



# **3D Reconstruction Using High Resolution Implicit Surface Representations and Memory Management Strategies**

**Thèse**

**Mana Eskandari**

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

Québec, Canada

# **3D Reconstruction Using High Resolution Implicit Surface Representations and Memory Management Strategies**

**Thèse**

**Mana Eskandari**

Sous la direction de:  
Denis Laurendeau, directeur de recherche

# Résumé

La disponibilité de capteurs de numérisation 3D rapides et précis a permis de capturer de très grands ensembles de points à la surface de différents objets qui véhiculent la géométrie des objets. La métrologie appliquée consiste en l'application de mesures dans différents domaines tels que le contrôle qualité, l'inspection, la conception de produits et la rétro-ingénierie. Une fois que le nuage de points 3D non organisés couvrant toute la surface de l'objet a été capturé, un modèle de la surface doit être construit si des mesures métrologiques doivent être effectuées sur l'objet.

Dans la reconstruction 3D en temps réel, à l'aide de scanners 3D portables, une représentation de surface implicite très efficace est le cadre de champ vectoriel, qui suppose que la surface est approchée par un plan dans chaque voxel. Le champ vectoriel contient la normale à la surface et la matrice de covariance des points tombant à l'intérieur d'un voxel. L'approche globale proposée dans ce projet est basée sur le cadre Vector Field. Le principal problème abordé dans ce projet est la résolution de l'incrément de consommation de mémoire et la précision du modèle reconstruit dans le champ vectoriel.

Cette approche effectue une sélection objective de la taille optimale des voxels dans le cadre de champ vectoriel pour maintenir la consommation de mémoire aussi faible que possible et toujours obtenir un modèle précis de la surface. De plus, un ajustement de surface d'ordre élevé est utilisé pour augmenter la précision du modèle. Étant donné que notre approche ne nécessite aucune paramétrisation ni calcul complexe, et qu'au lieu de travailler avec chaque point, nous travaillons avec des voxels dans le champ vectoriel, cela réduit la complexité du calcul.

# Abstract

The availability of fast and accurate 3D scanning sensors has made it possible to capture very large sets of points at the surface of different objects that convey the geometry of the objects. Applied metrology consists in the application of measurements in different fields such as quality control, inspection, product design and reverse engineering. Once the cloud of unorganized 3D points covering the entire surface of the object has been captured, a model of the surface must be built if metrologic measurements are to be performed on the object.

In real-time 3D reconstruction, using hand-held 3D scanners a very efficient implicit surface representation is the Vector Field framework, which assumes that the surface is approximated by a plane in each voxel. The vector field contains the normal to the surface and the covariance matrix of the points falling inside a voxel. The proposed global approach in this project is based on the Vector Field framework. The main problem addressed in this project is solving the memory consumption increment and the accuracy of the reconstructed model in the vector field.

This approach performs an objective selection of the optimal voxels size in the vector field framework to keep the memory consumption as low as possible and still achieve an accurate model of the surface. Moreover, a high-order surface fitting is used to increase the accuracy of the model. Since our approach do not require any parametrization and complex calculation, and instead of working with each point we are working with voxels in the vector field, then it reduces the computational complexity.

# Table of Contents

Résumé .....	iii
Abstract .....	iv
Table of Contents .....	v
List of Tables .....	vii
List of Figures .....	viii
Acknowledgments .....	xi
Introduction .....	1
1. Literature Review .....	4
1.1 Differential geometry .....	4
1.1.1 First fundamental form .....	4
1.1.2 Second fundamental form .....	5
1.1.3 Normal curvature .....	6
1.1.4 Principal curvature .....	6
1.1.5 Gaussian curvature .....	7
1.1.6 Mean curvature .....	8
1.1.7 Local description of smooth surfaces by classical differential geometry .....	8
1.2 Surface Representation Methods .....	9
1.2.1 Explicit surface representation .....	9
1.2.2 Implicit surface representation .....	10
1.2.3 Signed distance field .....	12
1.2.4 The Vector Field surface representation .....	13
1.3 3D Segmentation .....	15
1.3.1 Segmentation using covariance techniques .....	20
1.4 Surface fitting for 3D point clouds .....	23
1.4.1 Implicit polynomial fitting .....	24
1.5 Blending different curves and surfaces .....	26
1.5.1 Merging B-Spline curves or surfaces .....	27
1.5.2 Stitching Triangles .....	28
1.5.3 Zippered Polygon Meshes .....	28

2. Proposed Approach and Methodology .....	31
2.1 Data Acquisition and Sensor Position .....	31
2.2 Identification of the Different Surface Types in the Vector Field Framework Using Covariance Based Differential Geometry .....	36
Implementation.....	38
2.3 Surface Variation Method in the Vector Field Framework.....	42
Implementation.....	45
2.4 3D Segmentation method in the vector field.....	47
Implementation.....	49
2.5 Fitting a high order surface representation.....	52
Implementation.....	54
2.6 Merging different surface patches to obtain a single 3D mesh .....	56
Implementation.....	57
3. Experimental Results .....	60
3.1. Preparing the data.....	60
3.2. Identifying the surface type in the vector field.....	60
3.3. Surface variation method in the vector field framework.....	62
3.4. 3D segmentation method in the vector field .....	65
3.5. Fitting a high-order surface representation .....	66
3.6. Merging different surface patches to obtain a single 3D mesh .....	73
3.7. Experimental results achieved of the implementation in C++ .....	77
3.8. More results for different sample objects obtained in MATLAB .....	81
Conclusion.....	85
Bibliography.....	88

# List of Tables

Table 1.1 : Types of surface as a function of the signs of the Gaussian and Mean Curvatures .....	20
Table 2.1 : Color map corresponding to different surface types .....	40

# List of Figures

Figure 1.1 : Geometric illustration of the first fundamental form [9] .....	5
Figure 1.2: The second fundamental form keeps track of the change of the unit normal vector as a function of a displacement on the surface [12]. .....	5
Figure 1.3: Normal curvature. Figure taken from [13].....	6
Figure 1.4 : The definition of principal curvatures, $k_1$ and $k_2$ [16].....	7
Figure 1.5 : Gaussian curvature. a) Positive Gaussian curvature (tangent plane intersects the surface at one point). b) Negative Gaussian curvature (tangent plane intersects the surface along two curves). c) Zero Gaussian curvature (tangent plane intersects the surface along one curve). Figure taken from [13]. .....	7
Figure 1.6 : Mean curvature. Figure taken from [18].....	8
Figure 1.7 : Mesh definition; vertices: blue points, edges: green lines, faces: yellow triangles [22].	10
Figure 1.8 : Different forms of implicit surface representation. a) Point Cloud. b) Signed distanced field. c) Vector field. Figure taken from [24]. .....	11
Figure 1.9 : The Vector Field implicit representation [30]. .....	15
Figure 1.10 : A small patch near a point P on surface $\square(u,v)$ and the tangent plane $\square_t$ .....	17
Figure 1.11 : Geometry for the vectors in Equations (1.23), (1.24) and (1.25).....	21
Figure 1.12 : Inner offset (blue), outer offset (red) and original data (black) for 2D objects [45]...	25
Figure 1.13 : (a) Blending two functions. (b) Given blending function. Figure taken from [48]. ....	27
Figure 1.14 : Merging two B-spline curves with $C^0$ continuity (i.e. curves are connected at a joint). Figure taken from [49]. .....	28
Figure 1.15 : Searching for corresponding points for mesh registration. Dotted arrows show the matches that should be prevented since they cause the mesh K to be dragged up and left by mistake. Figure borrowed from [54]. .....	29
Figure 2.1 : The 3D scanning process using a handheld scanner [30] .....	31
Figure 2.2 : Multi view acquisition of range data [24].....	32
Figure 2.3: Obtain sensor positions in the vector field framework: Surface in green color represent the original data. P1 to P8 indicate the original input points. Dashed lines shown in Blue color present the sensor position's path. S1 to S8 shown in blue color indicate the sensor positions associated to each original points with the same indices number.....	33
Figure 2.4 : Position of the different surface type to the tangent plane: (a) Elliptic surface (b) Hyperbolic surface (C) Parabolic surface. Figure taken from [59] .....	37
Figure 2.5: An illustration of <b><i>Distancej</i></b> stored in the voxel of interest whose sides are in dotted lines. ....	39
Figure 2.6: Identifying the surface type at each voxel on a synthetic sphere. (a) Shows the peak in blue color. (b) Shows the peak in blue color and pit in yellow color.....	41
Figure 2.7: Identifying the surface types on synthetic cylinders. (a) Shows the ridge surface in green (b) Show the ridge surface in green and valley surface in black color.....	41
Figure 2.8: Identifying the surface types on 3D points provided by Stanford repository. (a) Object mesh data. (b) Shows the different surface types .....	42
Figure 2.9: Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): (a) Voxel size = 10 mm (b) Voxel size = 9 mm .....	45
Figure 2.10: Sample data point to show the K-means segmentation that uses the elbow method ....	48
Figure 2.11: The elbow method for determining the number of segments .....	49



Figure 2.12: Segmentation of a point cloud using K-means elbow method.....	49
Figure 2.13: Segmentation of the Ridge region. (a) Initial data points. (b) Segmented regions .....	50
Figure 2.14: Segmentation of the Peak region. (a) Initial data points. (b) Segmented regions .....	50
Figure 2.15: Segmentation of the valley region. (a) Initial data points in the valley region. (b) Segmented regions .....	50
Figure 2.16: Segmentation of the saddle region. (a) Initial data points in the saddle region. (b) Segmented regions .....	51
Figure 2.17: All segmented regions. (a) Segmented ridge and peak surface (b) Segmented Ridges, Peak and valley surface (c) Segmented Ridges, peak, valley and Pit region (d) Segmented Ridges, peak, valley, Pit and saddle region. ....	51
Figure 2.18: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Ridge segmented regions .....	54
Figure 2.19: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Valley segmented regions .....	54
Figure 2.20: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Peak segmented regions .....	55
Figure 2.21: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Pit segmented regions.....	55
Figure 2.22: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Saddle segmented regions .....	55
Figure 2.23: Fusion of field-functions for cross-sections through overlapping volumetric envelopes (a) Overlapping field-functions (M1, M2) (b) Integrated field-function (M). Taken from [67, 68].	56
Figure 2.24: Merge different meshes in the VXELEMENTS software platform. ....	58
Figure 2.25: Merge different meshes in the VXELEMENTS software platform. ....	58
Figure 2.26: The surface obtained from merging different meshes. (a) The result with less smoothing weight (b) The result with higher smoothing weight. ....	59
Figure 3.1 : The result of identifying the surface types on a 3D point cloud generated in MATLAB .....	61
Figure 3.2 : The result of the identifying the surface types on a 3D point cloud collected by HandyScan (Creaform) scanner.....	62
Figure 3.3 : Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): Voxel size =7 mm .....	64
Figure 3.4 : Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): Voxel size =6 mm .....	64
Figure 3.5 : Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): Voxel size =5 mm .....	64
Figure 3.6 : Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): Voxel size =4 mm .....	65
Figure 3.7: 3D segmentation of the object provided by Creaform according to the different surface types. (a) Ridge segments (b) Peak segments (c) Valley segments (d) Saddle segments .....	66
Figure 3.8 : 3L-IBS fitting for synthetic object. (a) Generating inner offset (red) and outer offset (blue), original data (green) for cylindrical segment. (b) IBS fitted to the cylindrical segment area. Voxel size = 3 mm .....	68
Figure 3.9 : 3L-IBS fitting for synthetic object. (a , b) Generating inner offset (red) and outer offset (blue), original data (green) for planar segment. (c , d) IBS fitted to the planar segment area. Voxel size = 3 mm .....	68

Figure 3.10 : (a) 2D illustration of simulated cylindrical data in larger voxel size (Red, green and purple line show the ripples). (b) 2D illustration of simulated cylindrical data in smaller voxel size (Purple and red line show the ripples).....	69
Figure 3.11: (a) Smoothing mesh in VXelements application. (b) The result of using smooth mesh tools in VXelements (Voxel size = 3mm). .....	70
Figure 3.12 : 3L-IBS fitting. (a) Generating inner offset (red) and outer offset (blue) for ridge segment. (b) IBS fitted to the ridge segment area. Voxel size = 5 mm .....	71
Figure 3.13 : 3L-IBS fitting. (a) Generating inner offset (red) and outer offset (blue) for valley segment. (b) IBS fitted to the valley segment area. Voxel size = 5 mm .....	71
Figure 3.14 : 3L-IBS fitting. (a) Generating inner offset (red) and outer offset (blue) for peak segment. (b) IBS fitted to the peak segment area. Voxel size = 5 mm.....	72
Figure 3.15 : 3L-IBS fitting. (a) Generating inner offset (red) and outer offset (blue) for saddle segment. (b) IBS fitted to the saddle segment area. Voxel size = 5 mm.....	72
Figure 3.16: Surface representation for planar regions of the synthetic object displayed in VXelements .....	73
Figure 3.17: Surface representation for the cylindrical region of the synthetic object displayed in VXelements.....	74
Figure 3.18: Merging the ridge and planar area obtained from the synthetic object.....	74
Figure 3.19 : The result of merging the different patches for the synthetic data.....	75
Figure 3.20: Merging patches for the 3D point collected by the HandyScan Creaform scanner. (a) Merging four different patches (b) 3D mesh achieved by merging (c) different view of the 3D mesh .....	76
Figure 3.21: Results of the proposed methods on a cube object. (a) Object mesh data (b) Identifying different surface type (c) constructed model with 40×40×40 resolution in vector field. ....	78
Figure 3.22: Results of the proposed methods for an electrical outlet. (a) , (b) Object mesh data (c) , (d) Identifying different surface types. (e) Constructed model with 50×50×50 resolution in vector field .....	79
Figure 3.23: Results of the proposed methods for a car body. (a) Object mesh data (b) Identifying different surface types. (c) Constructed model with 50×50×50 resolution in vector field.....	80
Figure 3.24 : Results of the proposed method on a teapot. (a) Object mesh data. (b) Identifying different surface types. (c) Different views of the constructed model after applying the post processing methods. Voxel size = 6mm.....	81
Figure 3.25: Results of the proposed method on a Bunny. (a) Object mesh data. (b) Identifying different surface types. (c) Constructed model after applying the post processing methods. Voxel size = 6mm .....	82
Figure 3.26 : Results of the proposed method on a synthetic saddle data. (b) Identifying different surface types. (c) Constructed model after applying the post processing methods. Voxel size = 6 mm .....	83
Figure 3.27 : Results of the proposed method on a synthetic plane data. (b) Identifying different surface types. (c) Constructed model. Voxel size = 6 mm.....	84

# Acknowledgments

I would like to express my special appreciation and thanks to my supervisor, Professor Denis Laurendeau. No words can express my sincere appreciation. I will forever be thankful for his endless patience, his guidance, his kind and supportive behaviour and his wisdom over the whole period of my PhD from inception to completion. I have been very lucky to be his student and I thank him for allowing me to grow, enlightening my world and for never giving up on me.

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 their valuable advice. In particular, I would like to thank Mrs. Annette Schwerdtfeger for her kind attitude, her precious advice, proofreading and correcting all my documents.

I take this opportunity to express my profound gratitude, from the bottom of my heart to my beloved parents, my husband, my sister, my brother-in law and my beloved niece. A very special thanks for their love and continuous support throughout this experience.

I am also grateful to NSERC/Creaform Industrial Research Chair on 3D Scanning for its financial support during this project. This study would not have been possible without this support.

# Introduction

Over the years, 3D sensors have become very popular because they are cheaper and easier to use than classical Coordinate Measuring Machines (CMM) while still achieving metrologic accuracy. In addition, 3D sensors can capture dense point clouds that convey the geometry of the object in real-time. Comparatively, capturing dense 3D point clouds with a CMM can be a very tedious and time consuming process. Capturing the geometry of parts is not only important in metrology but is also useful in many fields such as reverse engineering, design intent assessment and graphics rendering.

Applied metrology consists in the application of measurements in different fields such as quality control, inspection, product design and reverse engineering. By considering accuracy and precision as two important components of metrology, the time needed to achieve the measurements is a relevant issue, especially in the context of quality control in an industrial context for which a large number of parts have to be processed. The ease with which the scan can be obtained by users is also important when such users are domain specialists but not necessarily experts in 3D scanning. Handheld 3D sensors have become more popular because they allow the user to scan an object by moving the sensor around its surface while collecting the 3D coordinates of points on the surface.

Transforming an unstructured set of 3D points captured by these devices into a meaningful digital representation of the scanned object is called 3D surface reconstruction. This 3D surface reconstruction can be categorized into explicit surface reconstruction as summarized in [1, 2, 3] and implicit surface reconstruction as summarized in [4, 5].

The vector field representation is a novel real-time implicit representation approach proposed by Tubic, *et.al.* [5]. As described in the next chapters, the vector field is a volumetric structure composed of cubic voxels. The vector field has demonstrated its ability to support

in real-time the three most important steps in 3D modeling which are: view registration, view integration and model visualization. The vector field implicit surface representation enables all modeling steps to execute with linear complexity. However, if an accurate model needs to be built, the voxel size must be small which may lead to huge memory requirements specially where large objects must be modeled.

The overall goal of the research project presented in this thesis is to reconstruct an accurate 3D model using the vector field framework while keeping the memory requirements as low as possible and decreasing the processing time since the processing focuses on voxels instead of individual points.

The processing steps to achieve our goal are as follows:

- Identifying different surface types in the vector field

In this step, the proposed technique combines covariance-based differential geometry and the vector field implicit surface representation to identify the different surface types in each voxel of the vector field framework. Instead of working with each point and the neighborhood around that point, we rather propose to work with the voxels in the volumetric grid containing the vector field and their 26 possible neighbors in the grid. This step results into the identification of six (ridge, peak, valley, pit, planar and saddle) different surface types. We can consider this step as an initial segmentation.

- Selecting an optimal voxel size in the vector field representation

In the commercial metrology software, the voxel size in the implicit surface representation is usually selected arbitrarily by the user irrelevant of the number of points falling inside each voxel. This often results in huge vector fields.

In a metrology workflow using handheld 3D scanners based on an implicit surface representation, the processing step consists in selecting of the voxel size by the user, the scanning of a given object and then the observation of the model. Choosing the voxel size arbitrarily is not optimal. A good final model may not be achieved by choosing a large voxel size. In contrast, choosing a small voxel size results in the allocation of a large number of

voxels, which increases memory consumption. In addition, if very few points fall inside a voxel, the surface reconstruction in such a voxel may not be reliable. To solve such a problem we propose an approach that enables the voxel size to be selected systematically, which allows a good final model of the surface to be achieved while keeping the number of voxels as small as possible.

- 3D Segmentation method in the vector field

In this step, a K-means segmentation approach is implemented to group voxels belonging to the same surface type.

- Fitting a high order surface representation to the extracted segments

In this step, a high-order surface representation using 3L-Implicit B-Splines is implemented. This technique can provide shape descriptor through their zero-sets and reconstruct surfaces. These techniques are based on locally controlled functions that are combined via control points. This local control allows patch-based object representation. In this step, a high order surface is thus fitted to each segment corresponding to a given surface type.

- Merging different surface patches to obtain a single 3D mesh

The last step to achieve a final high order model is blending different surface patches obtained from the fitting step. To obtain such a model, Creaform's VXeElements is used to merge the different patches and results in a final mesh.

To test the proposed approach, we used a set of 3D data provided by the Stanford repository, a handheld 3D scanner provided by Creaform and synthetic data generated in MATLAB.

This thesis is organized as follows: Chapter 1 presents an overview of related works. The methodology for producing a high-order model based on the vector field framework is explained in Chapter 2. The experimental results are discussed in Chapter 3. Conclusion and future work complete the document.

# Chapter 1

## Literature Review

This chapter presents a review of the most relevant literature to our project.

### 1.1 Differential geometry

Differential geometry is a field of mathematics that focuses on the geometry of curves, surfaces, and manifolds. Riemannian computations based on the geometry of underlying manifolds are often faster and more stable than their classical, Euclidean match. Therefore, Riemannian computations and topological computing are becoming increasingly popular in the computer vision and machine learning communities [6].

In the following, more details on differential geometry will be described, such as the first fundamental form, second fundamental form, Normal, principal, mean and Gaussian curvatures.

#### 1.1.1 First fundamental form

The first fundamental form is a way of measuring the distance on a surface and allows the determination of the length of a tangent vector in a tangent plane [7, 8].

The first fundamental form is invariant to the translation and rotation of the surface and is invariant to surface parametric change as well. Therefore, it is an intrinsic property of a surface [3]. Let  $r: D \rightarrow S$  be a map of surface  $S$ . A geometric illustration of the first fundamental form is shown in Figure 1.1 [9].

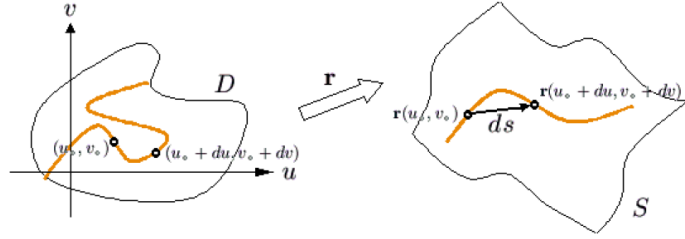


Figure 1.1 : Geometric illustration of the first fundamental form [9]

### 1.1.2 Second fundamental form

Like the first fundamental form, the second fundamental form is a symmetric bilinear form on each tangent space of a surface [10]. The second fundamental form is a way for measuring the correlation between the change in normal vector  $dn$  and the change in the surface position  $dx$  at a surface point  $(u, v)$  as a function of a small movement  $(du, dv)$  in parameter space [11]. The second fundamental form of a surface in the Euclidean space measures the change of the unit normal direction from point to point on the surface. Suppose  $V$  and  $W$  are tangent vector fields on a surface  $M \subseteq E^3$ , a geometric illustration of the second fundamental form is shown in figure 1.2.

