k}_1, \mathbf{k}_2)$ are computed at each point by paraboloid fitting [MSR07], which is a popular method for coping with noisy data. The main problem for surface normal and curvature estimation is the selection of the number of k-nearest neighbors, which has been a challenging issue for point cloud processing. In this chapter, we use the same number of k-nearest neighbors for both surface normal and curvature estimation. Table 3.2

lists the values that were used for all experiments. To guarantee the generality of the approach, no special approach was implemented for optimal k-nearest neighbor selection at each point.

Initial points $\mathbf{\Lambda}$ that potentially belong to spherical surface are extracted from point clouds if they satisfy the following condition

$$\mathbf{\Lambda} = \Big\{ i \Big| (k_{i1} \simeq k_{i2}) \&\& (k_{i1} > \gamma) \&\& (k_{i2} > \gamma) \Big\} \tag{3.10}$$

where γ is set to a low value to guarantee that all initial points are extracted. $\gamma = \sigma/3$ is set in all of the experiments.

σ **computation:** First, the signed distance of each point p_i is also computed as follows:

$$sd_i = (\mathbf{p}_i - \bar{\mathbf{p}}_i).\mathbf{n}_i \tag{3.11}$$

An unorganized point cloud including noisy data and outliers could be represented as $\mathbf{P} = \mathbf{P}_{inliers} \cup \mathbf{P}_{outliers}$. The fourth spread method is a standard statistical approach, which is used in recent works [Dev15, CNT$^+$14] to remove outliers. First, the signed distance is computed for every point (Eq. 3.11). Let L be the number of absolute values of signed distances. Then m_u is defined as the median of L/2 upper values, while m_l is defined as the median of L/2 lower values. Finally, a threshold $T_d = 1.5(m_u - m_l) + m_u$ is used to detect outliers that consist of the points whose absolute values of signed distance are larger than T_d. The remaining points are considered as inliers. Therefore, the average value of absolute signed distance of inliers σ is computed as follows:

$$\sigma = \frac{\sum_{i=1}^{|\mathbf{P}_{inliers}|} |sd_i|}{|\mathbf{P}_{inliers}|} \tag{3.12}$$

3.4.3 Efficient initial point sampling strategy

The purpose of the proposed method is to improve the drawbacks of sequential and greedy approaches. For instance, RANSAC-based [SWK07] and Hough-based [CVC14] methods extract a sphere and then remove its associated points immediately. Therefore, points removed at the intersection between two spheres no longer contribute to the sphere extraction process anymore. This can lead to inaccurate fitting for consecutive primitives.

The iterative sphere fitting process (Algorithm 4 in Section 3.3.2) is started at an initial point $\in \mathbf{\Lambda}$ to detect a sphere from the point cloud and compute its descriptive parameters. An important question is how many initial points should be chosen to detect all the spheres in a point cloud within reasonable execution time. On one hand, when all initial points $\mathbf{\Lambda}$ are executed sequentially, the sphere extraction process shows exponential time complexity.

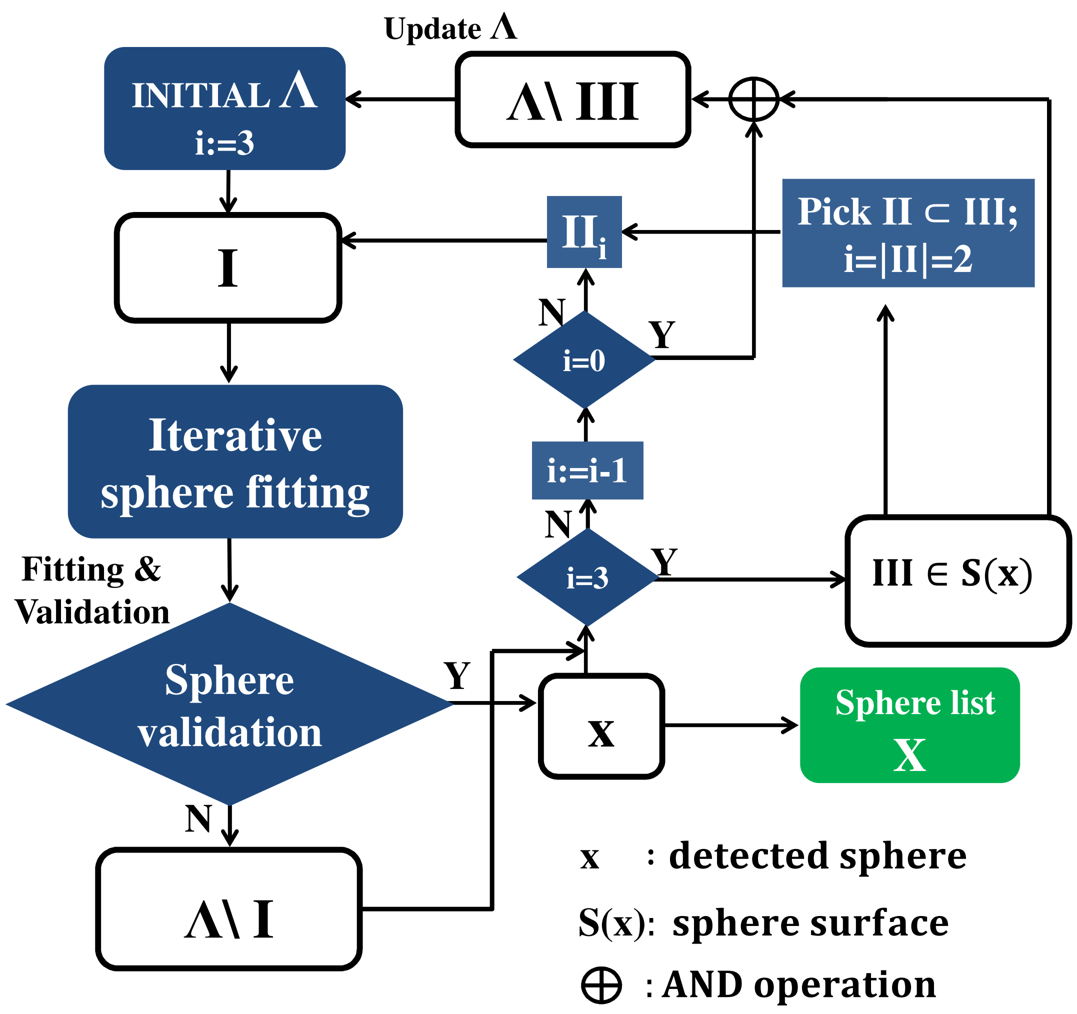


Figure 3.11. The efficient 3-sample approach for initial point sample strategy. Each sphere is extracted a maximum of three times because only three initial points (I) are selected. All initial points in **Λ** belonging to the surface of the detected sphere S(x) are stored into a set **III**.

On the other hand, when the set of initial points is downsampled (e.g. 5, 10 or 20 % of **Λ**), some spheres could be missed because their initial points are not chosen, while other spheres would be extracted several times. This section introduces an efficient sampling strategy which guarantees that all spheres are extracted with a reasonable time complexity regardless of the number of initial points.

In this section, a **3-sample** approach is proposed in which a maximum of three initial points is investigated to extract a sphere. The 3-sample approach is illustrated in Fig. 3.10. Starting at a random initial point ($I \in \mathbf{\Lambda}$), iterative fitting and validation is applied to its k-nearest neighbors to extract the sphere. If a reliable sphere x corresponding to I is detected, then all initial points in **Λ** belonging to that sphere are identified, these points are stored into a set **III**. From this **III** set, two other points of the 3-sample strategy are chosen randomly and stored into a set **II** ($|\mathbf{II}| = 2$). The loop of fitting and validation for this sphere is repeated and terminated after two iterations. $\mathbf{\Lambda}$ is updated by removing **III** and starts at another random initial point (I) that belongs to one of the other spheres in the point cloud. Each time a sphere is validated, it is considered as a valid sphere and its parameter vector **x** is stored in a list of sphere parameters **X** .

Considering the above 3-sample strategy, a sphere can be detected a maximum of three times. This redundancy in sphere detection is exploited at the clustering step described in Section 3.4.4. The advantages of the 3-sample approach are summarized as follows:

— Each sphere is extracted a maximum of three times because only three initial points

Algorithm 5: Multiple spheres extraction from point clouds.

Data: Point cloud; Normal vectors; Initial points $\mathbf{\Lambda}$ selected based on principal curvatures (Section 3.4.2)

Result: Parameters of detected spheres and their associated points

```
$\mathbf{X} := \{\varnothing\}$;                                          ▷ Sphere list
$i := 3$;                                                    ▷ dummy variable
while $\mathbf{\Lambda} != \{\varnothing\}$ do
    1. Start at a random initial point $I$ and its nearest neighbors;
    2. New sphere $\mathbf{x}$ detected by Algorithm 4;
    3. Sphere validation for $\mathbf{x}$ by computing $SC$ in Eq. 3.8 ;
    4. if $SC <= s_{sc}$ then
        $\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{x}$;                     ▷ insert new sphere x into X
    end
    5. if $SC > s_{sc}$ then
        Update $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} \setminus I$;                  ▷ remove bad initial point
        $i := 0$;                                            ▷ skip 6-7
    end
    6. if $i$=3 then
        6.1. Find initial points $\mathbf{III} \subset \mathbf{\Lambda}$ on the $\mathbf{S}(\mathbf{x})$-sphere surface;
        6.2. Pick a random set $\mathbf{II} \subset \mathbf{III}$; $|\mathbf{II}| = 2$;
                                                 ▷ Efficient 3-sample approach
        6.3. $i = |\mathbf{II}| = 2$;
    end
    7.while $i \neq 0$ do
        7.1. $I = \mathbf{II}_i$;
        7.2. Execute 1-4;                                    ▷ Storing x into X
        7.3. $i := i - 1$;
    end
    8. Reset $i := 3$;                               ▷ dummy control loop variable
    9. Update $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} \setminus \mathbf{III}$;                       ▷ remove initial points
end
10. Apply Mean Shift clustering to $\mathbf{X}$ list;
```

are selected. Therefore, running time is approximately constant regardless of the number of initial points. Instead it is dependent on the number of spheres and data points in the point cloud.

— All spheres in the point cloud are extracted because initial points are selected until the set $\mathbf{\Lambda}$ is empty.

3.4.4 Mean Shift clustering for sphere parameters estimation

Theoretically, initial points belonging to the same sphere share the same set of descriptive parameters. Consequently, each set of descriptive parameters is considered as a data point in the parameter space for the corresponding sphere as shown in Fig. 3.12(b). However, for point clouds with noise and outliers, these data points in parameter space distribute close to each other for a given sphere. Therefore, the next step of the proposed approach consists of clustering data in the $\mathbf{X}$ list of reliable spheres. This results in the estimation of multiple spheres from the point cloud. A clustering approach is used to find the cluster corresponding to the parameters of the sphere. Mean Shift clustering [CM02], a non-parametric clustering approach, is used because it does not require that the number of clusters be known a *priori*.

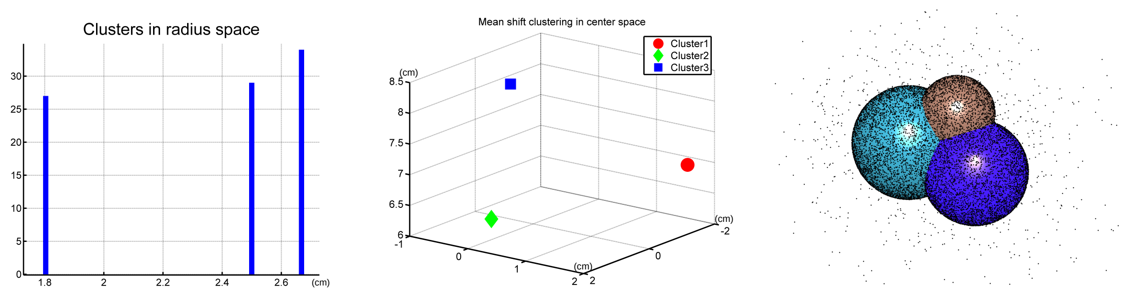
Mean Shift clustering in the sphere parameters space $\mathbf{X} \in \mathbb{R}^4$ is applied as follows. The descriptive parameters of all the spheres in the point cloud are stored in a vector $\mathbf{X} = \{\mathbf{x}_i = (\mathbf{c}_i^*, r_i^*) | i = 1, ..., n_s\}$. n_s is the number of detected spheres in the $\mathbf{X}$ list. Mean Shift clustering with separable Gaussian kernels for centers and radii is used (see Section 2.4.2 for the details on mean-shift clustering):

$$\hat{f}(\mathbf{x}) = \frac{1}{n_s h_1 h_2} \sum_{i=1}^{n_s} K_1(\frac{\mathbf{c}^* - \mathbf{c}_i{}^*}{h_1}) K_2(\frac{r^* - r_i{}^*}{h_2}) \tag{3.13}$$

The Mean Shift vector $\mathbf{m}$ is defined as:

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{n_s} \mathbf{x}_i g\left(\|\frac{\mathbf{c}^*-\mathbf{c}_i{}^*}{h_1}\|^2\right) g\left(|\frac{r^*-r_i{}^*}{h_2}|^2\right)}{g\left(\|\frac{\mathbf{c}^*-\mathbf{c}_i{}^*}{h_1}\|^2\right) g\left(|\frac{r^*-r_i{}^*}{h_2}|^2\right)} - \mathbf{x} \tag{3.14}$$

After clustering, the final number of clusters is the number of spheres in the point cloud and the parameters of the cluster centers are the descriptive parameters of the spheres. Only the bandwidth value $\mathbf{h}_s = \{h_1, h_2\}$ is defined for the Mean Shift algorithm. h_1 is the bandwidth for the center, and h_2 is the bandwidth for the radius. For all experiments reported in this chapter, the bandwidth is set automatically as $h_s = \{\frac{min(r_i{}^*)}{10}, \frac{min(r_i{}^*)}{10}\}$. Fig. 3.12 shows the process and the result of using Mean Shift clustering for sphere extraction on the three-sphere model.



(a) Histogram of the radii of detected spheres.

(b) Clusters in center space. Each color refers to a cluster.

(c) Three spheres are detected.

Figure 3.12. Result of Mean Shift clustering for the three-sphere model in Fig. 3.2 with noise and outliers. The final number of clusters is the number of spheres in the point cloud and the parameters of the cluster centers are the descriptive parameters of the spheres.

3.5 Results and discussion

The performance of the proposed method is tested for single and multiple spheres extraction. Moreover, the results are compared to state-of-the-art methods in terms of quantitative and qualitative experiments. These experiments demonstrate that the proposed method achieves more accurate results in center and radius estimation and is more robust to noise and outliers than other methods.

3.5.1 Single sphere extraction

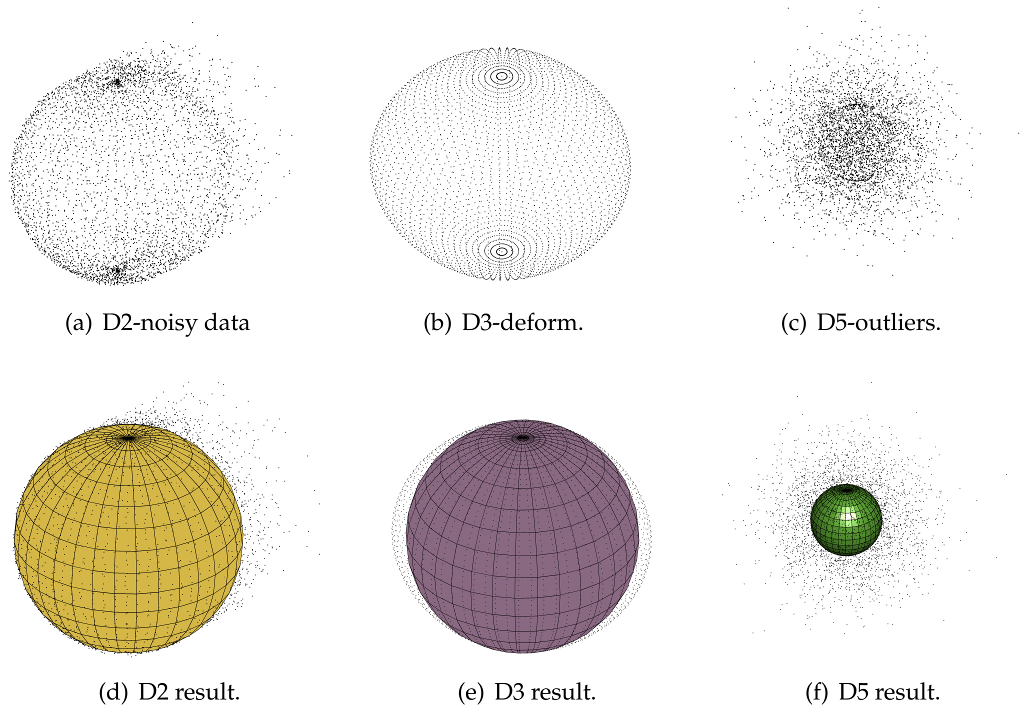
To illustrate the accuracy of eSphere for center and radius estimation, it is applied to fit a single sphere and estimate its parameters. The results are compared to three other methods: Hough-based method [CVC14], RANSAC-based method [FB81] and nonlinear least-squares (NLSQ) [For89] using Gauss-Newton optimization. Three types of datasets were taken from the work of Camurri et al. [CVC14] to test single sphere extraction. These datasets represent many challenges such as noisy data D2, deformation D3 and outliers D5 as shown in Fig. 3.13. Each type of data contains 100 spheres with different values of centers and radii and ground truth values are available. Fig. 3.13 shows some results achieved by our method. The results are compared based on the following criteria:

— The distance between the detected center $\mathbf{c}^*$ and the ground truth $\hat{\mathbf{c}}$ w.r.t. to the ground truth $\hat{r}$ (%).

$$E_c = \sum_{i=1}^{100} \frac{\|\hat{\mathbf{c}}_i - \mathbf{c}_i^*\|}{\hat{r}_i} \tag{3.15}$$

— The difference between the detected radius r^* and the ground truth $\hat{r}$ w.r.t. to the ground truth $\hat{r}$ (%).

$$E_r = \sum_{i=1}^{100} \frac{|\hat{r}_i - r_i^*|}{\hat{r}_i} \tag{3.16}$$


(a) D2-noisy data (b) D3-deform. (c) D5-outliers.

(d) D2 result. (e) D3 result. (f) D5 result.

Figure 3.13. Single sphere extraction using our method on the datasets in [CVC14]: noisy data D2, deformed sphere D3, sphere with outliers D5.

Fig. 3.14 and Fig. 3.15 show that Hough-based and RANSAC-based methods achieve acceptable results for D2 and D3, but fail for D5. The NLSQ method fails for the datasets with noise and outliers D2 and D5. The proposed method for single sphere extraction outperforms the other three methods and achieves good results close to ground truth. This demonstrates that iterative sphere fitting can reduce the effect of noise and outliers and guarantee accurate results for the estimation of the center and radius of the spheres.

3.5.2 Multiple spheres extraction

In this section, the performance of eSphere [TCL16b] for extracting multiple spheres is tested. The RANSAC-based method [SWK07] is known as being robust for primitive extraction from large point clouds. Although the RANSAC-based method can detect all types of primitives (not only spheres), for fair comparison with eSphere, it was used only to detect spheres in this experiment. The source code of this algorithm is available at `http://www.danielgm.net/cc/`. Some threshold values need to be set by the user to obtain good results depending on the level of noise, the size of the model and the variance of the radii. The results are shown in Fig. 3.16 and Table 3.2 summarizes the threshold values used for different models.

Our method achieves better results than the RANSAC-based method [SWK07]. As shown in Fig. 3.16(a), one sphere was detected twice and some detected spheres were not fitted well

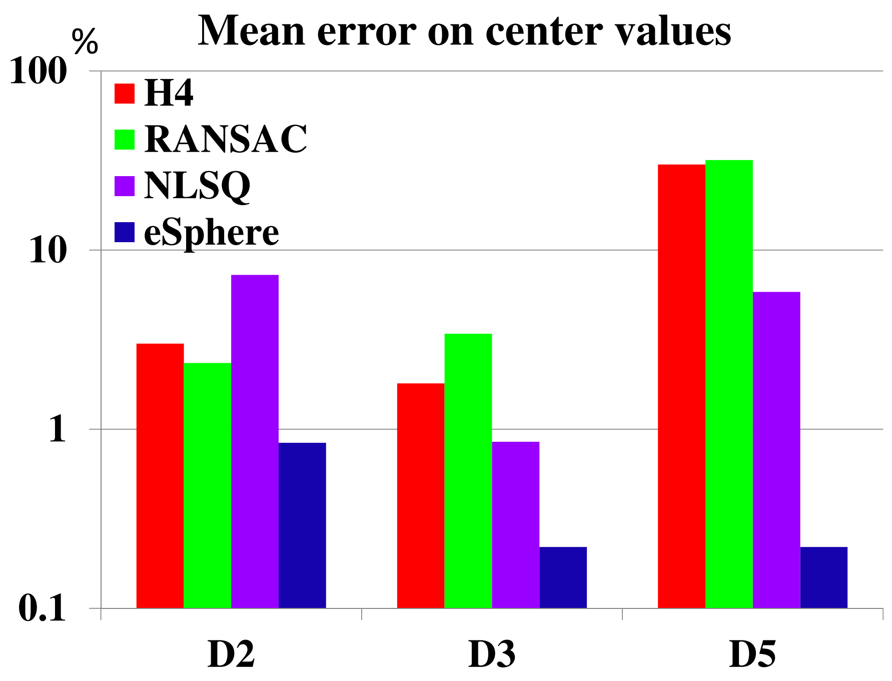


Figure 3.14. Logarithm of mean error on center values, $\log_{10}(E_c)$.

by RANSAC. This is caused by minimum random sampling and sequential point removal. Moreover, the RANSAC-based method misses two spheres as shown in Fig. 3.16(c). In comparison, the proposed method fits spheres well to the data points (Fig. 3.16(b)) by using iterative sphere fitting. In addition, as shown in Fig. 3.16(a)-(c), a set of the smallest spheres was detected by RANSAC with a large variance of radius values, while our method detects all of the smallest spheres with a small variance Fig. 3.16(b)-(d) (see Table 3.1).

Table 3.1. Mean and standard deviation of the results for RANSAC-based [SWK07] and eSphere methods: Radii of smallest detected spheres of the bracelet and sphere cluster datasets in Fig. 3.16 and Fig. 3.17 are evaluated. For the scanned data in Fig. 3.17(e)-(f), radii of two detected spheres are investigated.

Model Name	RANSAC		eSphere		Groundtruth radius
	Mean	Std	Mean	Std	
Bracelet Fig. 3.16	0.0362	4.251E-03	0.0312	1.380E-08	0.0312
Sphere cluster Fig. 3.16	0.2945	1.607E-03	0.2943	3.837E-04	0.2962
Sphere cluster Fig. 3.17(a)	0.2939	2.929E-03	0.2945	5.544E-04	0.2962
Scanned data Fig. 3.17(d)	21.1406	1.144	19.8637	0.172	20

We also applied the proposed method to other complex models in biochemistry, which are available at the 3D Warehouse. These models are comprised of different shapes (cylinders) and highly clustered and complex structures with various radii. Fig. 3.20 shows the spheres detected by our algorithm.

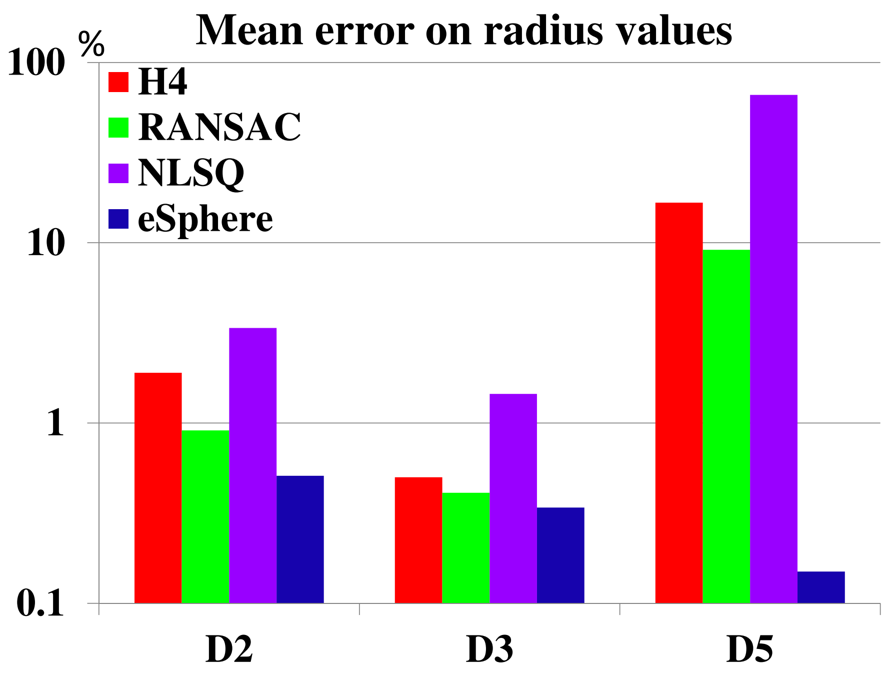
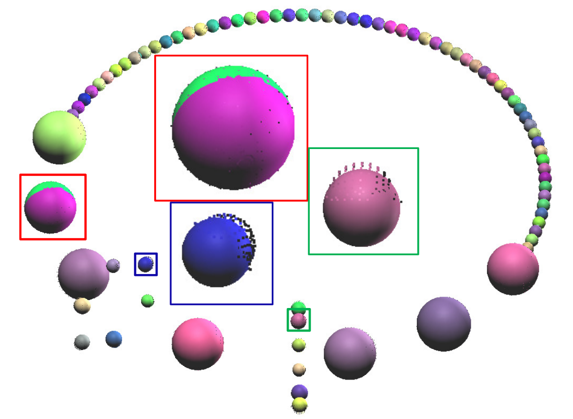


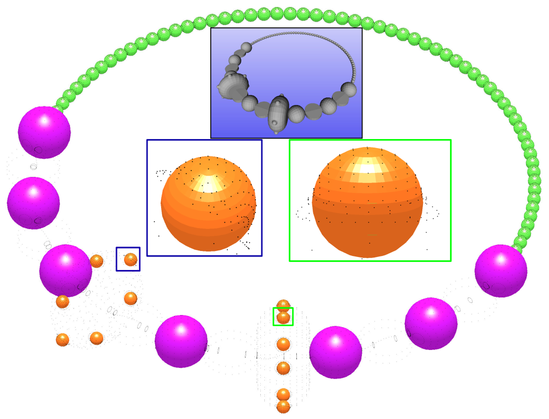
Figure 3.15. Logarithm of mean error on radius values, $\log_{10}(E_r)$.

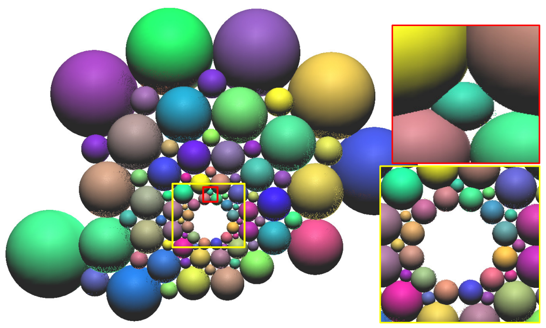
3.5.3 Noisy and real datasets

To illustrate the robustness of the method to noise and outliers, it was tested on noisy synthetic and real scanned models. In Fig. 3.17(a), 5% noise and outliers with 50 times the average space between points are applied to original datasets. Real scans were captured by the Go!SCAN handheld 3D scanner as shown in Fig. 3.17(d). In Fig. 3.17(g), the data points of the balls were captured by a Kinect sensor. The results show that the proposed method is able to cope with noise and outliers to extract multiple spheres and to estimate their parameters. The RANSAC-based method [SWK07] is greedy and sequential, so the radii of the detected spheres are larger than the ground truth (Fig. 3.17(b)-(e)). In comparison, our method (Fig. 3.17(c)-(f)) iteratively fits a sphere to inliers, so the result is close to the ground truth as shown in Table 3.1.

Table 3.2 summarizes the threshold values used in the experiments. It is difficult to choose optimal values for the RANSAC-based method [SWK07]. For instance, for complex data such as sphere packing, we were not successful in choosing a set of threshold values $\{\tau, \epsilon, \epsilon_{bitmap}\}$. While RANSAC requires that three global parameters be set, our method requires that only parameter s_{sc} be set with a simple selection strategy. s_{sc} is selected to evaluate the quality of the detected spheres and depends on the level of noise of the point clouds. Therefore, it is increased slightly with noisy data and outliers as summarized in Table 3.2. The number of spheres detected by the two methods are reported in Table 3.2. The proposed method detects a number of spheres closer to ground truth than the RANSAC-based method does. The novel sphere validation step and robust iterative sphere extraction help to reduce the


(a) Results of the RANSAC-based method for the bracelet model. Some spheres are not fitted well to data points. One sphere was extracted twice.


(b) Spheres detected using our method are fitted and extracted well to partial data of the sphere surface (a set of smallest spheres in orange).


(c) Results of the RANSAC-based method for the sphere cluster model. Smallest spheres are detected with large variance, see Table 3.1. Two spheres were missed in the region circumscribed by a yellow frame.

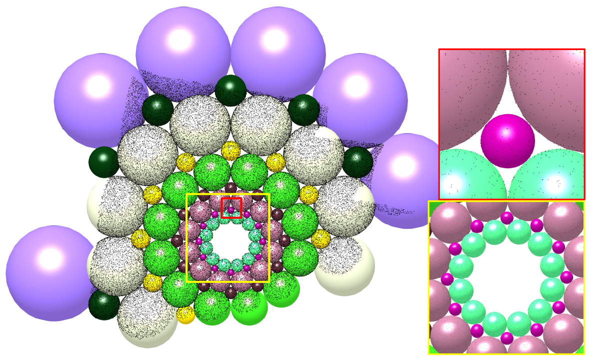
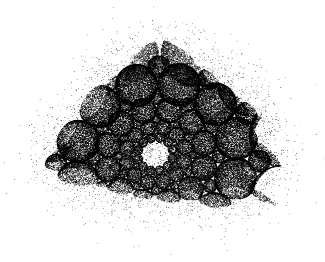
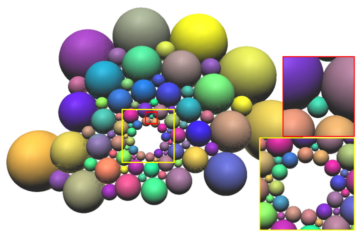

(d) Results of our method. The smallest spheres (purple) are detected with high accuracy and small variance (Table 3.1). Partial spheres at the outermost area of the point cloud are also detected.

Figure 3.16. The results of our method compared to a RANSAC-based method [SWK07]. Threshold values and the number of detected spheres are listed in Table 3.2. Mean and standard variation of the smallest spheres of bracelet and sphere cluster models are summarized in Table 3.1.

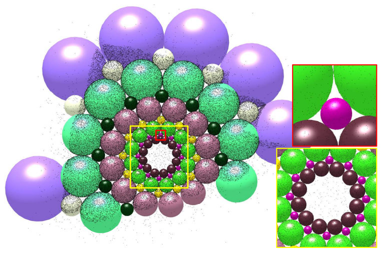
effect of noise and outliers for complex datasets.



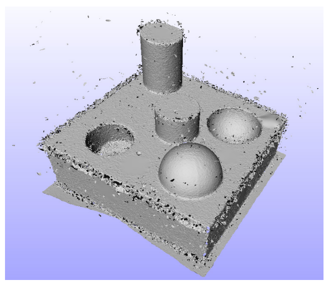
(a) Sphere cluster with 5% noise and 5% outliers. The magnitude of outliers is 50 times the average space between points.



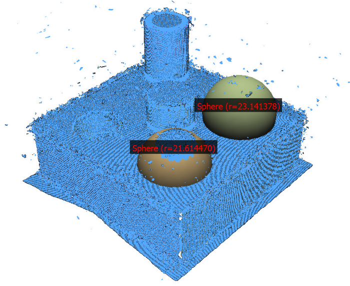
(b) Result for the RANSAC-based method. Mean and variance of smallest spheres are listed in Table 3.1. Three spheres were missed (Table 3.2).



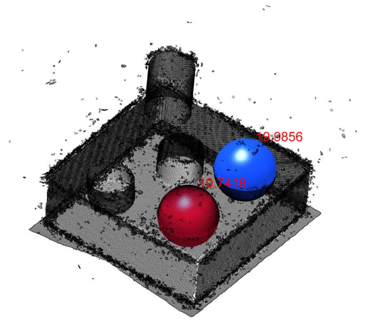
(c) Result of the proposed eSphere method. Mean and variance of smallest spheres (purple) are also listed in Table 3.1.



(d) Real data captured with the GO!SCAN 3D sensor. The model contains two spheres (concave, convex) with radius 20mm.

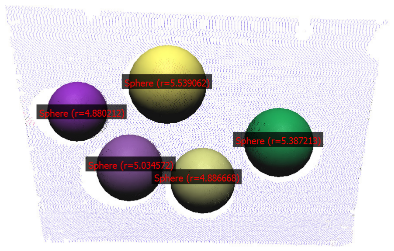


(e) Result of the RANSAC-based method. Radius values are larger than ground truth (20 mm).