The first fundamental form is an intrinsic property of a surface while the second fundamental form is extrinsic [10]. More detail in mathematical concept of the first fundamental form and second fundamental form is given in section 1.3.

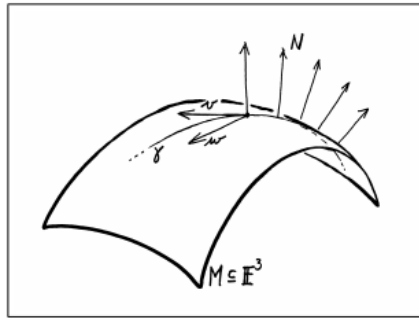


Figure 1.2: The second fundamental form keeps track of the change of the unit normal vector as a function of a displacement on the surface [12].



### 1.1.3 Normal curvature

The intersection of a surface and a plane containing the normal is described as a normal curve. As indicated in figure 1.3, the curvature of a normal curve, which is belonging to the plane containing the normal and the surface, is called a normal curvature  $k_n$  [13].

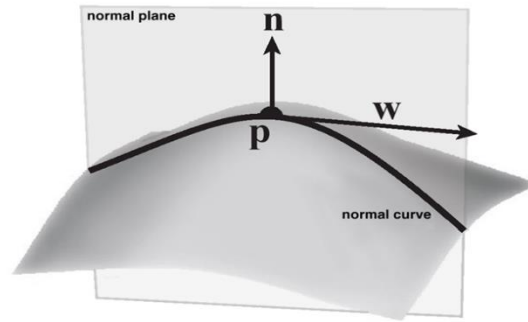


Figure 1.3: Normal curvature. Figure taken from [13]

The ratio of the second fundamental form to the first fundamental form as  $k_n = \frac{II}{I}$  is also called the normal curvature [14].

### 1.1.4 Principal curvature

At any given point on a surface, the maximum ( $k_1$ ) and minimum ( $k_2$ ) of the normal curvature are called the principal curvatures. The principal curvatures measure the maximum and minimum bending of a regular surface at each point [15].

In fact, the normal curvature varies between some maximum and minimum, which are the principal curvatures  $k_1$  and  $k_2$ . An illustration of the principal curvatures is given in figure 1.4.

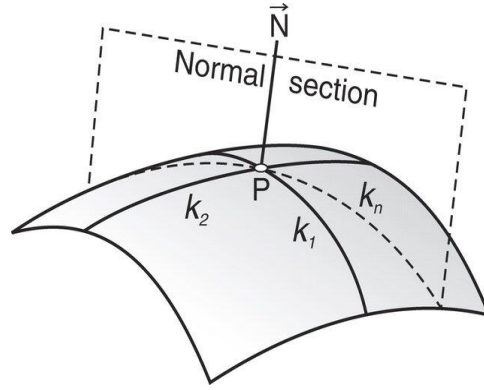


Figure 1.4 : The definition of principal curvatures,  $k_1$  (Minimum) and  $k_2$  (Maximum) [16]

### 1.1.5 Gaussian curvature

Gaussian curvature is an intrinsic property of a space independent of the coordinate system used to describe it. The Gaussian curvature of a regular surface in  $R^3$  at a point  $p$  is defined as

$$K(p) = \det(S(p)) \quad (1.1)$$

where  $S$  is the shape operator and  $\det$  denotes the determinant [17]. The Gaussian curvature ( $K$ ) is also defined as the product of the principal curvatures  $K = k_1 \cdot k_2$  [14]. An illustration of positive, negative and zero Gaussian curvature is given in Figure 1.5.

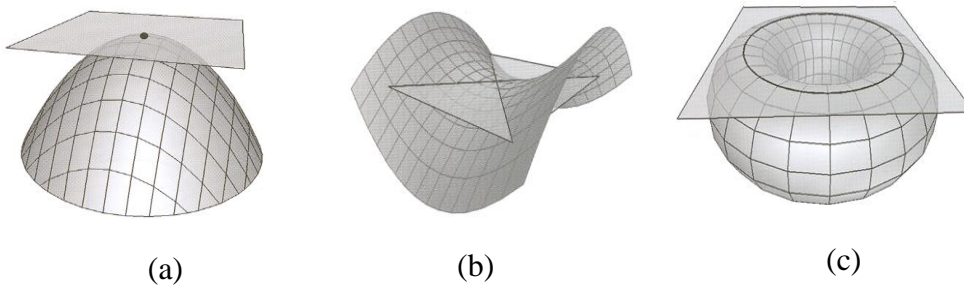


Figure 1.5 : Gaussian curvature. a) Positive Gaussian curvature (tangent plane intersects the surface at one point). b) Negative Gaussian curvature (tangent plane intersects the surface along two curves). c) Zero Gaussian curvature (tangent plane intersects the surface along one curve). Figure taken from [13].

### 1.1.6 Mean curvature

The mean curvature of a surface is an extrinsic measure of curvature that comes from differential geometry and that locally describes a surface embedded in some ambient space such as a Euclidean space [10]. See figure 1.6.

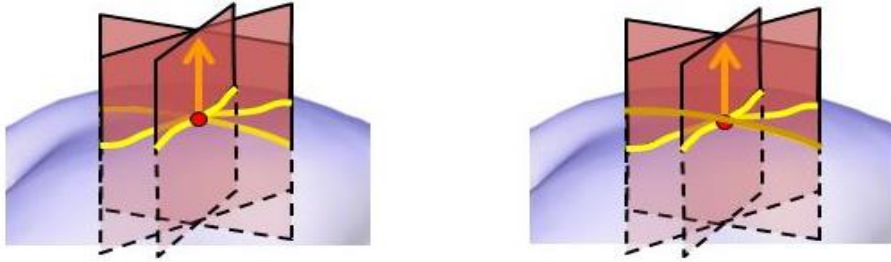


Figure 1.6 : Mean curvature. Figure taken from [18]

The mean of the principal curvatures  $H = \frac{1}{2}(k_1 + k_2)$  is also called the mean curvature.

### 1.1.7 Local description of smooth surfaces by classical differential geometry

Classical differential geometry provides a complete local description of smooth surfaces [11]. The first and second fundamental forms of surfaces provide a set of differential-geometric shape descriptors that capture domain independent surface information [11]. The Gaussian curvature, which refers to an isometric invariant of a surface, represents an intrinsic property of a surface and the mean curvature represents an extrinsic property of a surface. Both Gaussian and mean curvatures share the property of being translationally and rotationally invariant [19]. The Mean and Gaussian curvatures are important in computer vision because using the sign of these curvatures can be used to classify a range image into one of eight basic viewpoint-independent surface types [11]. More details for this approach are given in section 1.3.

## **1.2 Surface Representation Methods**

The set of unorganized 3D points covering the entire surface of the object generated by 3D scanning devices typically needs to be converted into a meaningful digital representation of the scanned objects to facilitate its use in various application domains such as computer-aided design, medical imaging, reverse engineering, virtual reality, and architectural modelling [20]. The representation of data determines the structure and complexity of the algorithms used in the classical sequential modeling steps in 3D view which are integration, surface integration and visualization [20]. In the following discussion, the surface representation methods are classified into explicit surface representation and implicit surface representation.

### **1.2.1 Explicit surface representation**

Explicit surface representation is a prescription of the precise location of the surface. It means that an explicit representation of a reconstructed surface from a point cloud lies on the points that are scanned from a real object and are present in the cloud [20]. For the explicit surface representation technique, no postprocessing is required to obtain the model. In this case, the representation of the surface is directly available and is defined by its geometry. In this method, when the surface evolves gradually, then all of the points on the surface are stored and these points define the entire surface. It means that the points that are captured with a 3D scanner are the ones that are forming the mesh. More formally, explicit surface representations rely on the points that are obtained by a scanner when scanning a real object. There are two different kinds of explicit surfaces: (1) parametric and (2) triangulated.

A parametric surface is the deformation of an initial model that covers an arbitrary part of the points. Some of the primitives being used are B-Spline, NURBS, plane, sphere and ellipsoid. By using parametric surfaces, complicated surfaces are not representable simply and they may need a mixture of primitives because the initial model topologically limits the parametric surfaces [20].

Triangulated surfaces are the most common version for explicit surface representation in which the surface is described by connected triangles made from the input points using k-nearest neighbors of a point to build the connectivity. A surface representation using a triangle mesh consists of a set of vertices  $V = \{v_1, \dots, v_n\}$  and these vertices are connected by a set of triangular faces  $F = \{f_1, \dots, f_m\}$ . The geometric embedding of a triangular mesh into  $R^3$ , which is shown in figure 1.7, is specified by assigning a 3D position  $p_i$  to each vertex [21].

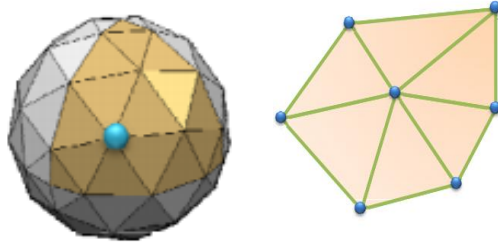


Figure 1.7 : Mesh definition; vertices: blue points, edges: green lines, faces: yellow triangles [22].

### 1.2.2 Implicit surface representation

Implicit surface representation is defined by a function which one of its isosurface is a close estimation to represent the input data. A surface can be described implicitly in a closed mathematical form by a single equation [20, 23]:

$$F(x, y, z) - T = 0 \quad (1.2)$$

Different surfaces by changing the value of constant  $T$  will be generated. Each generated surface is called an isosurface and function  $F$  always returns a scalar value. Function  $F$  is therefore called a scalar function [23]. The basic concept behind the implicit representation is to characterize the entire surface by classifying each 3D point to lie either inside, outside, or exactly on the surface. The negative value of function  $F$  defines points inside the object and positive value defines points outside the object. The zero-level isosurface contains the points located exactly on the surface, separating the inside from the outside.

Implicit surface reconstruction is actually the process of finding a function that best fits the input data and needs to be postprocessed in order to be visualized. The most common method to generate a triangulated surface from the implicit representation of the surface is the marching cubes algorithm [20].

There are different forms of implicit surface representation such as point clouds, signed distance fields and vector fields, see figure 1.8, which in order to be visualized as a surface, all need to be postprocessed.

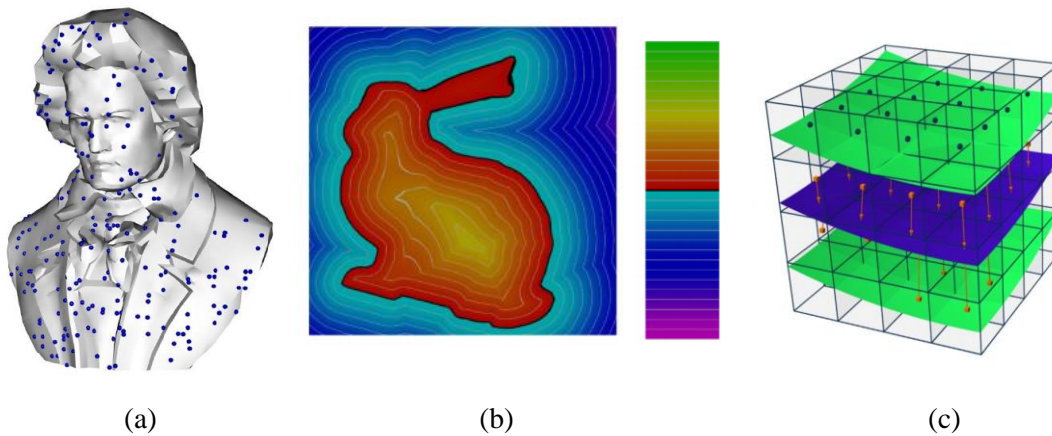


Figure 1.8 : Different forms of implicit surface representation. a) Point Cloud. b) Signed distanced field. c) Vector field. Figure taken from [24].

An implicit surface representation guarantees the most compact mathematical information representation. However, they do not guarantee memory compactness. This means that a large amount of memory space is consumed to represent the model. Generally, implicit surface representations are hard to deform and render, but they are well adapted to topological changes such as merging, twisting and pinching [23].

By contrast with the explicit representation, implicit surface representations are hard to deform and render, but they are well adapted to topological changes such as merging, twisting and pinching [25]. Explicit representations are not ideal for fitting surfaces to potentially noisy and incomplete data such as 3D points because fitting implies the minimization of a

non-differentiable distance function [20, 23, 25]. Besides that, building a model in real time is very difficult with current computer hardware because of the computational complexity.

### 1.2.3 Signed distance field

Masuda [26] proposed a framework named signed distance field in which the registration and integration were solved together to generate a geometric model from multiple range images. The signed distance field is a volumetric representation which is a scalar field determined by the signed distance of an arbitrary 3-D point from the object surface. In the signed distance field there is no need to consider the partial duplication such as the pairwise registration because an integrated shape will be generated in advance and each data can be registered to the integrated shape.

In the aforementioned method described in [26], first all the views of a given object were transformed to a reference coordinate system using the initial estimation of rigid motion. A set of key points is generated on a 3D space and the closest point to every key point is searched for in order to find the correspondences needed for the estimation of the rigid motion [27]. Note that this algorithm is not suitable for real time applications since all views should be available to compute this estimation [27, 28]. This process is repeated until convergence is achieved and an integrated model is built. At each iteration, the closest points are computed using the normal vectors and signed distance fields.

A major issue when performing view registration and view integration is to find the closest point on a surface to a given point. Defining the distance for the closest point in the signed distance field only means that the closest point to a surface can be anywhere on a sphere whose radius is equal to the distance. In addition, this method is not suitable for all of the 3D modeling steps such as visualization. It sometimes requires transforming a representation into another during the process which may cause errors [29, 24]. Therefore, the problem of computational complexity of finding the closest point to a surface still exists with this method and it cannot achieve real time performance since the complexity of finding the closest point grows exponentially with the number of data points. As a solution for the above problem, Tubic *et.al.* [5] proposed a framework based on the vector field.

### 1.2.4 The Vector Field surface representation

The vector field is a volumetric framework for 3D data representation proposed by Tubic *et.al.* [5]. In the vector field representation, the main point is that this representation allows registration, integration and visualization to be achieved in real time. For doing so, the key element is searching for the closest point efficiently. In the case of a large number of points, visiting all of the points to find the closest one would be very complex. But with the vector field representation as demonstrated below, finding the closest point is done simply by using Equation (1.7) which makes this process efficient. In the following, more details are given relative to finding the closest point using the vector field representation.

As shown in Figure 1.9 (a), the vector field is composed of a regular grid made of cubic voxels with side length  $L$ . Each voxel in the grid is addressed by the coordinate of its center  $v_{ijk}$  in a reference frame  $W_r$ . Let us assume that 3D points with coordinates  $p_u$  on the surface  $S$  are collected by the sensor in frame  $W_r$ . The covariance matrix  $C_{ijk}$  of the points falling in voxel  $ijk$  is defined as in Eq. (1.3):

$$C_{i,j,k} = \frac{1}{N} \sum_{u=1}^N (p_u - \bar{p})(p_u - \bar{p})^t \quad (1.3)$$

where  $\bar{p}$  is the mean vector of points  $p_u$  as defined in Eq. (1.4).

$$\bar{p} = \frac{1}{N} \sum_{u=1}^N p_u \quad (1.4)$$

If the voxel size  $L$  is small enough, it is assumed that the object surface in the voxel can be approximated by the plane  $\Pi_T$  tangent to the surface. The normal vector to tangent plane  $\Pi_T$  is the eigenvector corresponding to the smallest eigenvalue  $\lambda_{min}$  of  $C_{ijk}$ .

The eigenvector associated with the smallest eigenvalue of the covariance matrix represents the normal vector  $N$  to the tangent plane at the closest point [29]. The direction of the vector field  $F(v)$  at the voxel  $v$  is computed as



$$F(v_{ijk}) = N \langle N, \bar{p} - v_{ijk} \rangle \quad (1.5)$$

where  $\langle, \rangle$  is the scalar product.

$F(v_{ijk})$ ,  $c_v$  and  $C_{ijk}$  are stored in each voxel. In the voxel, point  $c_v$  on the tangent plane that is closest to  $v_{ijk}$  is given by Equation. (1.6).

$$c_v = F(V_{i,j,k}) + V_{i,j,k} \quad (1.6)$$

As new points are collected by the sensor,  $C_{ijk}$ ,  $\lambda_{min}$ ,  $F(v_{ijk})$  and  $c_v$  can be updated in real-time. Now, let us assume that the closest point  $c_t$  to the surface approximated by  $\Pi_T$  has to be found for a point  $b$  falling in voxel  $ijk$ . The coordinates of  $c_t$  can be computed from the content of the vector field with Equation. (1.7) where  $\langle, \rangle$  represents the scalar product and  $\| \cdot \|$  is the norm of a vector.

$$c_t = b + F(V_{i,j,k}) + \frac{\langle F(V_{i,j,k}), V_{i,j,k} - b \rangle}{\|F(V_{i,j,k})\|} \quad (1.7)$$

As shown in Figure.1.9 (b), the closest point  $c_t$  estimated by Equation. (1.7) is a very good approximation of the true closest point on the surface  $c_b$ . In addition, for a single unit of data  $b$ , the computational complexity of finding its closest point on the surface is constant  $O(1)$  and is of order  $O(n)$  for  $n$  units of data (i.e.  $n$  points for which the closest points on the surface needs to be computed). This is more efficient than classical nearest neighbor finding approaches which show  $O(n^2)$  or  $O(n \log(n))$  computational complexity. The vector field representation thus allows the view registration and view integration steps to execute in real-time. As mentioned above, with the vector field representation, the price to pay for computational efficiency is the amount of memory that is needed to store the voxel grid at a resolution for which the planar approximation is valid.

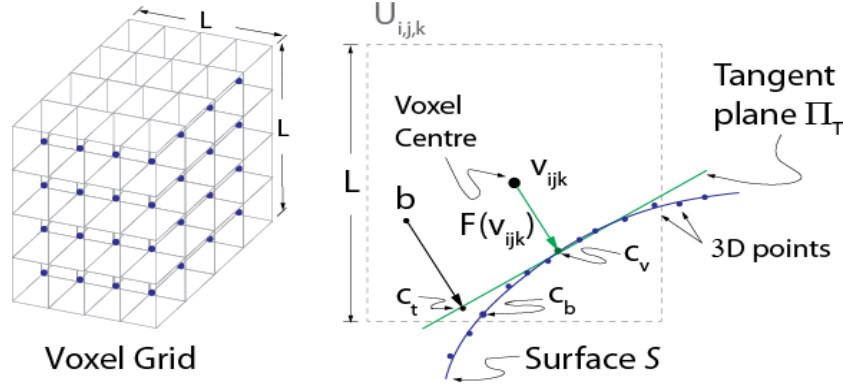


Figure 1.9 : The Vector Field implicit representation [30].

### 1.3 3D Segmentation

3D point cloud segmentation is the process of categorizing the point cloud into homogenous regions whereas classification is the step that tags these regions with the name of different surface categories depending of the application. In the following, more details on segmentation methods will be described, such as the surface based segmentation algorithm and covariance based segmentation algorithm.

Besl [31] proposed a surface-based segmentation algorithm in 1988 that simultaneously segments a large class of images into regions of arbitrary shape and approximates image data with bivariate functions. It makes possible to compute a complete, image reconstruction based on the extracted functions and regions. In the approach proposed by Besl, surface curvature sign labeling provides an initial coarse image segmentation, which can be refined by an iterative region growing method based on variable-order surface fitting. As presented by Besl [31], the surface can be segmented into a group of simple surface patches that are approximated by bivariate polynomials by using the sign of the Gaussian and mean curvatures at points on a surface. The detail of the aforementioned algorithm is given in the following.