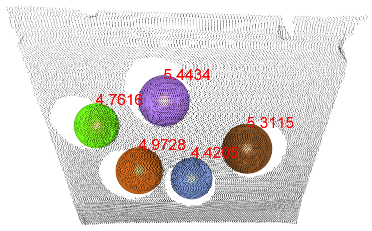


(f) Results achieved by eSphere method are close to ground truth. Mean and variance are reported in Table 3.1.

(g) Balls data captured by Kinect [CVC14].



(h) Sphere detected by RANSAC.



(i) Result of the eSphere method.

Figure 3.17. Performance of the RANSAC-based [SWK07] and eSphere methods for synthetic and real datasets with large noise and outliers. The RANSAC-based method results in a large variance for the detected spheres because of a greedy and sequential technique, which removes points immediately after sphere detection. In this experiment, RANSAC is used for sphere extraction only.

Table 3.2. Statistics on different models. N_S is the number of spheres in the point clouds. x means that this values is unavailable.

Model Name	#Points	N_S	RANSAC [SWK07]				eSphere [TCL16b]		
			τ	ϵ	ϵ_{bitmap}	N_{S1}	k-nn	s_{sc}	N_{S2}
Bracelet Fig. 3.16	27 089	79	50	0.05	0.1	80	10	1	79
Sphere cluster Fig. 3.16	149 864	90	50	0.1	0.2	88	10	1	90
Sphere cluster Fig. 3.17(a)	149 864	90	50	0.1	0.3	87	20	1.5	90
Kinect data Fig. 3.17(g)	257 698	5	200	0.75	1	5	50	2	5
Scanned data Fig. 3.17(d)	268 544	2	200	4.2	8.5	2	50	2	2
Carbon NanoTube Fig. 3.20(a)	127 064	448	100	1	2	448	20	1	448
ADN Fig. 3.20(b)	200 000	270	100	1	3	258	20	1	267
Sphere Packing Fig. 3.20(c)	682 374	x	x	x	x	x	20	1	7299

Table 3.3. Computation time for the eSphere method. Extraction includes the iterative sphere fitting and validation steps. Time results have been averaged over 5 runs.

Model Name	#Points	#N_{S2}	Normal (s)	Curvature (s)	Extraction (s)
Bracelet Fig. 3.16	27 089	79	2.12	2.57	13.81
Sphere cluster Fig. 3.16	149 864	90	11.84	20.01	54.26
Sphere cluster Fig. 3.17(a)	149 864	90	12.13	20.61	61.64
Kinect data Fig. 3.17(g)	257 698	5	20.81	35.93	93.36
Scanned data Fig. 3.17(d)	268 544	2	22.31	24.74	246.52
Carbon NanoTube Fig 3.20(a)	127 064	448	10.40	11.96	234.97
ADN Fig. 3.20(b)	200 000	267	15.71	27.01	312.78
Sphere Packing Fig. 3.20(c)	682 374	7299	60.22	96.32	5.14E+03

Some spheres were missed by eSphere for the ADN and sphere packing models (Table 3.2). There are several reasons for the problem. First, it is not possible to extract spheres whose radius is less than the noise magnitude. Moreover, we believe that this is caused by complex data for which the initial points are not extracted at the preprocessing step. Hence, some advanced methods could be used to achieve better results for normal [ZCL+13, BM12] and principal curvatures estimation [KNSS09, YQ07]. Another problem could be the selection of the global bandwidth parameter h in the Mean-Shift clustering in Section 3.4.4. Therefore, an adaptive Mean Shift clustering procedure [CRM01] might be the solution to this problem but this potential solution has not been investigated in this thesis.

3.5.4 Computation time of eSphere and 3-sample strategy

The framework in this chapter was implemented on MATLAB on a 3.2 GHz Intel Core i7 platform. Table 3.3 shows the computation time for the proposed method with different models. The bottleneck of eSphere is caused by the iterative sphere fitting and validation steps. Table 3.3 shows that the computation time depends on the size of the point cloud and the number of spheres in the point cloud.

The 3-sample method is proven to be efficient for computation burden reduction. As discussed in Section 3.4.3, if all initial points are processed sequentially, the extraction process is an exponential function of time. Another means of improving the computational performance would be to downsample set $\mathbf{\Lambda}$. However, note that doing so would no longer guarantee that all spheres would be detected. On this topic, the performance of the 3-sample method is compared to the uniform resampling method with 5, 10 and 20 % of initial points $\mathbf{\Lambda}$. Fig. 3.18 summarizes the running time for fitting and validation corresponding to the number of sampled initial points. As shown in Fig. 3.18, the 3-sample method is very efficient for the models including only spheres (Fig. 3.16 and Fig. 3.20(b)). For the noisy model in Fig. 3.17(f), the 3-sample method spends a lot of time validating poor initial points that are located in cylindrical and planar areas. This is a limitation of the proposed method. Therefore, a robust method for surface normal and principal curvature computation would improve the performance of initial point selection.

Another experiment was conducted to investigate the relationship between computation time and k-sample strategy (k=2,3, etc). With a robust iterative sphere fitting and an efficient sampling strategy, all spheres are extracted from point clouds with any k-sample strategy. However, the computation time depends on the number of samples. Therefore k is varied from 2 to 6 samples for the experiment. Fig. 3.19 shows the linear relationship between computation time and k-sample strategy with the slope depending on the number of spheres and the number of data points $|\mathbf{P}|$. For all experiments in this chapter, a 3-sample strategy was chosen.

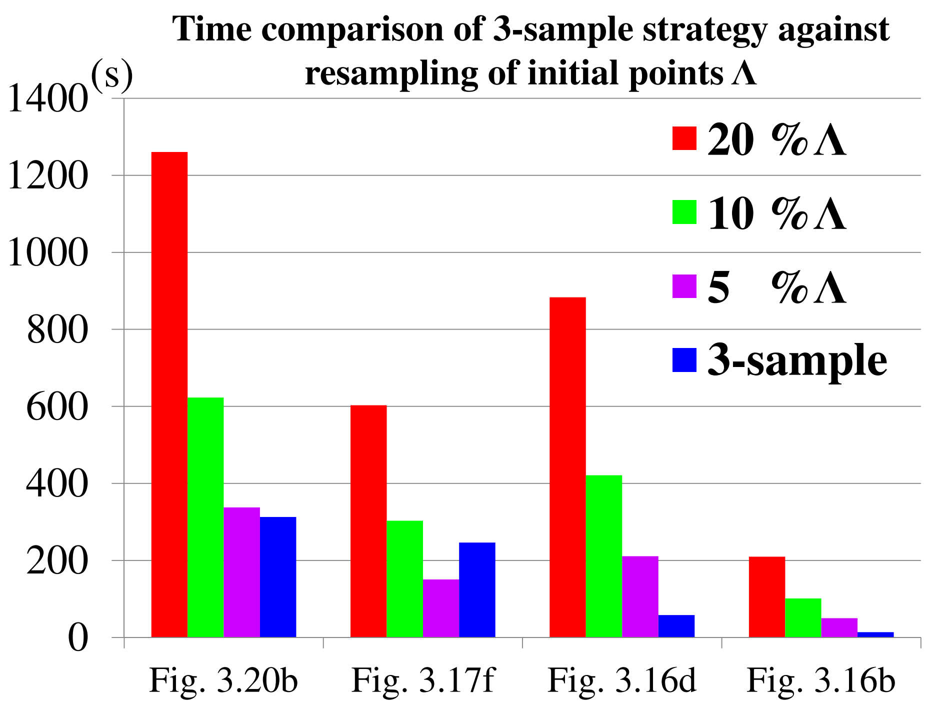


Figure 3.18. Time comparison of 3-sample approach against uniform resampling 5, 10 and 20 % of initial points **Λ**.

Theoretically, the computation time mainly depends on the number of spheres in the point cloud, not the number of initial points **Λ**. Fig. 3.16(d) and Fig. 3.17(c) have the same number of spheres, but a different number of initial points because of noise and outliers. Fig. 3.19 demonstrates that their computation times with the variants of the *k*-sample strategy are similar regardless of the number of initial points.

3.6 Conclusion

A robust approach is proposed for extracting multiple spheres and estimating their parameters from point clouds. First, an iterative sphere fitting algorithm is able to extract a sphere from noisy point clouds with outliers by using distance and normal filtering. A robust validation step is also proposed to evaluate the quality of the detected spheres. Especially, an efficient 3-sample strategy is proposed to extract all spheres with reduced computational burden. Finally, multiple spheres are extracted simultaneously through Mean Shift clustering. The performance of the method was tested on synthetic and real datasets with promising results. It is planned to use the method for reverse engineering and part-to-CAD analysis of point clouds of complex scenes containing many full or partial spherical components. In future work, the framework will be extended to the extraction of other types of primitives such as cones and tori.

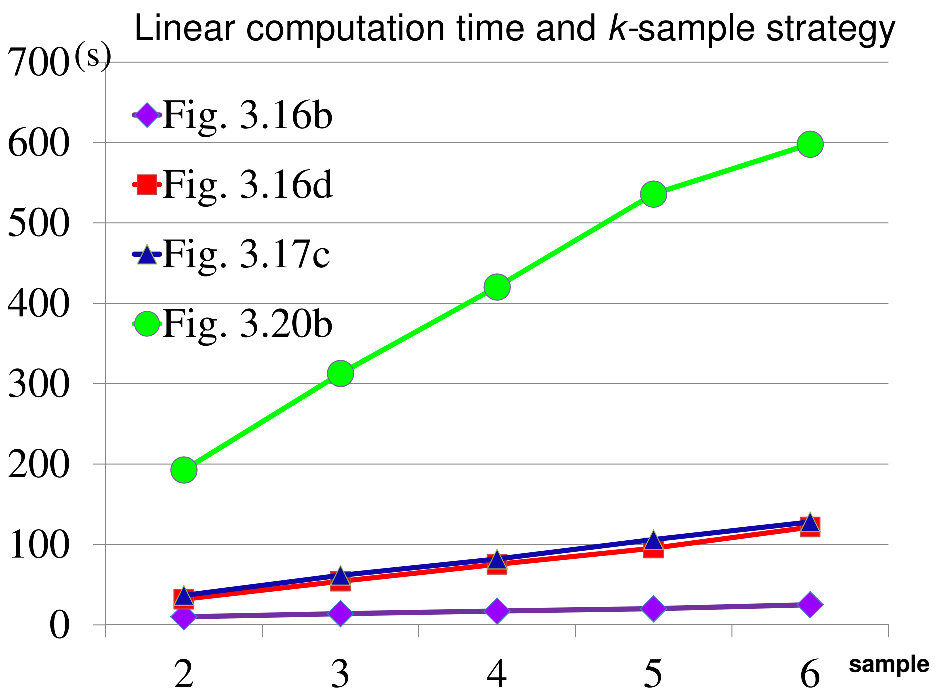
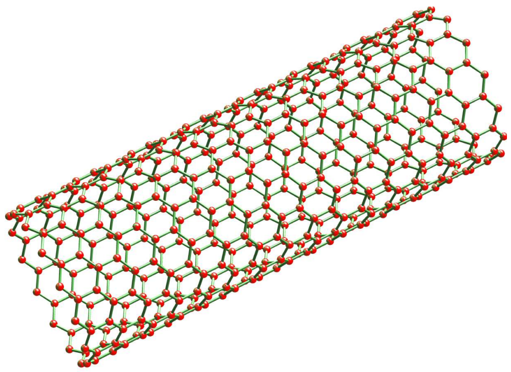
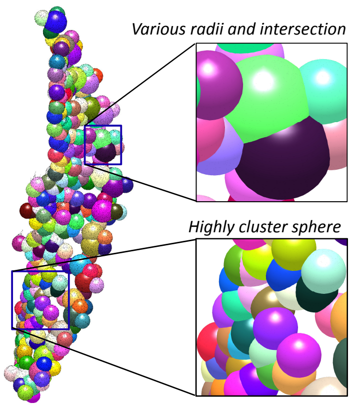


Figure 3.19. Linear relationship between computation time of the variants of the *k*-sample strategy. The computation time mainly depends on the number of spheres and the number of data points in the point cloud, not the number of initial points $\mathbf{\Lambda}$. For all experiments in this chapter, a 3-sample strategy was chosen.

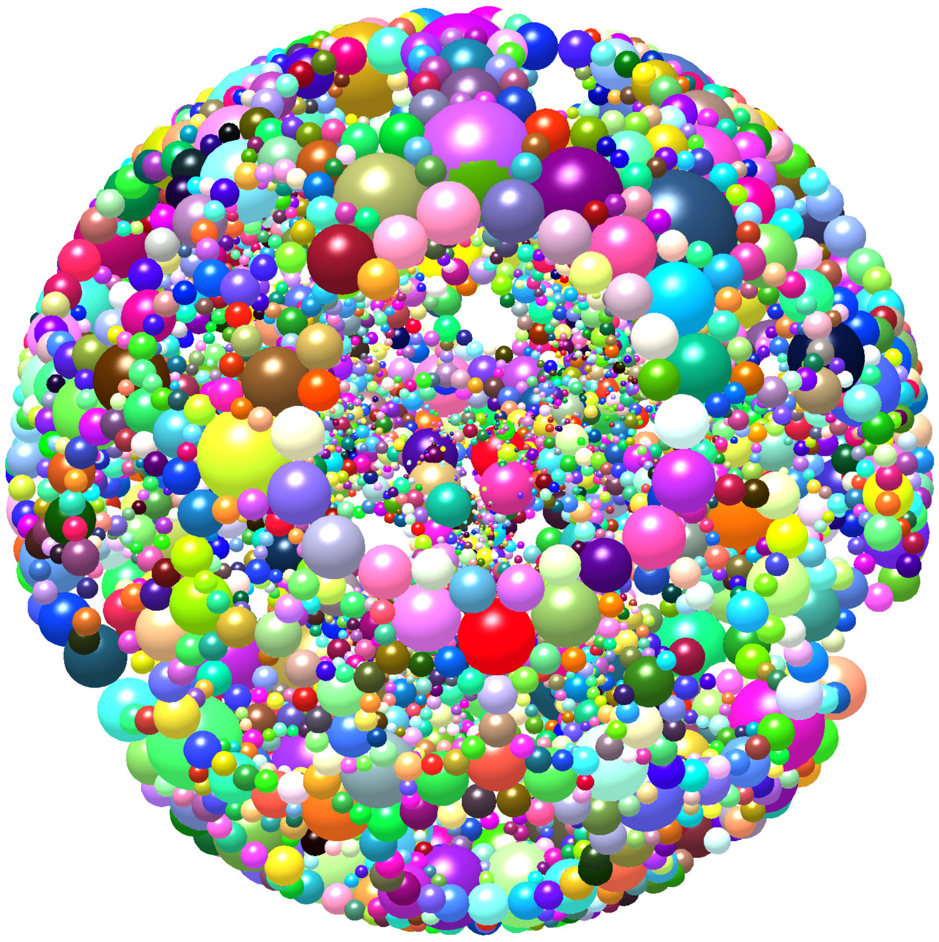
The next chapter will present a complete framework to extract multiple types of primitives simultaneously from unorganized point clouds.



(a) Carbon Nano Tube.



(b) ADN model with highly clustered spheres with intersections.



(c) Sphere Packing with various radii of scattered spheres.

Figure 3.20. Biochemistry (Fig. 3.20(a)-(b)) and complex sphere models (Fig. 3.20(c)). Detected spheres are colorized randomly.

Chapter 4

Geometric primitive extraction

4.1 Introduction and related work

A wide range of manufactured objects are comprised of simple geometric primitives such as planes, spheres, cylinders, etc. (i.e. engine parts of BMW X6M in Fig. 4.1). Geometric extraction plays a key role in surface reconstruction, reverse engineering and shape analysis. Recently, geometric primitive extraction and parameter estimation have been actively investigated but still remain an open problem for noisy scanned data with outliers. Existing methods can be classified in different ways according to the data input (point clouds or meshes), the specific applications or the extraction techniques. RANSAC-based [SWK07, LWC$^+$11], Hough-based [BELN11, RvdH05] and region growing methods [CSAD04, VS05, LM12] are applied widely for primitive extraction. All three of the above methods try to approximate data points by geometric patches and then use these patches to reconstruct the surface. In this chapter, a method for extracting reliable geometric primitives (planes, spheres and cylinders) from noisy data is presented and could be used in various applications such as primitive-based alignment [RDvdHV07] and quality control. The descriptive parameters of the extracted primitives are estimated more accurately than by other methods.

RANSAC [BF81] is the most popular technique for model selection and extraction. The method attempts to fit a primitive using a minimum number of data points selected randomly. The idea is used to detect primitives iteratively from point clouds that may include noisy data and outliers [SWK07, LHS08, LkW14]. Schnabel et al. [SWK07] proposed a general method for primitive extraction using RANSAC and the largest connected component. The primitive corresponding to the largest connected component is extracted first at each iteration. The concept of *largest connected component (lcc)* may achieve poor results when different types of primitives are compared. An example is shown in Fig. 4.6(b), which indicates that the size of the lcc of the false positive plane could be larger than the one of a real plane. RANSAC is known as a non-deterministic and greedy technique that immediately removes points associated with a detected primitive. This may cause a problem at the intersection of

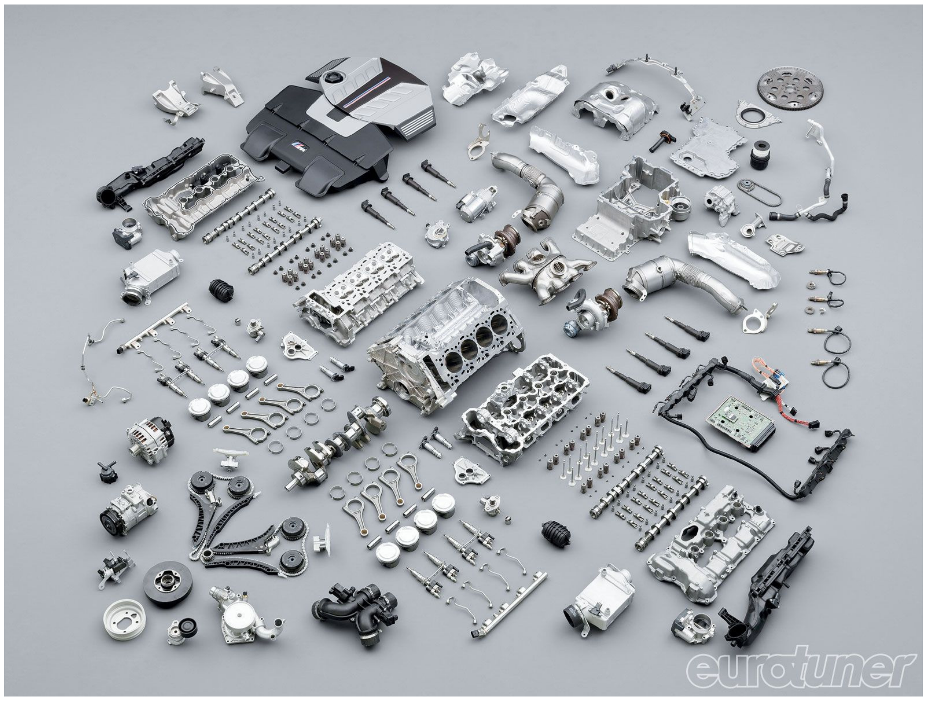


Figure 4.1. Most parts of the BMW X6M engine are composed of geometric primitives such as cylinders and planes. Image taken from Eurotuner.

primitives, since these removed points no longer contribute to the extraction process. Moreover, setting several threshold values for a RANSAC-based method [SWK07] is not trivial for complex and noisy point clouds. Therefore, it often requires post-processing to remove false positive primitives.

The Hough transform [DH72] extracts a primitive by using a voting scheme after transforming the data into a space that is representative of the primitive type. The method is used widely for the extraction of planes [BELN11, HB13], cylinders [RDvdHV07] and spheres [CVC14] from 3D data. Hough-based techniques are efficient for simple shapes such as lines and circles. However, memory requirements and parameter space quantization are the main issues for higher dimensional models such as cylinders and spheres. As described in Chapter 2-3 and references [TCL15a, TCL16b], we have proposed a robust framework outperforming the Hough transform for the extraction of cylinders and spheres from point clouds. The proposed framework for primitive extraction presented in this chapter exploits the approaches in [TCN$^+$14, TCL16b] to extract curved surfaces from point clouds.

Region growing [CSAD04, VS05] determines separated regions whose points share similar properties. The process starts by selecting a number of seeds which are usually based on principal curvatures or planarity criteria. Then, the points satisfying certain conditions are added to the seed regions. Some methods [LCL09, WGY$^+$12] have also used region growing to extract primitives sequentially (planar surfaces first, then curved surfaces). Differential geometry information such as normal vectors and curvatures are used to select seed points. The sequential approach can lead to *model confusion* in which parts of large cylinders or spheres are considered as planes (Fig. 4.2).

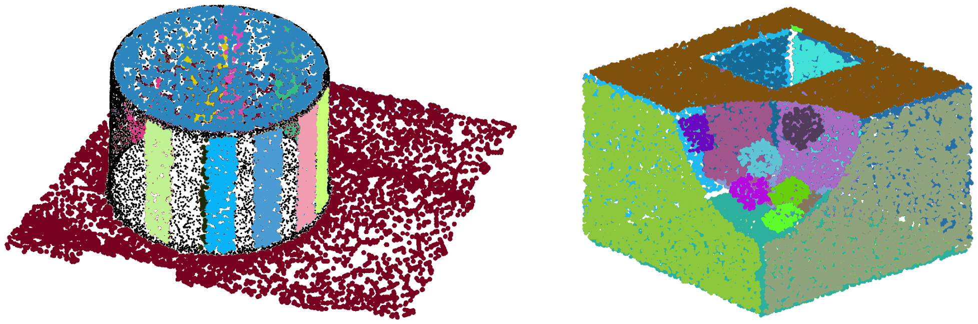


Figure 4.2. Cylindrical and spherical parts are approximated by planes. This problem is called *model confusion*.

The framework in this chapter proceeds in the opposite way by extracting curved surfaces first and planar surfaces afterward. This strategy helps to avoid model confusion as demonstrated in Section 4.3. Moreover, existing methods often require that meshes or point clouds be denoised beforehand. Contrarily, the proposed method processes point clouds directly despite the presence of noise and outliers.

Several approaches extract multiple types of primitives in one framework [AFS06, AP10]. Attene et al. [AFS06, AP10] cluster faces or vertices hierarchically according to a global model fitting priority until a desired number of clusters is reached. However, this can possibly lead to wrong model selection, especially with noisy data and outliers. Each vertex must be assigned to a cluster to deal with a predefined number of clusters. A primitive may thus be extracted approximately since it may include points belonging to another primitive. In the framework proposed here, a clever sequential process is carried out to avoid this problem. Curved surfaces are extracted first, then planar surfaces are extracted afterwards. Remaining points are considered as outliers.

The contributions of the chapter are summarized in the following:

— First, the proposed approach focuses on extracting reliable primitives, not approximate ones.

— An efficient sequential framework is proposed: (i) first curved surfaces are extracted robustly. (ii) planar surfaces are then extracted without model confusion.

— The proposed method works directly on unorganized point clouds without any preprocessing steps, a condition that is not met by other approaches.

— Combining advanced methods for curved surface extraction [TCL15a, TCL16b] and RANSAC-based plane extraction [SWK07] helps the proposed framework to achieve better results than other methods even in the presence of noise and outliers.

The chapter is structured as follows. First, curved surfaces such as cylinders and spheres are extracted reliably based on the strategy detailed in Section 4.2.1. Then planar surfaces are extracted using a RANSAC-based approach according to the method described in Section 4.2.2. Experimental results are shown and discussed in Section 4.3. The conclusion is presented in Section 4.4.

4.2 Framework for primitive extraction

In this section, the proposed framework for extracting reliable primitives is presented in detail. The overall diagram is shown in Fig. 4.3. The input is 3D scanned point cloud data captured from manufactured objects $\mathbf{P} = \{\mathbf{p}_i | i = 1, ..., N\}$ with normal vector $\vec{\mathbf{n}}_i$ at each $\mathbf{p}_i$. If $\vec{\mathbf{n}}$ is not available, it can be computed using the approach described in [HDD+92]. The framework is comprised of two sequential steps: curved surface extraction (Algorithm 6) and planar surface extraction (Algorithm 7). The output results are the descriptive parameters of primitives and their associated points.

4.2.1 Curved surface extraction

Efficient methods are proposed for the extraction of a *single primitive type* from point clouds (see Chapter 2 [TCL15a] for cylinder extraction and Chapter 3 [TCL16b] for sphere extraction). The extraction of a single primitive type is simple because the primitive sought is known a *priori*. First, initial points are extracted according to the type of primitive. These initial points are usually chosen based on principal curvatures. An iterative extraction process starts at an initial point and is then followed up by a robust validation method corresponding to the primitive type (cylinders in Chapter 2 and [TCL15a] and spheres in Chapter 3 and [TCL16b]). Finally, reliable primitives and their descriptive parameters are determined by Mean Shift clustering in the parameter space.

In the current framework, a more difficult task is addressed which consists of the *simultaneous extraction* of spheres and cylinders followed by a plane extraction step. As mentioned above, the algorithms in Chapter 2-3 and references [TCL15a, TCL16b] use initial points to start the extraction process. These initial points are chosen based on principal curvatures and can be affected by noise and outliers. Therefore, to guarantee the generality of the framework, initial points are chosen randomly in this chapter. As discussed in the simplification problem presented in [GH98], 0-3% of the points are enough to express a simplified model. In this chapter, 0.5% of the data points are used as initial points $\mathbf{I}$ for all of the experiments.

Starting at a given initial point in $\mathbf{I}$ and its neighbors, a hierarchical fitting procedure is applied to determine which type of primitive (cylinder or sphere) should be processed and extracted. Then the appropriate extraction algorithm for cylinders (see Chapter 2 and [TCL15a])

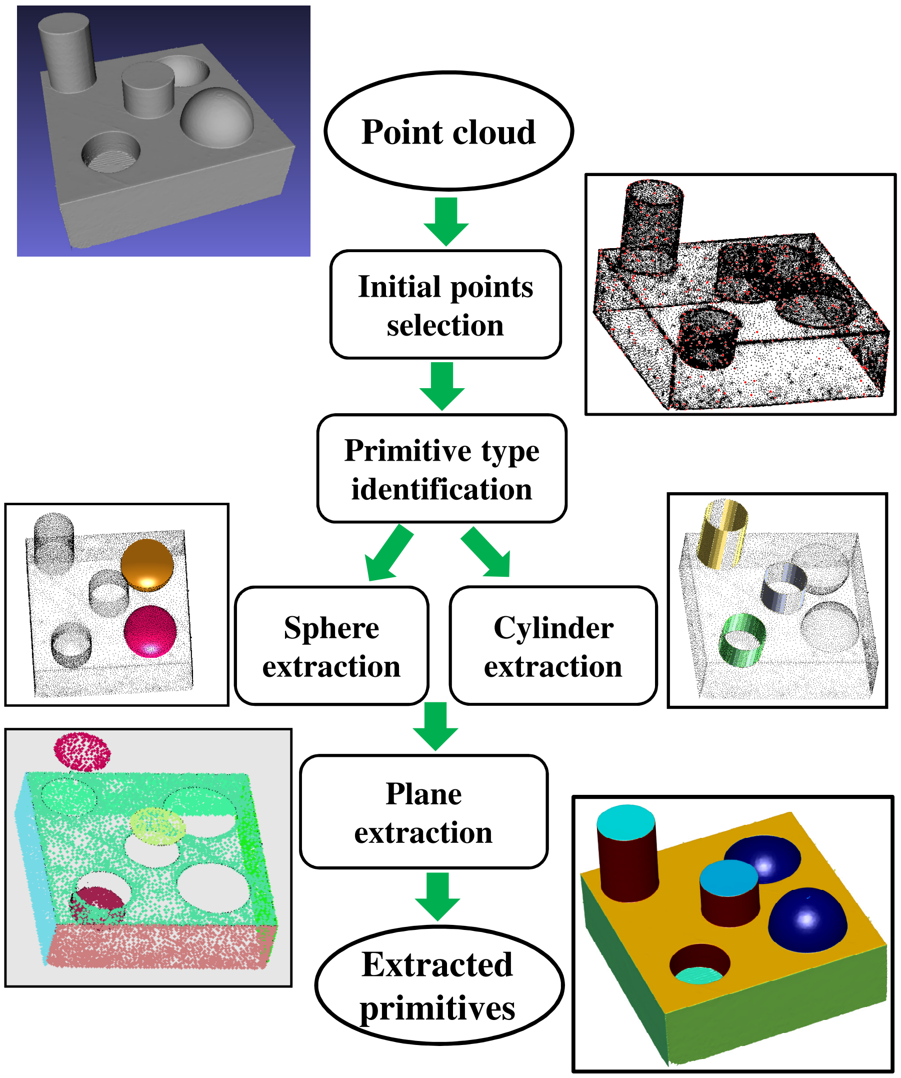


Figure 4.3. The proposed framework for reliable primitive extraction. First, the curved surfaces and their parameters are extracted robustly at initial points. Then planes are extracted using a RANSAC-based approach [SWK07].

or spheres (see Chapter 3 and [TCL16b]) is applied to extract the points and the parameters associated with each primitive for which the type has been identified by the hierarchical fitting procedure (Algorithm 6).

Primitive type identification

Hierarchical fitting is applied to determine the type of best primitive T at a given point and its neighbors. The primitive fitted with the smallest error value is considered as the best-fit primitive. Primitive fitting is discussed in [AP10] and the interested reader is referred to this paper for details. The main idea of this approach is summarized as follows. A given point I_k and its neighbors are used as the input and stored in the set $\mathbf{M}$, which is used

Algorithm 6: Algorithm for extracting curved surfaces (cylinders and spheres)

```
Data: Point cloud $\mathbf{P}$, normal vectors $\mathbf{n}$.
Result: Primitives' parameters $\mathbf{\Phi}^*$ and their associated points $P^*_{\mathbf{\Phi}}$.
I:=sampling 0.5 % of $N$;
while $k \leq |\mathbf{I}|$ do
    1. M:=a given point $I_k$ and its neighbors ($|\mathbf{M}|$=2$k$-nn);
    2. $T := primitive\_selection(\mathbf{M})$ in Section 4.2.1;
    3. if $T == 2$ then
        $\{\mathbf{\Phi}, P_{\mathbf{\Phi}}\} \leftarrow sphere\_extraction$ in Chapter 2 and [TCL16b];
    4. else if $T == 3$ then
        $\{\mathbf{\Phi}, P_{\mathbf{\Phi}}\} \leftarrow cylinder\_extraction$ in Chapter 3 and [TCL15a];
    end
    5. $k$++;
end
6. $\mathbf{\Phi}^*$:=Mean Shift clustering on $\mathbf{\Phi}$;
```

in Algorithm 6. Although the algorithms for cylinder and sphere fitting are presented in Chapter 2-3, these algorithms are presented again in this section to make it easier to follow for the reader. Primitive fitting errors for planes, cylinders and spheres are computed based on the following strategy.

Plane fitting: The centroid of the plane is calculated as $\overline{\mathbf{p}} = 1/|\mathbf{M}| \sum_{j\in\mathbf{M}} \mathbf{p}_j$ and the normal $\vec{\mathbf{n_p}}$ to the plane is the eigenvector associated with the smallest eigenvalue of the symmetric covariance matrix $\mathbf{CV} = \sum_{j\in\mathbf{M}}(\mathbf{p}_j - \overline{\mathbf{p}})^T(\mathbf{p}_j - \overline{\mathbf{p}})$, where $\mathbf{p}_j$ is a row vector. Therefore, the fitting error E_p of the plane is computed as:

$$E_p^2 = \sum_{j\in\mathbf{M}} |\langle \vec{\mathbf{n_p}}, \mathbf{p}_j - \overline{\mathbf{p}} \rangle_2|^2 \tag{4.1}$$

Sphere fitting: A sphere is represented by its center $\mathbf{c}$ and radius r, as shown in Fig. 5.1. They are computed by minimizing the algebraic distance [Pra87] as follows:

$$[c_x, c_y, c_z, r^2 - c_x^2 - c_y^2 - c_z^2]^T = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T b \tag{4.2}$$

where

$$\mathbf{H} = \begin{pmatrix} 2p_{1x} & 2p_{1y} & 2p_{1z} & 1 \\ 2p_{2x} & 2p_{2y} & 2p_{2z} & 1 \\ \vdots & \vdots & \ddots & \\ 2p_{\mathbf{M}x} & 2p_{\mathbf{M}y} & 2p_{\mathbf{M}z} & 1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} ||\mathbf{p}_1||_2^2 \\ ||\mathbf{p}_2||_2^2 \\ \vdots \\ ||\mathbf{p}_\mathbf{M}||_2^2 \end{pmatrix} \tag{4.3}$$

Therefore, the fitting error E_s for the sphere is computed as:

$$E_s^2 = \sum_{j\in\mathbf{M}} (\|\mathbf{p}_j - \mathbf{c}\|_2 - r)^2 \tag{4.4}$$

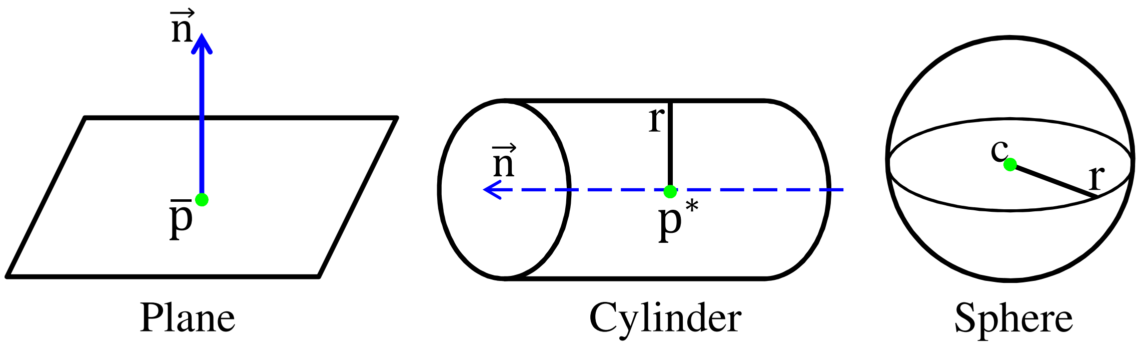


Figure 4.4. Models of the primitives of interest in this chapter (plane, sphere and cylinder).