In differential geometry, the coefficients of the first and second fundamental forms of a surface patch  $\sigma(\mathbf{u}, \mathbf{v})$  on a surface  $S$  in a differential neighborhood of a point  $P$  completely describe its intrinsic and extrinsic properties and, ultimately, its shape [14]. As shown in

Figure 1.10, given a parameterization  $(\mathbf{u}, \mathbf{v})$ , a differential surface patch  $\sigma(\mathbf{u}, \mathbf{v})$  at a point  $P$  has a surface normal  $\mathbf{N}_s$  that is orthogonal to the tangent plane  $\Pi_t$  at  $P$ . A tangent vector  $\mathbf{v}$  on  $\Pi_t$  can be expressed as a linear combination of  $\sigma_u$  and  $\sigma_v$ . Defining the linear maps  $du(v) = \lambda$  and  $dv(v) = \mu$ , we obtain Equation. (1.8).

$$\mathbf{v} = \lambda \sigma_u + \mu \sigma_v$$

Applying the inner product  $\langle \cdot \rangle$  to vector  $\mathbf{v}$  and using Equation. (1.8) yields Equation. (1.9).

$$\langle \mathbf{v}, \mathbf{v} \rangle = \lambda^2 \langle \sigma_u, \sigma_u \rangle + 2\lambda\mu \langle \sigma_u, \sigma_v \rangle + \mu^2 \langle \sigma_v, \sigma_v \rangle \quad (1.9)$$

Writing  $E = \|\sigma_u\|^2$ ,  $F = \sigma_u \cdot \sigma_v$ , and  $G = \|\sigma_v\|^2$  and using the maps  $du(v)$  and  $dv(v)$  above, the expression for  $\langle \mathbf{v}, \mathbf{v} \rangle$  writes as Equation. (1.10).

$$\langle \mathbf{v}, \mathbf{v} \rangle = E du^2 + 2F du dv + G dv^2 \quad (1.10)$$

Equation. (1.10) is referred to as the First Fundamental Form  $I$  of the surface. In an infinitesimal neighborhood of  $P$ ,  $I$  describes the measurement of a length on the surface. Although  $E, F, G$ ,  $\sigma_u$  and  $\sigma_v$  depend on the parameterization of the surface, the first fundamental form  $I$  depends only on  $S$  and  $P$ . It is an intrinsic property of the surface since it is independent of how the surface is embedded in 3D space. This can be better understood by visualizing the distance between two points on a flat sheet of paper. When the sheet is bent, the distance between the points remains the same. It is thus an invariant property of  $S$  and is not affected by rotations and translations.

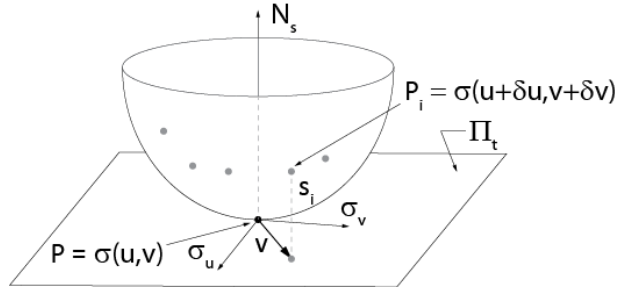


Figure 1.10 : A small patch near a point  $P$  on surface  $\sigma(u, v)$  and the tangent plane  $\Pi_t$

The Second Fundamental Form  $\mathbf{II}$  of  $S$  describes the extrinsic properties of the surface around a point and is linked to the curvature of this surface, i.e. the way the surface pulls away from the tangent plane  $\Pi_t$  at  $P$ . Considering again Figure.1.10, it can be seen that the surface at point  $P_i$  pulls away from the tangent plane at  $P$  by a distance  $S_i$  given in Equation. (1.11).

$$S_i = \left\langle \left( \sigma(u + \Delta u, v + \Delta v) - \sigma(u, v) \right), N_s \right\rangle \quad (1.11)$$

Approximating  $\left( \sigma(u + \Delta u, v + \Delta v) - \sigma(u, v) \right)$  by its Taylor expansion and neglecting the high order terms, one obtains Equation. (1.12).

$$\sigma_u \Delta u + \sigma_v \Delta v + \frac{1}{2} (\sigma_{uu} (\Delta u)^2 + 2\sigma_{uv} \Delta u \Delta v + \sigma_{vv} (\Delta v)^2) \quad (1.12)$$

The points that needs to be considered is that, at least two orders are needed to compute the curvature, because it is a second derivative. So the orders equal or greater than 3 are neglected.

Since  $\sigma_u$  and  $\sigma_v$  are tangent to the surface and are thus perpendicular to  $N_s$ , Equation. (1.12) becomes Equation. (1.13)

$$\frac{1}{2} \left( L (\Delta u)^2 + 2M \Delta u \Delta v + N (\Delta v)^2 \right) \quad (1.13)$$

with  $L = \sigma_{uu}$ ,  $M = \sigma_{uv}$ ,  $N = \sigma_{vv}$ . For small  $\Delta_u$  and  $\Delta_v$ , Equation. (1.13) can be written as Eq. (1.14) (if  $\frac{1}{2}$  is dropped)

$$\Pi = Ldu^2 + M du dv + Ndv^2 \quad (1.14)$$

Equation. (1.14) is called the Second Fundamental Form  $II$  of the surface at P.

The Shape Operator, also called the Weingarten Map,  $S$  in Equation. (1.15) can be defined at a point using the coefficients of the first and second fundamental forms [14].

$$S = (EG - F^2)^{-1} \begin{bmatrix} LG - MF & MG - NF \\ ME - LF & NE - MF \end{bmatrix} \quad (1.15)$$

The eigenvectors of  $S$  determine the directions in which the surface bends at each point and the eigenvalues  $k_1$  and  $k_2$  are the principal curvatures (i.e. the maximum and minimum normal curvatures at the point). It is possible to compute two very important invariant surface properties of a surface at a point: the Mean curvature  $H$  and the Gaussian curvature  $K$ .  $H$  and  $K$  are defined in Equation. (1.16) and (1.17) respectively.

$$H = \frac{\kappa_1 + \kappa_2}{2} \quad (1.16)$$

$$K = \kappa_1 \kappa_2 \quad (1.17)$$

Although Equation. (1.16) and (1.17) are useful, it is more convenient to use the coefficients of  $I$  and  $II$  to compute  $H$  and  $K$ . The Gaussian curvature  $K$  is given by Equation . (1.18) while the Mean curvature  $H$  is given by Equation . (1.19) [32].

$$K = \frac{LN - M^2}{EG - F^2} \quad (1.18)$$

$$H = \frac{LG - 2MF + NE}{2(EG - F^2)} \quad (1.19)$$

Looking at Equation. (1.18) and (1.19), K and H can be obtained from differentials. Using the signs of K and H, it is also possible to characterize the type of surface to which a 3D point on a surface belongs to [31]. As shown in Table 1.1, seven types of surface can be described by the combination of the signs of K and H.

The usual approach that was proposed for finding the type of surface at a given point consisted in fitting a quadratic surface model in the  $N \times N$  neighborhood of each point in a smoothed range map and then in computing the partial derivatives needed to extract K and H [31]. A range map is a 3D image defined as in Equation. (1.20) for which the surface parameterization is such that there is a depth value  $z$  corresponding to a coordinate pair  $(x,y)$  in a plane. The connectivity between 3D points in a range map is thus known compared to a point cloud for which the connectivity between points is unknown.

$$z(x, y) = f(x, y) \quad (1.20)$$

Although this fitting approach can achieve good results, computing the derivatives on the raw depth map (i.e. without fitting) is unpractical because of sensor noise. For large depth maps or, in a more general case for large point clouds, the fitting step is very time consuming. In addition, a different fit is implemented at each point even when points lie in the same neighborhood and may belong to the same surface type. However, as pointed out in [31], correcting this may require *a priori* assumptions on the surface type. Making such assumptions is very restrictive and does not allow generalization of the approach. When each point has been labelled with a given surface type, it is possible to group connected points sharing the same label and to fit a high-order polynomial (or spline model) to the region in order to obtain a high-level model.

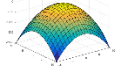
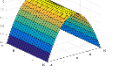
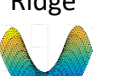
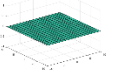
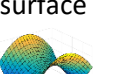
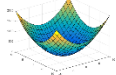
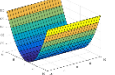
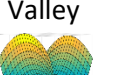
		K		
		+	0	-
H	-	Peak 	Ridge 	Saddle Ridge 
	0	none	Flat 	Minimal surface 
	+	Pit 	Valley 	Saddle Valley 

Table 1.1 : Types of surface as a function of the signs of the Gaussian and Mean Curvatures

### 1.3.1 Segmentation using covariance techniques

A different method using covariance-based differential geometry to segment range image was proposed by Berkmann and Caelli in 1994 [33]. Returning to Figure 1.10, one can compute a local covariance matrix  $C_I$  at point  $P$  of a range map as Equation. (1.21).

$$C_I = \frac{1}{N} \sum_{i=1}^N (\underline{P} - \underline{P}_m)(\underline{P} - \underline{P}_m)^t \quad (1.21)$$

$\underline{P}_m$  is defined as in Equation. (1.22) where  $\underline{P}_i$  is a point in the neighborhood of  $P$  on the range map. It is assumed that  $N$  points are selected in the neighborhood of  $P$ .

$$\underline{P}_m = \frac{1}{N} \sum_{i=1}^N \underline{P}_i \quad (1.22)$$

As described in [33], the eigenvectors of  $C_I$  are three orthogonal vectors, two of which,  $t_1$  and  $t_2$ , lie on the tangent plane to the surface at  $P$  (plane  $\Pi_t$  in Figure 1.10) and the third one, corresponding to the smallest eigenvalue of  $C_I$ , is the normal  $N_s$  to the tangent plane (and the surface) as suggested in [34]. In [33], the two-dimensional covariance matrix in Equation. (1.23) is defined. In Eq. (1.23),  $\underline{W}_i$  is a two-dimensional vector defined as in Equation. (1.24) with  $s_i$  being defined as in Equation. (1.25).

$$C_{II} = \frac{1}{N} \sum_{i=1}^N (W_i - W_m)(W_i - W_m)^t \quad (1.23)$$

$$W_i = s_i \begin{bmatrix} (P_i - P)^t t_1 \\ (P_i - P)^t t_2 \end{bmatrix} \quad (1.24)$$

$$s_i = (P_i - P)^t N_s \quad (1.25)$$

As shown in Figure 1.11, vector  $W_i$  is thus the difference between a point  $P_i$  in the neighborhood of  $P$  projected on the vectors in the tangent plane weighted by the distance  $s_i$  between  $P_i$  and the tangent plane  $\Pi_t$  at  $P$ .

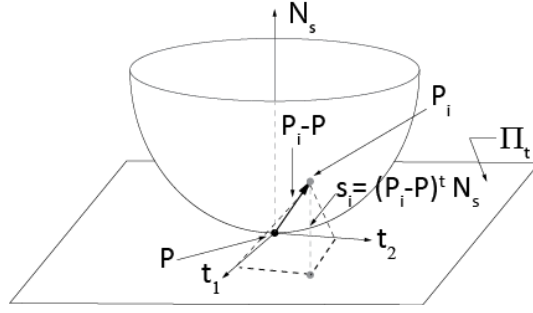


Figure 1.11 : Geometry for the vectors in Equations (1.23), (1.24) and (1.25)

Berckman *et al.* [33] define the quadratic form in Equation. (1.26) as a “covariance-based Weingarten map” for a vector  $v$  in the tangent plane.

$$H_C = \underline{v}^t C_{II} \underline{v} \quad (1.26)$$

Beckman *et al.* claim that the eigenvectors of  $C_{II}$  are the principal directions on the surface, *i.e.* the directions of minimum and maximum normal curvature. They also define a covariance-based approach analogous to the Gauss map at a point  $P$  of the range map as in Equation. (1.27) with vector  $v_i$  defined as in Equation. (1.28).



$$C_P = \frac{1}{N} \sum_{i=1}^N (v_i - v_m)(v_i - v_m)^t \quad (1.27)$$

$$v_i = \begin{bmatrix} n_i^t & t_1 \\ n_i^t & t_2 \end{bmatrix} \quad (1.28)$$

The 2 x 2 matrix  $C_P$  in Equation. (1.27) is the covariance matrix of the projections of the normal vector  $n_i$  at points  $P_i$  in the neighborhood of  $P$ ,  $v_m$  being the average vector of the projections. The eigenvectors of  $C_P$  are the principal directions. The eigenvalues of  $C_P$  provide information on the way the surface normal in the neighborhood of  $P$  project onto the tangent plane. For instance, if the surface in the neighborhood of  $P$  is a plane, the normal vectors all map into a single point, point  $P$  itself. When one eigenvalue is large and the other is small, the projection of the surface normal vectors map on a straight line and the underlying surface is a developable parabolic surface. Finally, when both eigenvalues are large, the surface is locally curved near  $P$ . *Berckman et al.* have applied the above covariance-based approach to segment points in a range map. In comparison with pure differential geometry approaches such as the one presented in [31], Berckman's covariance-based approach can only identify three different types of surface: planar, parabolic and curved.

A planar surface is identical to the “flat” surface type (with  $K = H = 0$ ) in Table 1.1. A parabolic surface covers the cases of ridge ( $K=0, H<0$ ) and valley ( $K=0, H>0$ ) in Table 1.1 while a curved surface covers the other in Table 1.1. Consequently, the segmentation obtained by covariance-based differential geometry is less rich than the one obtained with classical differential geometry for the reason that, as demonstrated in [35], even though the eigenvectors of  $C_P$  correspond to the principal directions, the eigenvalues of  $C_P$  are not equal to the principal curvatures  $k_1$  and  $k_2$  but are rather functions of their squared value as expressed in Equation. (1.29) and Equation. (1.30) (where  $r$  is the radius of the ball centered at  $P$ ).

$$\lambda_{1-C_P} = \frac{\kappa_1^2 r^4 \pi}{4} + o(r^4) \quad (1.29)$$

$$\lambda_{2-C_P} = \frac{\kappa_2^2 r^4 \pi}{4} + o(r^4) \quad (1.30)$$

Because of this, it is not possible to find the sign of  $H$  and, consequently, to differentiate between ridge or valley or peak / pit in Table 1.1.

## 1.4 Surface fitting for 3D point clouds

Surface fitting for scattered data sets with continuous representation is a problem frequently faced in the fields of geometric modeling, geometric processing, reverse engineering and computer vision [36]. The problem of surface fitting can be described as follows. Let assume a given set of data points  $p_r \in R^n$  and associated threshold  $\varepsilon_r \in R_+$ . Surface fitting consist in finding the most promising surface  $S$ , which can approximate each point  $p_r$  within threshold  $\varepsilon_r$  as described in Equation 1.31 [37].

$$\|S_{p_r} - p_r\| \leq \varepsilon_r, \quad r \in \{1, \dots, m\} \quad (1.31)$$

where  $S_{p_r}$  is a point on the fitting surface corresponding to the data point  $p_r$ . Note that this formulation is not a direct algorithm for the calculation of  $S$ . There are different methods for choosing  $S_{p_r}$  and the norm [37].

Well-known mathematical tools for continuous surface representation consist in implicit surface, subdivision surface, and parametric spline surface. There is a significant bulk of work dedicated to theoretical study or algorithm development for the surface fitting problem with different splines such as classical B-splines and Bézier basis [38], RBF (Radial Basis Function) [39], Triangular B-splines [40] and spherical volumetric simplex splines [41]. B-splines show the most attractive properties between the mentioned approaches. They are one of the industry standards for shape modeling. In B-spline surface fitting, the data points are usually estimated by using a least-squares formulation regarding to parameterization of the input data, knot vectors and control points of B- splines [36].

Since the parametric fitting methods suffer from parametrization problem and the focus of this project is on the fitting methods without requiring any parametrization, the reader is

referred to [20] for a good review of the parametric representation. In this section, two implicit fitting methods are explained: Implicit polynomial fitting and implicit B-spline fitting.

#### 1. 4.1 Implicit polynomial fitting

The goal of implicit fitting is to define a given cloud of points through the zero level set of an implicit function. An implicit polynomials (IP) provides one of the simplest solution to define curves and surfaces; it is described as [42, 43]:

$$f_c(x) = \sum_{\substack{(i+j+k) \leq d \\ \{i,j,k\} \geq 0}} c_{i,j,k} x^i y^j z^k \quad (1.32)$$

Where  $c_{i,j,k}$  are the coefficients of the implicit polynomial (IP),  $d$  is the degree of the IP and  $x^i y^j z^k$  are the monomials. The coefficients  $c_{i,j,k}$  must be solved through the values of  $f$  in the given data set close to zero [44].

The linear system of IP coefficients may lead to the lack of the geometric meaning and the instability problem in the classical algebraic methods. *Blane et.al.* [43, 45] proposed the 3L algorithm to solve the aforementioned shortcomings.

The 3L algorithm is an algebraic method for Implicit Polynomial fitting [43]. The processing steps of this algorithm are as follows. First constructing two additional level sets from inside and outside of the boundary. Second, the problem is modelled as the optimal IP, which comes close zero in the original set and leads to a sign transition from inside to outside of a region of space delimited by the surface .

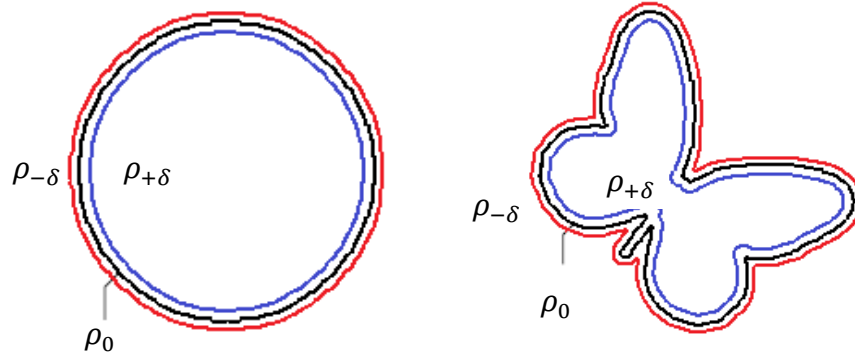


Figure 1.12 : Inner offset (blue), outer offset (red) and original data (black) for 2D objects [45].

The 3L algorithm generates an inner offset  $\rho_{+\delta}$  and an outer offset  $\rho_{-\delta}$  at the distance  $\pm\delta$  from the original data set  $\rho_0$  [45]. An illustration is given in Figure 1.12. The inner and outer offsets can be achieved through the triangulation or the PCA applied in each local neighborhood of a fixed point [46]. By computing three level sets  $\{\rho_{-\delta}, \rho_0, \rho_{+\delta}\}$ , an implicit polynomial can be found by approaching  $+\epsilon$  inside,  $-\epsilon$  outside and zero in the original data set. The 3L fitting algorithm can be formulated as an over determined system  $M_{3L}c = b$  where

$$M_{3L} = \begin{bmatrix} M_{\rho_{-\delta}} \\ M_{\rho_0} \\ M_{\rho_{+\delta}} \end{bmatrix}, b = \begin{bmatrix} +\epsilon \\ 0 \\ -\epsilon \end{bmatrix} \quad (1.33)$$

and where  $M_{\rho_{-\delta}}, M_{\rho_0}, M_{\rho_{+\delta}}$  are matrices of monomials calculated in the outer, original and inner set respectively. The  $\pm\epsilon$  are the corresponding expected measures in the inner and outer offsets. Solving Equation (1.33) is similar to minimizing Equation (1.34):

$$E(c) = \|M_{3L}c - b\|^2 = c^T M_{3L}^T M_{3L} c - 2c^T M_{3L}^T b + b^T b \quad (1.34)$$

The polynomial can be computed by finding the least-squares solution to this linear system of equations.

The 3L algorithm is a fast method for fitting a set of points through implicit curves or surfaces, but it is not a flexible representation for IP space because by increasing the IP degree and for complex objects, some artifacts and outliers can be created around the object [46].

### Implicit B-spline fitting

An implicit B-Spline is described as a zero set of B-Splines tensor products :

$$f(x) = \sum_{i,j,k=1}^N c_{i,j,k} B_i(x) B_j(y) B_k(z) \quad (1.35)$$

where  $c_{i,j,k}$  are the lattice of size  $N \times N \times N$  of control coefficients, and  $B_i(x)B_j(y)B_k(z)$  are the tensor product of spline basis functions.

To compensate the lack of flexibility of the 3L algorithm for IP space, Rouhani et.al. [42] proposed an approach which is extended to implicit B-spline curves and surfaces. Since the implicit B-Spline fitting is used in our project, more details for the extended 3L algorithm proposed by [46] are given in the next chapter.

## 1.5 Blending different curves and surfaces

A blending operation smoothly merges two or more primary surfaces and prevents intersections at the edges where the two surfaces met [47]. Blending two functions is possible by applying a method similar to the linear interpolation of two points. Consider  $f(x)$  and  $g(x)$  as two different functions, a blending could be achieved by using a third function  $a(x)$  with range  $[0,1]$  as in Equation 1.36 [48]:

$$h(x) = f(x)a(x) + g(x)(1 - a(x)) \quad (1.36)$$

$h(x)$  could be anywhere between  $f(x)$  and  $g(x)$  unless either  $a(x) = 1$  such that  $h(x) = f(x)$  or  $a(x) = 0$  then  $h(x) = g(x)$ . An illustration is given in Figure 1.13

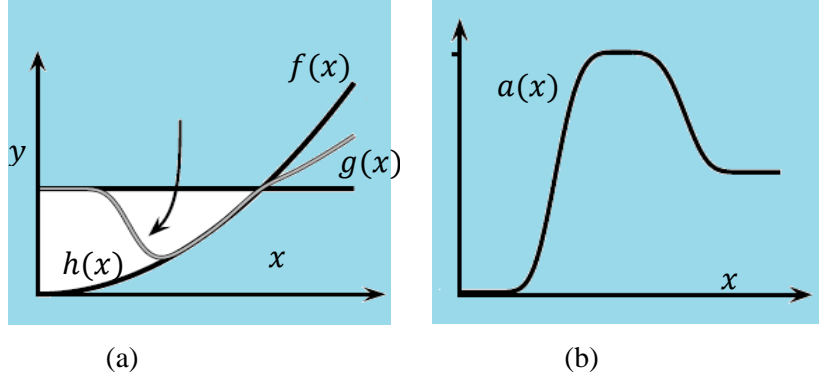


Figure 1.13 : (a) Blending two functions  $f(x)$  ,  $g(x)$ . (b) Resulting blended function  $a(x)$ . Figure taken from [48].