Cylinder fitting: A cylinder is described by 3 parameters: axis orientation $\overrightarrow{\mathbf{a}}$, a point on the axis $\mathbf{p}^*$ and radius r, as shown in Fig. 5.1. These descriptive parameters are computed in three steps.

The first step consists in the computation of the orientation of the axis of the cylinder $\overrightarrow{\mathbf{a}}$. The axis is the vector which is being the most orthogonal to all the normal vectors $\vec{\mathbf{n}}_j$ with $j \in \mathbf{M}$. Therefore, a positive semi-definite matrix $\mathbf{C} = \sum_{j=1}^{\mathbf{M}} \mathbf{n}_j^T \mathbf{n}_j$ is computed and analyzed in which $\vec{\mathbf{n}}_j$ is a row vector. The eigenvector of $\mathbf{C}$ corresponding to the smallest eigenvalue is considered as the axis orientation $\overrightarrow{\mathbf{a}}$, and $\mathbf{C_x}$ and $\mathbf{C}_y$ are the two other eigenvectors that create a coordinate frame in the plane normal to $\overrightarrow{\mathbf{a}}$.

In the second step, $\tilde{\mathbf{p}}_j = [\langle \mathbf{p}_j, \mathbf{C}_x \rangle_2, \langle \mathbf{p}_j, \mathbf{C}_y \rangle_2]$ is the 2D projection of $\mathbf{p}_j \in \mathbf{M}$ on the plane with normal $\overrightarrow{\mathbf{a}}$ that passes through the origin O. These projected points distribute on the plane as a circle. Then the problem of determining the radius r and a point $\mathbf{p}^*$ on the axis reduces to the one of determining the radius r and center $\tilde{\mathbf{c}}$ of the circle. Algebraic fitting [Pra87] is used to compute the radius r and center $\tilde{\mathbf{c}} = [\tilde{c}_x, \tilde{c}_y]$. Therefore, algebraic distances are minimized in the following equation:

$$[\tilde{c}_x, \tilde{c}_y, r^2 - \tilde{c}_x^2 - \tilde{c}_y^2]^T = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \tag{4.5}$$

where

$$\mathbf{A} = \begin{pmatrix} 2\tilde{p}_{1x} & 2\tilde{p}_{1y} & 1 \\ 2\tilde{p}_{2x} & 2\tilde{p}_{2y} & 1 \\ \vdots & \vdots & \ddots \\ 2\tilde{p}_{\mathbf{M}x} & 2\tilde{p}_{\mathbf{M}y} & 1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} ||\tilde{\mathbf{p}}_1||_2^2 \\ ||\tilde{\mathbf{p}}_2||_2^2 \\ \vdots \\ ||\tilde{\mathbf{p}}_{\mathbf{M}}||_2^2 \end{pmatrix} \tag{4.6}$$

The center $\tilde{c}$ of the circle is transformed back in 3D coordinates and is considered as the point $\mathbf{p}^*$ belonging to the axis of the cylinder, i.e., $\mathbf{p}^* = \tilde{c}_x \mathbf{C}_x + \tilde{c}_y \mathbf{C}_y$. Therefore, the fitting error E_c of the cylinder is computed by

$$E_c^2 = \sum_{j \in \mathbf{M}} (\|(\mathbf{p}_j - \mathbf{c}) \times \mathbf{a}\|_2 - r)^2 \tag{4.7}$$

Primitive type identification: The type of best-fit primitive T is based on the smallest value of fitting errors for plane E_p^2, sphere E_s^2 and cylinder E_c^2.

$$T = \operatorname{argmin}\{E_p^2, E_s^2, E_c^2\} \tag{4.8}$$

where T=1,2,3 corresponds to the planes, spheres and cylinders respectively. If T=1, the process is skipped, because planar surfaces are extracted last.

Otherwise, the parameters and associated points of a reliable primitive corresponding to a given initial point are extracted and stored into a set $\{\mathbf{\Phi}, P_{\mathbf{\Phi}}\}$. Initial points that belong to the same primitive share the same descriptive parameters. Hence, a Mean Shift clustering [CM02] technique is used to determine the final parameters of the extracted primitives. The reader is referred to Section 2.4.2 and references [TCL15a, TCL16b] for details.

When all of the curved surfaces (cylinders and spheres) are extracted, their associated points are removed from the point cloud before proceeding with the plane extraction step. The remaining points and their normal vectors are called $\mathbf{P}'$ and $\mathbf{n}'$ respectively.

4.2.2 Plane extraction

Once the processing described in Section 4.2.1 is complete, the hypothesis is made that the remaining points belong to planar surfaces. A RANSAC-based method is used to extract planes from these remaining points. The general idea for extracting a plane from the point cloud is summarized in Algorithm. 7. First, three random points are selected for plane fitting. Then inliers satisfying the conditions (*,**) below are used to decide whether the fitted plane is reasonably good or not. Afterwards, the detected plane is further evaluated by the largest connected component strategy proposed in [SWK07].

The inliers that are extracted at each iteration of Algorithm 7 satisfy the following conditions:

— The distance from an inlier to the fitted plane is smaller than a predefined distance $d(\mathbf{p'}_i, \mathbf{P}_f) < \varepsilon$ (*)

— The angle difference between the normal vector of an inlier and the normal to the fitted plane is smaller than a predefined angle $\arccos(\mathbf{n'}_i, \mathbf{n}_{\mathbf{P}_f}) < \alpha$ (**)

— In this chapter, $\alpha = 10$ and $\varepsilon = 3 * sd$ in which sd is the signed distance field value computed in [HDD+92, TAL13].

L **selection:** Parameter L in Algorithm 7, which is the number of iterations needed to find a good plane, is determined based on the following analysis. Three points are selected randomly for plane fitting ($m = 3$). Let p_t be the probability that RANSAC selects one good set of three random points in L trials and let w be the probability that an inlier is selected each time ($w := \frac{\#inliers \in \mathbf{P'}}{|\mathbf{P'}|}$). w is often guessed roughly from the data. Now, assuming that the m points needed for estimating a model are selected independently, w^m is the probability that all m points are inliers and $1 - w^m$ is the probability that at least one of the m points is an outlier, a case which implies that a bad model will be estimated from this point set. That

Algorithm 7: RANSAC-based method for single plane extraction.

```
Data: Remaining points $\mathbf{P}'$ after simultaneous cylinder and sphere extraction with
      associated normal vectors $\mathbf{n}'$.
Result: Plane and its associated points.
$k := 0$; $best\_model := \varnothing$; $best\_score := 0$ ;
$L$ computed in Eq. 4.10 ;
while $k \leq L$ do
    1. Pick 3 random points for plane fitting, with the procedure described in Section 4.2.1;
    2. Inlier:= points satisfying the conditions $\{\varepsilon, \alpha\}$;
    3. if $|Inlier| > best\_score$ then
        3.1. $best\_score := |Inlier|$ ;
        3.2. Fit a plane ($\mathbf{P}_f$) to inlier;
        3.3. $best\_model := P_f$;
    end
    4. $k$++;
end
5. Plane validation by largest connected component proposed in [SWK07];
```

probability to the power of L is the probability that the algorithm never selects a set of m points which are all inliers and this must be the same as $1 - p_t$. Consequently,

$$1 - p_t = (1 - w^m)^L \tag{4.9}$$

After taking the logarithm of both sides of Eq. 4.9, L is computed by

$$L = \frac{\log(1 - p_t)}{\log(1 - w^m)} \tag{4.10}$$

In Algorithm 7, the best plane is evaluated and extracted by the maximum number of inliers. In order to deal with noise and outliers, the concept of largest connected component (lcc) is used for plane validation and helps to discard poorly detected planes. First, a bitmap is created by projecting each inlier on the detected plane. The largest cluster among connected components extracted from the bitmap is then determined. If the size of the lcc group is larger than a predefined threshold (cc) and the number of inliers is larger than a predefined value (τ), the detected plane is reliable. Otherwise, it is considered as poorly detected and rejected. As noted in [SWK07], the cell resolution β of pixels in the bitmap should correspond to the distance σ between data points. Therefore, in this chapter, we set $cc = 30$, $\tau = 50$ and $\beta = 3 * \sigma$ for all of the experiments.

The extracted plane and its points are removed and $\mathbf{P}'$ is updated for the next plane extraction. Multiple planes are extracted from the remaining points $\mathbf{P}'$ using Algorithm 7 repeatedly until the number of remaining points is not sufficient for plane detection (i.e. $|\mathbf{P}'| < \tau$) or the maximum of iterations is reached.

4.3 Results and discussion

The proposed method was tested on many synthetic and real models. The performance on the estimation of primitive parameters is compared to the performance of other methods [AFS06, SWK07, CSAD04]. Moreover, the computation time and threshold values are also listed for different models.

4.3.1 Synthetic models

First, the proposed method was tested on synthetic models composed of spheres, cylinders and planes, as shown in Fig. 4.5. Observing the results, all primitives are extracted correctly without model confusion.

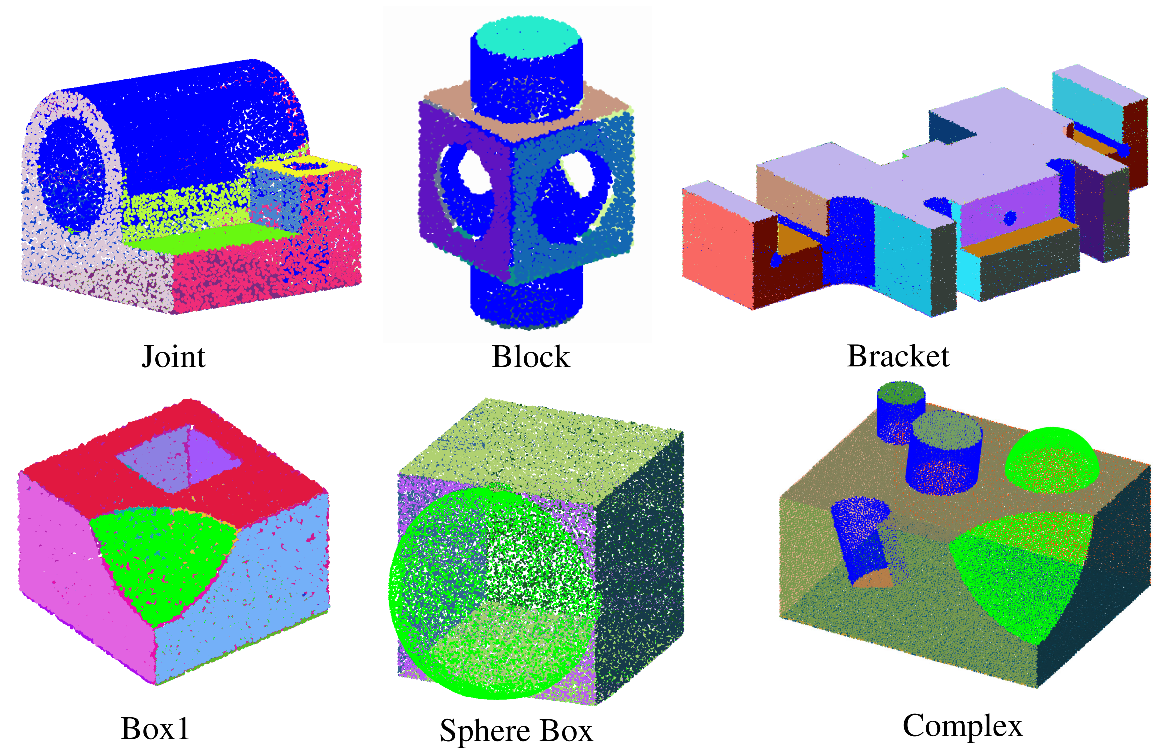


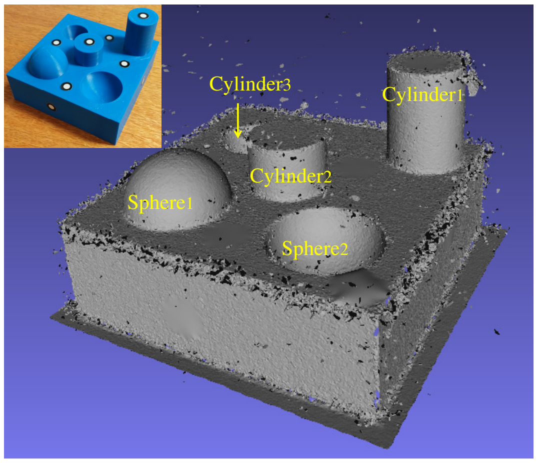
Figure 4.5. Results of the proposed framework on synthetic models. Blue: cylinders, green: spheres, randomly selected colors: planes.

4.3.2 Real scanned models

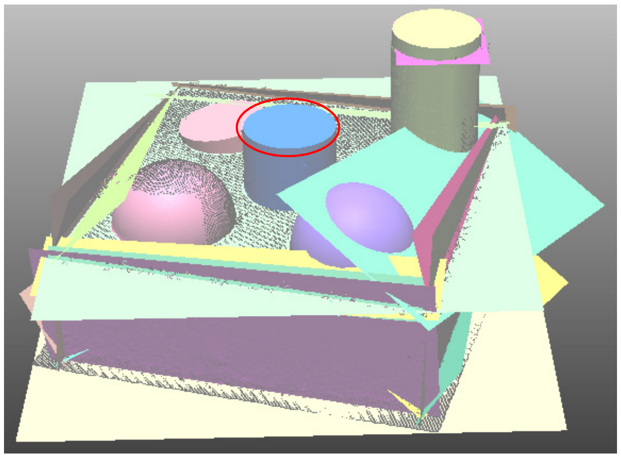
The proposed method was also tested on real data scanned with the Creaform GO!SCAN handheld scanner and compared to the RANSAC-based method, as shown in Fig. 4.6. Both methods work directly on the point cloud. The RANSAC-based method extracts primitives sequentially with the primitive corresponding to the largest connected component being extracted first. The primitive and its associated points are removed immediately once the primitive has been detected. The source code of the RANSAC-based method [SWK07] is available at http://www.danielgm.net/cc/. The quality of the results is evaluated by visual inspection. A plane on top of the cylinder at the center is missed (red ellipse) and several false positive planes in Fig. 4.6(b) are detected. The reason for this behavior is that the global threshold values used by RANSAC cannot deal with the complexity of the shapes. Therefore, the method requires a post-processing step to validate the extracted primitives and find missing ones. The proposed method succeeds in detecting all primitives without false positives and without post-processing. Since the curved surfaces were extracted first, they no longer affect the plane extraction step, and model confusion is avoided.

The proposed method is compared to existing methods based on region growing [CSAD04] and hierarchical face clustering [AFS06]. Variational shape approximation (VSA) [CSAD04] selects seeds using the smallest distortion error. The growing process is then carried on using the $\mathcal{L}^2$ and $\mathcal{L}^{2,1}$ norms. However, the hierarchical face clustering (HFC) method uses a global list of priority to merge vertices. Both methods require that the number of primitives be defined beforehand. Primitives are extracted but only approximately by these methods since the desired number of primitives must be met. VSA fails for the given model since two planes are merged in the same group (cyan primitive at the bottom of Fig. 4.7(a)). HFC extracts the different primitives correctly, but the intersections are not sharp, see Fig. 4.7(b). This leads to less accuracy for the estimation of primitive parameters (See Table 4.1). The proposed method finds primitives of all types successfully with sharp boundaries between primitives that are in contact with each other. Robust primitive extraction described in Section 4.2 results in sharp boundaries as shown in Fig. 4.7(c).

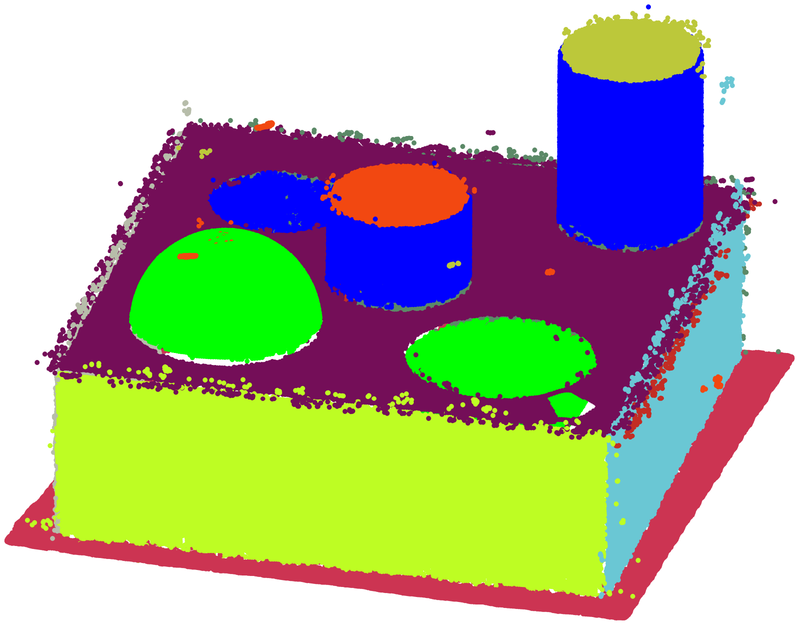
For quantitative comparison, the estimated radii for cylinders and spheres are listed in Table 4.1 for RANSAC, HFC and our method. Results show that the proposed framework outperforms the RANSAC-based and hierarchical face clustering methods.



(a) CAD model printed with a Stratasys Dimension 1200es 3D printer (top left of the figure) and scanned with the Creaform GO!SCAN handheld 3D scanner. Markers are used by the scanner for self positioning.

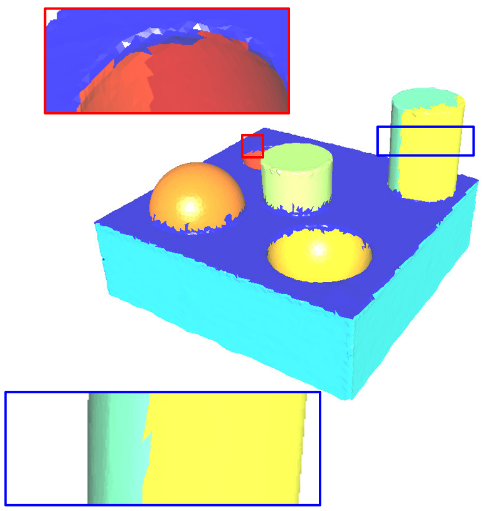


(b) Results of the RANSAC-based method [SWK07]. A plane is missed (red ellipse) and several false positive planes are extracted.

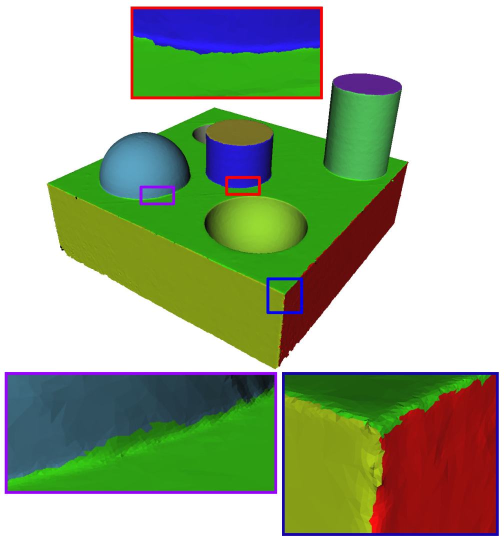


(c) Result obtained with the proposed method. All primitives are detected correctly.

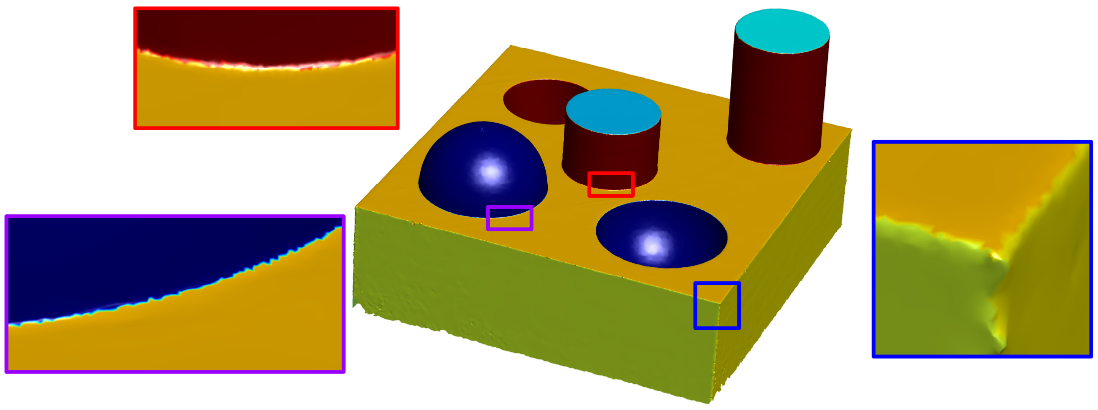
Figure 4.6. Primitives are detected by the RANSAC-based method [SWK07] (with $\epsilon = 3sd, \alpha = 10, \beta = 3\sigma, \tau = 200$) and the proposed approach (k-nn=50, $c_{sc} = 2, s_{sc} = 3$). Cylinders are in blue, spheres are in green and planes are in randomly selected colors.



(a) Variational shape approximation [CSAD04].



(b) Hierarchical face clustering [AFS06].



(c) Our method.

Figure 4.7. Primitives are detected by variational shape approximation (VSA) [CSAD04], hierarchical face clustering (HFC) [AFS06] and the proposed approach. Zoom-in parts show the intersection between primitives. Note that VSA and HFC require meshes with low level of noise and no outliers.

Table 4.1. Comparison of radii of detected cylinders and spheres estimated by RANSAC in Fig. 4.6(b), hierarchical face clustering in Fig. 4.7(b) and our method in Fig. 4.6(c). $\bar{r}$ is the estimated radius. $|E|$ is absolute error between ground-truth and the estimated radius.

Primitive	Groundtruth	RANSAC [SWK07]		HFC [AFS06]		OUR [TCL15b]	
	r (cm)	$\bar{r}$	$\|E\|$	$\bar{r}$	$\|E\|$	$\bar{r}$	$\|E\|$
Cylinder1	15	14.32	0.68	14.81	0.19	14.83	0.17
Cylinder2	15	15.12	0.12	14.95	0.05	15.04	0.04
Cylinder3	15	14.81	0.19	14.87	0.13	14.82	0.18
Sphere1	20	20.05	0.05	20.16	0.16	19.98	0.02
Sphere2	20	19.69	0.31	20.14	0.14	19.95	0.05
			0.232		0.134		0.092

Table 4.2. Threshold values used for the different models and the computation time. P-plane, C-cylinder, S-sphere.

Model Name	#Points	k-nn	c_{sc}	s_{sc}	#Primitives			Time (s)			
					N_P	N_C	N_S	T_P	T_C	T_S	$T_{\sum}$
Joint Fig. 4.5	50 000	20	1	-	9	3	-	11.4	23.6	-	40.75
Block Fig. 4.5	50 000	20	1	-	8	3	-	14.32	18.97	-	40.96
Bracket Fig. 4.5	130 460	20	1	-	22	9	-	71.24	22.63	-	110.37
Box1 Fig. 4.5	50 000	20	-	1	11	-	1	23.1	-	0.8	30.26
Sphere Box Fig. 4.5	62 183	20	-	1	6	-	1	25.9	-	8.97	38.91
Complex Fig. 4.5	73 671	20	1	1	9	3	2	25.1	6.1	4.3	42.98
Print Model Fig. 4.6(c)	230 738	50	2	3	8	3	2	101.42	29.32	15.6	164.67
Print Model1 Fig. 4.7(c)	117 858	30	1	1	8	3	2	30.9	16.32	12.25	65.71

4.3.3 Selection of threshold values

As presented in Algorithm 6 and 7, threshold values need to be set for the proposed framework.

— k-nn is the number of neighbors around a given point which are used for computing the normal. Therefore, $|\mathbf{M}|$ used in Section 4.2.1 for primitive type determination is set to $2k$-nn.

— c_{sc} is a validation score that is used for cylinder detection [TCL15a].

— s_{sc} is a validation score that is used for sphere detection [TCL16b].

Table 4.2 summarizes the threshold values chosen for the different models. The threshold values for plane extraction are described and set in Section 4.2.2.

For the synthetic model in Fig. 4.5, the data is noise-free and without outliers, $\{k\text{-nn}, c_{sc}, s_{sc}\}$ are set to small values, while $\{k\text{-nn}, c_{sc}, s_{sc}\}$ are set to larger values to deal with noisy data and outliers in the scanned models in Fig. 4.6(c) and Fig.4.7(c).

4.3.4 Computation time

The framework is implemented on MATLAB on a 3.2 GHz Intel Core i7 platform. The computation time of the main steps, which consist of plane, cylinder and sphere extraction, is listed in Table. 4.2. The computation time depends on the number of points and the number of primitives in the point clouds.

Several aspects could be improved to accelerate the extraction process. First, instead of using 0.5 % of the points as initial points, the points could be selected by differential geometry information. However, this could be affected by noise and outliers. Secondly, curved surface extraction could be processed in parallel, because this step works without point removal. The extraction at each initial point might be processed on separate processors. In this chapter, these ideas have not been investigated to guarantee the generality of the framework and the comparison with other methods. This could be the object of future work.

4.4 Conclusion

In this chapter, a novel framework is introduced to extract reliable primitives from unorganized point clouds. First, curved surfaces such as cylinders and spheres are extracted robustly and their associated points are removed from the point cloud. The robustness of primitive extraction is guaranteed (for cylinder [TCL15a] and sphere [TCL16b]). Planar surfaces are then detected using a RANSAC-based method for the remaining points. The framework is tested on many synthetic and real scanned data with good results. The extracted primitives could be used in several applications such as solid modeling and model abstraction, etc.

Future work will integrate other primitives such as cones and tori into the extraction framework. A C++ and parallel implementation is also planned to accelerate the process.

Part II: Primitive-based Alignment and Applications

The second part of the thesis focuses on primitive-based registration and its applications. As mentioned in Chapter 1, registration between the scans of manufactured objects is still a challenge for existing methods. In fact, registration of the scanned data with its CAD model is even a harder problem, which is frequently used in inspection application. Fortunately, as presented in previous chapters, geometric primitives are already extracted from unorganized point clouds [TCL15a, TCL16b, TCL16a]. Therefore, we have proposed a novel framework that uses the extracted primitives as a novel descriptor to solve these problems. First, the alignment of two synthetic geometric primitives is presented. Then the idea is applied to a real application, which registers two 3D data such as CAD models, meshes and point clouds. The content of this part is organized as follows:

— First, a novel approach to align single types of primitives will be presented in Chapter 5. With several significant improvements, the proposed approach achieves better a performance than the existing methods do.

— The proposed approach in Chapter 5 is then applied to align the scanned data of a product with its CAD model. The problem often exists in quality control and inspection applications, which helps to detect the difference or distortion between datasets. The complete framework and comparison result are described in more detail in Chapter 6.

Moreover, in this part, we have also described sharp feature extraction as an additional research topic. An automatic sharp feature extraction from 3D data is presented in Chapter 7. The extracted sharp features could be integrated into the framework proposed in Chapter 6 to create an automatic registration.

Chapter 5

Geometric Primitive Alignment Revisited

5.1 Introduction and related work

3D sensors are being used more and more widely in several applications, especially in industrial manufacturing. Since the field of view of these sensors is limited, it is necessary to scan an object from different viewpoints in order to cover its complete surface. These local scans are then used in the modeling and reconstruction process. Among common problems, a registration step is inevitable to transform all of the scans into a common coordinate reference frame. Registration is also referred to as alignment or matching, so these terms are used interchangeably in this chapter.

The 3D registration problem can be classified in several ways. On one hand, according to the rigidity of the objects, registration is categorized as rigid and non-rigid as summarized in [TCL$^+$13]. On the other hand, following the quality of the result, registration methods can also be classified as coarse registration [PMW05, DRLS15] and fine registration [BM92, RL01, SMFF07]. In this chapter, we investigate the problem of *coarse registration* in which the parts are aligned roughly with each other. From now on, the word registration will refer to the coarse registration problem.

Currently, registration methods often address the problem of aligning meshes or point clouds having similar densities of data points. 3D descriptors [DRLS15] are proposed and keypoints [TSDS13] are extracted from 3D data for correspondence and shape matching. However, most existing descriptors and keypoints fail to register two scans of man-made objects, because the main challenge is to propose a descriptor that is able to deal with symmetrical and similar geometries from the primitive surfaces.

Some research has been conducted to align datasets using primitives as descriptors. The primitives could be linear and circular features as in [SLC$^+$08] or planes as in [Bos12]. Linear features [CS05] between two planes and circular features [SLC$^+$08] from 2D and range images of buildings and urban scenes are used. Bosche's approach [Bos12] uses planes that are extracted at a single point selected by the user. The Iterative Closest Point (ICP) algorithm is then used to compute the transformation using three pairs of plane correspondences. Rabbani et al. [RDvdHV07] have also used geometric primitives such as planes, cylinders and spheres for registration and modeling. The method minimizes the sum of squares of the differences of parameters by using Levenberg-Marquardt optimization (LM) [Mar63]. The proposed approach exploits this idea and brings many improvements in terms of convergence and accuracy.

The contributions of the chapter are summarized as follows.

— A robust approach is proposed to align geometric primitives. Each type of primitive (plane, sphere, cylinder) is described in Section 5.2.3.

— A new minimization technique is exploited to achieve better results than other techniques with respect to convergence and error of descriptive parameters.

5.2 Primitive alignment

5.2.1 Quaternion representation

Rigid registration consists of finding a unique transformation $\mathbf{T} : \mathbb{R}^3 \mapsto \mathbb{R}^3$, which brings two datasets as close as possible. This transformation is expressed as a 3x3 rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t} = \{t_x, t_y, t_z\}$. Horn [Hor87] has proposed a closed-form solution for determining a rotation matrix using a quaternion representation. With a given quaternion $\mathbf{q} = \{q_0, q_1, q_2, q_3\}$, the rotation matrix $\mathbf{R}$ is computed as

$$\mathbf{R} = \begin{pmatrix} 2q_0^2 + 2q_1^2 - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2q_0^2 + 2q_2^2 - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2q_0^2 + 2q_3^2 - 1 \end{pmatrix} \tag{5.1}$$

with the constraint $\|\mathbf{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$ which guarantees a rotation operator in $\mathbb{R}^3$ having only three degrees of freedom.

5.2.2 Rabbani's approach and associated drawbacks

Rabbani et al. [RDvdHV07] have proposed an approach that combines registration and modeling in one framework. At the registration step, quaternion and geometric primitives are used to align two scans called $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. The transformation $\mathbf{T}$ is calculated using a least-squares algorithm in the parameter space of the primitives. In other words, this step minimizes the sum of the square of errors of primitive parameters as follows:

$$\min_{\{\mathbf{R}(\mathbf{q}),t\}} \sum_{i}^{|\mathbf{C}|} \sum_{j}^{\mathbf{M_i}} \boldsymbol{\Delta}_{u_{ij}}^2 \tag{5.2}$$

where $\boldsymbol{\Delta}_{u_{ij}} = u_{ij}^{\mathbf{S}_1} - \mathbf{T}(u_{ij}^{\mathbf{S}_2})$ and u_{ij} is the j-th parameter of the i-th correspondence that has $|\mathrm{M}_i|$ criteria of parameter convergence. $\mathbf{C}$ is the set of correspondences. For example, when the first correspondence pair is plane-plane, $|\mathrm{M}_1|$ is equal to 2 since there is a criterion for the error on the normal vector and another criterion for the error on the distance to the origin. The mathematical equations for each single primitive are expressed later in Section 5.2.3.

Levenberg-Marquardt optimization [Mar63] is used to solve the non-linear least-squares problem in Eq. 5.2. Partial derivatives of $\boldsymbol{\Delta}$ w.r.t of rotation $\mathbf{R}$ and translation $\mathbf{t}$ are required and computed as

$$\left\{ \frac{\partial \boldsymbol{\Delta}}{\partial q_0}, \frac{\partial \boldsymbol{\Delta}}{\partial q_1}, \frac{\partial \boldsymbol{\Delta}}{\partial q_2}, \frac{\partial \boldsymbol{\Delta}}{\partial q_3}, \frac{\partial \boldsymbol{\Delta}}{\partial t_x}, \frac{\partial \boldsymbol{\Delta}}{\partial t_y}, \frac{\partial \boldsymbol{\Delta}}{\partial t_z} \right\}$$

These derivatives are used to compute the Jacobian matrix $\mathbf{J}$ and update the transformation parameter $\boldsymbol{\Gamma} = \{\mathbf{q}, \mathbf{t}\}$ as follows:

$$\boldsymbol{\Gamma} = \{\mathbf{q}, \mathbf{t}\} = \{q_0, q_1, q_2, q_3, t_x, t_y, t_z\} \tag{5.3}$$

$$\boldsymbol{\Gamma}_{t+1} = \boldsymbol{\Gamma}_t - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}\boldsymbol{\Delta} \tag{5.4}$$

$$\mathbf{J} = \frac{\partial \boldsymbol{\Delta}}{\partial \boldsymbol{\Gamma}} \tag{5.5}$$

where t is the iteration step, and $\mathbf{J}$ is the Jacobian matrix. Observing Eq. 5.4, the Levenberg-Marquardt optimization consists of a combination of Gauss-Newton and Steepest descent methods by using a non-negative scalar λ, which is called the Levenberg-Marquardt parameter. The selection of λ and its impact on convergence are discussed in [PTVF07]. Generally, Levenberg-Marquardt optimization achieves better convergence compared to Gauss-Newton and Steepest descent methods taken separately.

Rabbani's approach faces two problems that have an significant impact on the convergence of the result.

— First, the LM optimization generally provides a solution to an unconstrained problem. Therefore, the method uses the quaternion method in Eq. 5.1 but it does not guarantee that the solution provides a unit quaternion $\|\mathbf{q}\| = 1$ for each iteration of the optimization loop. This leads to poor results for the rotation matrix $\mathbf{R}$. This behavior has been tested in the experiments reported in Section 5.3. The method proposed in this chapter uses an interior point technique with constraints to guarantee that condition ($\|\mathbf{q}\| = 1$) is satisfied.

— Second, the objective function for cylinders lacks a robust criterion for alignment, which has an adverse effect on convergence. The proposed approach complements the objective function with a new criterion (Eq. 5.9) that guarantees good convergence .

5.2.3 Proposed approach for primitive alignment

In this section, the detailed framework for registering two primitives is presented. These primitives can be planes, cylinders and spheres. We assume that the correspondence pairs $\mathbf{C} = \{\mathbf{S_1}, \mathbf{S_2}\}$ between primitives are already available. The transformation $\mathbf{T} = \{\mathbf{R}, \mathbf{t}\}$ is computed by minimizing the following cost function:

$$\begin{aligned} &\min_{\{\mathbf{R}(\mathbf{q}),\mathbf{t}\}} \sum_{i}^{|\mathbf{C}|} \sum_{j}^{\mathbf{M_i}} \boldsymbol{\Delta}_{u_{ij}}^2 \\ &\text{subject to:} \|\mathbf{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \end{aligned} \tag{5.6}$$

where the error is defined as $\boldsymbol{\Delta}_{u_{ij}} = u_{ij}^{\mathbf{S}_1} - \mathbf{T}(u_{ij}^{\mathbf{S}_2})$.

To solve the non-linear least-squares problem in Eq. 5.6, the interior-point method [FGW02] is used. The interior point method guarantees that the constraint $\|\mathbf{q}\| = 1$ is always satisfied at each iteration of the optimization loop. The mathematical equations for aligning each type of primitive are described in the following.

Plane

Planes are geometric primitives frequently found on manufactured objects. A plane is represented by a normal vector $\vec{\mathbf{n}}$ and its distance to the origin ρ, as shown in Fig. 5.1(a). Assume that there are two planes with parameters $\{\vec{\mathbf{n}}_1, \rho_1\}$ and $\{\vec{\mathbf{n}}_2, \rho_2\}$, respectively.

To align plane 2 to plane 1, the error on the normal vector and the error on the distance to the origin are minimized as follows:

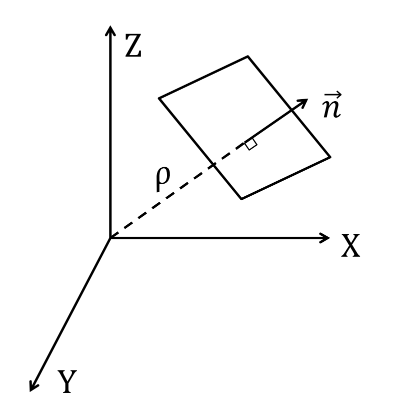
$$\vec{\mathbf{n}}_\delta = \vec{\mathbf{n}}_1 - \mathbf{R}\vec{\mathbf{n}}_2 \tag{5.7}$$

$$\rho_\delta = \rho_2 - \rho_1 + (\mathbf{R}\vec{\mathbf{n}}_2)^T\mathbf{t} \tag{5.8}$$

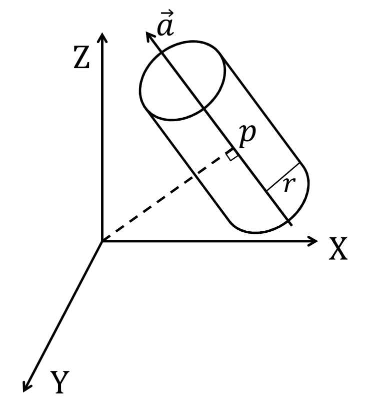
Eq. 5.7-5.8 are fed to Eq. 5.6 for the minimization step.

Cylinder

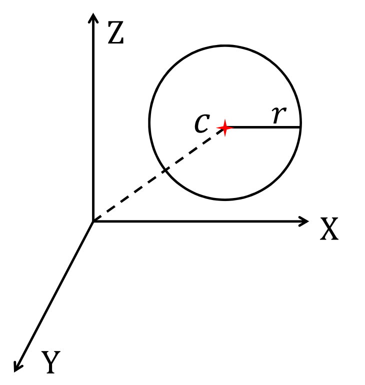
A cylinder is described by three parameters: its axis vector $\overrightarrow{\mathbf{a}}$, its radius r and a closest point to the origin on the cylinder axis $\mathbf{p}$, as shown in Fig. 5.1(b). To bring a cylinder $\mathbf{C}_2 = \{\overrightarrow{\mathbf{a}}_2; \mathbf{p}_2; r\}$ as close as possible to a cylinder $\mathbf{C}_1 = \{\overrightarrow{\mathbf{a}}_1; \mathbf{p}_1; r\}$, two points $\mathbf{p}_1$ and $\mathbf{p}_2$ and two



(a) Plane represented by normal vector $\vec{\mathbf{n}}$ and the distance to the origin ρ.



(b) Cylinder described by axis vector $\vec{\mathbf{a}}$, radius r and a closest point to the origin in axis $\mathbf{p}$.



(c) Sphere represented by center $\mathbf{c}$ and radius r

Figure 5.1. Representative parameters of primitives: plane, cylinder and sphere.

vectors $\overrightarrow{\mathbf{a}}_1$ and $\overrightarrow{\mathbf{a}}_2$ need to coincide.

$$\vec{\mathbf{a}}_\delta = \vec{\mathbf{a}}_1 - \mathbf{R}\vec{\mathbf{a}}_2 \tag{5.9}$$

$$\mathbf{p}_\delta = \mathbf{R}\vec{\mathbf{p}}_2 + \mathbf{t} - (\vec{\mathbf{a}}_\mathbf{R} * \mathbf{t}^T)\vec{\mathbf{a}}_\mathbf{R} - \vec{\mathbf{p}}_1 \tag{5.10}$$

where $\vec{\mathbf{a}}_\mathbf{R} = \mathbf{R}\vec{\mathbf{a}}_2$

Eq. 5.9-5.10 are fed to Eq. 5.6 for the minimization step.

Sphere

A sphere is simply represented by its center $\mathbf{c}$ and radius r, as shown in Fig. 5.1(c). To align a sphere 2 $\{\mathbf{c}_2, r\}$ to sphere 1 $\{\mathbf{c}_1, r\}$, the two centers need to coincide ($\|\mathbf{c}_\delta\|_2 = 0$).

$$\mathbf{c}_\delta = \mathbf{R}\mathbf{c}_2 + \mathbf{t} - \mathbf{c}_1 \tag{5.11}$$

Eq. 5.11 is fed to Eq. 5.6 for the minimization step.

The framework is applied to align a set of several multiple primitives simultaneously. The objective functions of each primitive pair are combined and minimized in Eq. 5.6. When the minimization ends, the transformation between the datasets is available.

5.3 Results and discussion

In this section, the results of the proposed approach are compared to the ones reported by Rabbani et al. [RDvdHV07]. Both methods use geometric primitives for alignment. The sets of primitives $\boldsymbol{\Psi}_1 = \{\mathbf{P}_1, \mathbf{C}_1, \mathbf{S}_1\}$ and their parameters are known *a priori*. Dataset $\boldsymbol{\Psi}_1$

is transformed by 100 sets of transformation $\mathbf{T}$ randomly to create a new set of primitives $\mathbf{\Psi}_2 = \{\mathbf{P}_2, \mathbf{C}_2, \mathbf{S}_2\}$. Then four experiments are conducted.

— Experiment 1: The transformation $\mathbf{T}'$ needed to transform plane $\mathbf{P}_2$ to plane $\mathbf{P}_1$ is estimated. The convergence is evaluated by

$$\cos(\angle(\mathbf{T}'(\mathbf{n}_{\mathbf{P}_2}), \mathbf{n}_{\mathbf{P}_1})) == 1 \tag{5.12}$$

— Experiment 2: The transformation $\mathbf{T}'$ needed to align cylinder $\mathbf{C}_2$ to cylinder $\mathbf{C}_1$ is estimated. The convergence is evaluated by

$$\cos(\angle(\mathbf{T}'(\mathbf{a}_{\mathbf{C}_2}), \mathbf{a}_{\mathbf{C}_1})) == 1 \tag{5.13}$$

— Experiment 3: The transformation $\mathbf{T}'$ to align a sphere $\mathbf{S}_2$ to sphere $\mathbf{S}_1$ is also computed. The convergence is then evaluated by

$$\|\mathbf{c}_{\mathbf{S}_1} - \mathbf{T}'(\mathbf{c}_{\mathbf{S}_2})\|_2 == 0 \tag{5.14}$$

— Experiment 4: All types of primitives are combined simultaneously into the framework. The convergence is reached if Eq. 5.12-5.14 are satisfied simultaneously.

The proposed framework is implemented on MATLAB on a 3.2 GHz Intel Core i7 platform. In their implementation of primitive alignment, Rabbani et al. [RDvdHV07] use Levenberg-Maquardt non-linear optimization which is implemented in the ***lsqnonlin*** function in MATLAB. The proposed framework rather uses the ***fmincon*** function which is implemented for the interior-point method. The percentage of success and the error on the parameters are investigated for both methods.

5.3.1 Success rate of the proposed method

With 100 random sets of transformations, four alignment experiments are carried out and the conditions in Eq. 5.12-5.14 are checked sequentially or simultaneously. Fig. 5.2 illustrates the success rate of both methods **LM_Rabbani** [RDvdHV07] and **IP_OUR** [TCL16a] in the four experiments.

Rabbani's method is able to align planar primitives, but fails with other primitives, as shown in Fig. 5.2. The approach proposed in this chapter achieves high success rates for all cases ($> 90\%$).

5.3.2 Errors on primitive parameters

The mean absolute error (MAE) values in Eq. 5.12-5.14 are computed for each of the 100 random sets of transformation.

— Experiment 1: The error on the plane normal after transformation is computed as follows:

$$\text{MAE}_P = \frac{1}{100}\sum_{i=1}^{100}\left(1 - |\cos(\angle(\mathbf{T_i}'(\mathbf{n}_{\mathbf{P}_2}), \mathbf{n}_{\mathbf{P}_1}))|\right) \tag{5.15}$$

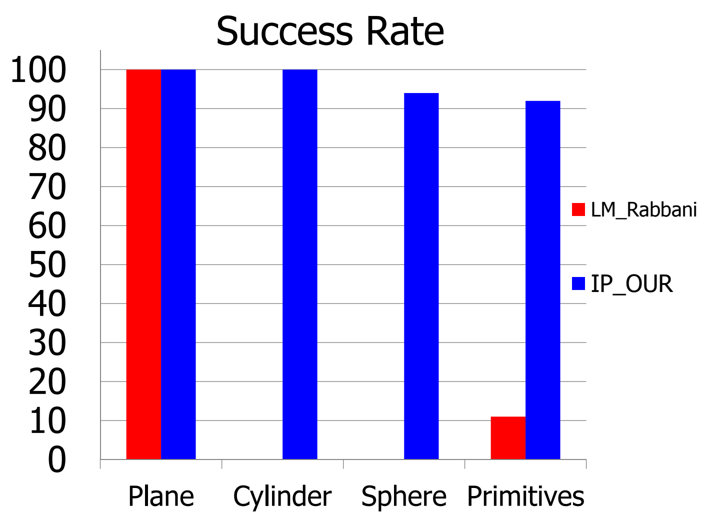


Figure 5.2. Success rate for Rabbani's method (LM_Rabbani) [RDvdHV07] and the proposed method (IP_OUR) [TCL16a]

— Experiment 2: The error on axis directions of the cylinders after transformation is computed as:

$$\text{MAE}_C = \frac{1}{100}\sum_{i=1}^{100}\left(1 - |\cos(\angle(\mathbf{T_i}'(\mathbf{a}_{\mathbf{C}_2}), \mathbf{a}_{\mathbf{C}_1}))|\right) \tag{5.16}$$

— Experiment 3: The error of the sphere centers after transformation is computed:

$$\text{MAE}_S = \frac{1}{100}\sum_{i=1}^{100}\left(\|\mathbf{c}_{\mathbf{S}_1} - \mathbf{T_i}'(\mathbf{c}_{\mathbf{S}_2})\|_2\right) \tag{5.17}$$

— Experiment 4: The error on combined primitives taken altogether is computed as:

$$\text{MAE}_{\sum} = \text{MAE}_P + \text{MAE}_C + \text{MAE}_S \tag{5.18}$$

A logarithm plot of the MAE for the four experiments is shown in Fig. 5.3. The proposed method clearly outperforms Rabbani's method in aligning primitives.

From the results of the experiments, it is concluded that the proposed method, which combines a new optimization technique and a new objective function, solves the problem of primitive-based alignment with better results with respect to convergence and error of descriptive parameters.

One more test is investigated to show the performance of the proposed method. The estimated transformation $\mathbf{T}'$ in the 4th experiment is compared to the ground-truth transformation $\mathbf{T}$ as follows:

— The error of the rotation matrix is defined as:

$$\text{E}_{\mathbf{R}} = \frac{1}{100}\sum_{i=1}^{100}\|\mathbf{R}_i - \mathbf{R}'_i\|_2 = 6.85e^{-7} \tag{5.19}$$

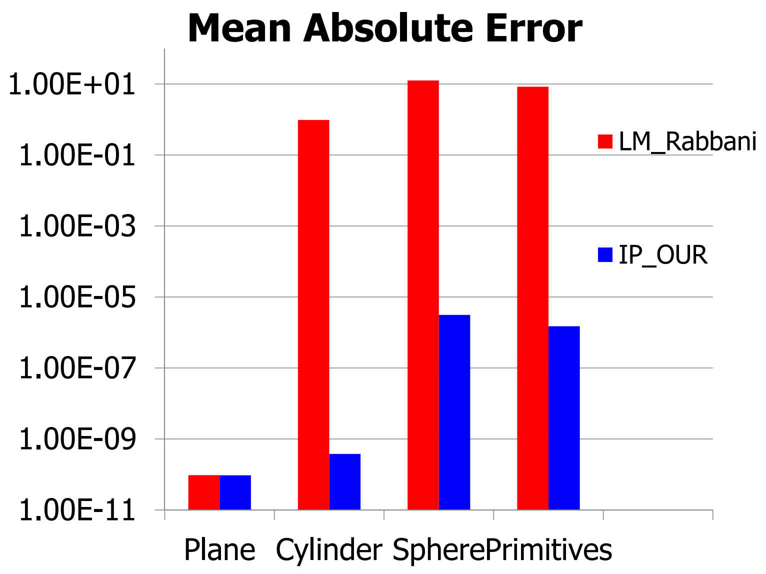


Figure 5.3. Logarithmic graph of mean absolute errors for Rabbani's method (LM_Rabbani) [RDvdHV07] and our method (IP_OUR) [TCL16a].

— The translation vector is commutative, so the error of translation is computed by:

$$\mathrm{E}_{\mathbf{t}} = \frac{1}{100} \sum_{i=1}^{100} \left| \|\mathbf{t}_i\| - \|\mathbf{t}'_i\| \right| = 2.68e^{-6} \tag{5.20}$$

The small values of $\mathrm{E}_{\mathbf{R}}$ and $\mathrm{E}_{\mathbf{t}}$ prove that the proposed method estimates the transformation between datasets accurately.

5.4 Conclusion

An efficient and robust method for registering two basic primitives together is proposed in this chapter. Using a new optimization technique and adding a complementary objective function, the convergence is guaranteed to achieve accurate results. The proposed method can benefit many other practical applications such as scan alignment, quality control and inspection, which are described and discussed in more detail in the next chapter.

Chapter 6

Primitive-based Registration and Applications

6.1 Introduction

Quality control and inspection play a key role in the manufactured parts production process. Through the quality control process, the quality of a product is improved and manufacturing errors are reduced or eliminated. For example, the difference in geometry between the product and its CAD design can be expressed by the surface distortion between them. Traditionally, a coordinate measuring machine (CMM) device is used to measure the distortion [MHSRP08, Li11]. The CMM collects several data points from the surface of the product. Data collection can be done manually by the user or by a robot programmed beforehand. Therefore, it is a time consuming process and requires that the surface of the manufactured object be hard enough to support the force of the CMM probe. Recently, 3D sensors have been used more widely in several applications, especially in industrial manufacturing. Several data points can be captured from the surface of manufacturing objects per second by such sensors.

No matter what the data collection approach is (CMM or 3D sensor), the emerging problem is how to align the data points with the CAD model to generate an error map. This alignment problem exists for part-to-CAD analysis, which determines the defect and distortion between a CAD model and a manufactured product. The bottom line for alignment is to find a minimum of three pairs of points to determine the rigid transformation [HZ03]. The possible solution is to use the traditional ICP or a 3D descriptor-based method for point pair correspondences. As demonstrated later in Section 6.2, the traditional ICP method [BM92] cannot be exploited in this case. Moreover, to the best of our knowledge, no descriptor-based or keypoint-based methods are able to deal with the above problem. Therefore, in this chapter, a novel approach is proposed to align scanned data of manufactured objects and their

CAD models using the primitives extracted by the methods presented in the previous chapters. This idea is generally applicable for the registration problem between point cloud-mesh, two point clouds and two meshes. This technique belongs to the *coarse & rigid registration* class of approaches, which computes an initial transformation for the fine registration step.

Some research has been conducted to align datasets using geometrical features such as linear and circular features [SLC$^+$08] or planes [Bos12]. Rabbani et al. [RDvdHV07] have used geometric primitives such as planes, cylinders and spheres for registration and modeling. The proposed approach exploits this idea and proposes many improvements in terms of convergence and accuracy.

The general idea of the proposed approach is to use common primitives between datasets for alignment. Therefore, in this Chapter, we assume that the primitives have already been extracted from the datasets (See Chapter 4 and [TCL15b] for primitive extraction from point clouds and [AFS06] for primitive extraction from meshes and CAD models).

The contributions of the proposed approach are summarized as follows.

— Most existing methods attempt to register two datasets by minimizing the distance (i.e. euclidean, geodesic) between data points. Therefore, the computation time is large even with several thousands of data points. The proposed method rather minimizes the difference of primitive parameters, so the computational time is significantly lower than for point-based methods [BM92, RL01] and descriptor-based methods [PMW05, DRLS15].

— The basic idea was proposed and tested in Chapter 5 for synthetic geometric primitives. Therefore, the idea is extended to real primitives extracted from the scanned data and the CAD model.

— Experiments on both synthetic and real scanned data show that the proposed method outperforms existing methods in all of the cases in terms of robustness and time complexity. The proposed framework is used in various applications such as data completion and part-to-CAD analysis.

6.2 Related work

In this section, related methods on point cloud to CAD alignment are discussed. The general idea of both fine and coarse registration approaches are summarized and presented with their shortcomings and advantages.

Fine registration is known as a method that brings two datasets as close as possible. Iterative closest point (ICP) [BM92] is the most popular approach for this problem. In the ICP algorithm, one point cloud, called the *target* (**S1**), is kept fixed, and the other one, called the *source* (**S2**) is transformed to align with the target. The algorithm iteratively estimates the transformation that minimizes the distance from the source to the target. The distance is computed for each point of the source data to its corresponding closest point on the target data. This iterative process is expressed as the following equation.

$$\min_{\{\mathbf{R},\mathbf{t}\}} \sum_{i=1}^{|\mathbf{S2}|} d(\mathbf{T}(\mathbf{p}_{i2}), \mathbf{p}^*_{i1}) = \min_{\{\mathbf{R},\mathbf{t}\}} \sum_{i=1}^{|\mathbf{S2}|} \|\mathbf{p}^*_{i1} - \mathbf{R}\mathbf{p}_{i2} - \mathbf{t}\| \tag{6.1}$$

where $\mathbf{T}$ is the rigid transformation between two datasets and $d(\mathbf{T}(\mathbf{p}_{i2}), \mathbf{p}^*_{i1})$ is the Euclidean distance between the transformed point of $\mathbf{p}_{i2} \in \mathbf{S2}$ and its corresponding closest point $\mathbf{p}^*_{i1} \in \mathbf{S1}$.

The distance $d(\mathbf{T}(\mathbf{p}_{i2}), \mathbf{p}^*_{i1})$ can be computed in several ways, and the methods are often classified according to the way $\mathbf{p}^*$ is determined such as point-to-point, point-to-plane, etc. The variants of ICP methods are summarized in [RL01]. In general, the idea is to establish 3D point correspondences between two overlapping point clouds and find the transformation $\mathbf{T}$. ICP methods have many advantages. They are easy to implement and applicable to various data types such as meshes, point clouds and range images. However, they also show several weaknesses. The computational burden is high because of the point correspondence search. Noise and outliers also has an adverse effect on convergence. Overlapping and direction requirements between datasets are necessary. Finally, a good initial transformation is needed in order to prevent the optimization from becoming trapped in a local minimum of Eq. 6.1.

More importantly, ICP cannot be applied for the alignment of scanned data and a CAD model because the point density between both types of structures are generally very different, which causes problems for the point-to-point search. Existing methods [BES11, YW14] often address the problem as follows: (i) data point density is increased by resampling the CAD model so it matches the density of data points of the scanned data. (ii) ICP is then applied to align two point clouds using a well-tuned set of parameters and a good initial transformation. The authors in [BES11] proposed a new data structure to improve the efficiency of closest point searching, while Li et al. [LS15] accelerates convergence by transforming the ICP algorithm into a variable step size iterative algorithm. However, a good initial transformation still remains a big challenge for these methods.

Coarse registration is the solution for some of the above registration problems. Two datasets are brought closer by the transformation computed by coarse registration. Several descriptors are proposed to encode the intrinsic and extrinsic properties of the surface of an object. Keypoints (interest points or feature points) are a subset of points that exhibit certain properties which distinguish them from the other points [TSDS13, Low04, BETVG08]. Correspondence pairs of the descriptors and keypoints from both datasets are then determined to compute the transformation. The computation of the transformation is carried out only once, not iteratively as for fine registration. However, not all of the detected descriptors or keypoints are used in the computation and some strategies such as RANSAC or the usage of a certain percentage of the found correspondences are applied to achieve more a robust performance. Excellent survey papers [PMW05, DRLS15, TSDS13] summarize the state-of-the-art descriptors and keypoints. Especially, some descriptors and keypoints such as spin-image [JH99], FPFH [RBB09] and SURF [BTVG06] are used widely in various applications. However, these methods require that the datasets be dense and overlapping.

The above descriptors and keypoints can fail with symmetric or isotropic surfaces such as planes and cylinders. *Spin image* [JH99] is investigated as an example. A spin image is created for an oriented point (i.e. a point with its surface normal) at a vertex on the surface mesh as illustrated in Fig. 6.1. All of the points within the support of the spin image are projected onto the local coordinate system at a vertex to create a 2D accumulator. This spin image is used to determine the corresponding position between different views of an object. However, surface of geometric primitives showing symmetries and similar geometries represents a challenge for the spin image approach. For instance, the spin image at any point on a plane is the same because the spin image descriptor is computed by using the height (α) to the local coordinate system (Fig. 6.1). Moreover, the bin size of the accumulator and the size of the support of the spin image also need to be considered carefully. Other descriptors (SURF [BTVG06], FPFH [RBB09] and etc.) also have similar problems.

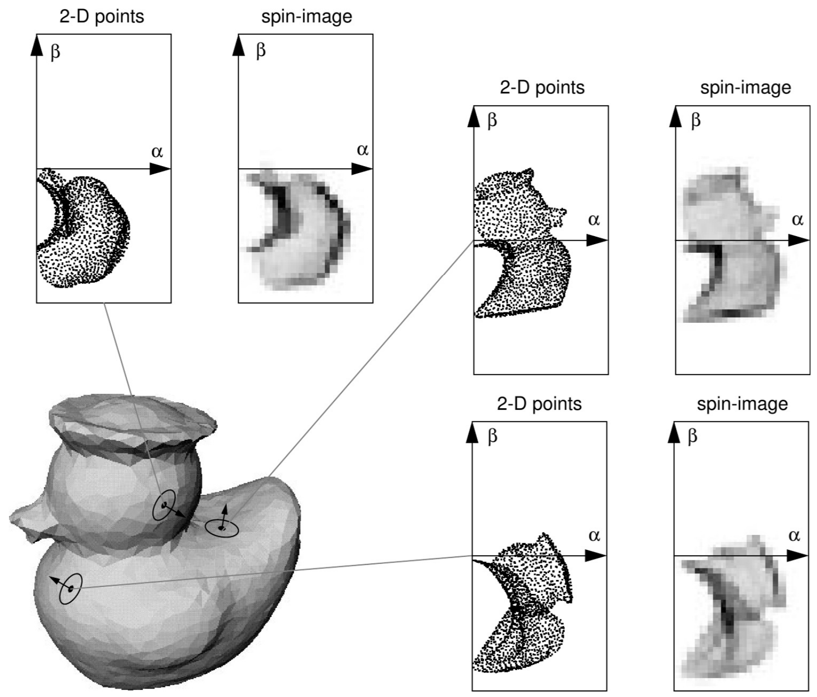


Figure 6.1. Examples of spin-images proposed in [JH99] for three oriented points on the surface of a rubber duck model.

Kim et al. [KLCK11] have proposed a method to align the 3D CAD model to the point cloud of a construction site. Principal component analysis (PCA) is first used for coarse registration followed by a Levenberg-Marquardt based optimization for fine registration. PCA [Jol86] determines three principal components that are used for coarse alignment. The complete point cloud is then required. Moreover, as mentioned earlier, vertices of the CAD model are distributed sparsely, so the method first applies a resampling of the CAD model beforehand. Since it is based on PCA analysis, the method is not suitable to align a partial point cloud (of a single view for instance) with the CAD model of the object.

Some methods use linear and circular features [SLC$^+$08] or planes [Bos12] for alignment. Linear features [CS05] between two planes and circular features [SLC$^+$08] from 2D and range images of buildings and urban scenes are used. Bosche [Bos12] uses planes that are extracted with the initial point selection made by user. ICP is then used to compute the transformation from three plane correspondences. Rabbani et al. [RDvdHV07] have used geometric primitives in registration and modeling. Their method minimizes the sum of squares of the differences of parameters using Levenberg-Marquardt optimization [Mar63]. The proposed approach in Chapter 5 has improved Rabbani's method using interior-point optimization and a new constraint function to achieve better convergence. This chapter exploits the approach on various applications, point cloud and CAD model alignment being an interesting instance. The complete framework is presented in the next section.

6.3 Framework for point cloud and CAD model alignment

The framework is illustrated in Fig. 6.2. First, geometric primitives from both the point cloud and the CAD model are extracted. For point clouds, the method described in Chapter 4 and [TCL15b] is used. For the CAD model, hierarchical face clustering [AFS06] is applied with a predefined number of clusters. Up to now, descriptive parameters and types of primitives are available. The user selects a set of primitive correspondences that are fed to the minimization step described in Section 5.2.3. Finally, after computing the transformation between the point cloud and the CAD model, an error map is computed to illustrate the difference between the transformed point cloud and the CAD model. Each step is presented in more detail in the next sections.

6.3.1 Hierarchical segmentation for CAD model

The algorithm of hierarchical face clustering (HFC) proposed in [AFS06] is summarized in this section. The algorithm belongs to a group of bottom-up approaches and attempts to merge neighboring triangles into representative clusters in which each cluster is composed of connected triangles approximated by a simple primitive such as a plane, sphere and cylinder. The detail of the clustering process is described as follows.

Let $\mathbf{M} = \{\mathbf{V}, \mathbf{F}\}$ be a CAD model that includes the set of vertices $\mathbf{V}$ and triangles $\mathbf{F}$. A dual graph $\mathbf{D} = \{\mathbf{C}, \mathbf{A}\}$ is then defined as follows: (i) each node $\mathbf{c}$ of $\mathbf{C}$ corresponds to a triangle in $\mathbf{F}$. (ii) the set $\mathbf{A}$ consists of the arcs that connect two nodes in $\mathbf{C}$ as shown in Fig. 6.3(a).

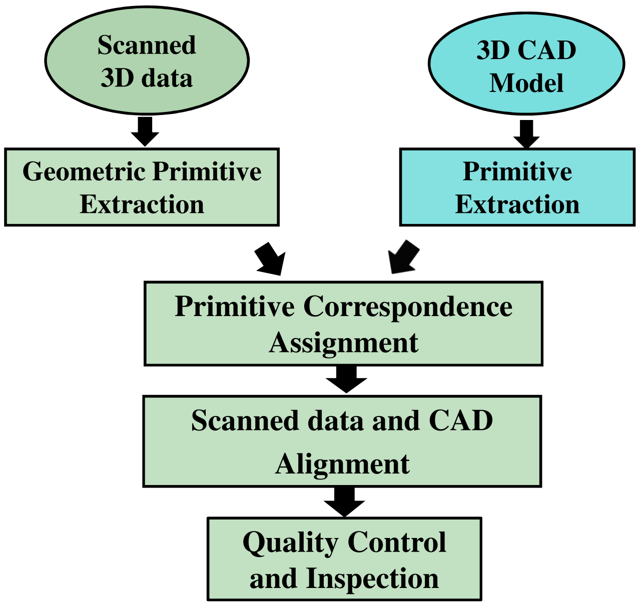

Scanned 3D data
3D CAD Model
Geometric Primitive Extraction
Primitive Extraction
Primitive Correspondence Assignment
Scanned data and CAD Alignment
Quality Control and Inspection

Figure 6.2. Complete framework for point cloud and CAD alignment. First, geometric primitives are extracted from both scanned data and the CAD model. Then a manual assignment is applied to find a set of primitive correspondences. Finally, the approach proposed in Chapter 5 is used to align the point cloud to the CAD model and to create an error map for further analysis such as inspection.

Merging two triangles into a single representative cluster corresponds to contracting a dual edge into a single node, as shown in Fig. 6.3. In the HFC approach, a priority queue is created in which all of the dual edges are sorted based on the cost of their contraction. At each step, the dual edge with the lowest cost is popped from the queue and contracted, and all of the edges incident to the new representative node are updated. Their costs are recomputed and their positions in the queue are updated according to the new cost.

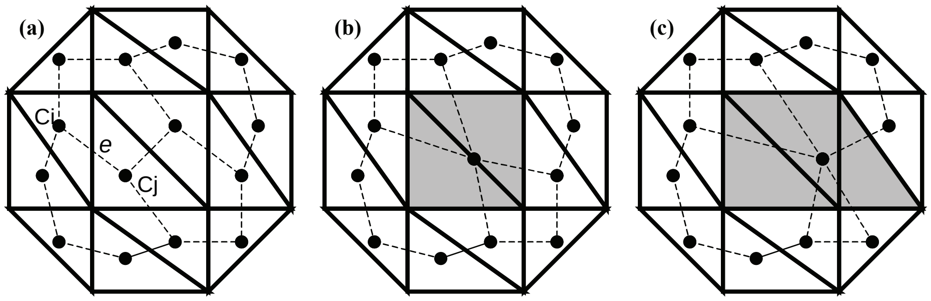


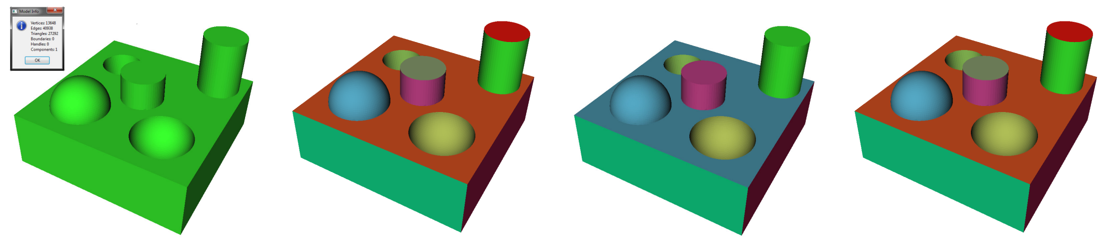
Figure 6.3. In (a) a triangle mesh and the corresponding dual graph are depicted. In (b) an arc of the dual graph has been contracted and the two triangles corresponding to the dual arc's end-points have been marked as belonging to the same single cluster. In (c) another arc has been contracted producing a resulting cluster made of three triangles. Figures taken from paper [AFS06].