### 1.5.1 Merging B-Spline curves or surfaces

Merging separate patches of B-Spline curves or surface is popular in geometric modeling. It is one of the main procedures to combine two or more given geometric models. Different approaches were developed to merge B-spline curves or surfaces. The method presented in [49] is a merging procedure of B-spline curves that is formulated as a linear optimization problem. This method is representing the spline curves or surfaces by explicit parametric vector functions in matrix form.

The matrix representation achieves the efficiency and stability of the integral computation while minimizing the objective function. In the aforementioned approach, the B-spline curve is updated each time by joining the new B-spline segment to the previous resulting curve, where the number of unknown control points is six at each step. In such a method, the number of linear equations to be solved is independent of the number of curves to be merged. Moreover, in this iterative procedure, by setting appropriate weighting of the error function, the shape of the final merged curve can be modified piecewise. For more detail, the interested reader is referred to [50], [51] and [52]. An illustration of merging B-spline curves is given in Figure 1.14.

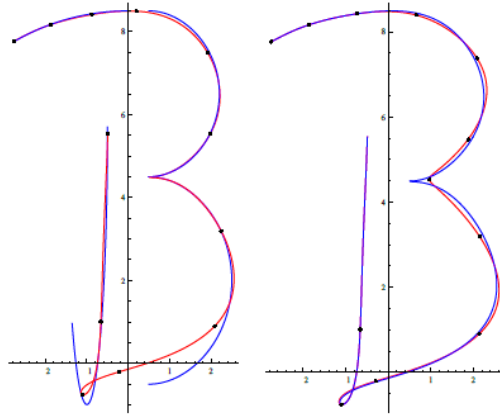


Figure 1.14 : Merging two B-spline curves with  $C^0$  continuity (i.e. curves are connected at a joint). Figure taken from [49].

### 1.5.2 Stitching Triangles

Another merging surface approach consist of the stitching of the triangles of the meshes such as the methods proposed in [53] and [54]. The method proposed by Cohen-Steiner and Da [53] uses a greedy approach to select the triangles. In this approach, the Delaunay triangulation of the input points is produced first. Then candidate triangles are selected according to some topological constraints. The output surface is reconstructed incrementally by choosing the triangles from the prioritized list of candidate triangles and stitching them together [20].

### 1.5.3 Zippered Polygon Meshes

Turk and Levoy [54] proposed an approach to combine a group of range images into a single polygonal mesh that completely represents an object to the extent that it is visible from the outside. This method combines different views of an object, each view being sampled by a range scanner, and blending these views into one unbroken continuous polygonal surface. Some applications for such an algorithm consist of [54]: “Digitizing complex objects for animation and visual simulation”. “Digitizing the shape of a found object such as an archaeological artifact for measurement and for dissemination to the scientific community”. “Digitizing human external anatomy for surgical planning remote consultation or the

compilation of anatomical atlases”. “Digitizing the shape of a damaged machine part to help create a replacement.”

The processing steps of this method are as follows:

- **Align the meshes with each other using a modified iterated closest-point algorithm**

Turk and Levoy [54] defined their own variant of the ICP algorithm. As shown in Figure 1.15, first nearest position on mesh H to each vertex of mesh K is found. Then pairs of points far apart or located on the mesh boundary are removed. The next step finds the rigid transformation that minimizes a weighted least-squared distance between the pairs of points. The process continues until convergence and the ICP is applied on a more detailed mesh in the hierarchy.

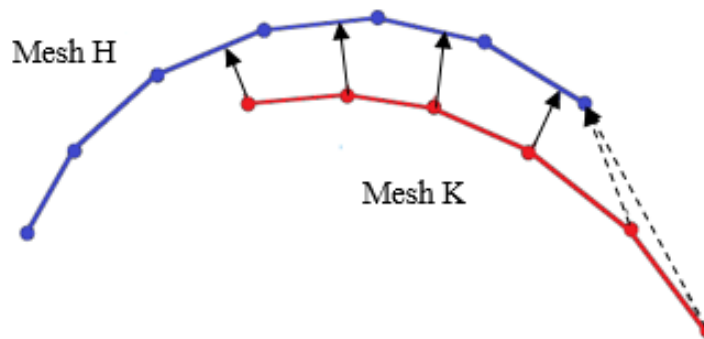


Figure 1.15 : Searching for corresponding points for mesh registration. Dotted arrows show the matches that should be prevented since they cause the mesh K to be dragged up and left by mistake. Figure borrowed from [54].

- **Zip together adjacent meshes to form a continuous surface that correctly captures the topology of the object**

The main level in combining range images is the integration of multiple views into a single model view. Integration aims to achieve the overall topology of the object being scanned. The complete topology of a surface is achieved by zippering new range scans one by one into the final triangle mesh. There are three steps to zip the two triangle meshes such as to: (1) Remove overlapping portions of the meshes. (2) Clip one mesh against another, and (3) Remove the small triangles introduced during clipping.



In addition, the last step of the proposed method in [54] is:

**-Compute local weighted averages of surface positions on all meshes to form a consensus surface geometry.**

The method proposed by Turk and Levoy [54] is used in the last step of our project to merge the different patches of the B-Splines. More details are given in the Chapter 2, proposed approach and methodology.

In the current chapter, the work related to the project is explained. The objective of this project is to achieve a model of a surface in an efficient way. In this chapter, explicit and implicit surface representation, segmentation as well as higher level modelling approach are described. In the next chapter, the proposed methodology to achieve the research objectives are explained in detail.

## Chapter 2

### Proposed Approach and Methodology

The recent availability of powerful 3D scanning devices has made it possible to capture very large sets of points at the surface of objects of various sizes such as man-made objects up to car-sized objects. Transforming an unstructured set of 3D points captured by these devices into a meaningful digital representation of the scanned object is called 3D surface reconstruction. The main objective of the current research is to build a 3D representation approach able to provide an accurate mesh everywhere on the surface with the less possible memory consumption. The methodology that is proposed to build the 3D representation model is presented in this chapter.

#### 2.1 Data Acquisition and Sensor Position

Among 3D sensors, handheld 3D sensors are of great interest because they allow the capture of 3D data on the specimens to be inspected in a very natural way, which, in many aspects, resembles spray painting. As shown in Figure 2.1, scanning an object with a handheld 3D sensor consists in moving the sensor around the surface of the object of interest while the 3D coordinates of points at the surface are collected.

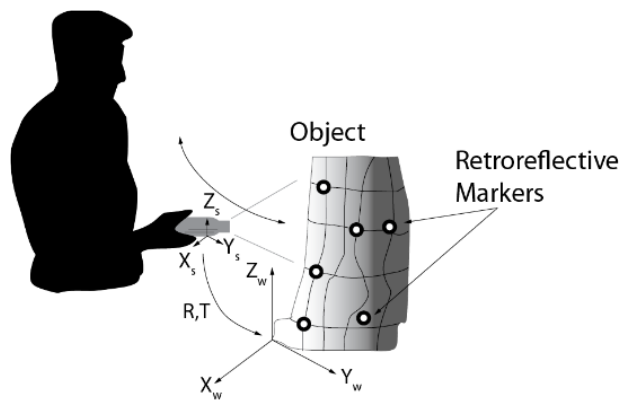


Figure 2.1 : The 3D scanning process using a handheld scanner [30]

Most 3D handheld sensors are active sensors that project some sort of light pattern (laser point, laser stripes, laser crosses, white light patterns, Moiré fringes, etc., see [55] for an overview of different 3D sensing technologies) on the object to ease the image analysis process leading to the measurement of 3D coordinates. The 3D coordinates of points acquired from a pose (*i.e.* position and orientation) of the sensor are expressed in this local reference frame. The estimation of the rigid transformation (rotation and translation) between each pose of the sensor and a “global” reference frame is needed if the 3D points are to be expressed in a common reference frame. This global reference frame can be the initial pose of the sensor when the scanning process starts. The estimation of the rigid transformations is made easier if a real-time self-positioning strategy is used to compute the position and orientation of the sensor with respect to the object. One way of implementing self-positioning is to install markers (often retroreflective markers (shown in Figure 2.1)) at the surface of the object and to estimate the pose of the sensor with respect to these markers.

Observing the whole surface of the object from a single point of view regardless the type of 3D sensor is impossible. It is thus necessary to move either the object or the sensor (or in some cases both) in order to capture the point from entire surface of an object, see Figure 2.2 [5].

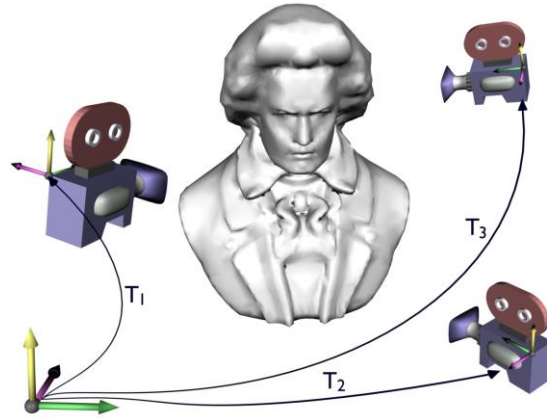


Figure 2.2 : Multi view acquisition of range data [24]

When the sensor is moving around the surface of the object thus, it is necessary to compute the sensor position to cover the whole surface and reconstruct a complete model of the desired object. In the proposed approach, the sensor position is computed as presented in Algorithm 2.1.

---

**Algorithm 2.1: Obtain sensor position in the vector field framework**

---

```

Input : 3D point clouds obtained by the scanner
Initialize the sensor position to zero
for i = 1 : Number of points do
    Sensor position = ((i-1)/i) * Sensor position + (1/i) * Input(:,i)
end

```

---

An illustration of obtain sensor position in the vector field framework using the Algorithm 2.1 is given in Figure 2.3. In which the surface shown in green color is the original data, the dashed lines shown in blue color indicate the sensor position's path. Points P1 to P8 are the points for which the sensor positions, S1 to S8, are computed.

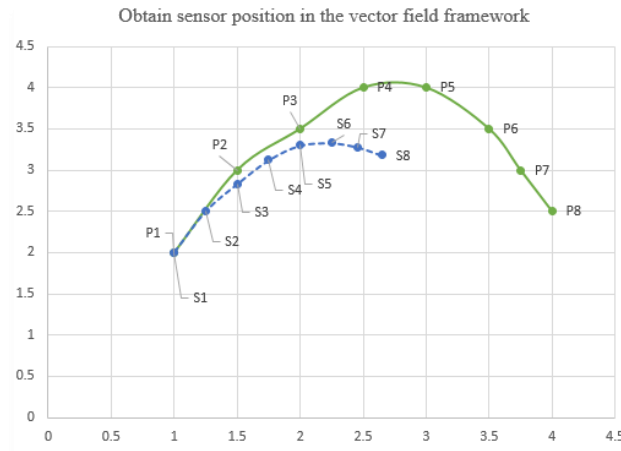


Figure 2.3: Obtain sensor positions in the vector field framework: Surface in green color represent the original data. P1 to P8 indicate the original input points. Dashed lines shown in Blue color present the sensor position's path. S1 to S8 shown in blue color indicate the sensor positions associated to each original points with the same indices number.

The information provided by Algorithm 2.1 is used to obtain the angle between the normal vector to the surface and the sensor position. This gives the information that is needed to distinguish surface shape (i.e. pit or peak).

Once the cloud of unorganized 3D points covering the entire surface of the object has been captured, a model of the surface must be built if metrologic measurements are to be performed on the object. This model can also be used to analyze the geometry of the object. Two main representations can be exploited to build this model: *explicit* representations and *implicit* representations. The explicit representation makes the geometry supported by the point cloud explicit, for instance by building a triangular mesh [56]. Such a mesh contains the connectivity between points in the cloud and is a compact representation of the geometry of the surface. This connectivity can take the form of a vertex-triangle list and a triangle-vertex list and can also include the information on the normal to the surface at each vertex.

The implicit representation rather encodes the geometric information contained in the point cloud implicitly into a volumetric structure composed of voxels. Two main types of voxel-based implicit representations have been proposed: the distance field [57] and the vector field [5]. The volumetric structure must be processed *a posteriori* to produce a mesh representing the geometry explicitly. The Marching Cubes algorithm is often used for this task [58]. More explanation on explicit and implicit representation is given in chapter 1 , section 1.2.

When handheld 3D sensors are used for *real-time* modelling (i.e. the model of the surface is built as the 3D points are measured), three tasks must be achieved: *i*) view registration, *ii*) view integration and *iii*) model visualization. View registration consists in the estimation of the rigid transformation between points of view from which the 3D data is collected. View integration aims at merging redundant 3D data common to two or more views. Finally, model visualization is the task of rendering the 3D model as it is being built so the user can observe the progression of the scan and plan the scanning strategy as points are being collected.

The advantage offered by an *explicit* representation is that a low-level model is readily available. However, it is not adapted to real-time modelling. The main reason for this is that the registration and integration steps rely on finding nearest neighbours and that the search for nearest neighbours to a point becomes too computationally expensive when the number

of points increases. Updating a mesh as new 3D points are being collected is also impossible to achieve in real time. The advantage of using *implicit* representations is that some, such as the vector field, have demonstrated the ability to support the three modelling steps in real-time. As described in chapter 1, a major advantage of the vector field representation is that it encodes the surface normal as well as information on the nearest neighbors in each voxel, thus enabling nearest neighbor search in linear time complexity.

However, the main problem with the current implementation of the vector field in the context of metrology using handheld 3D sensors is approximating a surface by a plane. For describing regions with small details, the planar approximation is not always sufficient in a metrology context. The first solution consist of decreasing the voxel size in the vector field. By doing so, two issues will arise. First, decreasing the voxel size causes an increase in the number of voxels, which increases memory requirement significantly. For instance, in a metrology context, decreasing the voxel size by an order of magnitude leads to an increase in the number of voxels by 1000. Therefore, the problem of memory requirement will also appear since it is unlikely that the vector field will fit in computer memory.

The second issue is that, by decreasing the voxel size, each voxel may not receive enough points. During 3D data collection, if the accumulated data in a voxel is not large enough (not enough points fall inside a given voxel), the covariance matrix may not be robust and reliable. For example, for current handheld sensors and with “regular” voxel sizes, a voxel can usually receive between 20 and 100 points. If the voxel size is decreased by an order of magnitude, several voxels on a surface being scanned would not receive any point ever though they are on the surface.

The first step of the proposed method to build the best possible surface representation is to identify the different surface types using the covariance based differential geometry in each voxel in the vector field framework.

## 2.2 Identification of the Different Surface Types in the Vector Field Framework Using Covariance Based Differential Geometry

As mentioned in section 1.3.2 the sign ambiguity of the eigenvalues obtained by  $CII$  in Equation 1.23 prevents us from distinguishing some surface types. We propose a new technique that combines covariance-based differential geometry and the vector field implicit surface representation to identify the different surface types in each voxel of the vector field framework. Instead of working with each point and the neighborhood around that point, we rather propose to work with the voxels in the volumetric grid containing the vector field and its 26 possible neighbors in the grid.

As mentioned above, handheld scanners for metrologic applications use retroreflective markers or natural features to estimate the pose of the sensor with respect to a reference frame chosen as the “world” reference frame. Secondly, using the vector field implicit surface representation, view registration, view integration and the estimation of the normal to the surface in each voxel of the field can be performed in real time as the 3D data is collected by the handheld sensor. The vector field is also built in the world reference frame. Since the pose of the sensor in the world reference frame is estimated in real-time, it is also possible to know on which side of the surface the sensor is when 3D data is collected and integrated in a voxel of the vector field.

The first step in computing the vector field consists in obtaining the normal at all points of a surface. All normal should be consistently oriented such that the scalar product between the direction of the optical axis of the sensor and the surface normal should be positive. When it is not the case, the orientation of the normal should be flipped.

Knowing on which side of the surface the sensor is located when the data is collected as well as the surface normal in the voxel allows the orientation of this normal to be defined with respect to the direction of the optical axis of the sensor.

This also allows the differentiation between peak/pit or ridge/valley and eliminates the limitations of covariance-based differential geometry (expressed in Equation 1.29 and Equation 1.30) for identifying the surface types. Based on the above, the strategy of the proposed method is to apply covariance-based geometry on the vector field implicit surface

representation instead of on individual points, thus reducing the computational load considerably since hundreds of points if not thousands fall in a single voxel.

In the context of the differential geometry, when the orientation of the normal vector is given, then it is possible to observe that if

- a) All points in a certain neighbourhood around the point of interest lie on only one side of the tangent plane (above or below) then it is an elliptic surface (Shown in Figure 2.4 (a)).
- b) The points in a certain neighborhood of the point of interest are distributed on both sides of the tangent plane then it is a hyperbolic surface (Shown in Figure 2.4 (b)).
- c) If the neighbourhood around the point of interest has a line in common with the tangent plane then it is a parabolic surface (Shown in Figure 2.4 (c))
- d) If the neighbourhood around the point of interest lie on the tangent plane then it is a planar surface.

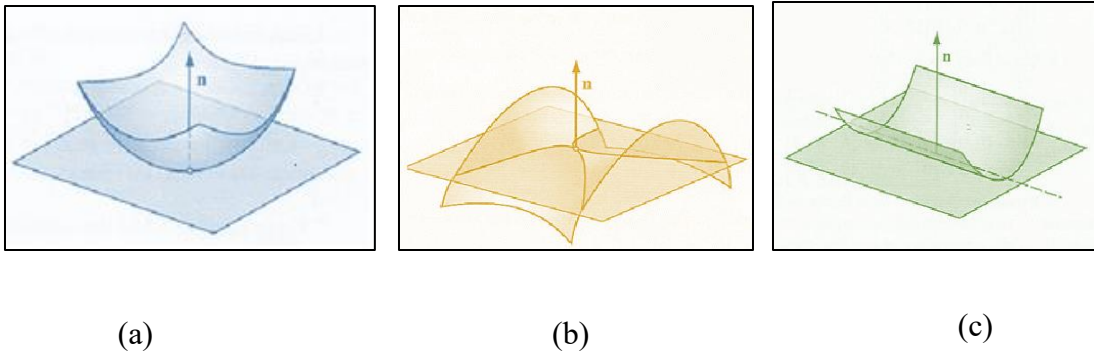


Figure 2.4 : Position of the different surface type to the tangent plane: (a) Elliptic surface (b) Hyperbolic surface (C) Parabolic surface. Figure taken from [59]

We extend this concept into the vector field framework. Therefore, instead of using points we rather use a voxel and the neighbours around that voxel to find the type of surface. With the additional knowledge of the direction of the surface normal (available in each voxel of



the field), it is possible to exploit Equation 1.25 on the neighborhood of a voxel to identify to which type of surface the points in this voxel belong to. Working on voxels instead of points significantly reduces the computation load associated with this tool.

## Implementation

Since the covariance matrix of the points falling inside a voxel is computed in real time, computation of the normal vector is also achieved in real time and is the eigenvector of the covariance matrix corresponding to the smallest eigenvalue. By implementing the vector field framework, the normal vector, the closest point to the surface (Equation 1.7) the sensor position and the voxel center are all stored in a voxel of the 3D grid. Therefore, all of the values needed to compute the orthogonal distance from the tangent plane of a particular voxel to the neighboring voxels are available. For each voxel in the 3D volumetric grid, there are 26 possible neighbors. We have to consider the neighboring voxels which contain much more than 3 points inside in order to have a reliable covariance matrix in each voxel. This is not a problem since modern scanners can capture 250,000 points per second and can thus provide a density of points that is sufficient for our purpose, assuming that the scanner is not moved too fast. With current handheld sensors with typical voxel size, a voxel can usually receive between 20 and 100 points.

We define the orthogonal distance form as Equation 2.1

$$Distance_j = (c_t(j) - c_t(0))^T \cdot n \quad (2.1)$$

where  $j = 1$  to 26 is the number of the neighbouring voxels around the voxel  $v_0$ .  $c_t(j)$  is the point on the plane approximating the surface in the  $j$ th neighboring voxel (point  $c_v$  in Figure 1.9),  $c_t(0)$  is the point on the plane approximating the surface in the voxel of interest,  $n$  is the normal vector obtained from the covariance matrix in the voxel of interest of the vector field.  $Distance_j$ , is a  $j \times 1$  vector stored in each voxel of the vector field framework. An illustration in 2D to simplify visualization is given in Figure 2.5.

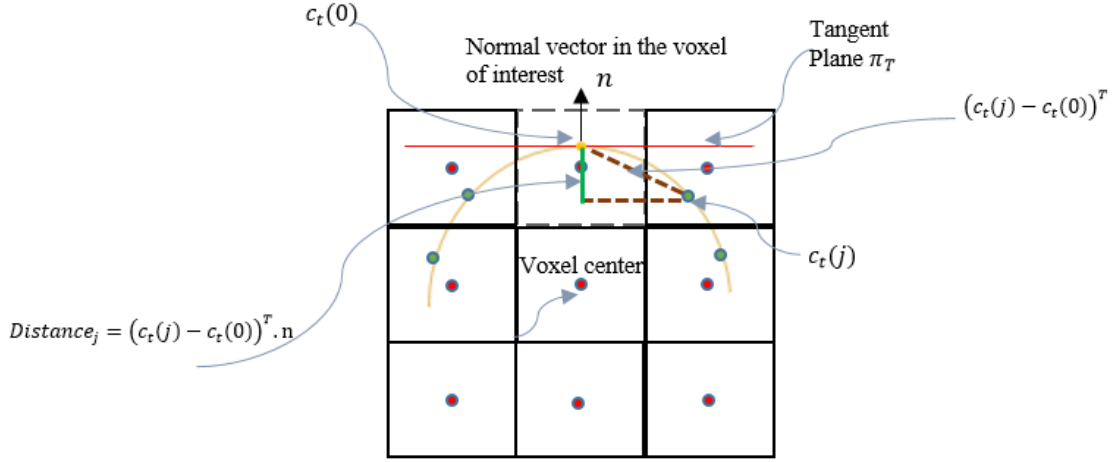


Figure 2.5: An illustration of ***Distance<sub>j</sub>*** stored in the voxel of interest whose sides are in dotted lines.

The information of the matrix *Distance<sub>j</sub>* allows us to recognize the surface type in a given voxel as follow:

- If all of the values of the *Distance<sub>j</sub>* are positive, then the surface in the particular voxel is peak surface.
- If all the values of the *Distance<sub>j</sub>* are negative, then the surface in the particular voxel is pit surface.
- If some of the values of the *Distance<sub>j</sub>* are negative while others are positive, then the surface in the particular voxel is a saddle surface.
- If some of the values of the *Distance<sub>j</sub>* are negative while others are positive, and the eigenvalues obtained by Equation 1.29 and 1.30 are equal then the surface in the particular voxel is a minimal surface. Which, in this paper are considered as belonging to the same category as the saddle surfaces.
- If the values of the *Distance<sub>j</sub>* are both positive and zero, then the surface in the particular voxel is ridge surface.

- If the values of the  $Distance_j$  are both negative and zero, then the surface in the particular voxel is valley surface.
- If all of the values of the  $Distance_j$  are zero within a given threshold, then the surface in the particular voxel is planar.

To demonstrate the performance of the method proposed for identifying the different surface types in the vector field framework, it has been applied to 3D synthetic data as well as 3D data which was obtained from a real metrologic scanner and from Stanford repository. The color map for different surface types is shown in Table 2.1.

Surface Type	Colour list
Pit surface	Yellow
Ridge surface	Green
Valley surface	Black
Saddle and Minimal surface	Cyan
Peak surface	Blue
Plane surface	Red

Table 2.1 : Color map corresponding to different surface types

The surface types of voxels belonging to the synthetic spherical surfaces are shown in Figure 2.6. The color in the voxels is coherent with the surface types in each voxel according to Table 2.1.

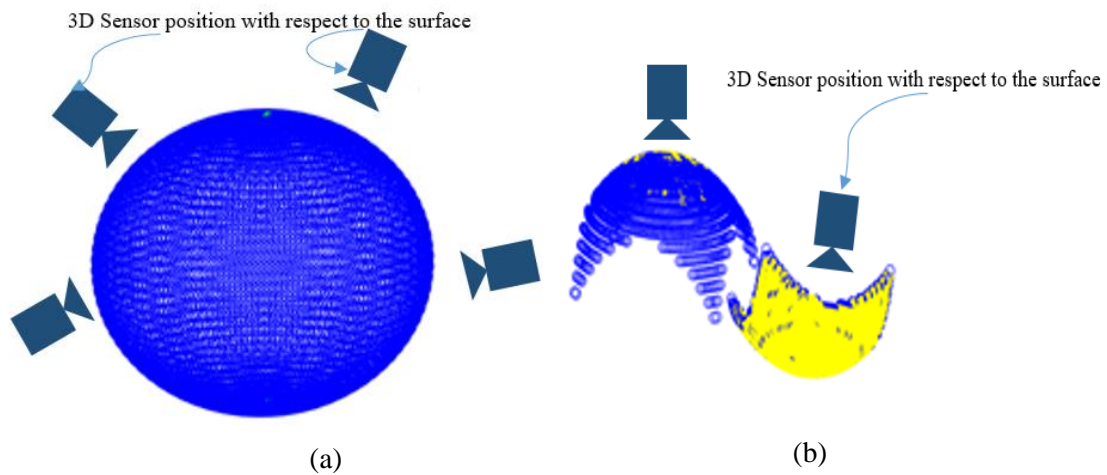


Figure 2.6: Identifying the surface type at each voxel on a synthetic sphere. (a) Shows the peak in blue color. (b) Shows the peak in blue color and pit in yellow color.

The results on the synthetic cylindrical surfaces, which are valley (black) and ridge (green) surfaces, are shown in Figure 2.7.

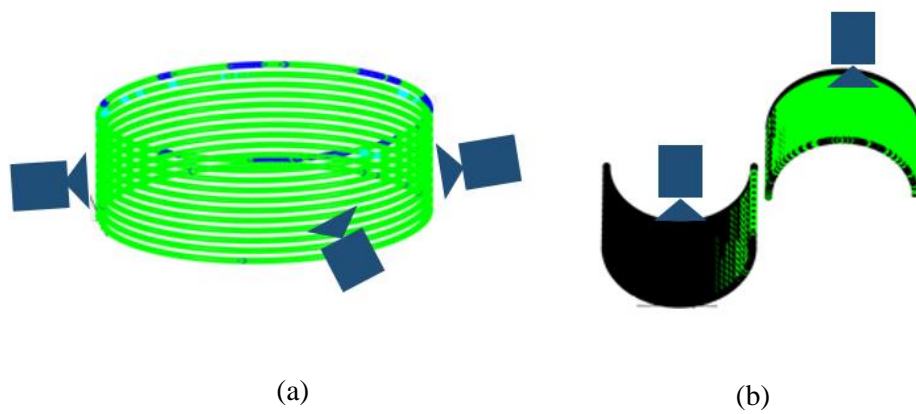


Figure 2.7: Identifying the surface types on synthetic cylinders. (a) Shows the ridge surface in green (b) Show the ridge surface in green and valley surface in black color

By referring to Figure 2.8 (a), it can be observed that the upper side of the bunny's belly is peak surface and under the belly is more ridge and the regions around the claws of the bunny are mostly pit and valley. Clearly the regions between the belly and claws are saddle, and Figure 2.8 (b) provides a qualitative validation of our approach on the 3D points of the bunny

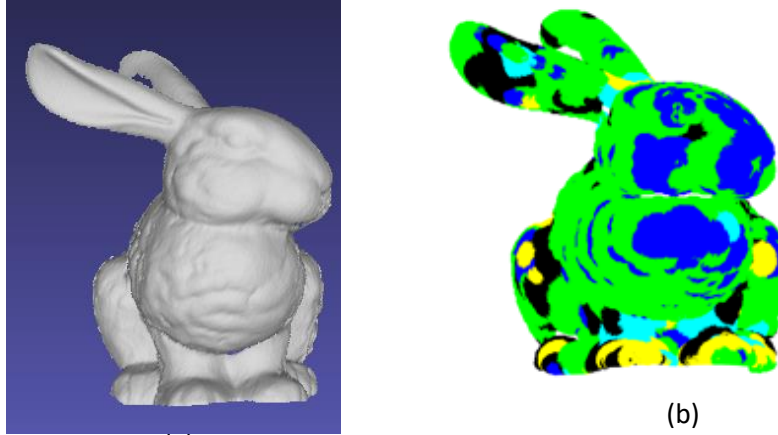


Figure 2.8: Identifying the surface types on 3D points provided by Stanford repository. (a) Object mesh data. (b) Shows the different surface types

More results for the synthetic data and the scanned objects are given in the next chapter on the Experimental Results.

In currently available commercial metrology software tools, the voxel size is selected arbitrarily irrelevant of the number of points falling inside each voxel for the verification of the planarity of the surface hypothesis. In the next section, the surface variation is given as an objective criterion to select an optimal voxel size systematically.

### 2.3 Surface Variation Method in the Vector Field Framework

In a metrology workflow using handheld 3D scanners based on an implicit surface representation, the first step consist in selecting the voxels size (*i.e.* the resolution of the model) and then start to collect the 3D data. For selecting the voxel size, two points need to be considered: the memory consumption and the accuracy of the reconstructed model that we plan to achieve. A fact that needs to be considered is that, in a metrology framework, the

size of the voxels of the vector field is usually chosen arbitrarily by the user. The process is that a user chooses the voxel size, scans a given object and then observes the model.

Choosing the voxel size arbitrarily is not optimal. By choosing a large voxel size, we may not get a good model. In contrast, choosing a small voxel size will require the allocation of a large number of voxels, which increases memory consumption. For instance, in a metrology context, decreasing the voxel size by an order of magnitude leads to an increase in the number of voxels by 1000. Therefore the problem of memory requirement will appear, since it is unlikely that the vector field fits in the computer memory. This is why we propose an approach that allows to select the voxel size systematically that is sufficient to achieve a good construction of the final model of the surface while keeping the number of voxels as small as possible. The approach exploiting covariance-based differential geometry proposed above can then be applied for building the model using high level surface reconstruction as presented in the next section.

The surface variation is defined as the ratio of the smallest eigenvalue over the sum of all eigenvalues of the covariance matrix of a set of 3D points. Pauly *et al.* [60] define the surface variation at point  $p$  in a neighbourhood of size  $n$  as:

$$\sigma_n(p) = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \quad (2.2)$$

where  $\lambda_1, \lambda_2, \lambda_3$  are the three eigenvalues of the covariance matrix of a set of 3D points such that  $\lambda_1 < \lambda_2 < \lambda_3$ .

An interesting property of  $\sigma_n(p)$  is that it is bounded between 0 and 1/3. All the points in a set are located on a plane if  $\sigma_n(p) = 0$ . When  $\sigma_n(p) = 1/3$  all the points in the neighbourhood are distributed evenly in space.

Note that  $\lambda_1$  defines the variation along the surface normal and  $\lambda_2, \lambda_3$  describe the variation of the sampling distribution in the tangent plane. These two latter eigenvalues can be used to estimate local point distribution anisotropy accordingly. If the surface in a given

neighborhood is highly curved, its surface variation should be high and if the neighborhood is flat the surface variation should be small [60].

These are two main reasons why the surface variation is of interest in our work: the fact that it is bounded between 0 and 1/3 and that it is simple to compute it in the vector field framework since the covariance matrix is already available in each voxel.

We thus propose to exploit surface variation to achieve the optimal selection of the voxel size in the vector field framework. In each voxel of the vector field, the surface variation, as defined in Equation 2.2, is computed and stored in the voxel. As mentioned in section 1.2.4, the eigenvalues and the covariance matrix are already available in each voxel of the vector field. Therefore, this implies that the surface variation is very easy to compute. This value  $\sigma_n(p)$  can be computed in real time without the need for surface fitting and other complex computations. It is important to mention that the vector field itself is computed and updated in real-time. See Tubic et al [5].

First based on the above reasoning, a reasonably large voxel size for the vector field is selected according to the size of the object and the size of the available computer memory, and the surface variation  $\sigma_n(p)$  is computed. Then the interval of possible values for  $\sigma_n(p)$ , which is  $[0, 1/3]$  is divided in 100 bins and the histogram of the number of voxels corresponding to each bin is computed on the vector field. In our work the interval values in  $[0, 0.0033]$  are considered as the planar area.

These aforementioned steps are repeated with the difference that each time the voxel size is decreased one unit. This method is repeated until it is observed that the number of the voxels falling in the planar area (first bin of each histogram) are larger than others at its right. In this stage, we can conclude that the size of the voxel is small enough to get an accurate model of the surface and the corresponding voxel size is selected as the optimal one. It is clear that choosing the voxel size smaller than the aforementioned size, the number of voxels corresponding to the first bin (Planar region) will increase with no significant improvement in model quality. Note that in the proposed method, the size of the side of a voxel is measured in mm.

## Implementation

To visualize the performance of the method proposed for selecting the optimal voxel size in the vector field, it has been applied to different 3D objects. In this section, the results related to the bunny provided by the Stanford repository are shown in Figure 2.9 (a) to (g). The results shown in Figure 2.9 are related to the surface variation computed for voxel sizes between 10 mm to 3 mm.

To show the results of the value of the surface variation we have considered 100 bins to represent  $\sigma_n(p)$  between  $0 < \sigma_n < 1/3$ . Each colour corresponds to a bin.

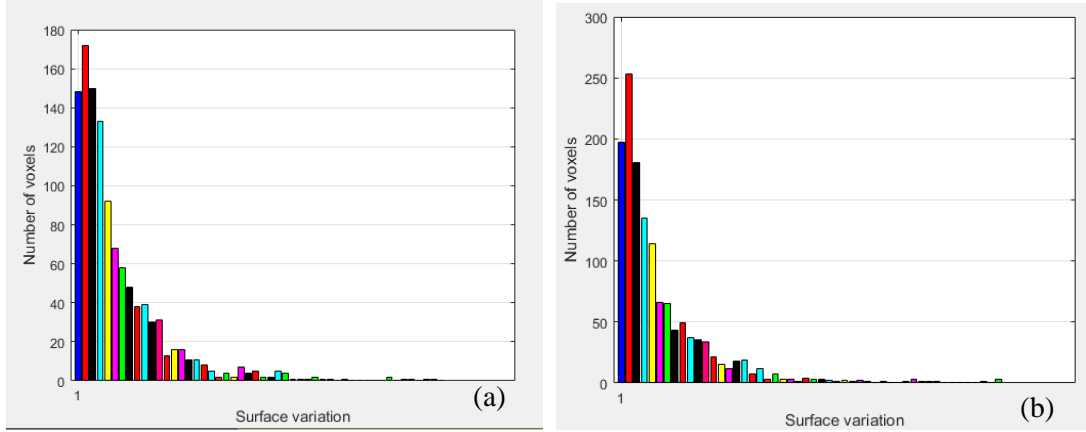


Figure 2.9: Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): (a) Voxel size = 10 mm (b) Voxel size = 9 mm

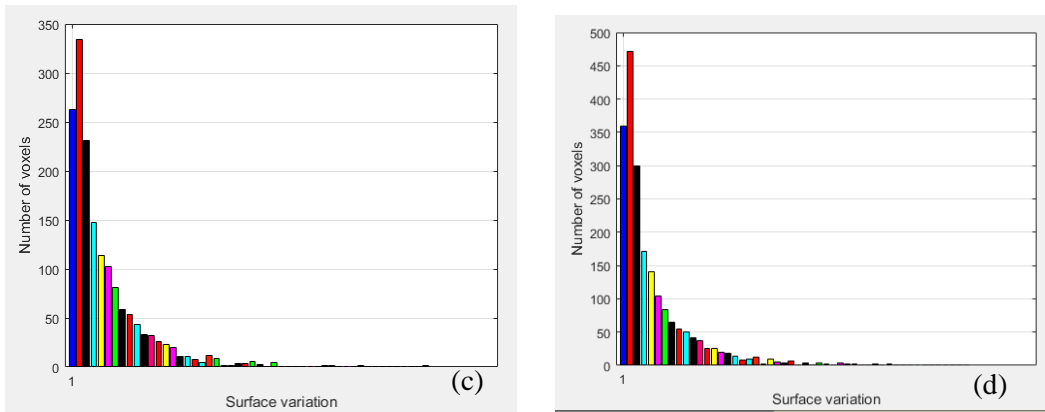


Figure 2.9. Number of voxels corresponding to the surface variation: (c) Voxel size = 8 mm (d) Voxel size = 7 mm



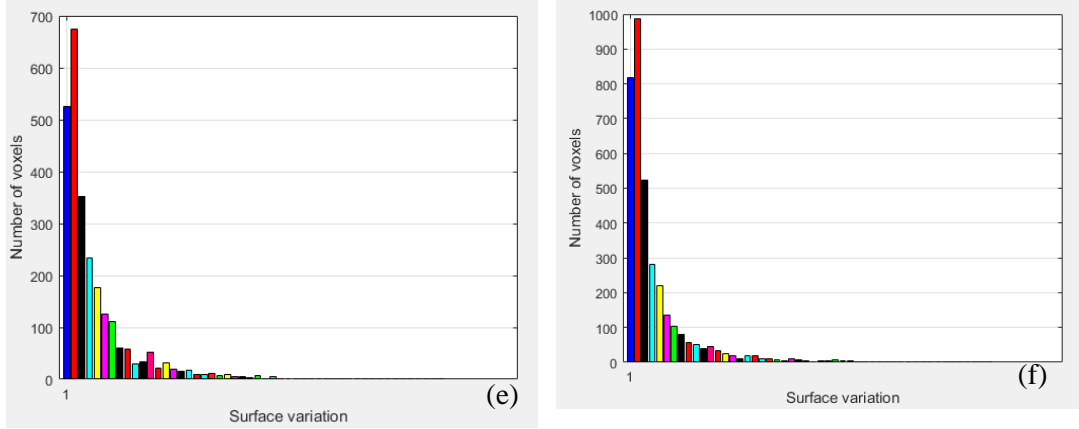


Figure 2.9. Number of voxels corresponding to the surface variation: (e) Voxel size = 6 mm (f) Voxel size = 5 mm

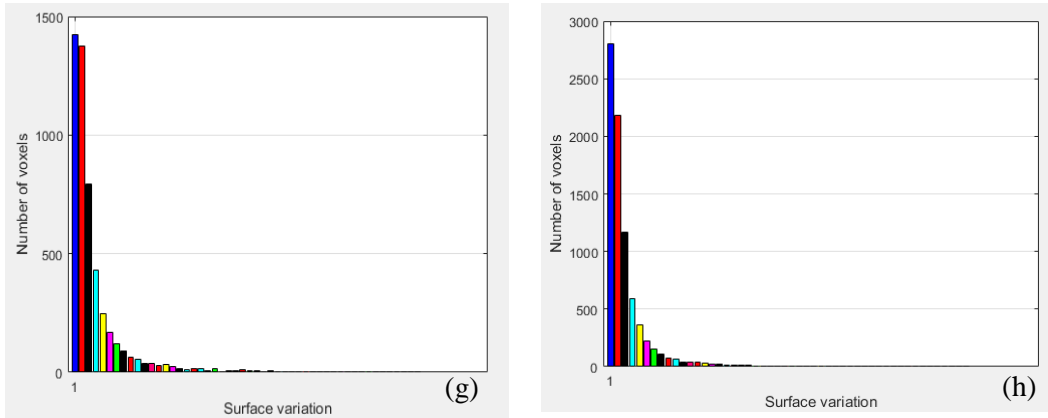


Figure 2.9. Number of voxels corresponding to the surface variation: (g) Voxel size = 4 mm (h) Voxel size = 3 mm

As shown in Figure 2.9 (g) the number of the voxels corresponding to the planar area is larger than the second bin. Therefore, it is decided that, 4 mm is the optimal voxel size for the 3D model of the Bunny model. As shown in Figure 2.9 (h), if the voxel size still decreases to 3 mm, it is obvious that the number of voxels related to the planar areas is still greater than the second bin and that there is no need to decrease the voxel size anymore. In addition of voxel size of 3 mm increases the number of voxels which causes a large memory consumption. In addition, decreasing the voxel size may result in voxels containing less points, then decreasing the robustness of the covariance matrix.

In the next sections, a method is proposed to compensate the fact that the planar approximation may not be verified in all voxels by fitting and then blending the patches. Before doing fitting, we need to connect the surface patches that share the same surface type. To do so, a segmentation approach is proposed in section 2.4 that aims to group the patches with the same surface type into connected regions.

## **2.4 3D Segmentation method in the vector field**

3D point cloud segmentation is the process of categorizing the point cloud into homogenous regions whereas classification is the step that tags these regions with the name of different categories depending of the application. The aim of segmentation is to group data such that similar objects are in one segment and the objects of different segments are dissimilar. As explained in section 2.2, identifying the surface type for each voxel is first applied to the vector field built from the scan of the 3D object. Then six different surface types are identified. Each surface type is a classification, the regions belonging to the same classification need to be connected together. Since the initial step of the segmentation has been described in section 2.1, the K-means algorithm is an appropriate method to connect voxels with the same surface type into a region (or segment), this procedure is described below.

K-means is a simple algorithm that groups a point cloud into a number of  $K$  clusters specified by a user. In other words, the K-means method defines  $K$  centroids to define clusters and then allocates an element to a particular cluster if it is closer to that cluster's centroid than any other centroid [61]. This method segments the data into  $K$  segments even if  $K$  is not the correct number of segments to use. Therefore, users need to find a way to identify whether the right number of clusters  $K$  is used or not [61].

The method that is used in our work to validate the number  $K$  is the elbow method. The idea behind the elbow method is to run the K-means method on the data with a range of values for  $K$  [62, 63]. Then for each value of  $K$ , the sum of squared distances from each point to its assigned center is computed. Then the sum of squared distances for each value of  $K$  is plotted. If the plot looks like an arm, then the elbow on the arm is the best value for  $K$  [62, 63].

To visualize the concept of the K-means using the elbow method, a simple sample of cloud of 3D points is considered as shown in Figure 2.10.