The computation of the cost assigned to an edge is implemented as follows. Let $e = (\mathbf{c}_i, \mathbf{c}_j)$ be an edge ($e \in \mathbf{A}$) and let $\mathbf{M(c)}$ denote the set of vertices aggregated within the cluster represented by the node **c** of **C**. Therefore, $\mathbf{M} = \mathbf{M}(\mathbf{c}_\mathrm{i}) \cup \mathbf{M}(\mathbf{c}_\mathrm{j})$ is the set of points that would form a cluster as the result of the contraction of e. Let us consider the fitting error E of the best-fit primitive. This error value is computed as the sum of the squared residuals between the points of **M** and their fitting primitive described in Section 4.2.1. Using the values of fitting errors, the value (E) and type of best-fit primitive (T) are determined by

$$\begin{aligned} E &= \min\{E_p^2, E_s^2, E_c^2\} \\ T &= \operatorname{argmin}\{E_p^2, E_s^2, E_c^2\} \end{aligned} \tag{6.2}$$

where $\{E_p^2, E_s^2, E_c^2\}$ are the fitting errors of a plane, sphere and cylinder described in Section 4.2.1. The type of best-fit primitive T is based on the smallest value of fitting errors for a plane E_p^2, sphere E_s^2 and cylinder E_c^2.

At each step, the dual edge with the lowest cost is popped from the queue and contracted, and all of the edges incident to the new representative node are updated. Their cost is recomputed and their position in the queue is updated according to the new cost. The process is terminated when the desired number of remaining clusters is reached. The above algorithm is used to extract primitives **S1** and their parameters from the CAD model. The algorithm requires the desired number of segmented primitives. Fig. 6.4 shows an example of CAD segmentation with different values of cluster numbers k.



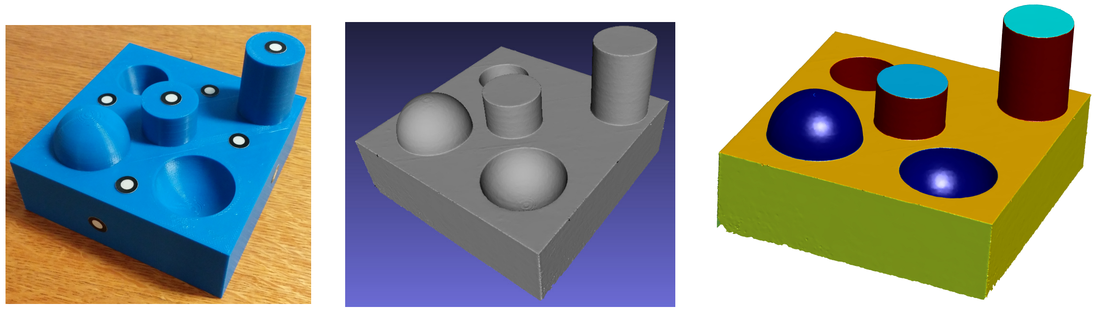
(a) CAD model and its detailed information. (b) Segmentation result with k=5. (c) Segmentation result with k=10. (d) All primitives are segmented separately with k=14.

Figure 6.4. CAD model and the results of HFC [AFS06] with different values of k. $k = 14$ is optimal value to segment primitives separately. The color of each primitive is assigned randomly.

6.3.2 Primitive extraction from the point cloud

3D geometric primitives are extracted from unorganized point clouds using the method proposed in Chapter 4 and [TCL15b]. The interested reader is referred to Chapter 4 for more details. The main algorithm is summarized as follows: (i) curved surfaces are extracted robustly in a first step. (ii) planar surfaces are then extracted without model confusion.

An example of primitive extraction is shown in Fig. 6.5. The output results are the descriptive parameters of primitives **S2** and their associated points. They are fed to the next step for correspondence assignment and minimization.



(a) The plastic object printed with a Stratasys Dimension 1200es printer.

(b) The data scanned by the GO!SCAN 3D sensor for the plastic object in Fig. 6.5(a).

(c) Primitives extracted by the method proposed in Chapter 4 and [TCL15b].

Figure 6.5. An example of primitive extraction detected from the scanne data using the method in [TCL15b]. Cylinders are in red, spheres are in blue and planes are in randomly selected colors.

6.3.3 Primitive correspondence and alignment

Chapter 5 has proposed an novel approach to align two sets of synthetic primitives consisting of planes, spheres and cylinders. The idea is exploited to align real primitives extracted from 3D data. The approach also determines the transformation between two sets of primitives by minimizing the difference of primitive parameter values. The minimization step is revisited briefly in this section to make reading easier.

Let **S1** and **S2** be two sets of primitive correspondences, then the set of correspondence pairs $\mathbf{C} = \{\mathbf{S_1}, \mathbf{S_2}\}$ is established. The transformation $\mathbf{T} = \{\mathbf{R}, \mathbf{t}\}$ between **S1** and **S2** is computed by minimizing the following cost function:

$$\min_{\{\mathbf{R}(\mathbf{q}),\mathbf{t}\}} \sum_{i}^{|\mathbf{C}|} \sum_{j}^{\mathbf{M}_i} \boldsymbol{\Delta}^2_{u_{ij}} \qquad \text{subject to:} \quad \|\mathbf{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{6.3}$$

with the error defined as $\boldsymbol{\Delta}_{u_{ij}} = u_{ij}^{\mathbf{S}_1} - \mathbf{T}(u_{ij}^{\mathbf{S}_2})$ and u_{ij} is the j-th parameter of the i-th correspondence that has $|\mathrm{M}_i|$ criteria of parameter convergence. For a given quaternion $\mathbf{q} = \{q_0, q_1, q_2, q_3\}$, the rotation matrix **R** is computed as

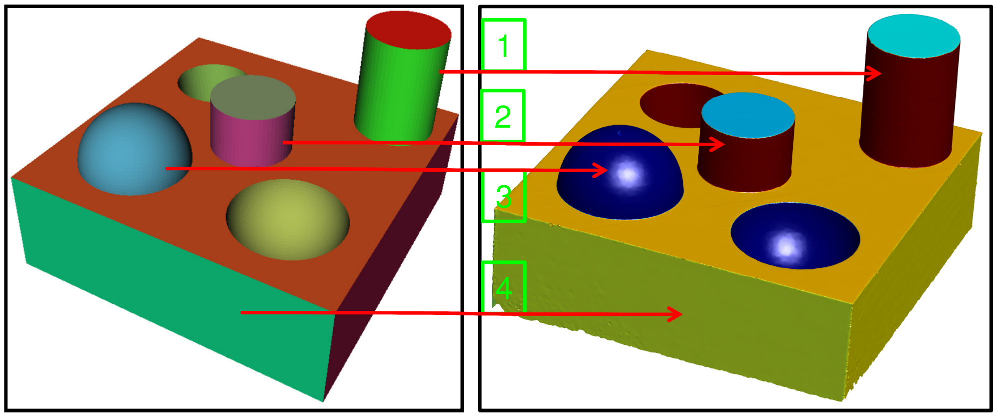
$$\mathbf{R} = \begin{pmatrix} 2q_0^2 + 2q_1^2 - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2q_0^2 + 2q_2^2 - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2q_0^2 + 2q_3^2 - 1 \end{pmatrix} \tag{6.4}$$

and the translation vector is $\mathbf{t} = \{t_x, t_y, t_z\}$.

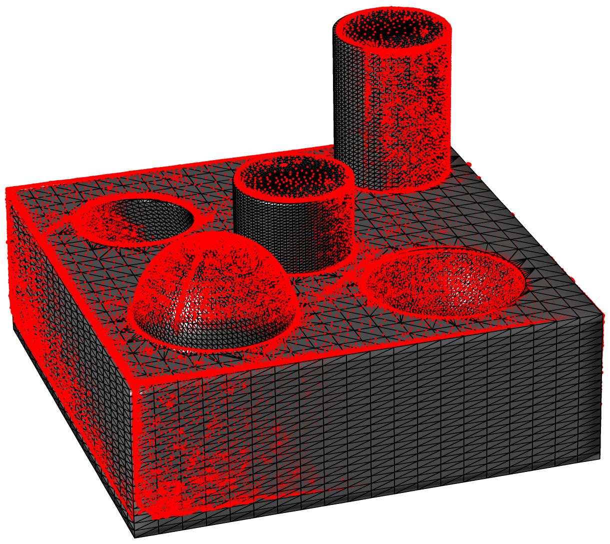
To solve the non-linear least-squares problem in Eq. 6.3, the interior point method [FGW02] is used. The expression of $\boldsymbol{\Delta}_{u_{ij}}$ for each type of primitive is given in Section 5.2.3. The objective functions of all of the primitive pairs are combined and minimized in Eq. 6.3. When the minimization is completed, the transformation between datasets is achieved.

Manual assignment of primitive correspondences

The minimization in Eq. 6.3 requires that the input be a set of primitive correspondences (ex. plane-plane, cylinder-cylinder and sphere-sphere). In this section, a manual assignment of primitive correspondence is described. The user decides which primitive in **S1** corresponds to a primitive in **S2** and the pair of primitives is stored into **C** (Fig. 6.6).



(a) An example of manual assignment of primitive correspondences from CAD model to scanned data. A set of four primitives including two cylinders, one sphere and one plane is illustrated.



(b) Good alignment result of the scanned data and its CAD model.

Figure 6.6. Manual assignment of primitive correspondences and alignment result. The scanned data (red) is superimposed on the CAD model, which is represented by triangular facets.

The following criteria are used to help the user assign primitive correspondences.

1. For curved surfaces such as cylinders and spheres, the radius is used as the first criterion for correspondence assignment.

2. Then, the relative positions among primitives are considered (i.e. parallel, orthogonal and coplanar).

3. An interactive GUI could be implemented to support the user, an example is shown in Fig. 6.6(a). From known parameters and the relative position between primitives, the user can select a set of correspondences easily.

Compared to raw point cloud, geometric primitives are a high level of representation used to abstract the 3D data. Therefore, the selection of primitive correspondences is much easier than the selection of point pairs in existing methods such as manual ICP and Bosche's method [BM92, Bos12].

An example of manual assignment of primitive correspondences is illustrated in Fig. 6.6. The alignment result is shown in Fig. 6.6(b) where the scanned data is superimposed on the CAD model.

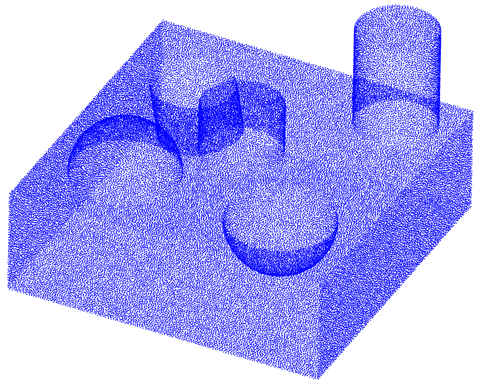
6.4 Results and discussion

In this section, we compare the performance of the proposed method to that of other methods such as ICP, Rabbani and a commercial Polyworks software (Section 6.4.1). Computation times for ICP and the proposed framework are compared in Section 6.4.2. Moreover, the proposed framework is applied in practical applications such as inspection and data completion.

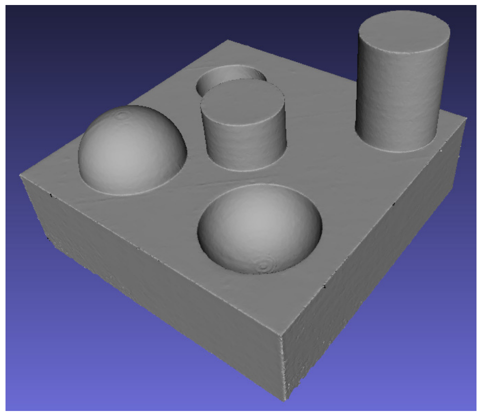
6.4.1 Comparison with ICP and Rabbani's method

First, the performance of our method [TCL16c] is compared to that of other methods such as ICP [BM92] and Rabbani's method [RDvdHV07]. However, as mentioned earlier, traditional ICP cannot be applied directly to register the scanned data with its CAD model because of the different density of data points. Therefore, the experiment is implemented as follows. Dense data points are resampled from the CAD model with 100 000 points, called **S1** and shown in Fig. 6.7(a). The scanned data captured by the GO!SCAN 3D sensor are shown in Fig. 6.7(b). Our ultimate goal is to bring two datasets as close as possible. To investigate the robustness of the different methods with respect to the initial relative position of **S1** and scanned data, the scanned data is transformed by 100 random transformations to create new datasets (**S2**). However, these transformations still guarantee that the datasets remain within the range of direction $(0 - 90^\circ)$, which increases the chance of convergence of ICP. ICP is then applied to determine the transformations between **S1** and **S2**.

For fair comparison, both Rabbani's and our methods also work on the datasets (**S1** and **S2**). Primitives are extracted from both datasets by using the method [TCL15b] and are then used for alignment.



(a) **S1:** 100 000 data points resampled from the CAD model in Fig. 6.4(a).



(b) **S2:** The data scanned by the GO!SCAN 3D sensor.

Figure 6.7. Resampled data points from 3D CAD model and scanned data from printed model (Fig. 6.5(a)).

The performance of the three methods is evaluated by how well the scanned data (**S2**) is aligned with the resampled data (**S1**). The distance from each point of transformed **S2** to its closest point in (**S1**) is computed. Therefore, the mean error of the distance E_d is computed as follows:

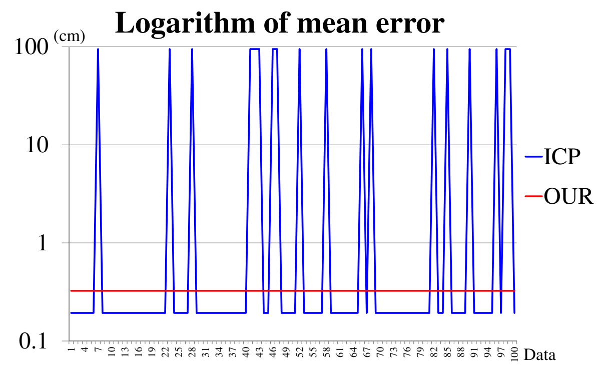
$$E_d = \frac{1}{|\mathbf{S2}|} \sum_{i=1}^{|\mathbf{S2}|} \|\mathbf{T}(\mathbf{p}_2) - \mathbf{p}_1^*\|_2 \tag{6.5}$$

where $\mathbf{T} = \{\mathbf{R}, \mathbf{t}\}$ is the transformation between **S2** and **S1**. $\mathbf{p}_2$ is a point of **S2** and $\mathbf{p}_1^*$ in **S1** is the closest point of transformed $\mathbf{p}_2$.

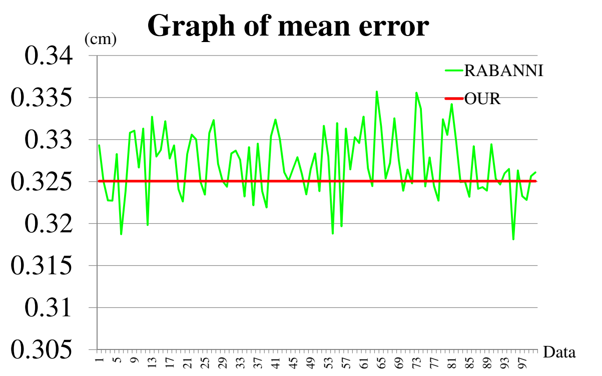
Observing Fig. 6.8, ICP and Rabbani's methods fail to align the datasets. Although the position and direction of both datasets are only slightly different, these performances are not robust with respect to the relative position between the point cloud and the CAD model. On the other hand, the proposed method always achieves robust results regardless of the random relative position between the two datasets.

A more difficult problem is addressed to prove the robustness of the proposed method. A set of 100 random transformations, which invert the direction of the scanned data with respect to that of the CAD model, is generated. The three methods are applied to these datasets. As predicted, ICP and Rabbani's method fail and a large value of distance error is observed and as well as a high rate of failure, as shown in Fig. 6.9. The proposed method still achieves the same value (0.32mm) obtained in the previous case.

Let us recall that contrarily to ICP, Rabbani's and our methods do not require a resampling step, but work directly on the original CAD model. The problem is to register the scanned data of the printed model with its CAD model. The plastic model is printed with a Stratasys Dimension 1200es printer (Fig. 6.5(a)) and scanned by the Creaform GO!SCAN 3D scanner.



(a) Comparison of ICP and OUR in terms of mean error.



(b) Comparison of RABBANI and OUR in terms of mean error.

Figure 6.8. Comparison of the robust performance of the proposed method (OUR) [TCL16c] with ICP [BM92] and Rabbani's method (RABBANI) [RDvdHV07]. The proposed method is robust and always achieves the same result of mean error (0.32mm) regardless of the transformation.

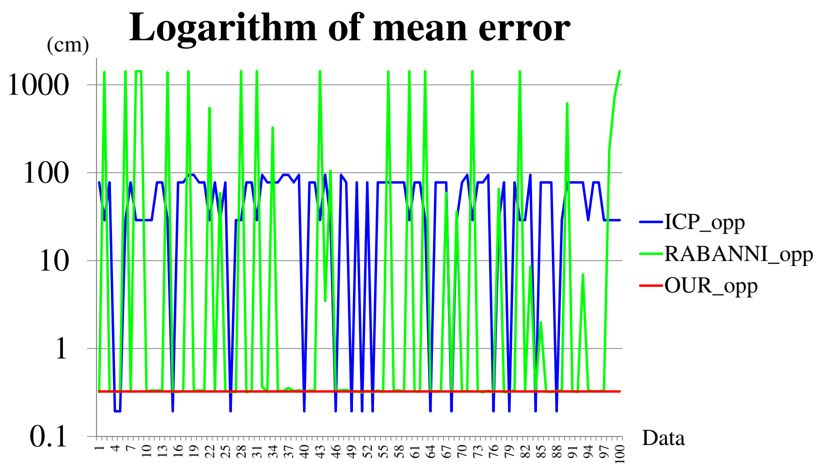
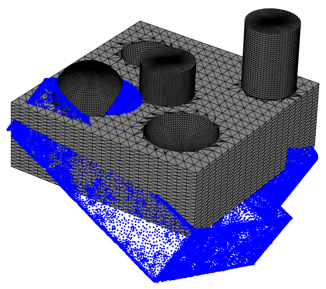
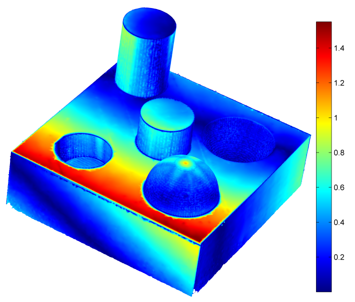


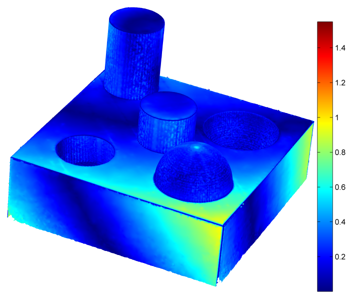
Figure 6.9. Comparison of three methods in the case of opposite direction of the point cloud and CAD model. The proposed method always achieves the same result of mean error (0.32mm).

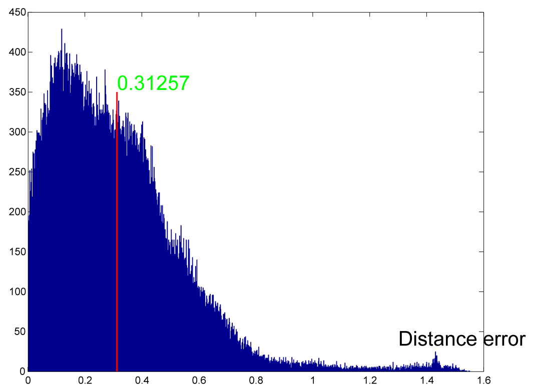
Markers are used by the scanner for self positioning. The proposed framework for primitive alignment is applied in the application as follows. First, primitives are extracted from the point cloud by using the method in Chapter 4 and [TCL15b], and the primitives from the CAD model are extracted by the approach proposed in [AFS06]. The correspondence pairs of primitives are selected manually. After alignment, the distances between the point cloud and the CAD model are computed to generate an error map, as shown in Fig. 6.10.

Observing the results from Fig. 6.10, the proposed method clearly outperforms Rabbani's approach in terms of alignment errors (Fig. 6.10(b)-6.10(c)). Observing the histograms of error maps, the error of the proposed method is smaller than the one given by Rabbani's approach. The reason is that Rabbani's approach does not guarantee the condition ($\|\mathbf{q}\| = 1$)

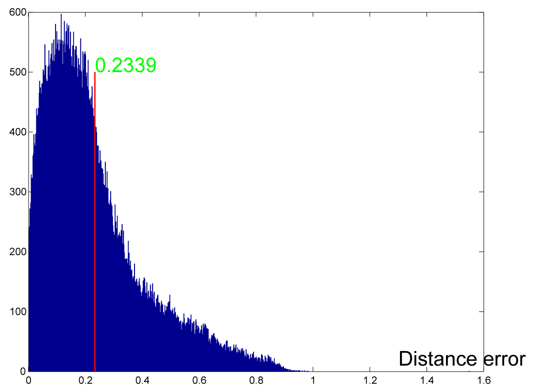

(a) Scanned data and CAD model.


(b) Error map of the result from Rabbani's approach.


(c) Error map of the result from the proposed approach.



(d) Histogram of error map created from Rabbani's approach, largest value (1.55mm) mean error value (0.31mm).



(e) Histogram of error map generated from the proposed method, largest value (0.98mm) and mean error value (0.23mm).

Figure 6.10. Alignment of the scanned data with its CAD model. Error maps of both methods are created and analyzed. The distance error of our proposed method is less than the one of Rabbani's approach [RDvdHV07].

when the LM optimization updates the parameter in the minimization process. Moreover, the lack of objective function for the cylinder has a significant influence on the convergence of the result. The proposed method solves these two problems and achieves better alignment.

6.4.2 Computing time comparison with ICP

The computation time of the proposed method is claimed to be faster than ICP because ICP works directly on data points, while the proposed method works on the primitive parameters. KDtree search [Wei94] is often used as a data structure to accelerate ICP [BM92]. Although the proposed framework includes primitive extraction and minimization steps, only the computation time of the minimization step is considered. Two experiments are conducted by resampling from the CAD model with various numbers of data points 100 000 (**Exp1**) and 50 000 (**Exp2**), respectively, while the scanned data is used directly for alignment (|**S2**|=117 858).

The computation time for ICP and the proposed method in the two experiments is shown in Fig. 6.11. The result shows that the computation time for ICP depends on the number of data points. However, the proposed method requires less time to achieve convergence. The more important point is that the computational load of the proposed method does not depend on the number of data points.

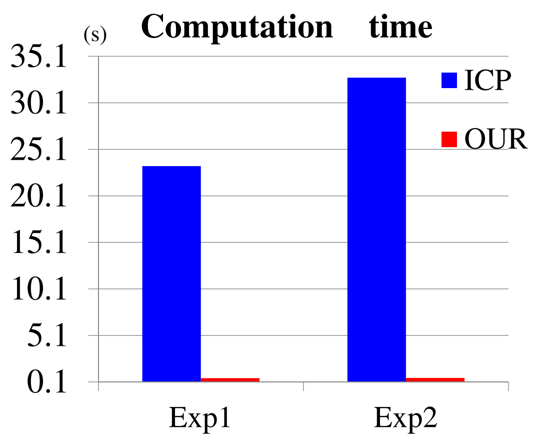


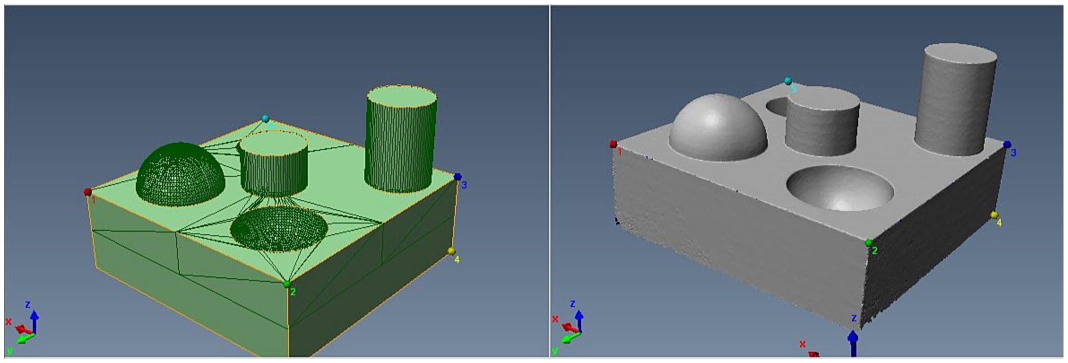
Figure 6.11. The computation time of the minimization step for ICP and the proposed method (OUR) [TCL16c] in two experiments.

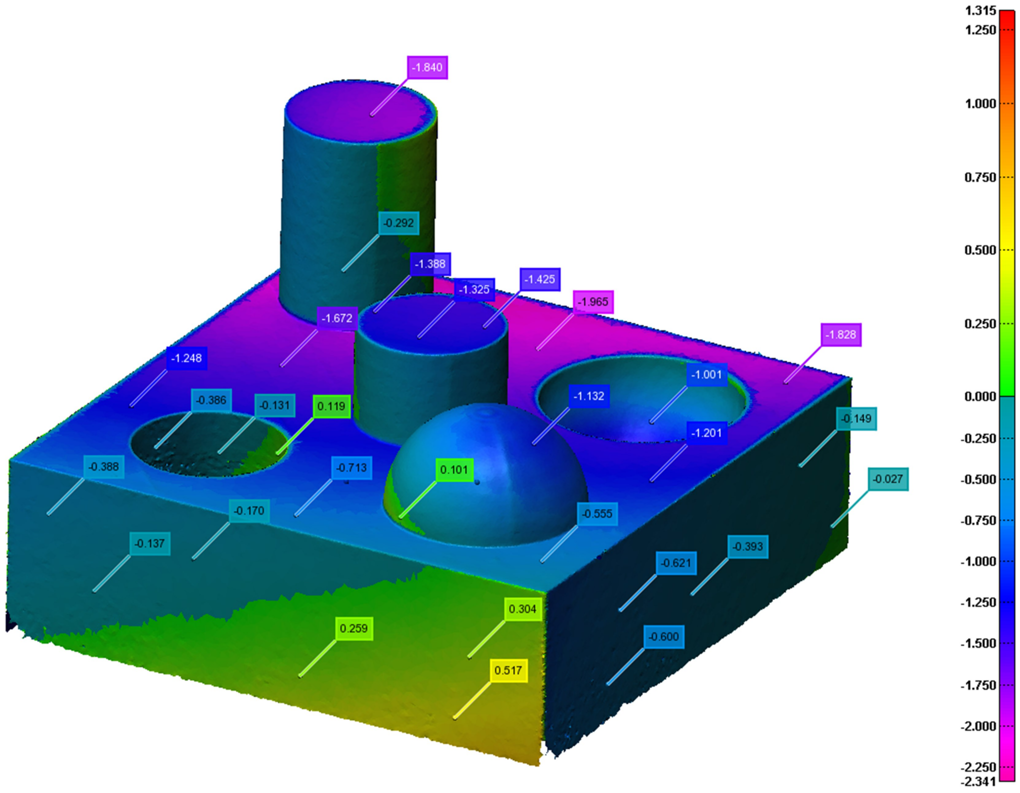
6.4.3 Comparison with the Polyworks 3D inspection software

Currently, Polyworks is known as a powerful software for processing 3D data in reverse engineering and quality control. To register a point cloud to a CAD model, Polyworks also depends on their initial relative positions, so automatic alignment fails. However, a function is available that allows the user to align two datasets manually by selecting multiple point pairs. In the proposed framework in Section 6.3, we also allow the user to choose correspondence pairs of primitives. Therefore, the performance of both methods can be compared under similar conditions (Fig. 6.12).

Observing the error map in Fig. 6.12(b), we see that Polyworks software has large value of error compared to the method proposed in this chapter. Although the data in Fig. 6.12(a) consists of corners that helps the user to select the matching points, selecting points is still not easy. Contrarily, in our method, the user selects the correspondences between primitives easily. With a robust minimization step, the proposed method achieves excellent results with respect to accuracy of alignment. The maximum value of difference between datasets obtained by our proposed method (0.98mm) in Fig. 6.10(c) is lower than the one achieved by Polyworks (2.341mm) (Fig. 6.12(b)).

We also tested the framework on the scanned data with noise and outliers, as shown in Fig. 6.13. This complex data is challenging for other methods such as ICP and Polyworks with manual point correspondence selection because of noise and outliers. Our method still achieves good alignment results (Fig. 6.13(b)) because the framework uses geometric primi-

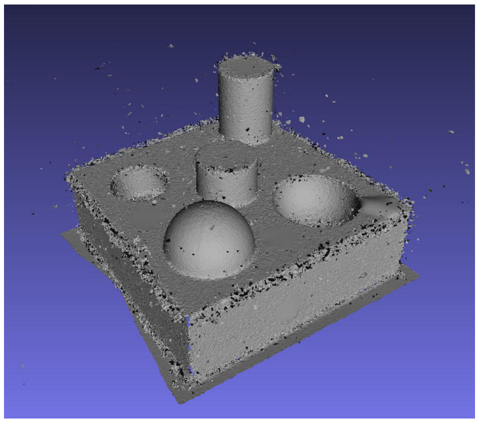

(a) Point pairs from both CAD model and scanned data.

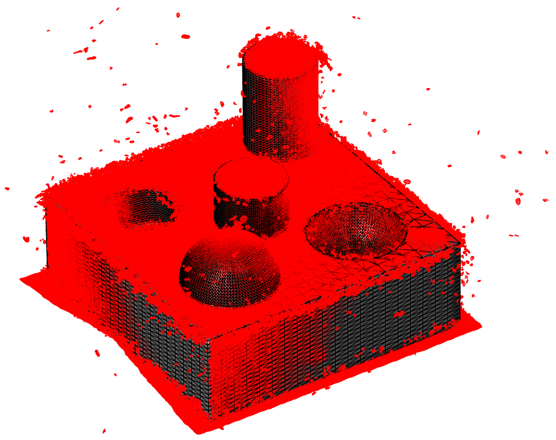


(b) Error map created from alignment result.

Figure 6.12. An example of point pair selection and error map generated by Polyworks software. The maximum of absolute value of the distance error (2.341mm) is greater than the one obtained with our method (0.98mm) in Fig. 6.10(c)

tives as a descriptor for alignment. With the help of a robust primitive extraction method in Chapter 4 and [TCL15b], geometric primitive selection is clearly easier than data point selection. In addition, the above experiment proves that the proposed method is able to handle noisy data and outliers efficiently.


(a) Noisy data and outliers scanned from printed object.


(b) Alignment result of scanned data and CAD model.

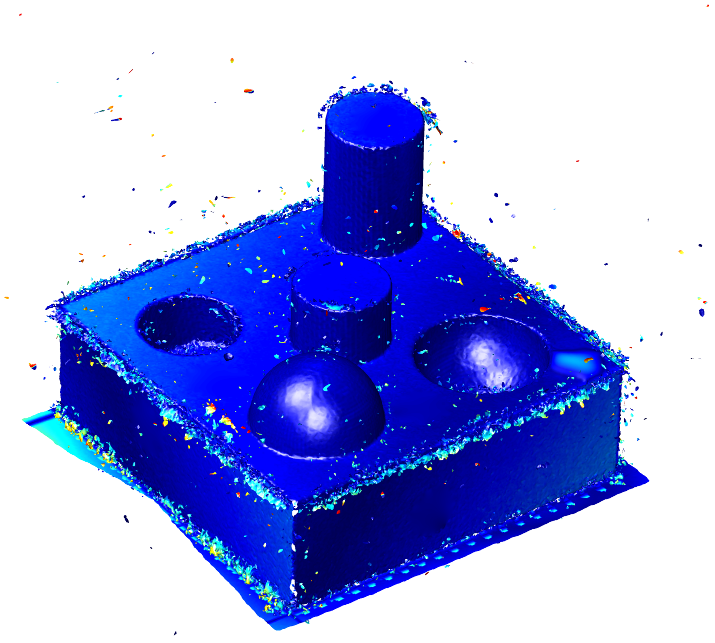
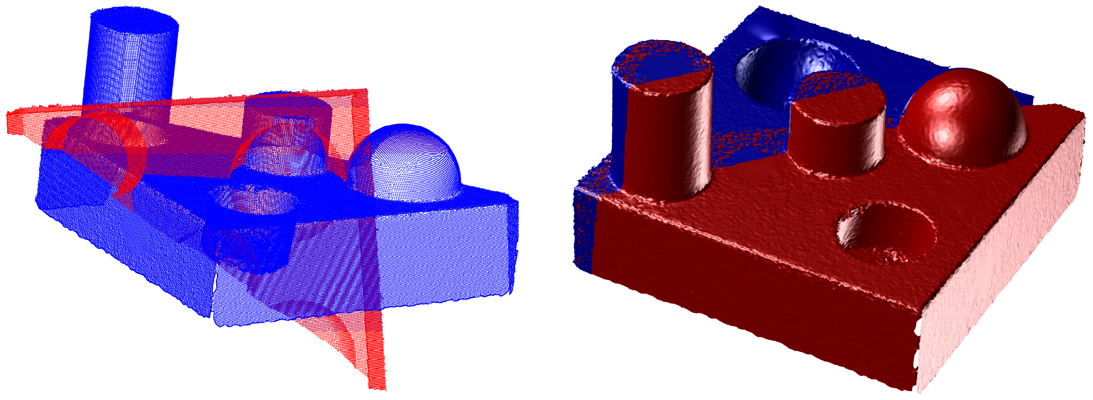

(c) Error map from alignment result.