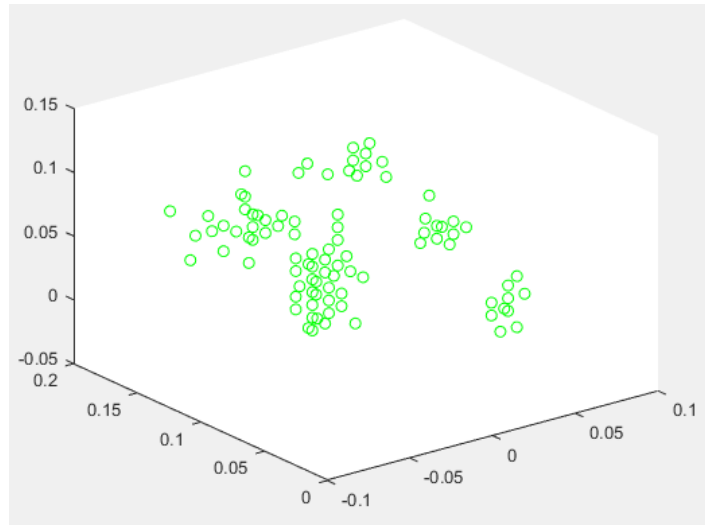


Figure 2.10: Sample data point to show the K-means segmentation that uses the elbow method

Then the elbow method to obtain the best  $K$  value is applied on the sample points shown in Figure 2.10. The result of the elbow method is shown in Figure 2.11. The elbow in the arm is shown with a red circle which has assigned the value 9 to  $K$ .

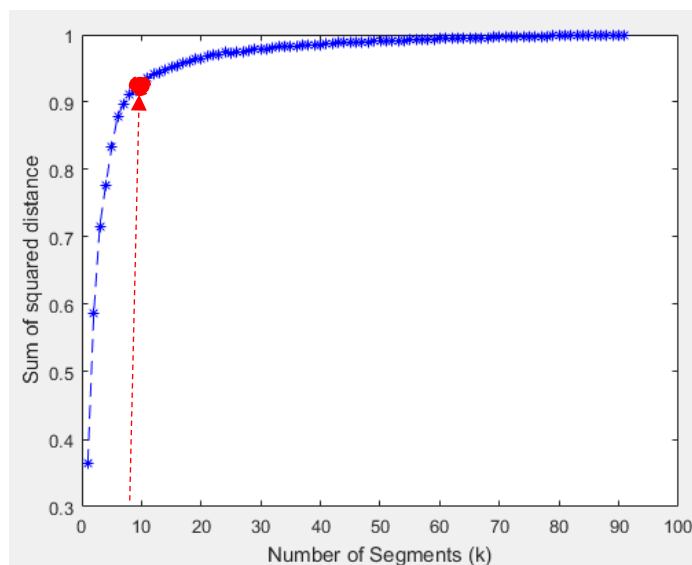


Figure 2.11: The elbow method for determining the number of segments

When the best value for  $K$  is determined using the elbow method, the K-means is used to obtained value of  $K$  for segmentation. The result of the segmentation for the sample points shown in Figure 2.9 is shown in Figure 2.12.

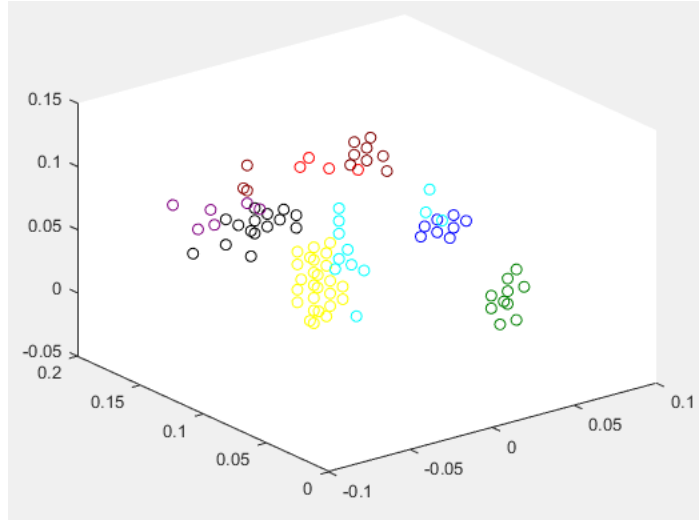


Figure 2.12: Segmentation of a point cloud using K-means elbow method.

## Implementation

As explained above, since the initial classification of voxels is done using the surface type method described in section 2.1. Then segmenting a 3D object into the regions of the same surface type by using the aforementioned segmentation algorithm is very simple.

In this section, the results of the segmentation are given in Figure 2.13 to Figure 2.16 for the Bunny model. More results will be presented in the next chapter. Figure 2.13 illustrates the segmented area for ridge surface, Figure 2.14 shows the result of the segmentation for the peak surface. The segmentation results for valley surface are given in Figure 2.15. The segmentation of the saddle region is shown in Figure 2.16.

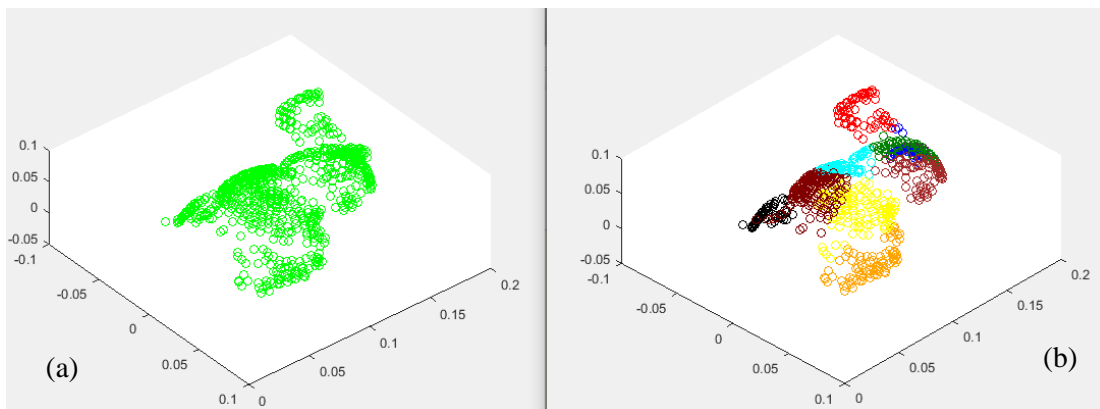


Figure 2.13: Segmentation of the Ridge region. (a) Initial data points. (b) Segmented regions

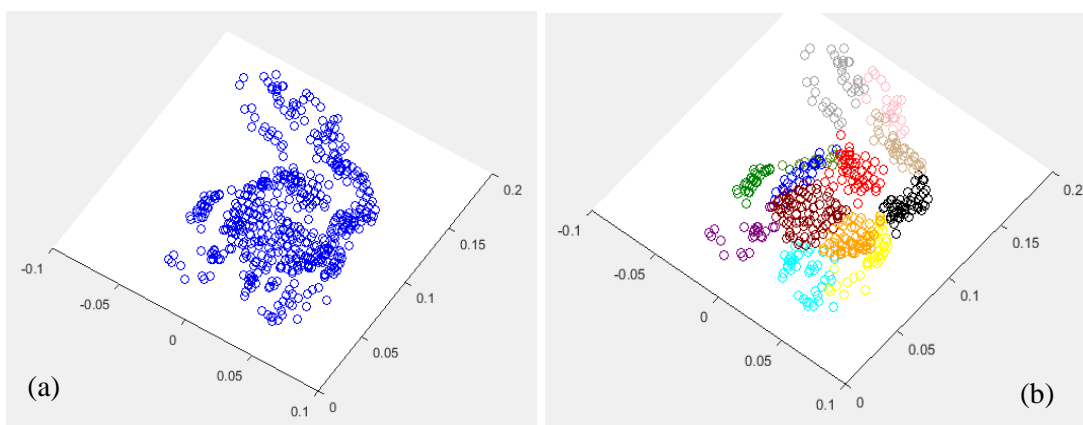


Figure 2.14: Segmentation of the Peak region. (a) Initial data points. (b) Segmented regions

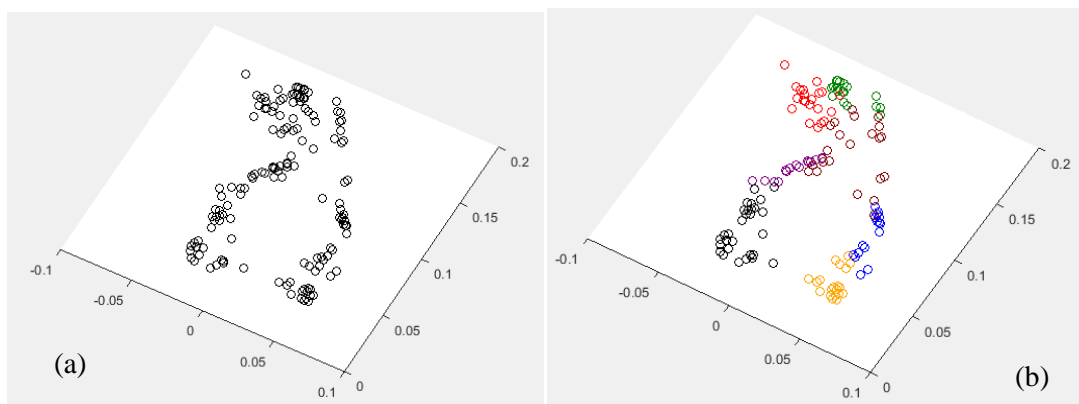


Figure 2.15: Segmentation of the valley region. (a) Initial data points in the valley region. (b) Segmented regions

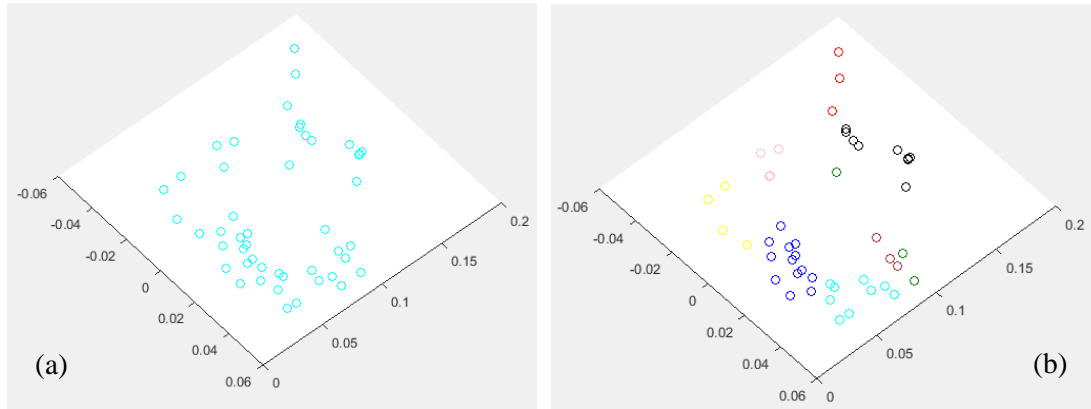


Figure 2.16: Segmentation of the saddle region. (a) Initial data points in the saddle region. (b) Segmented regions

In Figure 2.17, the workflow of segmentation of all different surface types is shown.

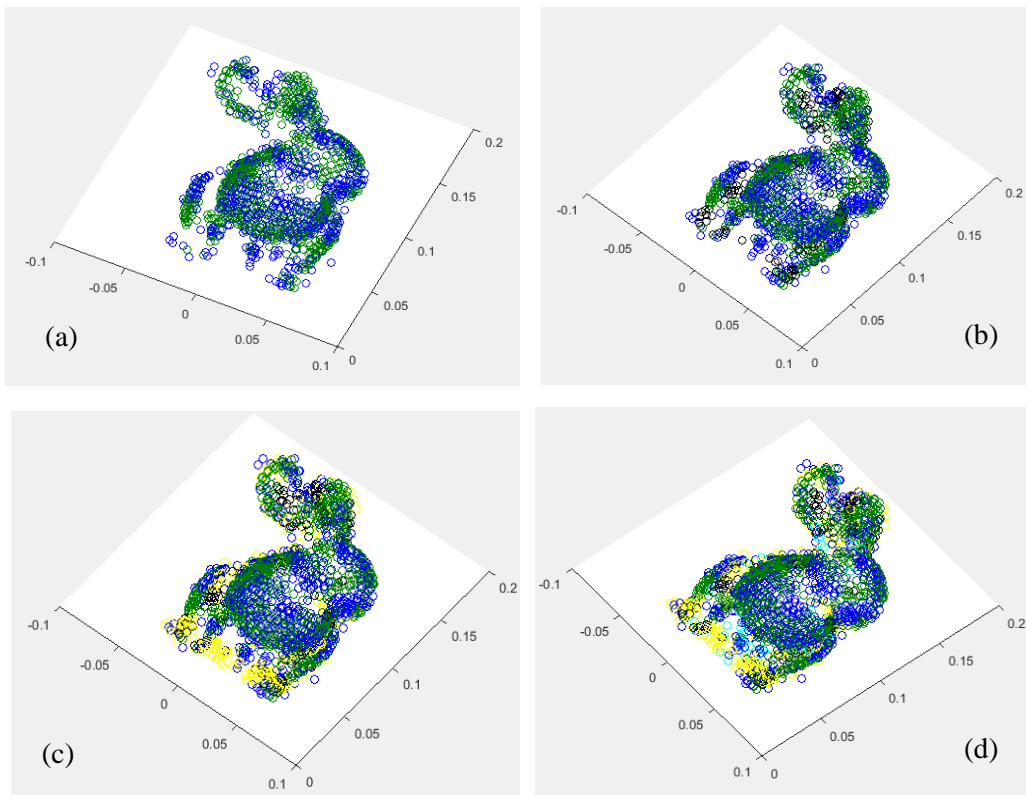


Figure 2.17: All segmented regions. (a) Segmented ridge and peak surface (b) Segmented Ridges, Peak and valley surface (c) Segmented Ridges, peak, valley and Pit region (d) Segmented Ridges, peak, valley, Pit and saddle region.

3D points falling into a voxel need to be presented by a high order surface representation. The advantage of the segmentation step in this project is that instead of fitting the surface in each voxel of the vector field separately, a high order surface representation can be built on a segment of the connected voxels with the same surface types. Which results in decreasing the computational complexity.

The next section presents the fitting method that has been implemented to build a high order surface representation.

## **2.5 Fitting a high order surface representation**

In this step, an approach is proposed to build a high order representation of each segment obtained from the previous step (K-means segmentation).

Implicit B-splines (IBS) are a way of building a representation that can provide shape descriptors through their zero-sets and reconstruct surfaces. This technique is based on locally controlled functions that are combined via control points. This local control allows patch-based object representation [64]. In such a method, each parameter has a local contribution to the shape, which makes it useful for algebraic fitting methods, although it can be also used in a geometric framework [65].

To represent the surfaces in the segmented regions obtained from the segmentation step (section 2.4), we adapt an implicit B-spline method proposed by [42, 44]. The method proposed by Rouhani et al. presented an extension of the 3L Implicit Polynomial algorithm (section 1.4) to the implicit tensor-product B-spline. One limitation of the 3L- IP approach is that although increasing the IP degree enables the description of objects with more complex geometries, it causes some artifacts and outliers to appear on the object. This limitation has been eliminated by 3L- Implicit B-spline (IBS).

The IBS definition in Equation 1.35 can be reformulated (see Equation 2.3) as the inner product of the coefficient vector  $c$ , which includes control values  $\{c_{i,j,k}\}$  and the basis vector  $m(x)$  which is a vector form of monomials  $\{B_i(x)B_j(y)B_k(z)\}$  [34]:

$$f(x) = c^T m(x) = m(x)^T c \quad (2.3)$$

According to Rouhani et al. [44], the B-spline functions can be constructed through the following blending patches:

$$\begin{aligned} b_0(u) &= (1 - u^3)/6, & b_1(u) &= (3u^3 - 6u^2 + 4)/6, \\ b_2(u) &= (-3u^3 + 3u^2 + 3u + 1)/6, & b_3 &= u^3/6 \end{aligned} \quad (2.4)$$

To determine the control point vector  $c$ , the point cloud is normalized into a unit cube  $[0 \ 1]^3$ . Then the IBS definition in Equation 1.35 can be directly reformulated based on the blending functions as:

$$f(x) = \sum_{r,s,t=0}^3 c_{i+r,j+s,k+t} b_r(u) \cdot b_s(v) \cdot b_t(w) \quad (2.5)$$

where

$$i = \lceil x/\Delta \rceil, j = \lceil y/\Delta \rceil, k = \lceil z/\Delta \rceil$$

$$u = \frac{x}{\Delta} - \left\lfloor \frac{x}{\Delta} \right\rfloor, v = \frac{y}{\Delta} - \left\lfloor \frac{y}{\Delta} \right\rfloor, w = \frac{z}{\Delta} - \left\lfloor \frac{z}{\Delta} \right\rfloor$$

$$\Delta = 1/(N - 3)$$

Therefore, the unit cube is split into a  $N \times N \times N$  voxel grid where  $N$  is the IBS resolution. Each control point in  $c$  is defined with an index number, which indicates the vertex of this grid at which the related control point is located [66]. For more detail on the 3L algorithm, the reader is referred to section 1.4.1.



## Implementation

The IBS method proposed by Rouhani *et al.* and adapted to our problem has been implemented on the segmented regions that were obtained from the segmentation of the vector field into regions composed of connected voxels sharing the same surface type. To show the performance of the method in this context, fitting results are presented in Figure 2.18 to Figure 2.22.

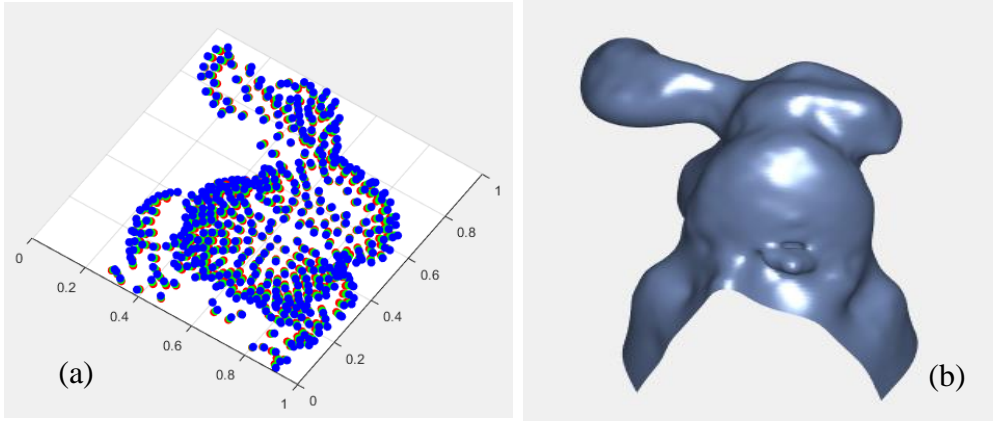


Figure 2.18: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Ridge segmented regions

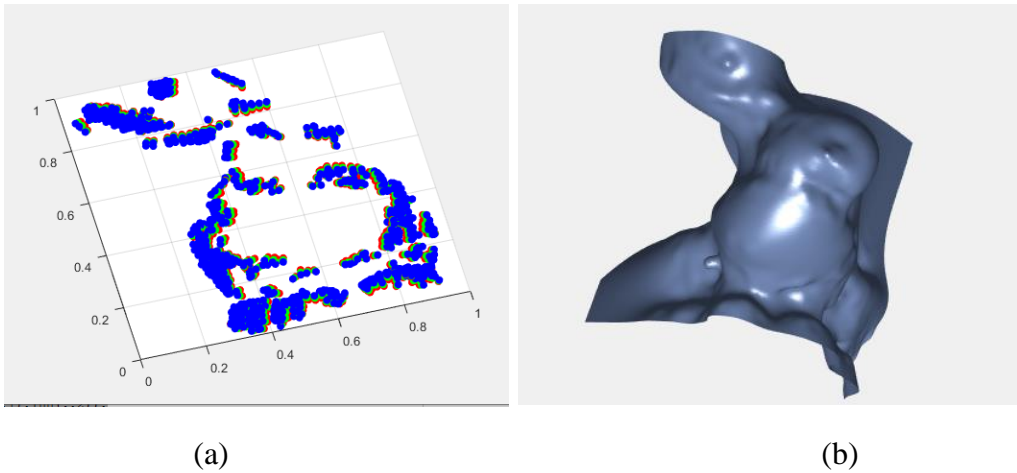


Figure 2.19: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Valley segmented regions

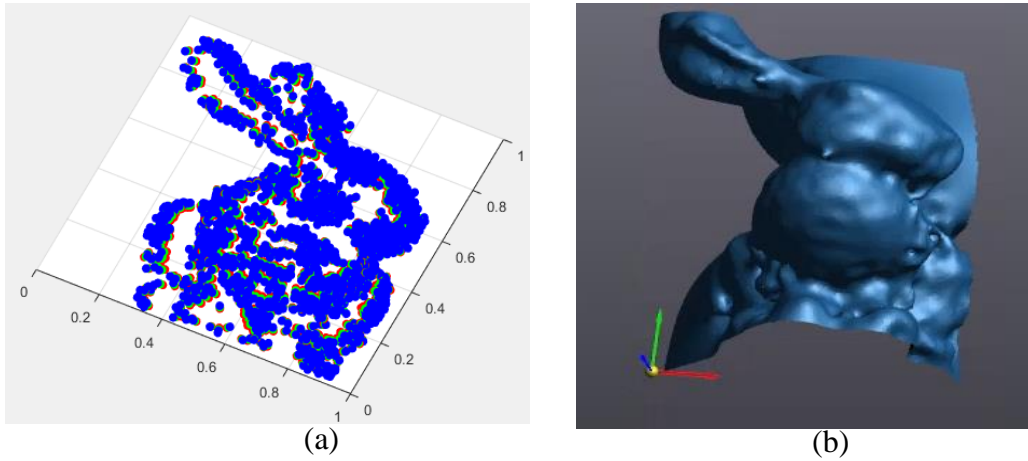


Figure 2.20: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Peak segmented regions

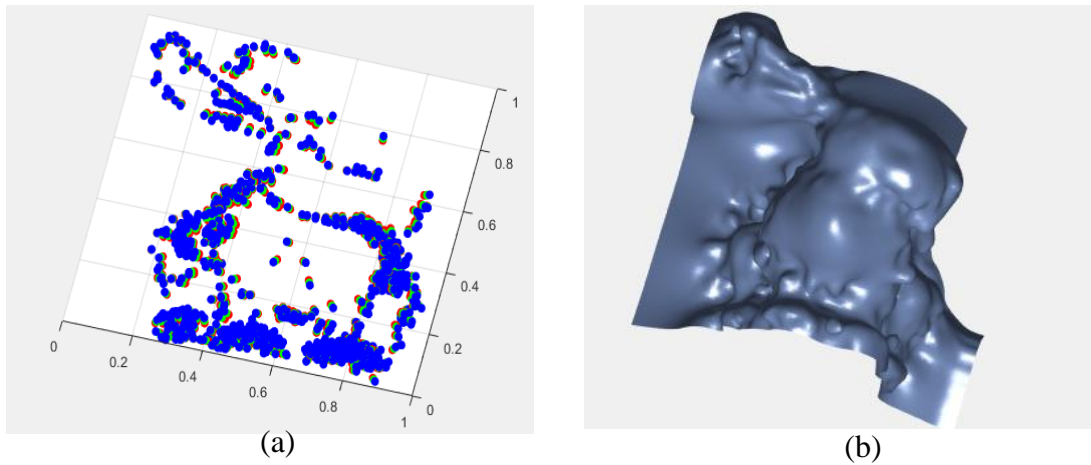


Figure 2.21: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Pit segmented regions

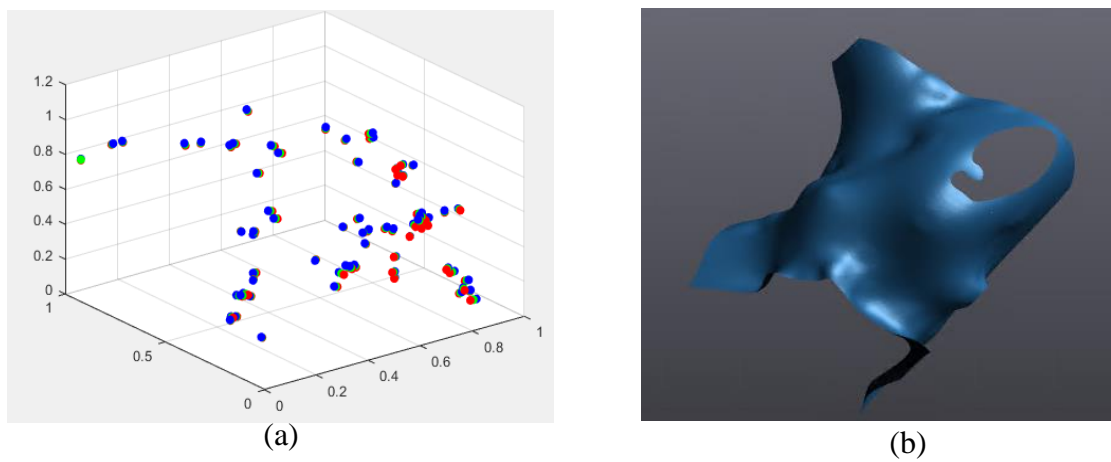


Figure 2.22: IBS fitting. (a) Generating inner offset (red) and outer offset (blue). (b) IBS fitted to the Saddle segmented regions

Since each segmented region is represented by IBS, there are thus multiple B-spline patches which need to be merged together in order to represent a single surface. In the next section, the proposed approach to blend the surface patches is described.

## 2.6 Merging different surface patches to obtain a single 3D mesh

Once the different segments have been fitted, merging the different fitted segments is the last step in our workflow. Once the patches are constructed by fitting the B-spline in each segmented region, then they need to be blended to build a single surface representation. To merge these patches, the approach proposed by [54] called “Zippered polygon meshes” is applicable to our work. It uses geometric fusion to integrate multiple surface patches to obtain a single surface. Thus, the output of this method is a merged mesh. The details on this approach are given in section 1.5.3.

Fusion of multiple meshes requires an ‘overlap’ test to determine if surface measurements from different meshes in close spatial proximity correspond to the same or different regions of the measured object surface [67]. See Figure 2.23. Geometric constraints are used to estimate if overlapping measurements correspond to the same surface region. To find more details on geometric constraints to estimate the overlap the reader is referred to [67].

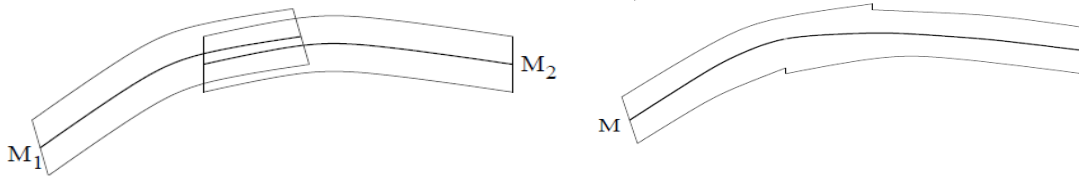


Figure 2.23: Fusion of field-functions for cross-sections through overlapping volumetric envelopes (a) Overlapping field-functions ( $M_1$ ,  $M_2$ ) (b) Integrated field-function ( $M$ ). Taken from [67, 68].

Instead of coding from scratch to blend the patches and guarantee the continuity between the patches, we rather work with Creaform's VXelements (3D Software Platform) in which different patches can be merged to achieve a single 3D mesh and works similarly to the method proposed by Levoy et al. [54].

## **Implementation**

To merge different implicit B-spline patches, the first step consist in the conversion of these aforementioned patches into a mesh. This step has been implemented in MATLAB by a simple code to extract isosurface data from volume data and then converting this data into a "PLY" or "STL" mesh file format. Then, the triangular meshes are imported into VXelements. Next, the "Merge" option in this tool allows us to select different 3D meshes and then blend them together to achieve a final surface.

In the case of the "Bunny" object, five different patches shown in Figure 2.18 (b), Figure 2.19 (b), Figure 2.20 (b), Figure 2.21 (b) and Figure 2.22 (b), are merged to achieve the final model. There is a red right brace shown in Figure 2.24 which indicates "VXmodel" section that shown different patches which are fitted in each segment of the bunny object that are imported into VXelements. The next step is choosing "Merge" as indicated by a red rectangle in Figure 2.24. Then the red box shown in Figure 2.25 will appear. From the list of meshes that are shown in the red box, the desired surface patches should be selected to be merged. Each patch is shown with a different color as illustrated in the black box shown in Figure 2.25.

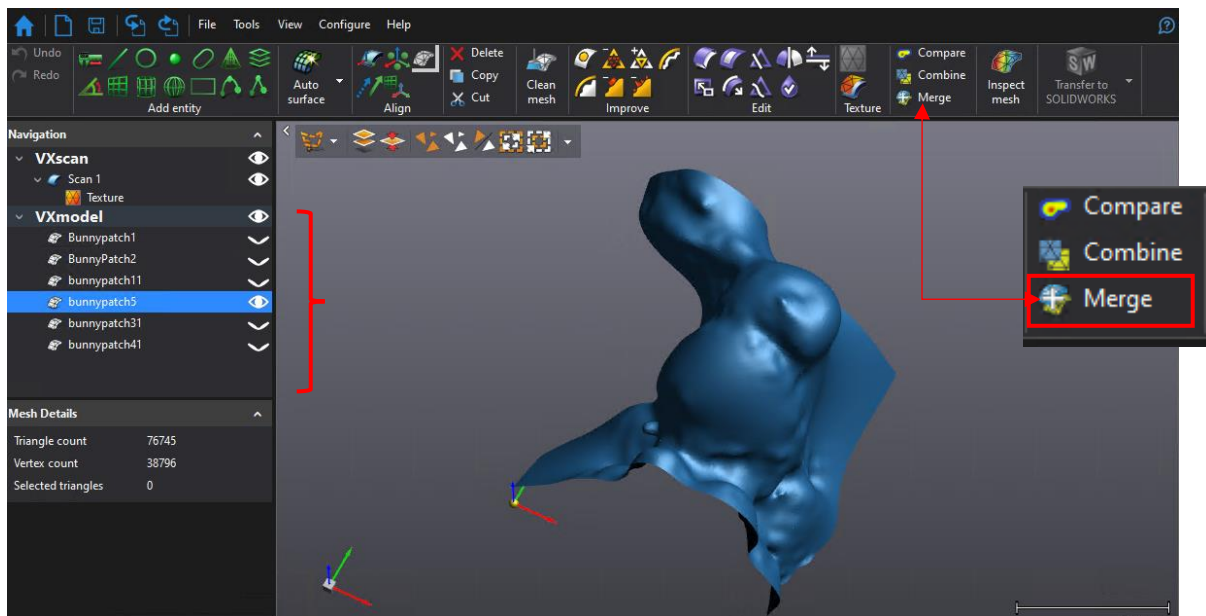


Figure 2.24: Merge different meshes in the VXelements software platform.

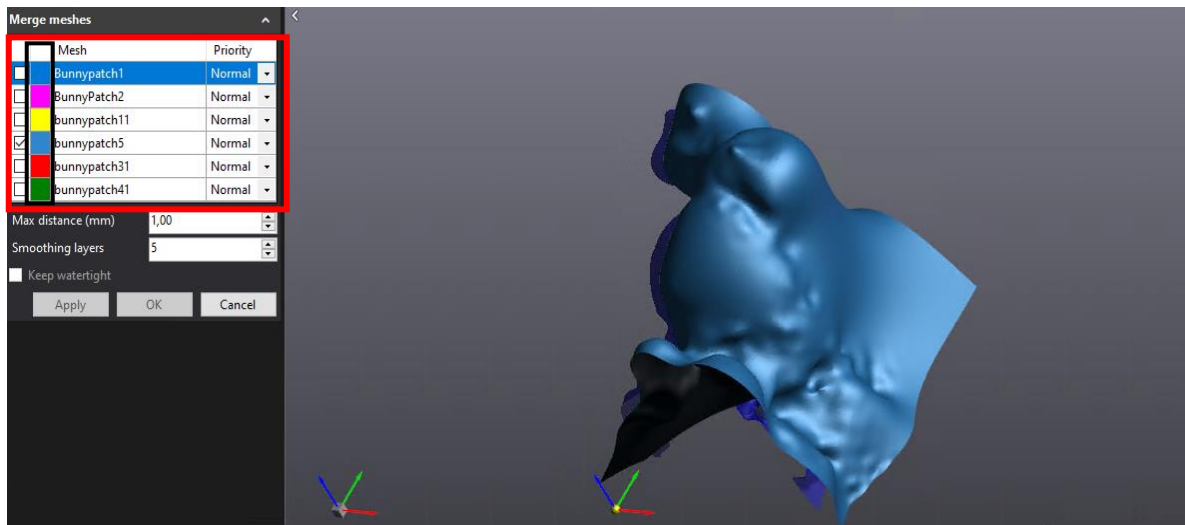


Figure 2.25: Merge different meshes in the VXelements software platform.

The surface obtained by merging the different surface patches belonging to the “Bunny” object is shown in Figure 2.26.

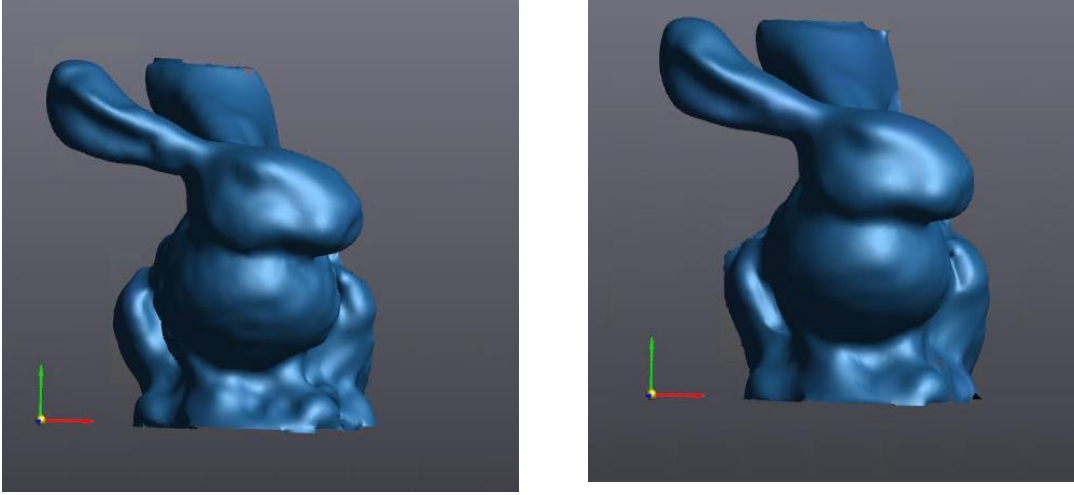


Figure 2.26: The surface obtained from merging different meshes. (a) The result with less smoothing weight (b) The result with higher smoothing weight.

In this chapter, the proposed approach on how the vector field is combined with covariance based differential geometry to build the model of an object from an unstructured point cloud is explained and how the proposed surface variation method can reduce memory requirement through the selection of an adequate resolution of the implicit representation.

In the next chapter, more results are presented to validate the surface representation approach proposed in this thesis. A brief discussion is also given for each 3D object reconstructed by this approach.

## Chapter 3

# Experimental Results

In this chapter, we present and discuss the experimental results obtained with the approaches mentioned in chapter 2. The ordering of the sections is the same as the one in the previous chapter in order to present the results of the different steps coherently. We have implemented the proposed approach on MATLAB R2016a (9.0.0.341360). The experiments are executed on a computer running Windows 7 with an Intel® Core(TM) i7-5820K CPU @ 3.30 GHz and 48GB of RAM.

### 3.1. Preparing the data

To test the proposed approach, we used a set of 3D data provided by the Stanford repository, a Creaform handheld 3D scanner and synthetic data generated in MATLAB.

### 3.2. Identifying the surface type in the vector field

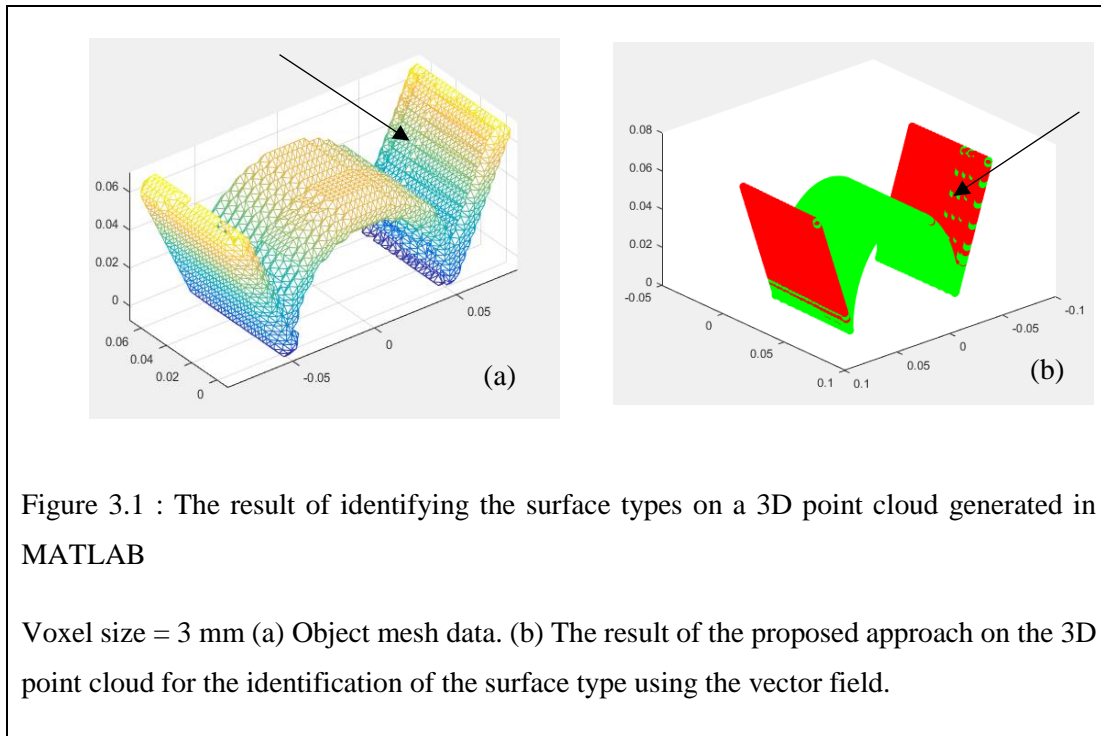
This section presents the results on the identification of the different surface types using the vector field framework. Figure 3.1 and 3.2 illustrate the results of surface type identification on a model captured by a handheld 3D scanner provided by Creaform and a synthetic data generated in MATLAB. The generated synthetic data is a combination of a cylinder and two planes. The color map for different surfaces is the same as the one shown in Table 2.1 such that Peak surfaces are shown in blue, Pit in yellow color, Ridge, Valley shown in green and black color respectively. Saddle and minimal surface in cyan color and Planar region shown in red color.

The object shown in Figure 3.1 is a synthetic object generated in MATLAB and which is a combination of a cylinder and two planes.

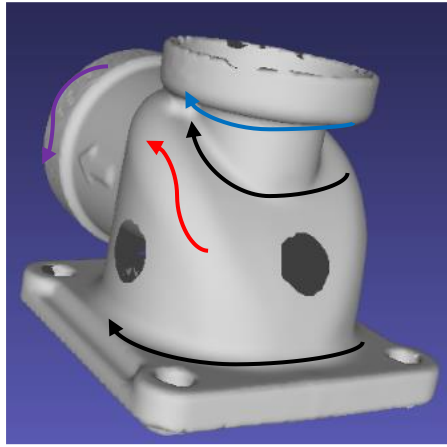
The result shown in Figure 3.1 (b) demonstrates that we obtained the corresponding ridge (cylindrical) surface type in green color and the corresponding planar surface type in red as expected. The black arrow in Figure 3.1 (a) shows that some stripes exist in the object mesh data. The black arrow in Figure 3.1 (b) presents some green points in the planar region, which demonstrate that the proposed approach is sensitive enough to detect the surface types other than a plane.

A HandyScan scanner by Creaform collected the 3D points of the object shown in Figure 3.2. Since the scanner is designed for metrology applications, the accuracy of the 3D data is high and the level of noise is low. As labeled in Figure 3.2 (a), the red arrow points to the regions with high curvature and belonging to the spherical surface type. The region shown by the purple arrow correspond to the ridge type. The blue arrow corresponds to a saddle surface. In addition, the black arrows points to the valley surface type. The corresponding surface types shown in Figure 3.2. (b) show a graphic representation of these surface types.

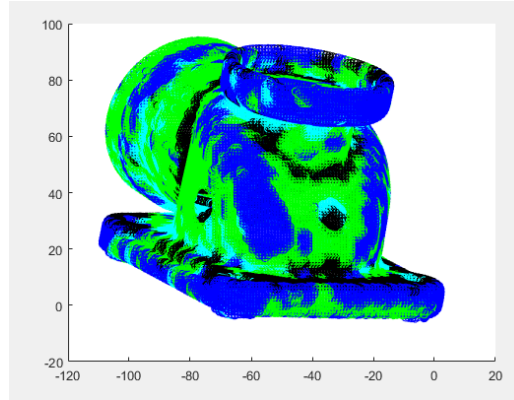
The ground truth surface types were established manually since the scan was obtained from a real engine part for which “independent” ground truth does not exist.







(a)



(b)

Figure 3.2 : The result of the identifying the surface types on a 3D point cloud collected by HandyScan (Creaform) scanner.

Voxel size = 5mm (a) Object mesh data. (b) The result of the proposed approach on the 3D point cloud for the identification of the surface type using the vector field.

### 3.3. Surface variation method in the vector field framework

As mentioned earlier in this document, MATLAB was used for the development of the algorithms and for the experiments. However, it happens that memory limitation associated with MATLAB prevent the use of very large vector fields. Although this could be seen as a problem in the general case, it is rather considered as an interesting limitation in the context of this project and serves the purpose of demonstrating the approach for the selection of the optimal voxel size. There is no need to use very large point clouds to show that the approach based on surface variation for the automatic selection of the size of the voxels works well and that it could translate to large point clouds (of large objects) if the approach was fully implemented in C++ for instance. Consequently, the reader should not be surprised if the object used in the experiments are not large.

To further validate the approach proposed for choosing an optimal voxel size and keep the number of voxels as small as possible, the approach using surface variation was used (see section 2.3). The method is applied on different objects as shown in Figure 3.3 to Figure 3.6.

As mentioned in section 2.3, when the surface variation in each voxel is computed, then the interval of the surface variation value  $[0, 1/3]$  is divided into 100 bins and the number of voxels with a value of surface variation falling in each bin is computed.