Figure 6.13. Alignment result and error map for noisy data and outliers. This complex dataset is still a challenge for methods such as ICP and a method using commercial Polyworks inspection software.

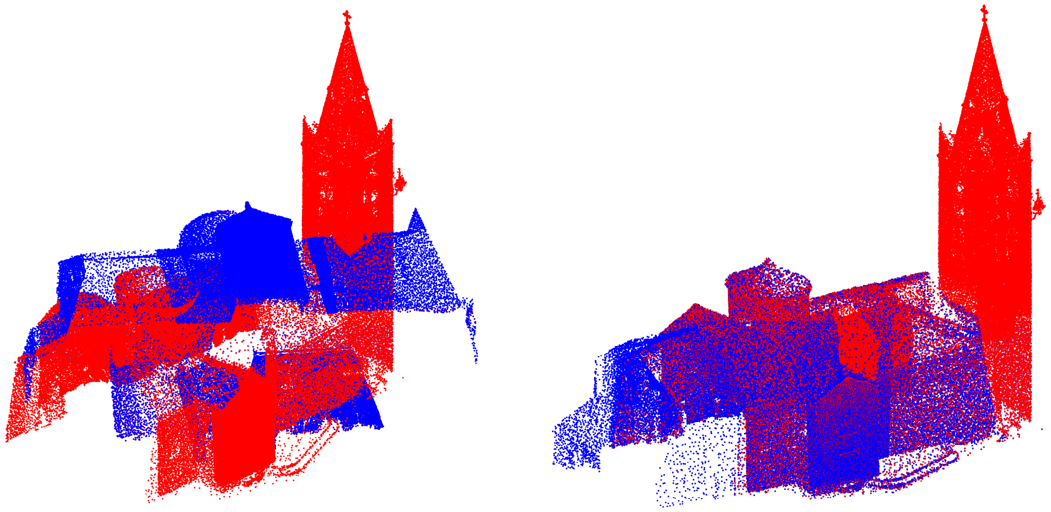
6.4.4 Applications of the proposed approach on real 3D scans

The second application of the proposed approach is data completion. The complete model of an object is often built from multiple partial scans. Separate scans need to be expressed in a common coordinate frame using registration. In this section, the proposed method is used to register two real scans. In Fig. 6.14, two scans of the object in Fig. 6.14(a) are shown. First, planar, cylindrical and spherical primitives are extracted from the scans using the method reported in Chapter 4 and [TCL15b]. These primitives are then fed to the alignment approach presented in Section 5.2.3 to build a complete model of the object. The resulting model is shown in Fig. 6.14(b). The same strategy is used for the planes on two point clouds of the church in Fig. 6.14(c) with the resulting model in Fig. 6.14(d) after alignment using planar primitives.



(a) Two partial scans of the plastic model scanned by the GO!SCAN 3D sensor.

(b) Complete model created using the extracted primitives for alignment.

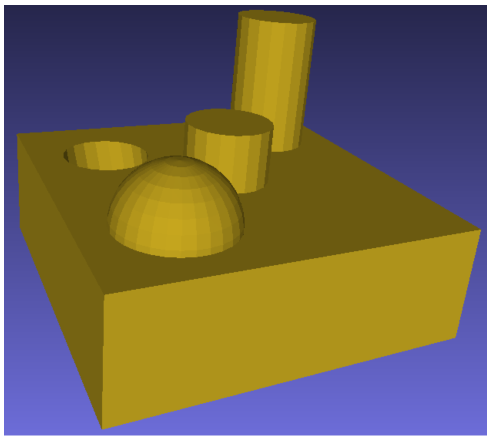


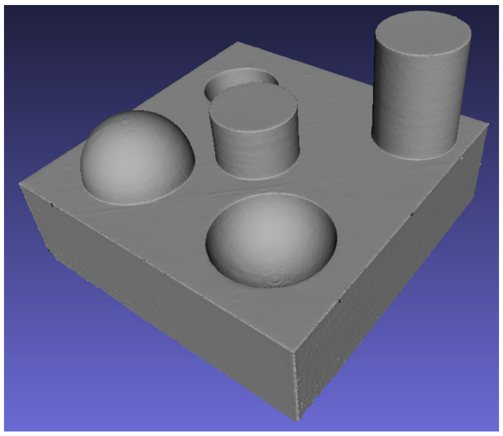
(c) Two partial point clouds of the church of Lans le Villard (France) taken from the Aim@Shape repository.

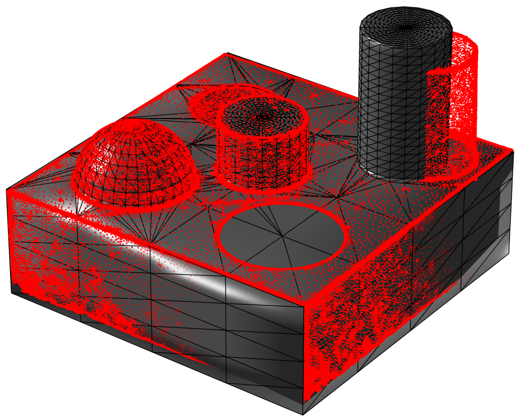
(d) Church model is built by aligning the extracted planes from partial data.

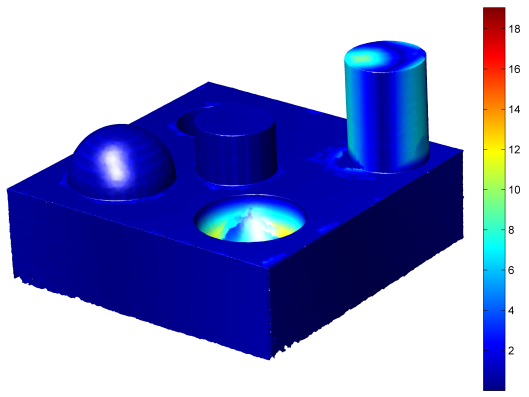
Figure 6.14. Complete models are created from partial data using the proposed method of primitive alignment.

Although primitive correspondences are selected manually by the user, this operation is not difficult even for a novice. In addition, this operation is easier than choosing the points used in manual ICP and Bosche's methods [BM92, Bos12]. Moreover, the manual assignment of primitive correspondence also has the advantage of detecting different components between primitives (Fig. 6.15). This is another application of the proposed framework. For instance, we assume that Fig. 6.15(a) and Fig. 6.15(b) are the CAD model and scanned data from the production line. By using only primitive correspondences provided by the user, the alignment result and error map are created (Fig. 6.15(d)). From the error map, the different parts between the CAD model and the object built by the production line is made explicit. In this experiment, not all of the extracted primitives should be used for alignment because two cylinders having the same radius value are not a correct correspondence. Therefore, manual assignment in Section 6.3.3 helps to solve this type of problem. An automatic and elegant method for correspondence assignment will be a topic for future work.


(a) A CAD model designed by 3DS Max.


(b) The data scanned from a printed object.


(c) The alignment result expressing the difference between the CAD model and the scanned data.



(d) Error map expressing the difference clearly.

Figure 6.15. Manual assignment of primitive correspondence also has an advantage of detecting different components between primitives.

6.5 Conclusion

A complete framework of primitive-based registration is presented in this chapter. The general idea is to use extracted primitives as a robust descriptor for the alignment problem. The problem of registering the scanned data and the CAD model is considered as an example. First, primitives from the point cloud and the CAD model are extracted. Then a set of primitive correspondence pairs is chosen by the user. The set of primitive correspondences is fed to a minimization step to find the transformation between the CAD model and a scan. The performance of the proposed method is compared to existing methods and tested on both synthetic and real scanned data. The promising results of the proposed method could be used in many applications such as data completion and inspection applications.

Although manual primitive correspondence assignment has its own advantages, an automatic assignment approach will be considered for future work. Preliminary results of this idea are described and shown in Section 7.4 of Chapter 7.

Chapter 7

Sharp Feature Extraction

7.1 Introduction and related work

Among emerging problems in 3D data processing, sharp feature extraction from scanned data of CAD models has received much significant from the research community. Sharp features help to understand the structure of the underlying geometry of a surface. Furthermore, it is very important for many issues such as segmentation [LDB05], surface reconstruction [WHHB12] and resampling [HWG$^+$13]. Especially, most manufactured objects consist of the combination of common geometric primitives such as planes, cylinders, spheres, cones or toruses and the intersection between these primitives can be considered as being composed of sharp features. The input 3D data consists of meshes or point clouds. Therefore, existing methods of sharp feature extraction are classified into two categories: mesh-based and point-based methods. Most existing methods focus only on point clouds or meshes which limits their flexibility and generality.

Several techniques [HG01, OBS04, HPW05] use polygonal meshes as input. In [HG01], a framework to extract mesh features from surfaces is presented. However, this approach is semi-automatic, that is, the user is requested to input a few control parameters before a solution is found. Hildebrandt et al. [HPW05] have also proposed a new scheme based on discrete differential geometry, avoiding costly computations of higher order approximating surfaces. This scheme is augmented by a filtering method for higher order surface derivatives to improve both the stability and the smoothness of feature lines. Ohtake et al. [OBS04] propose to use an implicit surface fitting procedure for detecting view and scale independent ridge-valley structures on triangle meshes. Nevertheless, this method still requires a threshold value for a scale-independent parameter tobe set in order to keep the most visually important features. Moreover, the methods only show the results of feature lines for graphics models.

Point clouds preserve the structure of the underlying surface, so they have been recently used at 3D modeling and shape processing. Sharp features are normally found at the points that show variations in curvature or discontinuities in the orientation of the normal to the surface. Consequently, several approaches such as [DVVR07] detect sharp features using differential geometry principles exploiting surface normal and curvature information. Nevertheless, the first step of this method is a normal-based segmentation that does not guarantee the quality of results because of noisy point data. Merigot et al. [MOG11] compute principal curvatures and normal directions of the underlying surface through a Voronoi covariance measure (VCM). Then they estimate the location and direction of sharp features using Eigen decomposition. However, this method requires significant computational time for calculating the VCM.

Recently, Gumhold et al. [GWM01] use PCA to calculate the penalty function that takes into consideration curvature, normal and correlation. Pauly et al. [PKG03] calculate surface variation using the same mathematical technique. Weber et al. [WHH10] have proposed to use statistical methods to infer local surface structure based on Gaussian map clustering. Park et al. [PLL12] used tensor voting to infer the structure near a point. However, all these methods require that many parameters be set manually.

In this chapter, a novel algorithm for extracting sharp features automatically from 3D data is proposed [TAL13, TCN+14]. The scanned data is preprocessed by finding the neighborhood and estimating the surface normal at each point. First, the projected distance is calculated for each point at a given scale (See Section 7.2.2). Then an automatic threshold value is selected by Otsu's method to detect potential sharp features at a given scale. The process is iteratively applied for incrementing scales. In the end, valid sharp features are determined by multi-scale analysis as a refinement. Generally, our method focuses on two main computation steps: (i) calculating the projected distance at a single scale and automatically extracting potential sharp features. (ii) applying these calculations at multiple scales. Therefore, the strong points of the proposed method are summarized in the following:

— The proposed method is very simple and fast. Furthermore, the method accepts both meshes and unstructured point clouds as input.
— No parameter needs to be set by the user since the entire process is completely automatic. This is a significant advantage over existing methods.
— Our approach explores multiple scales in order to extract sharp features accurately with certain levels of noise. In most cases, the detected features are one-point wide.
— Accurate and robust results are achieved by the method even for noisy data and complex object geometry.
— The extracted sharp features could be used to create a graph that describes the structure of scanned data. Graph matching is proposed and integrated into the proposed approach in Chapter 6 to create an automatic registration between two 3D datasets (Section 7.4).

7.2 Proposed method for extracting sharp features

In this section, the proposed algorithm for extracting sharp features at multiple scales is described. Fig. 7.1 illustrates a block diagram of the method. First, the projected distance definition and the calculation procedure for each point are introduced in Section 7.2.2. Then a threshold value is selected automatically by Otsu's algorithm in Section 7.2.3. After this step, potential sharp features at a given scale are detected. These steps are applied iteratively for multiple scales. Finally, the refinement approach for detecting valid and reliable sharp features is described in Section 7.2.4.

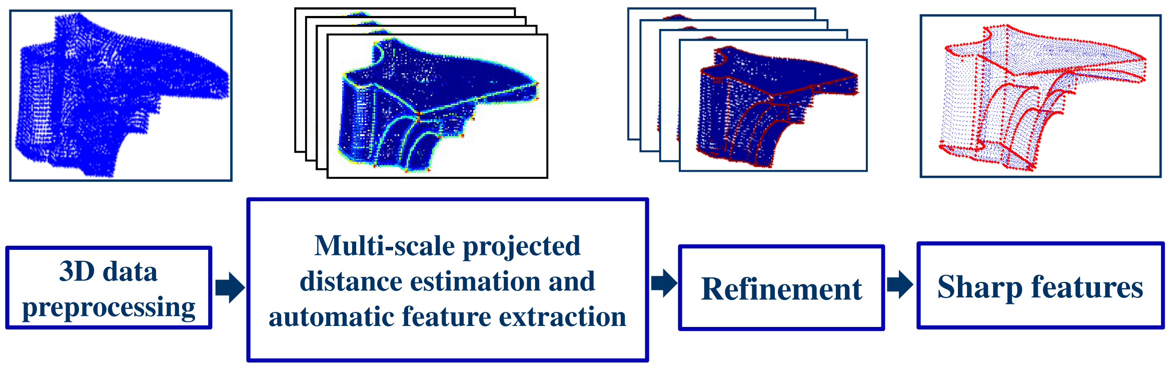

3D data preprocessing
Multi-scale projected distance estimation and automatic feature extraction
Refinement
Sharp features

Figure 7.1. Overall procedure of sharp feature extraction.

7.2.1 Preprocessing step

The input 3D data can consist of meshes or point clouds that have different data structures to represent the underlying surface of a given object. The neighborhood around a point is used to estimate the normal at that point. Therefore, the problem of finding the neighborhood of each point is addressed first and the problem of estimating the normal vector for each point is then described according to the type of data. Appendix .1 and Appendix .2 describe in more detail the way to determine the neighorhood and to compute surface normal vector according to mesh and point cloud formats, the reader is referred to the appendices for details.

7.2.2 Projected distance calculation

The projected distance is a value calculated by projecting the vector between a given point and the centroid of its neighborhood at a given scale on the surface normal as defined in Eq. 7.1. The procedure is shown in Fig. 7.2.

$$pd_i = \|\overrightarrow{(\mathbf{p}_i - \overline{\mathbf{p}}_i)} \cdot \overrightarrow{\mathbf{n}_i}\| = \|\|\overrightarrow{\mathbf{p}_i - \overline{\mathbf{p}}_i}\| \cdot \cos\theta_i\| \tag{7.1}$$

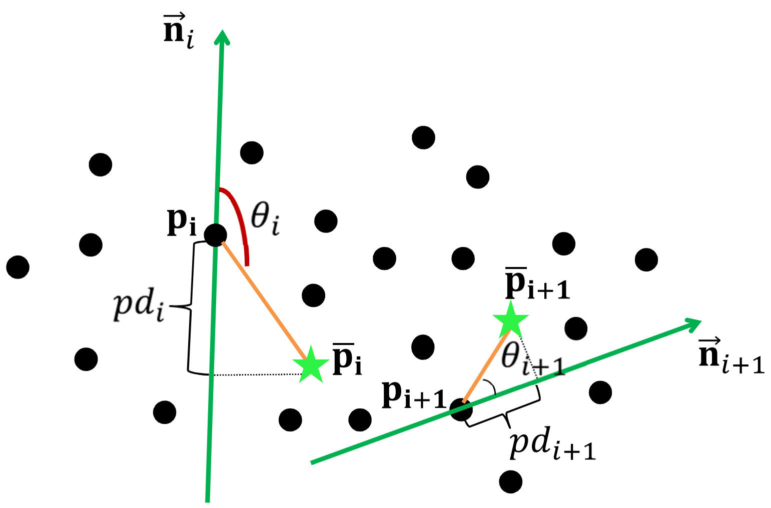


Figure 7.2. Procedure for calculating the projected distance at each point. The green star is the centroid of the neighborhood.

where θ_i is the angle between vector $\overrightarrow{(\mathbf{p}_i - \overline{\mathbf{p}_i})}$ and vector $\vec{\mathbf{n}_i}$, which is the surface normal at point $\mathbf{p}_i$ calculated in Appendix .1 or Appendix .2 according to the type of input data. $\overline{\mathbf{p}_i}$ is the centroid of neighborhood points determined with the k-d tree for point clouds (Appendix .1) or k-ring for meshes (Appendix .2). Angle θ_i between vector $\overrightarrow{(\mathbf{p}_i - \overline{\mathbf{p}_i})}$ and vector $\vec{\mathbf{n}_i}$ can be greater or smaller than 90 degrees, but the projected distance is always a positive number. That is why the direction of the normal vector is not considered here as it is in [HDD+92], only the orientation is used in our work.

The projected distance value expresses the structure of the underlying surface supported by the neighborhood. Therefore, the central idea of our method is that the projected distance is almost zero for a point lying on a smooth surface area. However, the projected distance has a large value if a point is located on or near a sharp feature, as shown in Fig. 7.3. Our method uses this property of the projected distance to assess whether a point is located on a sharp feature or not.

7.2.3 Automatic selection of a threshold value for the detection of sharp features

The projected distance for every data point (point cloud or mesh) was computed at the same scale in the previous section. Potential sharp features are detected at a given scale. First, every projected distance is normalized between [0,1] as displayed in Fig. 7.4(a). Then a threshold value for the projected distance needs to be calculated to evaluate whether or not a given point is located on a sharp feature. Otsu's method [Ots79] is an optimal method for this nontrivial task and was chosen because of its simplicity and efficiency. Furthermore, it contributes in making our method completely automatic. The input to Ostu's method being a histogram, a histogram of the normalized projected distance values is generated and

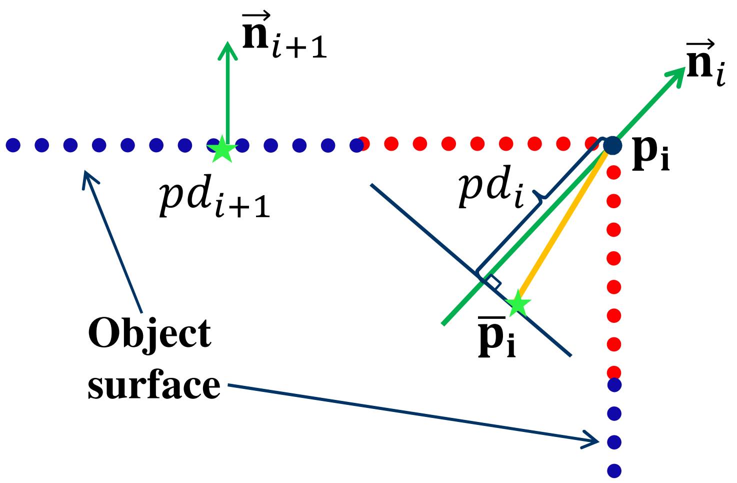


Figure 7.3. Projected distances calculated at smooth (blue points) and sharp areas (red points). Green points are the centroids of the data.

shown in Fig. 7.4(b).

Otsu's algorithm is based on the very simple idea of finding the threshold value that minimizes the weighted within-class variance. This turns out to be the same as maximizing the between-class variance. Threshold *Th* allows the generation of a binary version of projected distance values by setting all vertices below the threshold to zero and all vertices above that threshold to one Eq. 7.2.

$$sf(i) = \begin{cases} 1 & \text{if } pd_i \geq Th \\ 0 & \text{if } pd_i < Th \end{cases} \tag{7.2}$$

where **sf** is a binary version of pd_i with global threshold *Th* at the red line in Fig. 7.4(b). The points for which $f(i)$ is equal to one are considered as potential sharp features. Potential sharp features are extracted by Eq. 7.2 and displayed in Fig. 7.4(c).

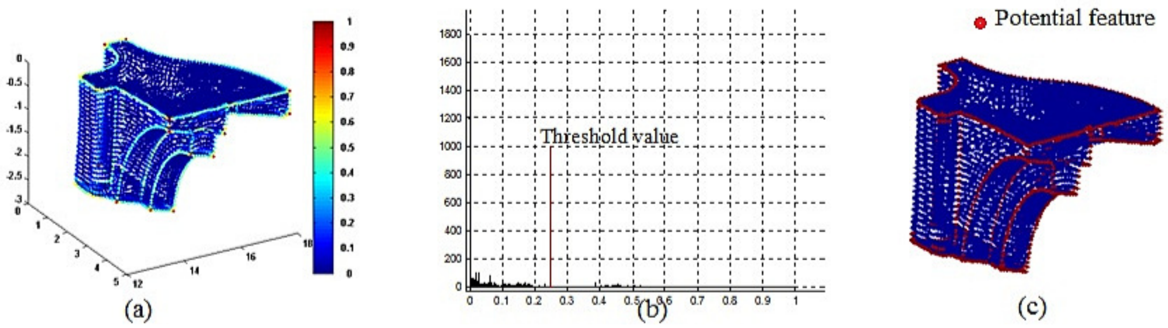


Figure 7.4. Automatic threshold estimation and feature extraction. (a) Normalized projected distance. (b) Histogram and threshold value. (c) Sharp feature candidates.

7.2.4 Multiscale processing and refinement

Potential sharp features are extracted from the steps described in the two previous sections at a single scale. A real feature can be recognized at multiple scales by a human. Besides, the result at a single scale may contain several false feature points because of noise. Therefore, the proposed method calculates the projected distance and extracts sharp features automatically at multiple scales to eliminate false sharp features.

Valid sharp features are those recorded at all scales. Hence, Eq. 7.3 is used to decide whether or not a point is valid sharp feature:

$$SF(i) = \begin{cases} 1 & \text{if } \sum_{i=1}^{ns} f(i) = ns \\ 0 & \text{if } \sum_{i=1}^{ns} f(i) < ns \end{cases} \tag{7.3}$$

where **SF** is the final sharp feature map, ns is the number of scales that are investigated for point clouds or meshes in Section 7.3.2 and $f(i)$ is calculated from Eq. 7.2 in Section 7.2.3. The results of the multiscale sharp feature extraction method are shown in Fig. 7.9 and Fig. 7.10 for point clouds and meshes, respectively.

7.3 Results and discussion

We have tested the proposed method [TAL13, TCN$^+$14] on various complex models corrupted by different noise levels. Furthermore, the results provided by our method are compared with some other methods using mean curvature [YL99, WB01], normal vector [HG01] and surface variation in Pauly's method [PKG03] for point-based methods and Ohtake's method [OBS04] for mesh-based methods. Computation time and limitations of our method are also reported.

7.3.1 Comparison with other methods

Point-based method

As mentioned above, a sharp feature is a point at which the surface normal vector [DVVR07] or the curvature [YL99] is discontinuous in value, so the Mean curvature and the normal difference between adjacent triangles are used as factors for determining sharp feature locations. A value Δn_i is computed by normal difference between given point and its neighborhood [HG01].

$$\Delta n_i = \frac{1}{|\mathbf{N}(i)|} \sum_{j \in \mathbf{N}(i)} \cos\left(\frac{\mathbf{n}(i)}{\|\mathbf{n}(i)\|} \cdot \frac{\mathbf{n}(j)}{\|\mathbf{n}(j)\|} \right)^{-1} \tag{7.4}$$

where $\mathbf{n}(i)$, $\mathbf{n}(j)$ and $\mathbf{N}(i)$ are defined in Appendix .1.

In [PKG03], Pauly et al. use surface variation σ_i, which is calculated as the ratio between the smallest eigenvalue and the sum of the eigenvalues of the covariance matrix $\mathbf{CV}_i$ in Eq. 2

established by the neighborhood around a given point.

$$\sigma_i = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{7.5}$$

Therefore, we will compare our method based on the projected distance with these three parameters. We investigate these methods by applying them to the fandisk point cloud at a single scale (k=16), that was mentioned as the optimal one in [WHH10]. Moreover, the other three methods are not automatic and require some threshold values to be set, so our automatic threshold estimation is applied to them. The results of the four methods are shown in Fig. 7.5 and Fig. 7.6.

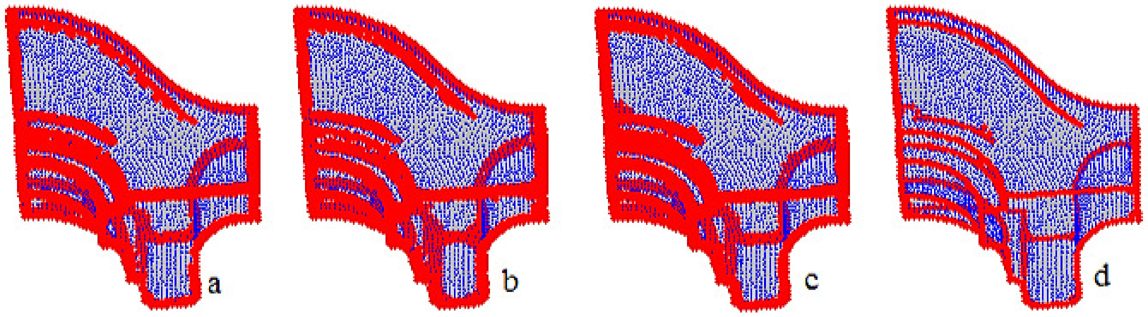


Figure 7.5. Sharp feature extraction using different parameters for the fandisk model. (a) Normal. (b) Mean curvature. (c) Surface variation [PKG03]. (d) Projected distance [TCN+14].

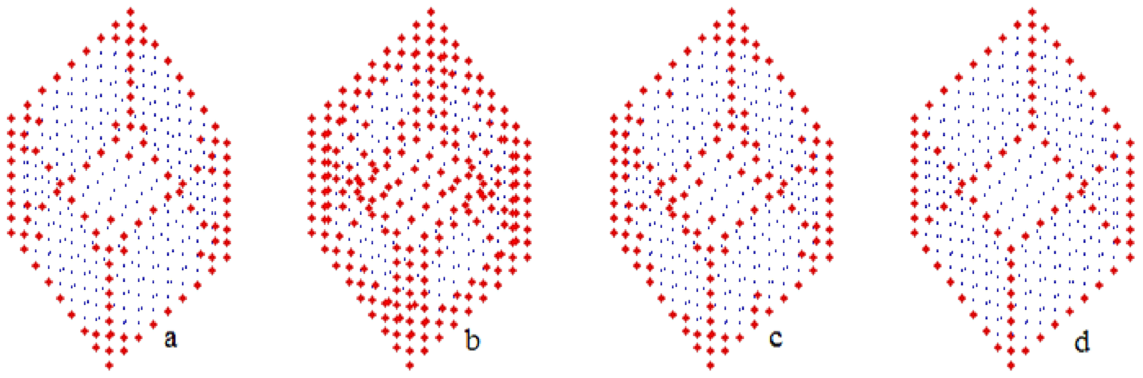


Figure 7.6. Sharp feature extraction using different parameters for the synthetic cube model. (a) Normal. (b) Mean curvature. (c) Surface variation [PKG03]. (d) Projected distance [TCN+14].

Observing Fig. 7.5 and Fig. 7.6 indicate that our method is able to detect sharp features that are one point wide. Mean curvature, normal and Pauly's methods detect rather thick features including many potential feature points, since the mean curvature and the surface normal do not reveal shape information of the underlying surface as the projected distance does. The projected distance is small at smooth and near-edge points and much smaller than the value at sharp features. Therefore, the feature lines detected by the proposed method are thinner and more accurately located. The results prove that the projected distance is a better factor that can be used for detecting sharp features.

Fig. 7.7 shows that even when tuning the threshold value manually to improve its performance, Pauly's method still fails to produce thin and accurate features. For instance on threshold values for surface variation set at T=0.1, 0.15 and 0.2, the results contain many spurious sharp features. With threshold T=0.25, some good features are missing and the result still contains spurious sharp features (black ellipse in Fig. 7.7).

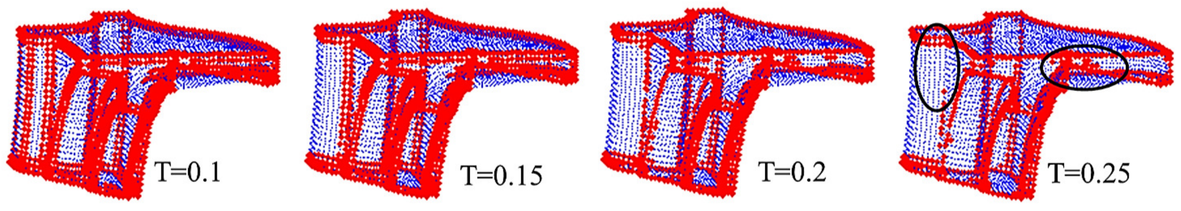


Figure 7.7. Sharp features extracted with manual tuning of the threshold for Pauly's method.

Furthermore, the absolute accuracy of our method is demonstrated in Fig. 7.6 and Table 7.1. A synthetic cube point cloud was generated with 91 sharp features composed of 386 data points. Mean curvature, normal and Pauly's method detect many spurious sharp features while our method extracts all 91 sharp features correctly and at the right location.

Table 7.1. Number of detected sharp features on the cube model by 4 methods.

Normal	Mean curvature	Surface variation	Projected distance
116	248	135	91

Mesh-based method

In this section, we compare the results of the mesh-based method proposed in this thesis to Ohtake's method [OBS04], which uses an implicit surface fitting procedure for detecting ridge-valley structure on a surface approximated by a dense triangulation. The implementation code for Ohtake's method is available from the author's website and the selected threshold settings are the default values for scale-independent parameter, connecting and iterations. The results are shown in Fig. 7.8. For these results, ridge and valley points are displayed in red and blue, respectively. We find sharp features accurately and does not include valley points (blue). Moreover, Ohtake's method detects many spurious sharp features at the small and thin holes as shown in Fig. 7.8.

7.3.2 Results for various models

The method is applied on other meshes and point clouds. Most models are taken from the AIM@SHAPE Shape Repository and the Octa-flower model from Ohtake. For point

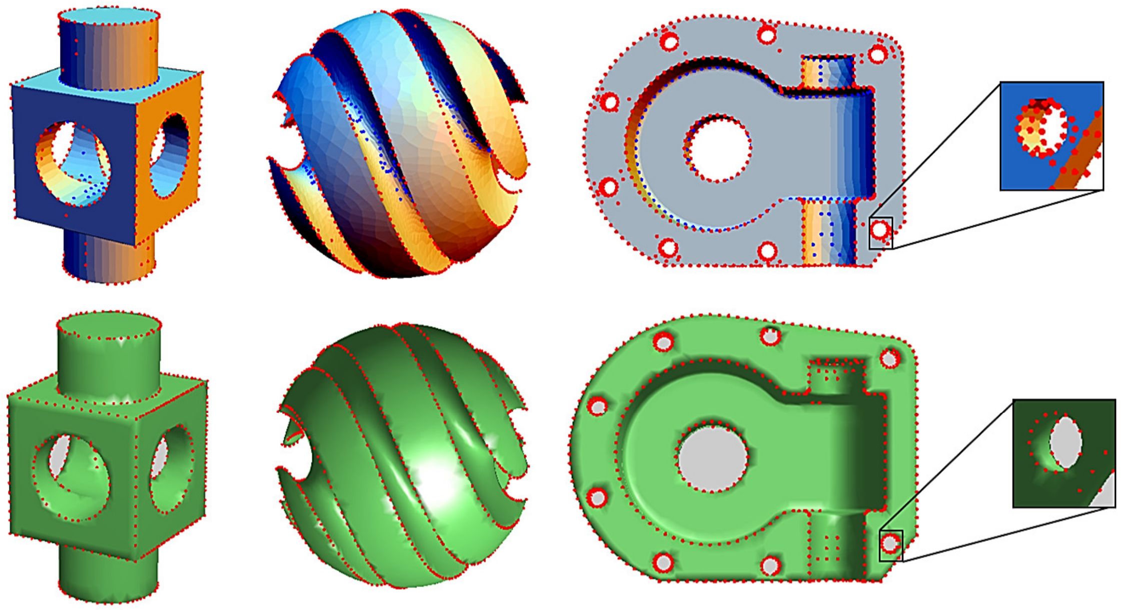


Figure 7.8. Sharp features extracted from Block, Sharp-feature and Casting models using Ohtake's method [OBS04] (top row) and our mesh-based method (bottom row) [TCN+14].

cloud models, the behavior of the proposed method was investigated at multiple scales *k*= (12,16,...40) and Fig. 7.9 shows the results for several point cloud models. Although these

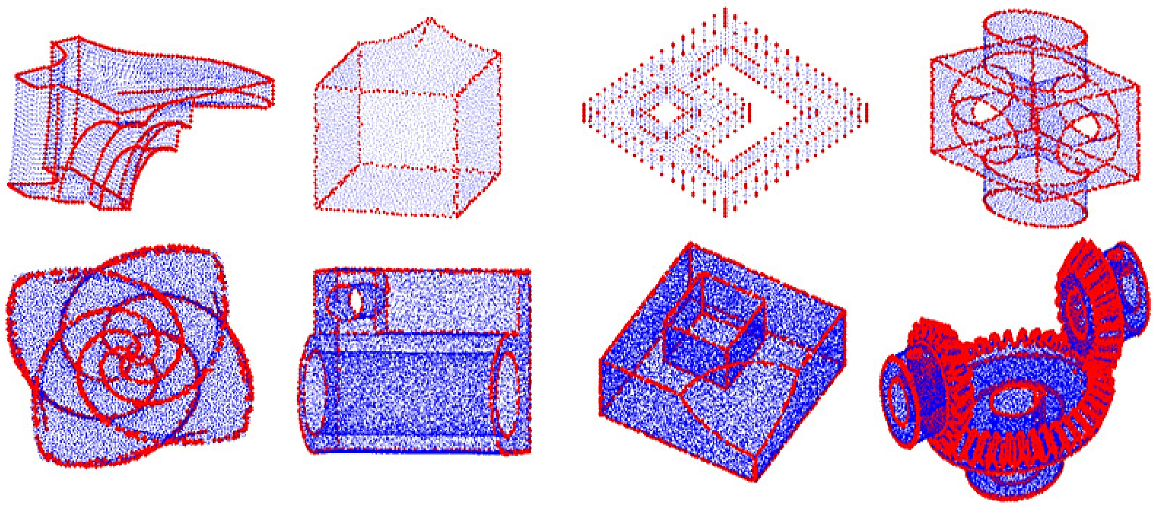


Figure 7.9. Sharp features extracted from point clouds: top row (Fandisk, Smooth-feature, Double-torus2) and bottom row (Block, Octa-flower, Cylinder, Box, BEVEL2) respectively.

models contain many different primitives including planes, spheres, cylinders and even free-form primitives, our method can detect sharp features accurately in each model. For meshes, 1,2,3 and 4-ring neighborhoods are used to calculate the projected distances for every vertex. Fig. 7.10 shows the results for some mesh models and demonstrates the efficiency of the method to detect thin sharp features.

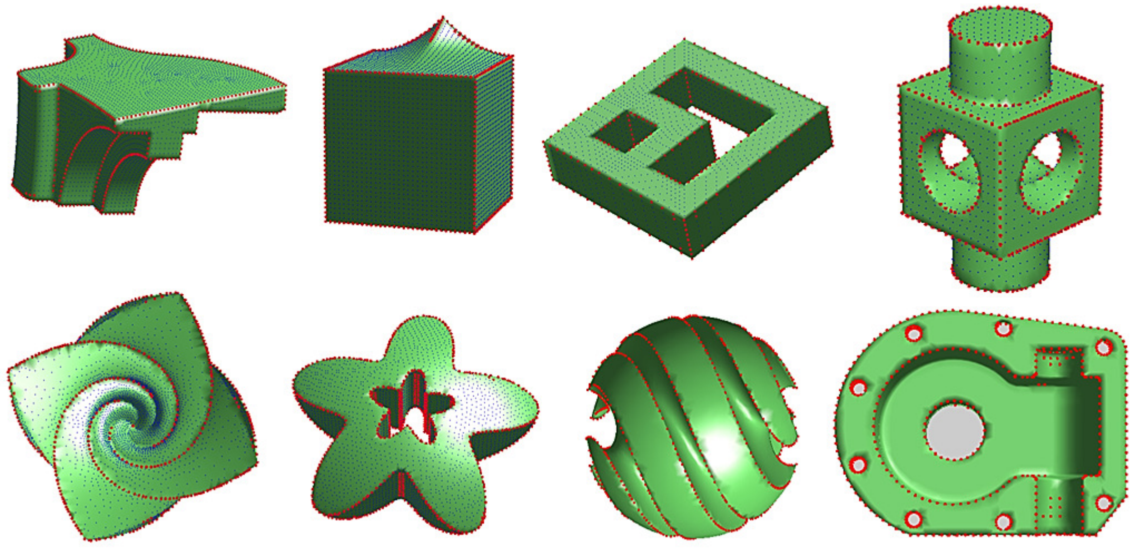


Figure 7.10. Sharp features extracted from meshes: top row (Fandisk, Smooth-feature, Double-torus2, Block) and bottom row (Octa-flower, Trim-star, Sharp-sphere, Casting) respectively.

7.3.3 Computational cost

The proposed method is executed on MATLAB on a 3.2 GHz Intel Core i7 platform. Table 7.2 for point clouds and Table 7.3 for meshes show the computation time of the method. This process is implemented without parallel processing. It is hypothesized that a C++ implementation would speed up the process significantly and parallel processing could be applied to process multiple scales simultaneously.

Table 7.2. Timing performance of our method for point cloud models.

Model name	Points	Total time (s)
Fandisk	6 475	1.05
Smooth-feature	6 177	1.01
Double-torus2	2 668	0.44
Block	12 909	2.01
Octa-flower	12 868	2.11
Cylinder	50 000	8.34
Box	50 000	8.21
BEVEL2	64 250	4.92
Real data1	108 349	17.4
Real data2	30 751	4.99

7.3.4 Robustness to noise

To illustrate the robustness of the proposed method to a various levels of noise, some results for noisy fandisk point cloud models and real data are presented in this section. First, Gaussian noise with zero mean and standard deviations of 1-5 % of the average distance

Table 7.3. Timing performance of our method for mesh models.

Model name	Points	Faces	Time(s)
Fandisk	6 475	12 946	1.48
Smooth-feature	6 177	12 350	1.40
Double-torus2	2 668	5 340	0.61
Block	2 132	4 272	0.52
Octa-flower	7 919	15 834	1.79
Trim-star	5 192	10 384	1.18
Sharp-sphere	8271	16538	1.87
Casting	5 086	10 204	0.42
Real data1	108 349	214 189	21.4
Real data2	30 751	60 783	6.84

between points was added to the point cloud data. Then our method was applied to this data corrupted by noise. Fig. 7.11 shows the results for different noise levels. For a noise level of (1-3)% acceptable results are still achieved. For a noise level of (4-5)% a small part of the model is contaminated by spurious feature points. Therefore, applying a prior denoising algorithm could help the proposed method to obtain better results.

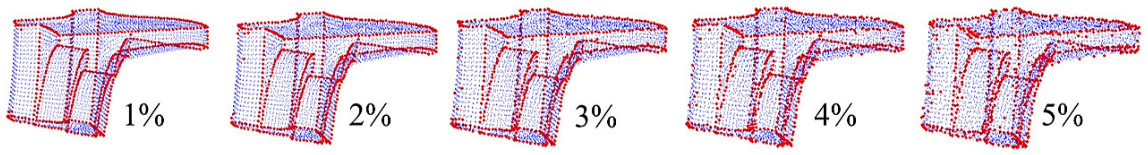


Figure 7.11. Sharp features extracted from the fandisk model with different levels of noise.

Fig. 7.12 shows the results on real scanned data, which contains many outliers and holes Fig. 7.12(b). The data is captured by the GO! scan sensor (`http://www.goscan3d.com`). Other methods such as Pauly's (Fig. 7.12(c)) and Ohtake's (Fig. 7.12(d)) methods are also tested. Observing the results indicates that Pauly's method misses several sharp features, while Ohtake's method fails completely for noisy data.

7.3.5 Limitations

Although the proposed method can extract sharp features accurately, it depends on the level of noise of the captured data and also on the distribution of the data. For instance, for areas with low point density, the improvement brought by the weighting factor is minor. Therefore, preprocessing stages such as resampling and pre-filtering would be necessary in these cases to improve the results. Fig. 7.13 shows the results after removing outliers and filling holes on real scanned data. Sharp features are detected accurately.

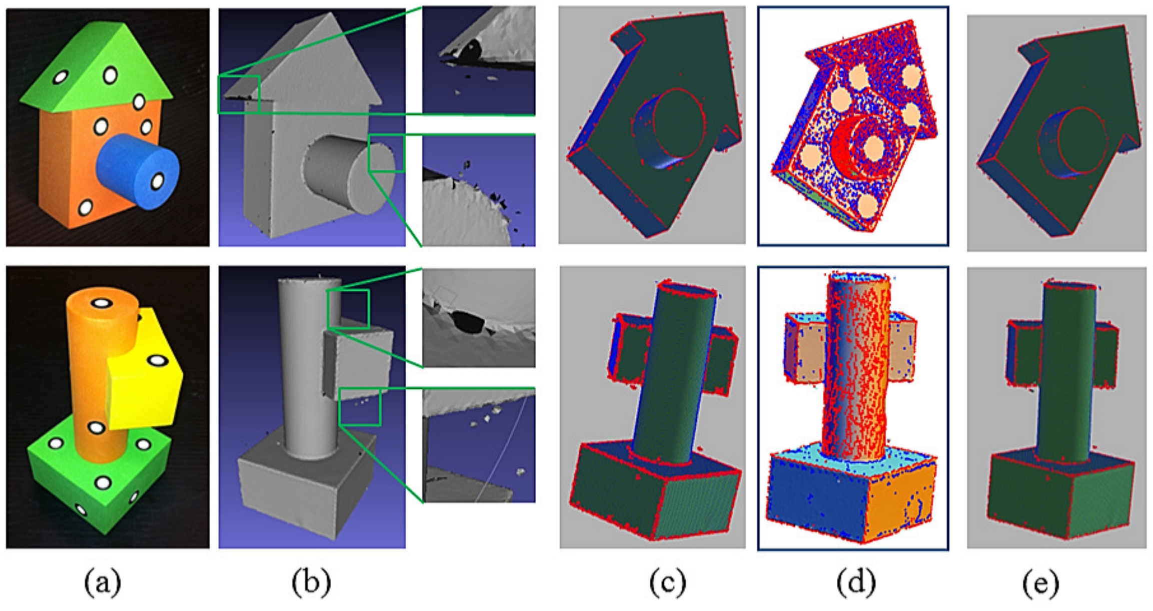


Figure 7.12. Results on noisy data with outliers and holes: (a) Original object and markers for self-positioning of the GO!SCAN sensor. (b) Scanned data with outliers and holes. (c) Pauly's method [PKG03]. (d) Ohtake's method [OBS04]. (e) Our Method [TCN+14].

7.4 Preliminary results of automatic assignment of primitive correspondences

In Chapter 6, a primitive-based approach for registration between models was proposed in which the correspondences of primitives between datasets are provided by the user (Section 6.3.3). In this section, we present a potential *graph matching* solution for automatic assignment of primitive correspondences using the result of sharp feature extraction. First, an introduction and background on graph matching are presented. More details of the application of graph matching for finding correspondences are then described.

Graph matching plays a key role in solving correspondence problems in computer vision [DSD14]. Graphs are used to represent objects or images in which vertices or nodes usually represent regions (or features) of the object or images, and edges between them describe the relations between regions (or features). A graph can be either directed or undirected. When edges are undirected, they often express the presence of a relation between two vertices. On the other hand, directed edges are used when relations between vertices are considered as being asymmetric.

Existing methods for graph matching are classified into exact and inexact approaches [Ben02], as shown in Fig. 7.14. In this section, we investigate the problem of attribute relation graph (ARG) matching of 3D data (inexact and undirected). Each primitive is considered as a node of a graph and an attribute relational graph is created to provide a structural and relational

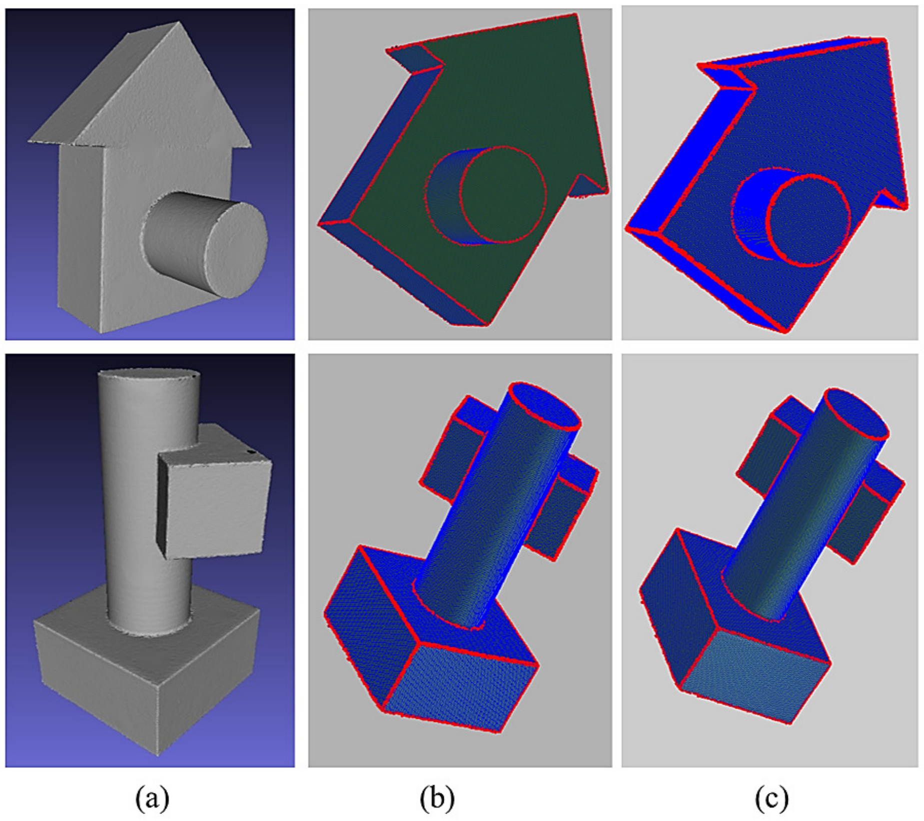


Figure 7.13. Results on preprocessed real data: (a) Smoothed and hole-filled data. (b) Point-based method. (c) Mesh-based method.

description between primitives. First, primitives were extracted from the point cloud and the CAD model using the approaches described in Section 6.3 of Chapter 6.

As described in Section 6.3.1, hierarchical face clustering [AFS06] attempts to segment triangular meshes into separated clusters that are possibly fitted by geometric primitives. This method belongs to the class of bottom-up approaches. Two clusters of faces are contracted into one at each iteration. The process terminates when the desired numbers of clusters are reached. At the end, each cluster corresponds to a type of geometric primitive such as a plane, cylinder and sphere, as shown in Fig. 7.15(a). At this point, each cluster is considered as a node of graph (**G**), and the connections between clusters are considered as edges in the graph (Fig. 7.15(b)).

Another graph (**g**) is extracted from the scanned data by using the results obtained by the sharp feature extraction described in this chapter. As mentioned earlier, two or more primitives intersect at sharp features. Therefore, from the results of sharp features and primitive extraction, a graph that describes the connection between extracted primitives is established and shown in Fig. 7.16.

ARG graph matching: Let **G** and **g** be undirected graphs. Each node corresponds to a type

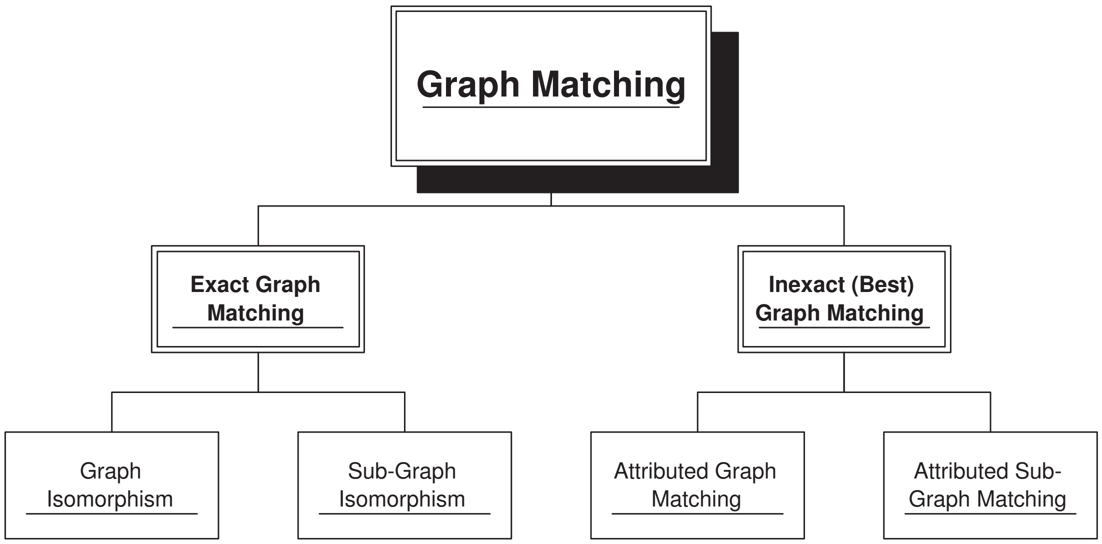

Graph Matching
Exact Graph Matching
Inexact (Best) Graph Matching
Graph Isomorphism
Sub-Graph Isomorphism
Attributed Graph Matching
Attributed Sub-Graph Matching

Figure 7.14. Classification of graph matching types into two main classes: exact graph matching and inexact graph matching [Ben02].

of primitive that is called an *attribute*. Links or relations between graph nodes are encoded by link types. These link types may consist of topological or geometrical relations. The matrix **Q** indicates which nodes in the two graphs match

$$Q_{ai} = \begin{cases} 1 & \text{if node } a \text{ in } \mathbf{G} \text{ corresponds to node } i \text{ in } \mathbf{g} \\ 0 & \text{otherwise,} \end{cases} \tag{7.6}$$

The matrix **Q** is computed by minimizing the following objective function:

$$\begin{aligned} E(\mathbf{Q}) &= \frac{1}{2}\sum_{a,b}^{A}\sum_{i,j}^{I} Q_{ai}Q_{bj}\sum_{r}^{R} C^{(r)}_{aibj} + \sum_{a}^{A}\sum_{i}^{I} Q_{ai}\sum_{s}^{S} C^{(s)}_{ai} \\ \text{subject to:} &\quad \forall a, \forall i, \sum_{i=1}^{I} Q_{ai} \leq 1, \sum_{a=1}^{A} Q_{ai} \leq 1, Q_{ai} \in \{0,1\}. \end{aligned} \tag{7.7}$$

where

— R is the number of link types and S is the number of attributes.

— Graphs **G** and **g** have A and I nodes respectively.

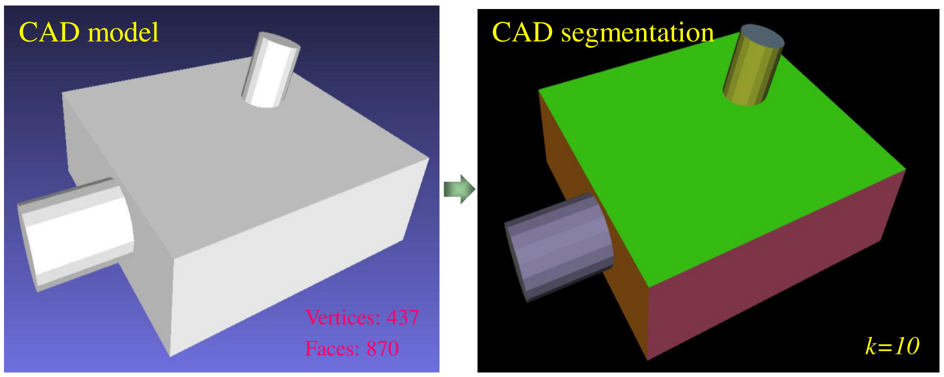
— $C^{(r)}_{aibj}$ is the comparability matrix for a r-link and is defined by

$$C_{aibj} = \begin{cases} 0 & \text{if either } \mathbf{G}_{ab} \text{ or } \mathbf{g}_{ij} \text{ is NULL} \\ c_1(\mathbf{G}_{ab}, \mathbf{g}_{ij}) & \text{otherwise,} \end{cases} \tag{7.8}$$

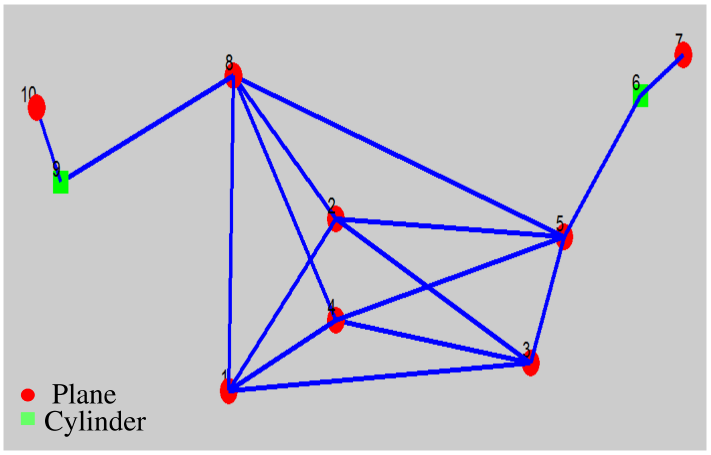
c_1 is a measure of compatibility between a r-link in **G** and a r-link in **g**.

— $C^{(s)}_{ai}$ is the similarity matrix for an attribute of type s and is defined by

$$C_{ai} = \begin{cases} 1 & \text{if node } a \text{ in } \mathbf{G} \text{ has the same attribute as node } i \text{ in } \mathbf{g} \\ c_2(\mathbf{G}_a, \mathbf{g}_i) & \text{otherwise,} \end{cases} \tag{7.9}$$



(a) CAD model and its primitives extracted by the HFC method [AFS06].



(b) Graph connecting the extracted primitives. Each primitive is considered as a node.

Figure 7.15. CAD model segmentation and graph created from the connection between extracted primitives.

c_2 is a measure of similarity between a node in **G** and a node in **g**, with respect to the same attribute s.

The graduated assignment algorithm in [GR96, SASH12, Tao15] is used to solve the above problem of ARG graph matching. These methods have three advantages compared to previous methods:

— The *softassign* and Sinkhorn's method [Sin64] are combined to satisfy the constraint of two-way assignment. The assignment constraint requires the nodes of both graphs to be equally constrained. A node in one graph can be matched to at most one node in the other graph and vice versa.

— A continuation method, called graduated non-convexity, is used in an effort to avoid local minimum with a parameter controlling the convexity.

— Sparsity is encoded to increase efficiency.

More detailed explanations concerning softassign, graduated non-convexity and sparsity can be found in paper [GR96].

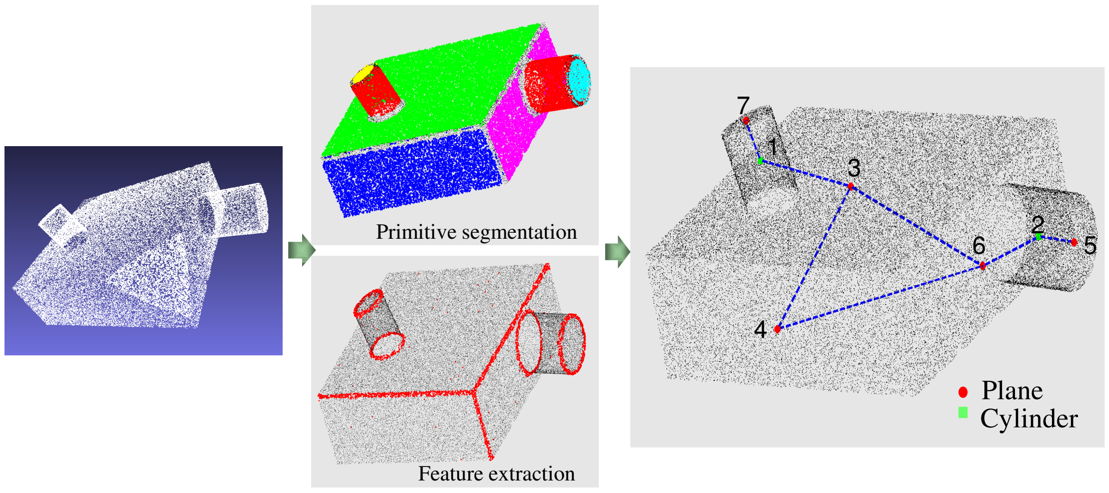


Figure 7.16. Creation of scan graph from point cloud. From the results of primitive segmentation and feature extraction, a graph **g** connecting primitives is created. Each node corresponds to a primitive. Edges connecting nodes are determined by common sharp features between extracted primitives.

An example of graph matching between scan graph **g** and CAD graph **G** is shown in Fig. 7.17. In this experiment, $|R| = 1$ is the connection property between geometric primitives with $c_1(\mathbf{G}_{ab}, \mathbf{g}_{ij}) = 1$. And $|S| = 1$ is the type of primitive in which $c_2(\mathbf{G}_a, \mathbf{g}_i) = 0$ if node a in **G** has the different attribute than node i in **g**. Eq. 7.7 is simply converted to

$$\begin{aligned} E(\mathbf{Q}) &= \frac{1}{2}\sum_{a,b}^{A}\sum_{i,j}^{I} Q_{ai}Q_{bj}C_{aibj} + \sum_{a}^{A}\sum_{i}^{I} Q_{ai}C_{ai} \\ \text{subject to:} &\quad \forall a, \forall i, \sum_{i=1}^{I} Q_{ai} \le 1, \sum_{a=1}^{A} Q_{ai} \le 1, Q_{ai} \in \{0,1\}. \end{aligned} \tag{7.10}$$

Primitive correspondences of graph matching, which are determined from the result **Q** of Eq. 7.10, are fed to Eq. 6.3 in Chapter 6 for the minimization step.

The attribute and link used in the above experiment are applicable for the simple dataset shown in Fig. 7.15 and Fig. 7.16. For more complex datasets including multiple primitives with various primitive types, the approach requires more attributes and types of links (i.e. angle between links, length of links...) embedded into Eq. 7.7 to improve the quality of graph matching. This problem will be considered in future work.

7.5 Conclusion

We have introduced an automatic method for extracting sharp features from 3D data. This method accepts both meshes and point clouds as input. The projected distance is calculated at multiple scales, which are supported by the k neareast neighborhood in point clouds or the k-ring neighborhood in meshes. Then reliable sharp features are extracted automati-

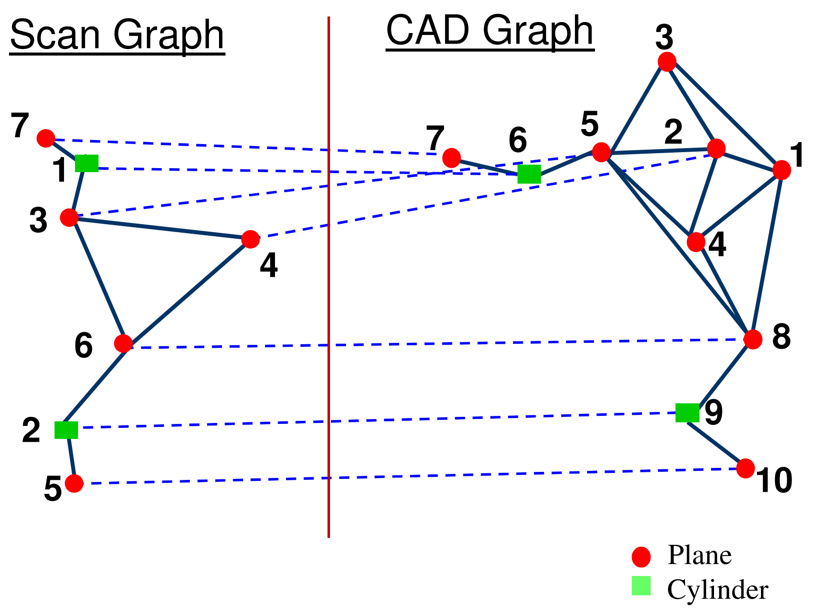


Figure 7.17. Result of graph matching between scan graph **g** in Fig. 7.15 and CAD graph **G** in Fig. 7.15. Primitive correspondences between graphs are determined and fed to Eq. 6.3 in Chapter 6 for the minimization step.

cally using Otsu's method. In addition to its simplicity, the proposed method outperforms other methods presented in the literature. In the future, the algorithm could be improved by connecting discrete sharp features into parametrized curves for obtaining high-level descriptions. Furthermore, we plan to use the results of our algorithm for practical problems such as remeshing or mesh generation.

By integrating the extracted sharp features into the graph matching problem, it is possible to create an automatic assignment for primitive correspondences used for registration (See Chapter 6. For future work, it is planned to improve the performance of graph matching by adding more types of links between primitives.

Chapter 8

Conclusions and Future work

8.1 Overview of Contributions

In the thesis, we presented a set of novel algorithms for primitive and feature extraction from unorganized point clouds of man-made objects. Moreover, an approach is proposed to align 3D datasets using primitives as a robust descriptor. Although the algorithms were proposed to work directly on point clouds, they could be extended easily to adapt to various types of data formats. The main contributions of the thesis are summarized in the following

First, a robust approach is proposed to extract a single type of primitive from a point cloud (Chapter 2 for cylinder extraction and Chapter 3 for sphere extraction). The novel validation method converts the problem into a simpler one consisting of circle detection. With the validation method, primitives are extracted reliably and with high accuracy. Moreover, accurate parameters of extracted primitives are estimated from noisy data with outliers. The method has been tested on both synthetic and real scanned data with convincing results.

A complete framework to extract multiple types of primitive is described in Chapter 4. This framework implements a sequential extraction approach in which curved surfaces (cylinders and spheres) are extracted first using the approaches in Chapter 2 and Chapter 3. Planar primitives are then extracted sequentially using a RANSAC-based approach. The advantage of the sequential approach is to avoid model confusion, which is challenging for other methods. The approach could be useful for various applications such as reverse engineering and primitive-based registration.

The registration problem between the scans of manufactured objects is still a challenge for existing methods. Fortunately, 3D geometric primitives can be extracted from 3D data using the methods described in the thesis. These primitives can be used to solve the alignment problem. Chapter 5 and 6 introduced the proposed approach to align two 3D datasets using common primitives, especially the alignment of scanned data and its corresponding CAD model. Several other applications such as data completion and inspection benefit from the proposed approach.

Chapter 7 finally proposed an automatic approach to extract sharp features from 3D data (point clouds, meshes). First, a novel method is proposed to detect potential sharp features at a given scale. The process is iteratively applied for increasing scales. In the end, valid sharp features are determined by multi-scale analysis as a refinement. The extracted sharp features could be used to create a graph describing the structure of the scanned data. Graph matching is proposed and integrated into the proposed approach in Chapter 6 to create an automatic registration between two 3D datasets.

8.2 Future Work

As summarized in Section 8.1, several contributions have been proposed in the thesis. However, there are still many aspects that could be explored or improved. The future work that could complement the thesis is described.

The other types of primitives such as cones and tori will be investigated. The proposed framework in Chapter 2 and Chapter 3 can possibly be extended to extract cones and tori from point clouds. With this extension, the framework would be able to process objects with more complex shapes.

All of the algorithms are currently implemented in MATLAB on a 3.2 GHz Intel Core i7 platform so it is time consuming. The second task would be to implement them in C++ and paralle or GPU programming. We hypothesize that with these improvements they could be applied to larger datasets and used in real time applications.

The third task would be to work on the graph matching problem addressed briefly in Chapter 7. As mentioned earlier, graph matching is one of the most popular research topics in computer vision and graphics. We believe that the proposed approach for primitive-based graph matching could be used intensively in other applications, for instance 3D data retrieval.

Since the thesis covers a wide range of research fields, its results could contribute to other research topics. For example, geometric primitives extracted from point clouds could be the input of Boolean operation in constructive solid geometry [Fol96, Ope]. Through the process, a reconstructed CAD model could be used for further steps of reverse engineering applications. Another interesting application would be to apply geometric primitive extraction for scene and building reconstruction from large scanned datasets, a topic that has recently been investigated in the last decade [MWH$^+$06, MWA$^+$13].

In conclusion, the problems related to reverse engineering, quality control and inspection are still challenging topics that are far from being solved. For instance, reconstruction is a challenging topic due to the various types of input data [MWA+13] (multi-view imagery, range image, and etc.). Quality control merging problem of 2D and 3D data [VAA14] still needs considerable attention from the research community before being implemented in practical applications. Therefore, we believe that the combination of different types of approaches and different types of input data and features should be used to achieve these long term goals.

Appendix

.1 Normal vector computation for meshes

A polygonal mesh is a collection of vertices, edges and faces that represents the shape of a object. This type of data is different from point clouds because it contains edges connecting the vertices. Therefore, an approach is needed to find the neighborhood of a point based on this connection. A k-ring neighborhood of a vertex $\mathbf{p}_i$ is defined as a set of vertices that are connected to $\mathbf{p}_i$ by at most k edges.

A simple and effective method for normal vector estimation is to calculate it as the weighted average of the normal vectors of the triangles formed by $\mathbf{p}_i$ and pairs of its neighbors inside the 1-ring. The basic averaging method is based on Eq. 1:

$$\mathbf{n}_i = \frac{1}{|\mathbf{N}(i)|} \sum_{j \in \mathbf{N}(i)} \omega_j \frac{[\mathbf{p}_j - \mathbf{p}_i] \times [\mathbf{p}_{j+1} - \mathbf{p}_i]}{|[\mathbf{p}_j - \mathbf{p}_i] \times [\mathbf{p}_{j+1} - \mathbf{p}_i]|} \tag{1}$$

where $\mathbf{p}_j$ is one of the points inside 1-ring. ω_j is a normalized weighting factor that can be calculated by angle-weighted, area-weighted and centroid-weighted methods. A good summary of these methods can be found in [JLW05].

.2 Normal vector computation for point clouds

A point cloud is defined as a set of points within a three-dimensional coordinate system. This type of data acquired from the scanners represents noisy samples of the object surface. The information about connectivity and surface normal of underlying surfaces is lost completely in the sampling process. Moreover, it resides implicitly in the relationship between a sampled point and its neighborhood. Hence, the first step in surface normal estimation at a given point is to find the number of points around this point. In our works, the data is structured using k-d trees [Wei94] to search for the k-nearest neighbors.

Classical principal component analysis (PCA) in [HDD+92] is usually selected for surface normal estimation because of its simplicity and efficiency. However, a point cloud contains a set of non-uniformly distributed points, so a weight that is derived from the distance to the centroid should be assigned to each point in the neighborhood in order to improve the robustness of the method. A covariance matrix **CV** of point $\mathbf{p}_i$ is established in Eq. 2 and analyzed and the eigenvector corresponding to the smallest eigenvalue is considered as the normal vector $\mathbf{n}_i$ at a point $\mathbf{p}_i$.

$$\mathbf{CV}_i = \sum_{j \in \mathbf{N}(i)} \mu_j (\mathbf{p}_j - \overline{\mathbf{p}}_i)^T (\mathbf{p}_j - \overline{\mathbf{p}}_i) \in \mathbb{R}^{3\times 3} \tag{2}$$

where $\overline{\mathbf{p}}_i = \frac{1}{|\mathbf{N}(i)|} \sum_{j\in\mathbf{N}(i)} \mathbf{p}_j$ is $\mathbf{p}_i$'s local data centroid, $\mathbf{N}(i)$ is the set of k points in the neighborhood around point $\mathbf{p}_i$. μ_j is calculated by the Gaussian function $\mu_j = e^{\frac{-d_j^2}{k^2}}$ of the distance $d_j = \|\mathbf{p}_j - \overline{\mathbf{p}}_i\|$. The scale $k = |\mathbf{N}(i)|$ is the number of neighboring points at a given point $\mathbf{p}_i$.

.3 Principal curvature computation

Taubin's method [Tau95] is referred to as an efficient and reliable method for principal curvature estimation [MSR07], which was improved by Hameiri et al. [HS03]. The main idea is summarized as follows. Let $\mathbf{N}(i)$ be the set of points in the neighborhood of point $\mathbf{p}_i$. $\overrightarrow{\mathbf{T}_{ij}}$ is the tangent being a projection of the vector between $\mathbf{p}_i$ to $\mathbf{p}_j \in \mathbf{N}(i)$ to the tangent plane at $\mathbf{p}_i$. k_{ij} is the normal curvature associated with $\overrightarrow{\mathbf{T}_{ij}}$, so $\mathbf{M}_{\mathbf{p}_i}$ is defined as:

$$\begin{aligned}\mathbf{M}_{\mathbf{p}_i} &= \sum_{j\in\mathbf{N}(i)} w_{ij} k_{ij} \overrightarrow{\mathbf{T}_{ij}} \overrightarrow{\mathbf{T}_{ij}}^T \\ &= [\overrightarrow{\mathbf{T}_1}\overrightarrow{\mathbf{T}_2}\overrightarrow{\mathbf{N}}] \begin{pmatrix} \mathbf{e}_1 & 0 & 0 \\ 0 & \mathbf{e}_2 & 0 \\ 0 & 0 & 0 \end{pmatrix} [\overrightarrow{\mathbf{T}_1}\overrightarrow{\mathbf{T}_2}\overrightarrow{\mathbf{N}}]^T\end{aligned} \tag{3}$$

where $k_{ij} \simeq \frac{2\overrightarrow{\mathbf{n}_i}^T(\mathbf{p}_{ij}-\mathbf{p}_i)}{\|\mathbf{p}_{ij}-\mathbf{p}_i\|^2}$ and $w_{ij} \sim \frac{1}{\|\mathbf{p}_{ij}-\mathbf{p}_i\|}$ is proportional to the inverse of the geometric distance from $\mathbf{p}_i$ to $\mathbf{p}_{ij}$ with $\sum_{j\in\mathbf{N}(i)} w_{ij} = 1$. $\overrightarrow{\mathbf{T}_1}$ and $\overrightarrow{\mathbf{T}_2}$ denote the principal directions associated with the principal curvatures k_1 and k_2, respectively. $\overrightarrow{\mathbf{N}}$ is the normal vector to the surface at a given point $\mathbf{p}_i$. Assume that $\hat{\theta}$ is the angle between $\overrightarrow{\mathbf{T}}$ and $\overrightarrow{\mathbf{T}_1}$, and θ_{ij} is the angle between $\overrightarrow{\mathbf{T}_{ij}}$ and $\overrightarrow{\mathbf{T}}$, so

$$\begin{aligned}\mathbf{e}_1 &= \overrightarrow{\mathbf{T}_1^T} \mathbf{M}_{\mathbf{p}_i} \overrightarrow{\mathbf{T}_1} = \sum_{j\in\mathbf{N}(i)} w_{ij} k_{ij} \overrightarrow{\mathbf{T}_1^T} \overrightarrow{\mathbf{T}_{ij}} \overrightarrow{\mathbf{T}_{ij}}^T \overrightarrow{\mathbf{T}_1} \\ &= \sum_{j\in\mathbf{N}(i)} w_{ij} [k_1 cos^4(\theta_{ij} + \hat{\theta}) + k_2 sin^2(\theta_{ij} + \hat{\theta}) cos^2(\theta_{ij} + \hat{\theta})] \\ &= Ak_1 + Bk_2\end{aligned} \tag{4}$$

and

$$\begin{aligned}
\mathbf{e}_2 = \overrightarrow{\mathbf{T}_2^T}\mathbf{M}_{\mathbf{p}_i}\overrightarrow{\mathbf{T}_2} &= \sum_{j\in\mathbf{N}(i)} w_{ij}k_{ij}\overrightarrow{T_2^T}\overrightarrow{T_{ij}}\overrightarrow{T_{ij}}^T\overrightarrow{T_2} \\
&= \sum_{j\in\mathbf{N}(i)} w_{ij}[k_1 sin^2(\theta_{ij}+\hat{\theta})cos^2(\theta_{ij}+\hat{\theta}) + k_2 cos^4(\theta_{ij}+\hat{\theta}) \\
&= Bk_1 + Ck_2
\end{aligned} \tag{5}$$

where

$$\begin{aligned}
&A = \sum_{j\in\mathbf{N}(i)} w_{ij}cos^4(\theta_{ij}); B = \sum_{j\in N(i)} w_{ij}sin^2(\theta_{ij})cos^2(\theta_{ij}); \\
&C = \sum_{j\in N(i)} w_{ij}sin^4(\theta_{ij})
\end{aligned} \tag{6}$$

The principal curvatures are computed by solving Eq. 4 and Eq. 5 with A, B, C defined in Eq. 6. The reader is referred to [HS03] for more detail.

List of Publications

1. [TCL16a] **Trung-Thien Tran**, Van-Toan Cao and Denis Laurendeau. **3D geometric primitive alignment revisited**. *Proceedings of the 11th International Conference on Computer Graphics Theory and Applications*, 2016.
2. [TCL15b] **Trung-Thien Tran**, Van-Toan Cao and Denis Laurendeau. **Extraction of reliable geometric primitives from unorganized point clouds**. *3D Research*, 6-4, 2016.
3. [TCL16b] **Trung-Thien Tran**, Van-Toan Cao and Denis Laurendeau. **eSphere: Extraction of multiple spheres from point clouds**. *The Visual Computer*, 1-18, 2016.
4. [TCL15a] **Trung-Thien Tran**, Van-Toan Cao and Denis Laurendeau. **Extraction of cylinders and estimation of their descriptive parameters in point clouds**. *Computers & Graphics*, 46:345-357, 2015.
5. [TCN+14] **Trung-Thien Tran**, Van-Toan Cao, Van Tung Nguyen, Sarah Ali and Denis Laurendeau. **Automatic method for sharp feature extraction from 3D data of man-made objects**. In *Proceedings of the 9th International Conference on Computer Graphics Theory and Applications*, Lisbon, Portugal, 112-9, 2014.
6. [TAL13] **Trung-Thien Tran**, Sarah Ali, and Denis Laurendeau. **Automatic sharp feature extraction from point clouds with optimal neighbor size**. In *Proceedings of the Thirteenth IAPR International Conference on Machine Vision Applications*, Kyoto, Japan, 165-8, 2013.
7. [CTL15a] Van-Toan Cao, **Trung-Thien Tran** and Denis Laurendeau. **Non-rigid Registration with a Compact Deformation Model**. *Proceedings of the 11th International Conference on Computer Graphics Theory and Applications*, 2016.
8. [CTL15b] Van-Toan Cao, **Trung-Thien Tran** and Denis Laurendeau. **A two-stage approach to align two surfaces of deformable objects**. *Graphical Models*, 85, 13-28, 2015.
9. [CNT+14] Van-Toan Cao, Van Tung Nguyen, **Trung-Thien Tran**, Sarah Ali and Denis Laurendeau. **Non-rigid registration for deformable objects**. In *Proceedings of the Ninth International Conference on Computer Graphics Theory and Applications*, Lisbon, Portugal, 43-52, 2014.
10. [NTCL13] Van Tung Nguyen, **Trung-Thien Tran**, Van-Toan Cao, and Denis Laurendeau. **3D Point Cloud Registration based on the Vector Field Representation**. In

Proceedings of the Second IAPR Asian Conference on Pattern Recognition, Okinawa, Japan, 2013.

11. [ATL13] Sarah Ali, **Trung-Thien Tran** and Denis Laurendeau. **A Comparative Survey on 3D Models Retrieval Methods**. *REV Journal on Electronics and Communications*, 3(2):1-11, 2013.

12. [ATCL13] Sarah Ali, **Trung-Thien Tran**, Van-Toan Cao and Denis Laurendeau. **An approach for local comparison of deformable 3D models**. In *Proceedings of Recent Advances in Computer Vision and Pattern Recognition*, Okinawa, Japan, 2013.

Bibliography

[3DS] 3DSMAX. http://www.autodesk.com/store/3ds-max.

[AD03] Motilal Agrawal and Larry S Davis. Camera calibration using spheres: A semi-definite programming approach. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 782–789. IEEE, 2003.

[AFS06] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006.

[ANC13] Anas Abuzaina, Mark S Nixon, and John N Carter. Sphere detection in Kinect point clouds via the 3D Hough transform. In *Computer Analysis of Images and Patterns*, pages 290–297. Springer, 2013.

[AP10] Marco Attene and Giuseppe Patanè. Hierarchical structure recovery of point-sampled surfaces. *Computer Graphics Forum*, 29(6):1905–1920, 2010.

[APP$^+$07] Alexander Agathos, Ioannis Pratikakis, Stavros Perantonis, Nikolaos Sapidis, and Philip Azariadis. 3D mesh segmentation methodologies for CAD applications. *Computer-Aided Design and Applications*, 4(6):827–841, 2007.

[ATCL13] S. Ali, T.-T. Tran, Van-Toan Cao, and D. Laurendeau. An approach for local comparison of deformable 3d models. In *Proceedings 2nd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 852–856, Okinawa, Japan, Nov 2013.

[ATL13] Sarah Ali, Trung-Thien Tran, and Denis Laurendeau. A comparative survey on 3D models retrieval methods. *REV Journal on Electronics and Communications*, 2(5):–, 2013.

[BELN11] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3D Hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):1–13, 2011.

[Ben02] E. Bengoetxea. *Inexact Graph Matching Using Estimation of Distribution Algorithms*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France, Dec 2002.

[BES11] Per Bergström, Ove Edlund, and Inge Söderkvist. Repeated surface registration for on-line use. *The International Journal of Advanced Manufacturing Technology*, 54(5-8):677–689, 2011.

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[BF81] Robert C. Bolles and Martin A. Fischler. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 637–643, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[BKV$^+$02] Pal Benko, Géza Kós, Tamás Várady, László Andor, and Ralph Martin. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19(3):173–205, 2002.

[Ble] Blender. https://www.blender.org/.

[BM92] P.J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[BM12] Alexandre Boulch and Renaud Marlet. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum*, 31(5):1765–1774, August 2012.

[Bos12] Frédéric Bosché. Plane-based registration of construction laser scans with 3D/4D building models. *Advanced Engineering Informatics*, 26(1):90–102, 2012.

[BR02] Fausto Bernardini and Holly Rushmeier. The 3D model acquisition pipeline. volume 21, pages 149–172. Wiley Online Library, 2002.

[BSG$^+$13] Roseline Bénière, Gérard Subsol, Gilles Gesquière, François Le Breton, and William Puech. A comprehensive process of reverse engineering from 3D meshes to CAD models. *Computer-Aided Design*, 45(11):1382–1393, 2013.

[BTS$^+$14] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. State of the Art in Surface Reconstruction from Point Clouds. In Sylvain Lefebvre and Michela

Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the Ninth European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[BV04] Pál Benkő and Tamás Várady. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design*, 36(6):511–523, 2004.

[CG01] Thomas Chaperon and François Goulette. Extracting cylinders in full 3D data using a random sampling method and the Gaussian image. In *Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 35–42, 2001.

[CGF09] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. In *ACM Transactions on Graphics*, volume 28, page 73. ACM, 2009.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[CNT$^+$14] Van-Toan Cao, Van-Tung Nguyen, Trung-Thien Tran, Sarah Ali, and Denis Laurendeau. Non-rigid registration for deformable objects. In *Proceedings of the Ninth International Conference on Computer Graphics Theory and Applications, Lisbon, Portugal, 2014.*, pages 43–52, 2014.

[CRM01] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. The variable bandwidth mean shift and data-driven scale selection. In *Proceedings Eighth IEEE International Conference on Computer Vision ICCV 2001*, volume 1, pages 438–445. IEEE, 2001.

[CS05] Chen Chao and Ioannis Stamos. Semi-automatic range to range registration: a feature-based method. In *Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*, pages 254–261. IEEE, 2005.

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, 2004.

[CTL15a] Van-Toan Cao, Trung-Thien Tran, and Denis Laurendeau. Non-rigid registration with a compact deformation model. In *Proceedings of the Eleventh International Conference on Computer Graphics Theory and Applications, in submission*, volume 85, pages 13 – 28, 2015.

[CTL15b] Van-Toan Cao, Trung-Thien Tran, and Denis Laurendeau. A two-stage approach to align two surfaces of deformable objects. *Graphical Models*, pages 13 – 28, 2015.

[Cur99] Brian Curless. From range scans to 3D models. *ACM SIGGRAPH Computer Graphics*, 33(4):38–41, November 1999.

[CVC14] Marco Camurri, Roberto Vezzani, and Rita Cucchiara. 3D Hough transform for sphere recognition on point clouds. *Machine Vision and Applications*, 25(7):1877–1891, 2014.

[CYDL06] MY Cao, CH Ye, O Doessel, and C Liu. Spherical parameter detection based on hierarchical Hough transform. *Pattern recognition letters*, 27(9):980–986, 2006.

[CZ09] Will Chang and Matthias Zwicker. Range scan registration using reduced deformable models. In *Computer Graphics Forum*, volume 28, pages 447–456. Wiley Online Library, 2009.

[dC76] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[Dev15] Jay Devore. *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 2015.

[DG10] Jean-Emmanuel Deschaud and François Goulette. A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing. In *3DPVT*, 2010.

[DH72] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, January 1972.

[DRLS15] Yago Díez, Ferran Roure, Xavier Lladó, and Joaquim Salvi. A qualitative review on 3D coarse registration methods. *ACM Computing Surveys*, 47(3):45, 2015.

[DSCJRN14] Paulo Dias, Jo˜ao Silva, Rafael Castro, and Anto´nio J. R. Neves. Detection of aerial balls using a Kinect sensor. In *The 18th annual RoboCup International Symposium*, 2014.

[DSD14] M. Fatih Demirci, Ali Shokoufandeh, and Sven J. Dickinson. Many-to-many graph matching. In *Computer Vision, A Reference Guide*, pages 472–477. 2014.

[DVVR07] Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design*, 39(4):276–283, April 2007.

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FCSW09] M. Franaszek, G.S. Cheok, K.S. Saidi, and C. Witzgall. Fitting spheres to range data from 3-d imaging systems. *IEEE Transactions on Instrumentation and Measurement*, 58(10):3544–3553, Oct 2009.

[FGW02] Anders Forsgren, Philip E Gill, and Margaret H Wright. Interior methods for nonlinear optimization. *SIAM review*, 44(4):525–597, 2002.

[FO01] Clark F Olson. Locating geometric primitives by pruning the parameter space. *Pattern Recognition*, 34(6):1247–1256, 2001.

[Fol96] James D. Foley. 12.7 Constructive Solid Geometry. In *Computer Graphics: Principles and Practice*, page 557–558. Addison-Wesley Professional, 1996.

[For89] Alistair B Forbes. *Least-squares best-fit geometric elements*. National Physical Laboratory (NPL) Division of Information Technology and Computing, 1989.

[GGS94] Walter Gander, GeneH. Golub, and Rolf Strebel. Least-squares fitting of circles and ellipses. *BIT Numerical Mathematics*, 34(4):558–578, 1994.

[GH98] Michael Garland and Paul S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the Conference on Visualization '98*, VIS '98, pages 263–269, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[GMGP05] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05. Eurographics Association, 2005.

[GR96] Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, April 1996.

[GWH01] Michael Garland, Andrew Willmott, and Paul S Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58. ACM, 2001.

[GWM01] Stefan Gumhold, Xinlong Wang, and Rob Macleod. Feature extraction from point clouds. In *Proceedings of the Tenth International Meshing Roundtable*, pages 293–305, 2001.

[HB13] Dirk Holz and Sven Behnke. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In *Intelligent Autonomous Systems 12*, pages 61–73. Springer, 2013.

[HDD$^+$92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the Nineteenth annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, USA, 1992.

[HG01] Andreas Hubeli and Markus Gross. Multiresolution feature extraction for unstructured meshes. In *Proceedings of the conference on Visualization '01*, VIS '01, pages 287–294, Washington, DC, USA, 2001. IEEE Computer Society.

[HHRB12] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using RGB-D cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317. Springer, 2012.

[Hor87] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[HPW05] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. Smooth feature lines on surface meshes. In *Proceedings of the Third Eurographics symposium on Geometry processing*, SGP '05, pages 85–90, Aire-la-Ville, Switzerland, 2005. Eurographics Association.

[HR84] Donald D Hoffman and Whitman A Richards. Parts of recognition. *Cognition*, 18(1):65–96, 1984.

[HS03] Eyal Hameiri and Ilan Shimshoni. Estimating the principal curvatures and the Darboux frame from real 3-d range data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(4):626–637, 2003.

[HWG$^+$13] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao (Richard) Zhang. Edge-aware point set resampling. *ACM Transactions on Graphics*, 32(1):9:1–9:12, February 2013.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[JH99] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, May 1999.

[JLW05] Shuangshuang Jin, Robert R. Lewis, and David West. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21:71–82, 2005.

[Jol86] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[KFH09] David Koller, Bernard Frischer, and Greg Humphreys. Research challenges for digital archives of 3D cultural heritage models. *Journal on Computing and Cultural Heritage*, 2(3):7, 2009.

[KHS10] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3D mesh segmentation and labeling. In *ACM Transactions on Graphics*, volume 29, page 102. ACM, 2010.

[KLCK11] Changmin Kim, Joohyuk Lee, Minwoo Cho, and Changwan Kim. Fully automated registration of 3D CAD model with point cloud from construction site. In *Proceedings of 28th International Symposium on Automation and Robotics in Construction*, pages 917–922, 2011.

[KNSS09] Evangelos Kalogerakis, Derek Nowrouzezahrai, Patricio Simari, and Karan Singh. Extracting lines of curvature from noisy point clouds. *Computer-Aided Design*, 41(4):282–292, 2009.

[LCL09] Hou-Chuan Lai, Yi-Hong Chang, and Jiing-Yih Lai. Development of feature segmentation algorithms for quadratic surfaces. *Advances in Engineering Software*, 40(10):1011–1022, 2009.

[LDB05] Guillaume Lavoué, Florent Dupont, and Atilla Baskurt. A new cad mesh segmentation method, based on curvature tensor analysis. *Computer-Aided Design*, 37(10):975–987, 2005.

[Lev98] David Levin. The approximation power of moving least-squares. *Mathematics of Computation of the American Mathematical Society*, 67(224):1517–1531, October 1998.

[LGF07] Bart Lamiroy, Olivier Gaucher, and Laurent Fritz. Robust circle detection. In *Proceeding of the 9th International Conference on Document Analysis and Recognition, ICDAR'07*, volume 1, pages 526–530, 2007.

[LHS08] Irina Lavva, Eyal Hameiri, and Ilan Shimshoni. Robust methods for geometric primitive recovery and estimation from range images. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(3):826–845, 2008.

[Li11] Da Peng Li. Application of coordinate measuring machine in reverse engineering. *Advanced Materials Research*, 301-303:269–274, 2011.

[LkW14] Jin Liu and Zhong ke Wu. An adaptive approach for primitive shape extraction from point clouds. *Optik - International Journal for Light and Electron Optics*, 125(9):2000 – 2008, 2014.

[LM12] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation. *International Journal of Computer Vision*, 99(1):69–85, 2012.

[LMM98] Gabor Lukács, Ralph Martin, and Dave Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In *Proceeding of the 5th European Conference on Computer Vision*, pages 671–686. Springer, Freiburg, Germany, 1998.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[LPGW87] T. Lozano-Perez, W. Grimson, and S. White. Finding cylinders in range data. In *Proceedings of the IEEE International Conference on Robotics and Automation.*, volume 4, pages 202–207, 1987.

[LS15] Weimin Li and Pengfei Song. A modified {ICP} algorithm based on dynamic adjustment factor for registration of point cloud and {CAD} model. *Pattern Recognition Letters*, 65:88 – 94, 2015.

[LWC$^+$11] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, 30(4):52:1–52:12, July 2011.

[LZH$^+$13] Yong-Jin Liu, Jun-Bin Zhang, Ji-Chun Hou, Ji-Cheng Ren, and Wei-Qing Tang. Cylinder detection in large-scale point cloud of pipeline plant. *IEEE Transactions on Visualization and Computer Graphics*, 19(10):1700–1707, 2013.

[Mar63] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[May99] Roy Mayer. Scientific Canadian : Invention and innovation from Canada's National Research Council. 1999.

[MHSRP08] M Manzoor Hussain, CH Sambasiva Ra, and K. E. Prasad. Reverse engineering: point cloud generation with CMM for part modeling and error analysis. *Journal of Engineering and Applied Sciences*, 2008.

[MOG11] Q. Merigot, M. Ovsjanikov, and L.J. Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):743–756, june 2011.

[MS11] Gerald F Marshall and Glenn E Stutz. *Handbook of optical and laser scanning*. CRC Press, 2011.

[MSR07] Evgeni Magid, Octavian Soldea, and Ehud Rivlin. A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data. *Computer Vision and Image Understanding*, 107(3):139 – 159, 2007.

[MWA$^+$13] Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, L Gool, and Werner Purgathofer. A survey of urban reconstruction. In *Computer Graphics Forum*, volume 32, pages 146–177. Wiley Online Library, 2013.

[MWH$^+$06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM Transactions On Graphics*, 25(3):614–623, 2006.

[NTCL13] Van Tung Nguyen, Trung-Thien Tran, Van-Toan Cao, and D. Laurendeau. 3d point cloud registration based on the vector field representation. In *Proceedings 2nd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 491–495, Okinawa, Japan, Nov 2013.

[OBS04] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics*, 23(3):609–612, August 2004.

[OCBH07] Olatokunbo O Ogundana, C Russell Coggrave, Richard L Burguete, and Jonathan M Huntley. Fast Hough transform for automated detection of spheres in three-dimensional point clouds. *Optical Engineering*, 46(5):051002–051002, 2007.

[OGG09] A. C. Oztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009.

[Ope] OpenSCAD. http://www.openscad.org/.

[Ots79] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:62–66, 1979.

[PKG03] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–289, 2003.

[PLJ$^+$13] Min Ki Park, Seung Joo Lee, In Yeop Jang, Yong Yi Lee, and Kwan H Lee. Feature-aware filtering for point-set surface denoising. *Computers & Graphics*, 37(6):589–595, 2013.

[PLL12] Min Ki Park, Seung Joo Lee, and Kwan H Lee. Multi-scale tensor voting for feature extraction from unstructured point clouds. *Graphical Models*, 74(4):197–208, 2012.

[PMW05] Birgit M Planitz, Anthony J Maeder, and JA Williams. The correspondence framework for 3D surface matching algorithms. *Computer Vision and Image Understanding*, 97(3):347–383, 2005.

[Pra87] Vaughan Pratt. Direct least-squares fitting of algebraic surfaces. In *Proceedings of the Fourteenth Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 145–152, New York, NY, USA, 1987. ACM.

[PTVF07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[RA11] Helena Rua and Pedro Alvito. Living the past: 3D models, virtual reality and game engines as tools for supporting archaeology and the reconstruction of cultural heritage–the case-study of the Roman villa of Casal de Freiria. *Journal of Archaeological Science*, 38(12):3296–3308, 2011.

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Proceedings of the International Conference on Robotics and Automation (ICRA*, pages 3212–3217. IEEE, 2009.

[RDvdHV07] Tahir Rabbani, Sander Dijkman, Frank van den Heuvel, and George Vosselman. An integrated approach for modelling and global registration of point clouds. *{ISPRS} Journal of Photogrammetry and Remote Sensing*, 61(6):355 – 370, 2007.

[RH14] Minghao Ruan and Daniel Huber. Extrinsic calibration of 3D sensors using a spherical target. In *Proceedings of the International Conference on 3D Vision*, pages 187–193, Tokyo, Japan, Dec 2014.

[RL01] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.

[RvdH05] T. Rabbani and F.A. van den Heuvel. Efficient Hough transform for automatic detection of cylinders in point clouds. In *ISPRS WG III/3, III/4*, volume 3, pages 60–65, 2005.

[SASH12] Gerard Sanromà, René Alquézar, Francesc Serratosa, and Blas Herrera. Smooth point-set registration using neighboring constraints. *Pattern Recognition Letters*, 33(15):2029–2037, 2012.

[SBMM15] Aaron N. Staranowicz, Garrett R. Brown, Fabio Morbidi, and Gian-Luca Mariottini. Practical and accurate calibration of RGB-D cameras using spheres. *Computer Vision and Image Understanding*, 137:102 – 114, 2015.

[SCC$^+$11] Roberto Scopigno, Marco Callieri, Paolo Cignoni, Massimiliano Corsini, Matteo Dellepiane, Federico Ponchio, and Guido Ranzuglia. 3D models for cultural heritage: beyond plain visualization. *Computer*, (7):48–55, 2011.

[Sha08] Ariel Shamir. A survey on mesh segmentation techniques. In *Computer Graphics Forum*, volume 27, pages 1539–1556. Wiley Online Library, 2008.

[Sin64] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, pages 876–879, 1964.

[SLC$^+$08] Ioannis Stamos, Lingyun Liu, Chao Chen, George Wolberg, Gene Yu, and Siavash Zokai. Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes. *International Journal of Computer Vision*, 78(2-3):237–260, 2008.

[SMFF07] Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578–596, 2007.

[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

[TAL13] Trung-Thien Tran, Sarah Ali, and Denis Laurendeau. Automatic sharp feature extraction from point clouds with optimal neighbor size. In *Proceedings of the Thirteenth IAPR International Conference on Machine Vision Applications*, MVA '13, Kyoto, Japan, 2013.

[Tao15] Songqiao Tao. CAD model retrieval based on graduated assignment algorithm. *3D Research*, 6(2), 2015.

[Tau91] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.

[Tau95] Gabriel Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 902–907. IEEE, 1995.

[TCL$^+$13] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. Registration of 3D point clouds and meshes: a survey from rigid to nonrigid. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1199–1217, 2013.

[TCL15a] Trung-Thien Tran, Van-Toan Cao, and Denis Laurendeau. Extraction of cylinders and estimation of their parameters from point clouds. *Computers & Graphics*, 46:345 – 357, 2015.

[TCL15b] Trung-Thien Tran, Van-Toan Cao, and Denis Laurendeau. Extraction of reliable primitives from unorganized point clouds. *3D Research*, 6(4), 2015.

[TCL16a] Trung-Thien Tran, Van-Toan Cao, and Denis Laurendeau. 3D geometric primitive alignment revisited. In *Proceedings of the Eleventh International Conference on Computer Graphics Theory and Applications*, 2016.

[TCL16b] Trung-Thien Tran, Van-Toan Cao, and Denis Laurendeau. esphere: extracting spheres from unorganized point clouds. *The Visual Computer*, pages 1–18, 2016.

[TCL16c] Trung-Thien Tran, Van-Toan Cao, and Denis Laurendeau. A novel framework for registration of point cloud and CAD model. *in preparation*, 2016.

[TCN$^+$14] Trung-Thien Tran, Van-Toan Cao, Van-Tung Nguyen, Sarah Ali, and Denis Laurendeau. Automatic method for sharp feature extraction from 3D data of man-made objects. In *GRAPP 2014 - Proceedings of the Ninth International Conference on Computer Graphics Theory and Applications, Lisbon, Portugal, 5-8 January, 2014.*, pages 112–119, 2014.

[TPT15] Panagiotis Theologou, Ioannis Pratikakis, and Theoharis Theoharis. A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation. *Computer Vision and Image Understanding*, 135:49–82, 2015.

[TSDS13] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision*, 102(1-3):198–220, 2013.

[VAA14] B. Verney, M. Akhloufi, and O. Aubreton. Multimodal fusion system for ndt and metrology. In *Proceedings of 12th Quantitative InfraRed Thermography Conference, QIRT 2014*, 2014.

[vdGVBV02] Marjolein van der Glas, Frans M. Vos, Charl P. Botha, and Albert M. Vossepoel. Determination of position and radius of ball joints. In *Proceedings of the SPIE on Medical imaging 2002: Image Processing*, volume 4684, pages 1571–1577. International Society for Optics and Photonics, 2002.

[VMC97] Tamas Varady, Ralph R Martin, and Jordan Cox. Reverse engineering of geometric models—an introduction. *Computer-Aided Design*, 29(4):255–268, 1997.

[VRKS01] Jens Vorsatz, Christian Rössl, Leif P Kobbelt, and H-P Seidel. Feature sensitive remeshing. In *Computer Graphics Forum*, volume 20, pages 393–401. Wiley Online Library, 2001.

[VS05] Miguel Vieira and Kenji Shimada. Surface mesh segmentation and smooth surface extraction through region growing. *Computer Aided Geometric Design*, 22(8):771–792, 2005.

[WB01] Kouki Watanabe and Alexander G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, 20(3):385–392, 2001.

[Wei94] Mark Allen Weiss. *Data structures and algorithm analysis (2. ed.)*. Benjamin/Cummings, 1994.

[WGY$^+$12] Jun Wang, Dongxiao Gu, Zeyun Yu, Changbai Tan, and Laishui Zhou. A framework for 3D model reconstruction in reverse engineering. *Computers & Industrial Engineering*, 63(4):1189–1200, 2012.

[WHH10] C. Weber, S. Hahmann, and H. Hagen. Sharp feature detection in point clouds. In *Shape Modeling International Conference (SMI '10)*, pages 175–186, june 2010.

[WHHB12] Christopher Weber, Stefanie Hahmann, Hans Hagen, and Georges-Pierre Bonneau. Sharp feature preserving MLS surface reconstruction based on local feature line approximations. *Graphical Models*, 74(6):335–345, 2012.

[WSZZ14] Yanmin Wang, Hongbin Shi, Yanyan Zhang, and Dongmei Zhang. Automatic registration of laser point cloud using precisely located sphere targets. *Journal of Applied Remote Sensing*, 8(1):083588, 2014.

[WZC11] Kwan-Yee Wong, Guoqiang Zhang, and Zhihu Chen. A stratified approach for camera calibration using spheres. *IEEE Transactions on Image Processing*, 20(2):305–316, Feb 2011.

[YL99] M. Yang and E. Lee. Segmentation of measured point data using a parametric quadric surface approximation. *Computer-Aided Design*, 31(7):449–457, 1999.

[YQ07] Pinghai Yang and Xiaoping Qian. Direct Computing of Surface Curvatures for Point-Set Surfaces. In *Proceedings of IEEE/Eurographics Symposium on Point-based Graphics (PBG 2007)*, pages 29–36, 2007.

[YW14] Quan Yu and Kesheng Wang. A hybrid point cloud alignment method combining particle swarm optimization and iterative closest point method. *Advances in Manufacturing*, 2(1):32–38, 2014.

[YWLY12] Dong-Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design*, 44(11):1072–1082, 2012.

[ZCL$^+$13] Jie Zhang, Junjie Cao, Xiuping Liu, Jun Wang, Jian Liu, and Xiquan Shi. Point cloud normal estimation via low-rank subspace clustering. *Computers & Graphics*, 37(6):697–706, 2013.