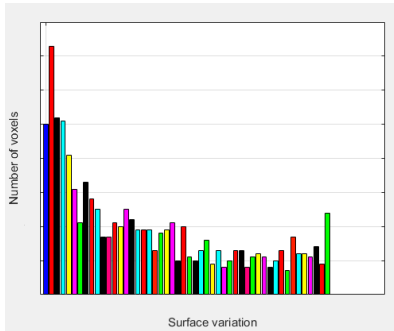
Figure 3.3 to 3.6 illustrate the histograms of the number of the voxels assigned to each bin. The strategy in our proposed approach is to stop decreasing the voxel size when it is observed that the number of the voxels falling in the first bin (planar region) is greater than others.

Figure 3.3 to 3.6 show the surface variation histogram corresponding to the object scanned by the HandyScan Creaform scanner. At the first step, 7 mm is selected for the starting voxel size as shown in Figure 3.3. Then the size is decreased progressively by one mm down to 4 mm. Figure 3.5 illustrates the results of the surface variation related to “voxel size = 5 mm”. It is clear that the number of the voxels in the planar bin (shown in Blue color) is greater than the bin on its right. So, as described in section 2.3, this is the size that is selected to reconstruct the model.

To illustrate the reason to justify the implementation of this method, the size of the voxels is decreased one more unit to 4 mm as shown in Figure 3.6. The result shown in this figure is close to the results shown in Figure 3.5 (the number of voxels in the planar region is greater than the bin next right to it). The difference is that with a voxel size of 5 mm, the number of voxels in the vector field is 35805. In contrast, the number of the voxels in the vector field for a voxel size of 4 mm is 57561. The largest point-to-point measure of the object collected by Creaform’s scanner (shown in Figure 3.1(a)) is 125 cm and reduction of the number of voxels would be even great for objects with a large size.

The reason why a voxel size equal to 5 mm is selected to perform further processing to construct the model is thus clear: choosing 5 mm as the voxel size reduces the number of voxels by 22K, which is a significant reduction in size and reduces memory consumption. With a voxel size of 5 mm, there are enough voxels to achieve a good construction of a model

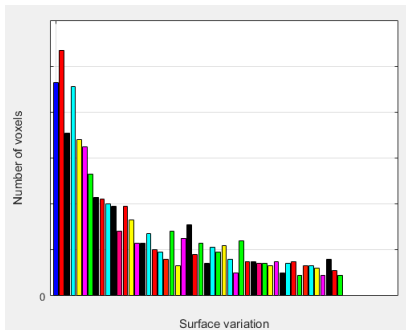
while keeping the memory usage as low as possible. The fitting/blending step will contribute to enhance the final model.



Number of voxels in the vector field:

$$29 \times 27 \times 31$$

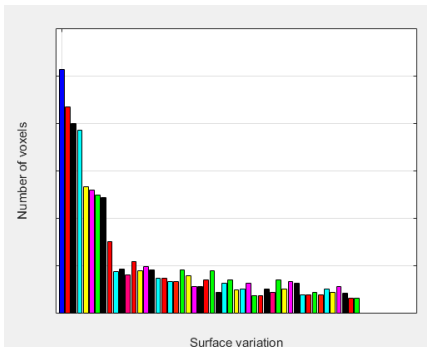
Figure 3.3 : Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): Voxel size =7 mm



Number of voxels in the vector field:

$$33 \times 31 \times 35$$

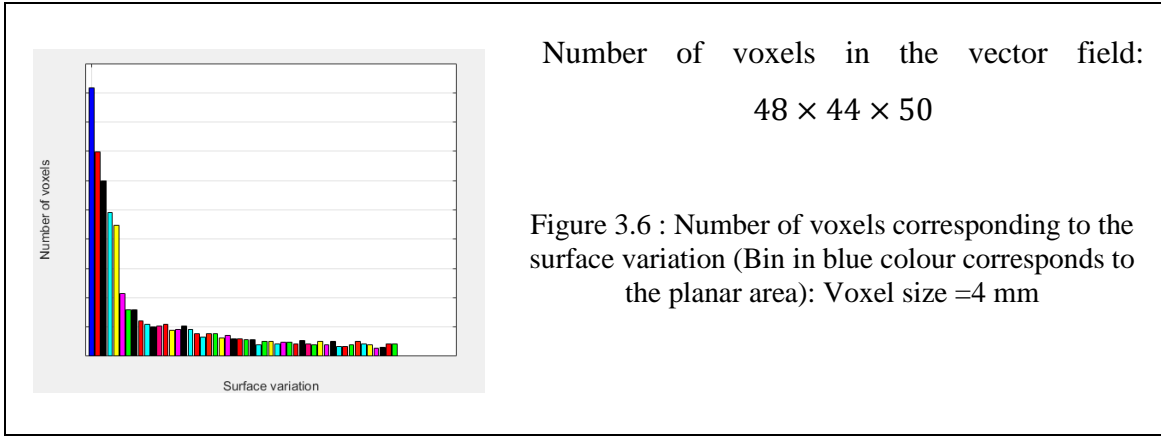
Figure 3.4 : Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): Voxel size =6 mm



Number of voxels in the vector field:

$$39 \times 36 \times 41$$

Figure 3.5 : Number of voxels corresponding to the surface variation (Bin in blue colour corresponds to the planar area): Voxel size =5 mm



### 3.4. 3D segmentation method in the vector field

Once the optimal voxel size has been selected with the approach presented in the previous section then the next processing step consists of the construction of a model using this voxel size. As presented in section 3.2, the surface types have been identified in each voxel of the vector field. To connect the neighbouring voxels with the same surface type, a K-means segmentation algorithm is used as explained in section 2.4. In the segmentation step, the elbow method is applied to achieve the best  $K$  value. This section presents additional example of the segmentation in the vector field on the object collected by the Creaform HandyScan in Figure 3.7. The result shown in Figure 3.7 (a) is the segment region of ridge surface type that contains different connected sub-segments (shown in different colors). The region related to the segmented peak surface type contains different sub-segments (illustrated in different colors) and is shown in Figure 3.7 (b). Figure 3.7 (c) and 3.7 (d) show the segments of valley surface type and segmented saddle surface type respectively. Each different color in the segmented area correspond to a connected sub-segment of the given surface type. As mentioned earlier, these aforementioned sub-segments in each specific surface type area are connected to each other.

Once the segmentation is complete and connected regions with the same surface type have been found, the next step is to build the high order representation of the surface for each segment.

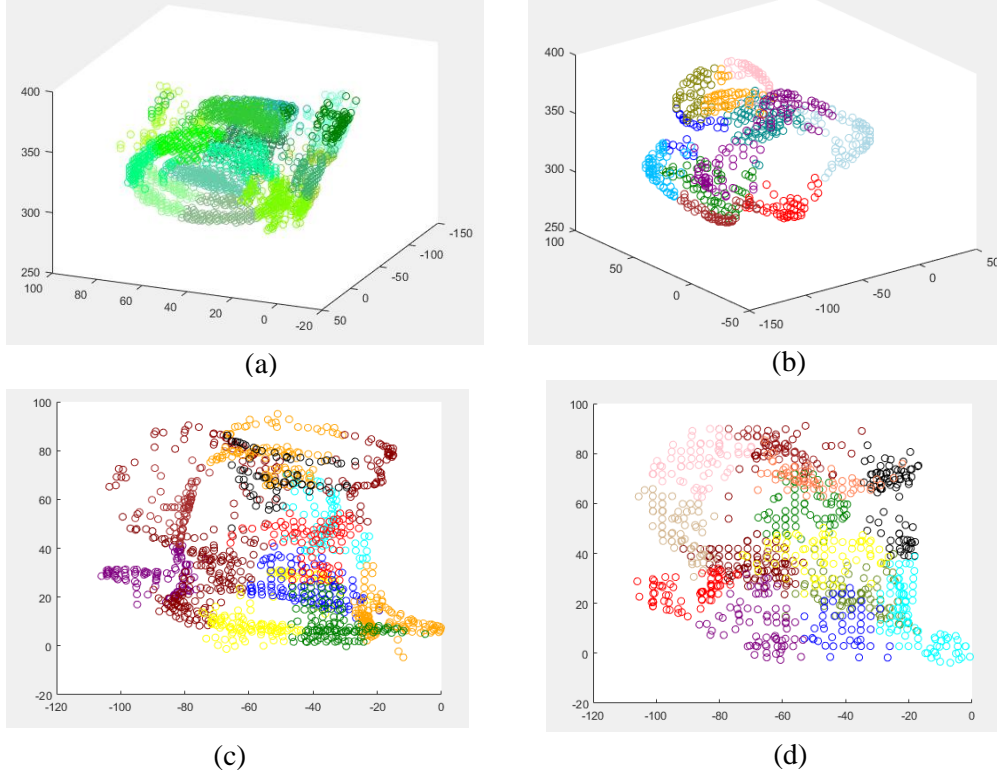


Figure 3.7: 3D segmentation of the object provided by Creaform according to the different surface types. (a) Ridge segments (b) Peak segments (c) Valley segments (d) Saddle segments

The next section presents the results of the surface representation using the 3L-IBS method presented in section 2.5 to represent each segment area associated to a certain different surface types (Ridge, valley, peak, pit, saddle and plane).

### 3.5. Fitting a high-order surface representation

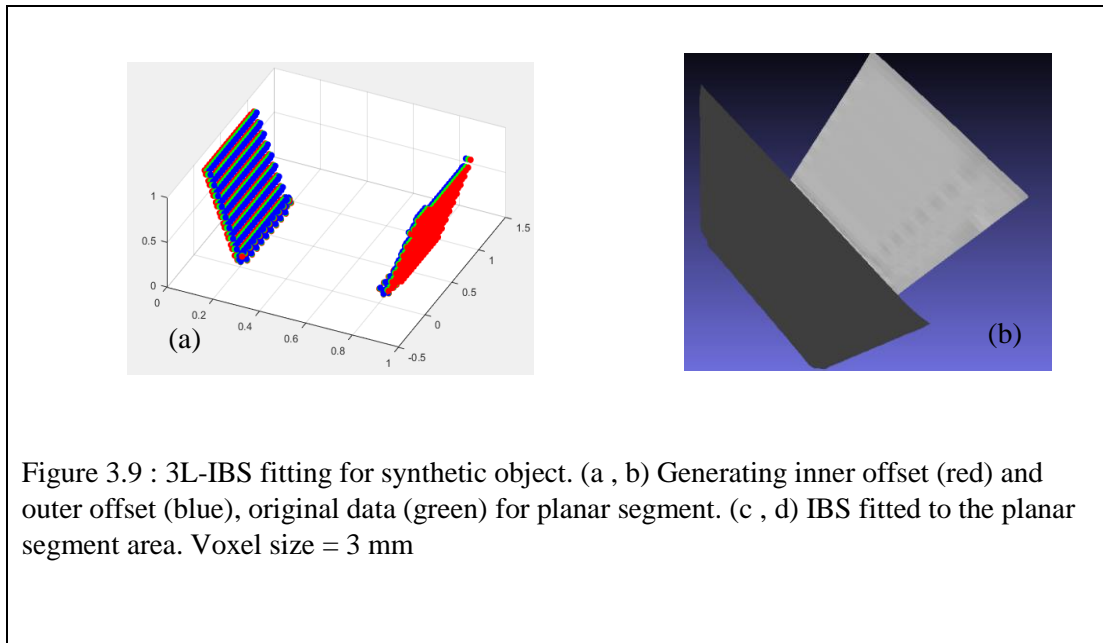
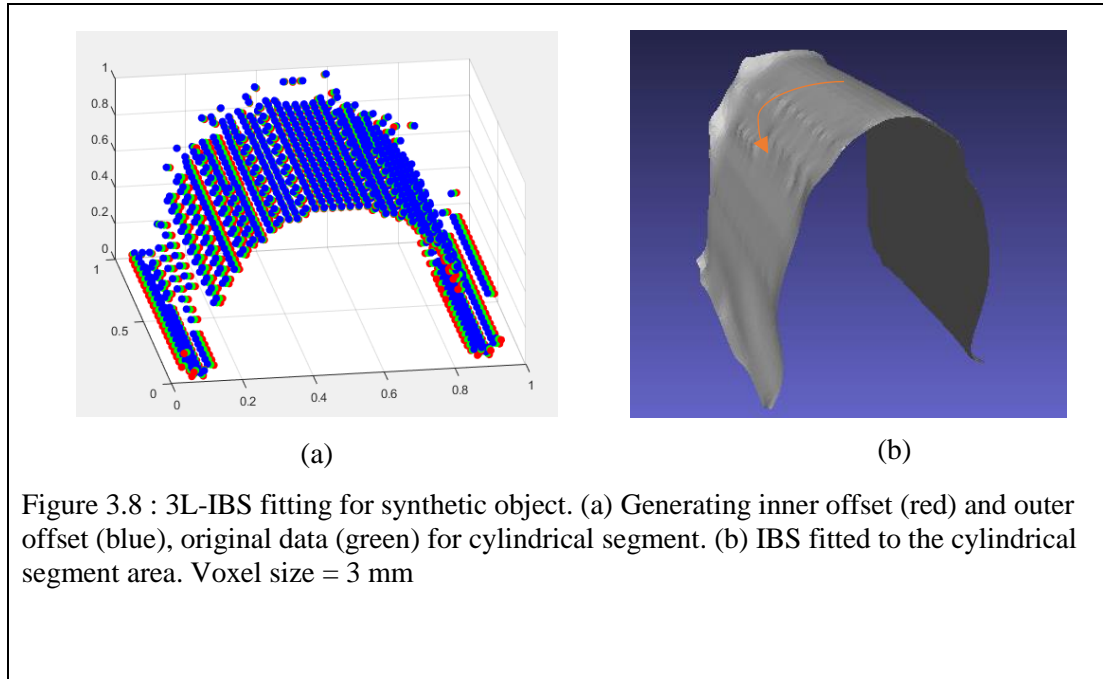
Once the set of segmented regions has been found, we represent the points of each segment with the control points of Implicit B-Splines (IBS). The detail about the IBS fitting method that is used for this purpose has been presented in section 2.5. This method represents the surface without requiring any parametrization. This aforementioned method proposed by Rouhani, *et.al* [44] is used to describe the surface in each segment through its zero-level set. Then, to visualize the surface, the marching cubes algorithm is used to build a mesh. For a better visualization of the fitted surface the results shown in this section on Figure 3.8 to 3.9 are using a proper regularization term. This term is added to the objective function of the 3L-

IBS which has been shown in Equation 1.34. This term which uses regularization parameter  $\mu$  to control the rigidity of the implicit fitted function, is shown in Equation 3.1. Increasing the value of this regularization parameter results in smoother surface and vice versa [34].

$$c = (M_{3L}^T M_{3L} + \mu D)^{-1} M_{3L}^T b \quad (3.1)$$

In Equation (3.1),  $D$  is a  $N^3 \times N^3$  sparse matrix including the integral of overlapping basis functions and their derivatives [33, 53]. For more detail on the regularization term, the reader is referred to [55]. However, the point that need to be considered is that before using the regularization term, the IBS is obtained and control points and the basis functions are computed. Then the IBS patches are converted into meshes in order to blend the patches together. Figure 3.8 and Figure 3.9 illustrate the fitting process in each segment of the synthetic data.

Figure 3.8 (a) and 3.9 (a) shown the two offsets supporting the original data (shown in green color). The inner offset is shown in red color and the outer one is shown in blue color. Figure 3.8 (a) refer to the cylindrical segmentation that we want to represent by a high order surface description (using IBS fitting). The representation results achieved for this segment using the 3L-IBS method is shown in Figure 3.8 (b). Figure 3.9 (a) show the segmented region corresponding to the planar surface type. Figure 3.9 (b) illustrates the fitting result for this planar segment using the 3L-IBS method to represent the surface. In this workflow, the voxel size for processing the synthetic data (cylinder and two planes) has been set to 3 mm.



There are some ripple areas pointed by the orange arrow in Figure 3.8 (b). The following explanation describes the presence of this ripple. Figure 3.10 illustrates a 2D image of a synthetic cylinder on a 2D grid. Each small square is a voxel (a voxel is shown with the red arrow). As shown by the black arrows from top to the bottom of the object in Figure 3.10 (a), (b) the step of moving to the next voxel changes with the size of the voxel. Since we are

working with closest points in voxels instead of dense point cloud, these steps lead to small ripples in the fitted surface shown in Figure 3.8 (b). In other words, the location of the closest points in neighbouring voxels and how they are placed relative to each other are the reason for these ripples. The results in Figure 3.10 were also generated to illustrate the importance of choosing a good voxel size. The voxels in Figure 3.10 (a) are larger than the voxels in Figure 3.10 (b). It is shown that the ripple is more visible in Figure 3.10 (a) than in Figure 3.10 (b). It means the ripple effect is more visible for larger voxel sizes.

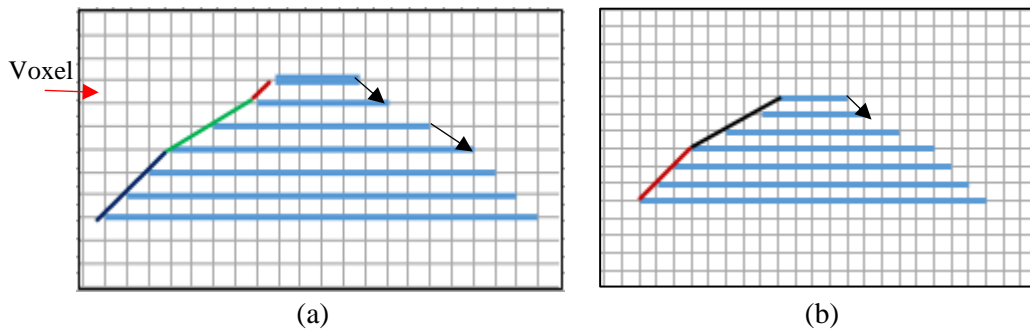


Figure 3.10 : (a) 2D illustration of simulated cylindrical data in larger voxel size (Red, green and purple line show the ripples). (b) 2D illustration of simulated cylindrical data in smaller voxel size (Purple and red line show the ripples).

In the Creaform VXelement application, there are functions such as “clean mesh”, “remove spike”, “fill holes” and etc. that could help to remove the spikes, ripples and smooth the surface. The Figure 3.11(a) shows how the ripple on the mesh shown in Figure 3.8 (b) can be disappear and how the aforementioned surface could be made smoother. In the area indicated by green brace, the smoothing weight could be selected. In this case, the smoothing weight is 25, this value can change depend to the object and the level of the ripple appearing on each object.



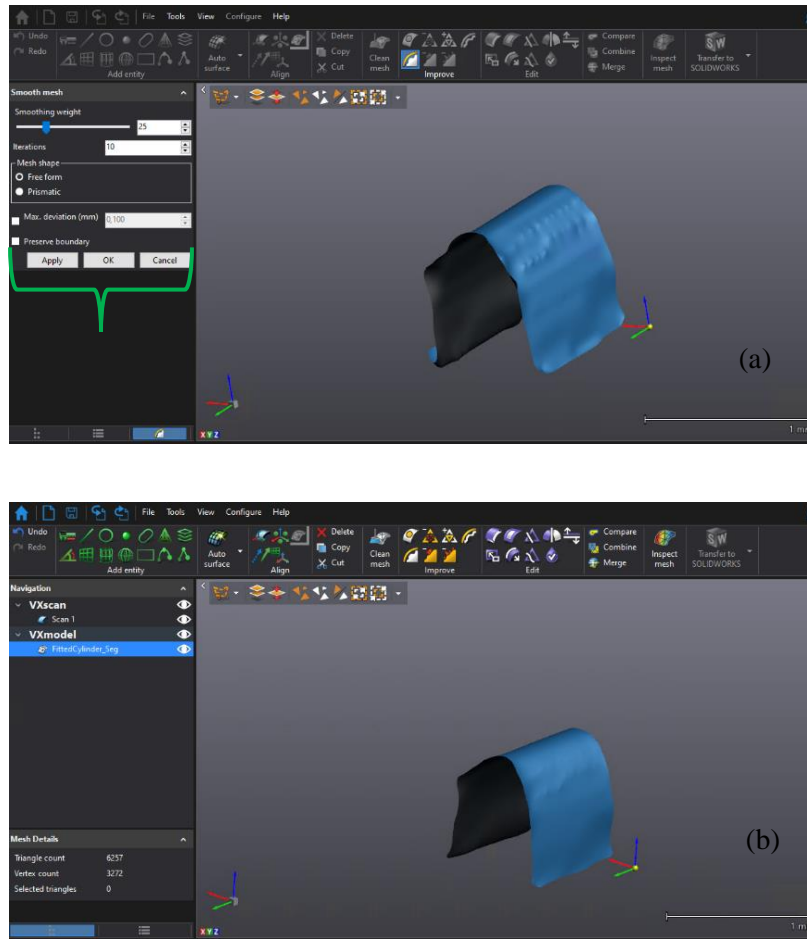


Figure 3.11: (a) Smoothing mesh in VXelements application. (b) The result of using smooth mesh tools in VXelements (Voxel size = 3mm).

Figure 3.12 to Figure 3.15 shows the fitting process for the object provided by the Creaform HandyScan scanner.

Figure 3.12 (a) shows the two offset (in blue and red) generated for the ridge segmentation area that need to be represented by a high order surface representation method. The fitting results for this segment using the 3L-IBS method are shown in Figure 3.12 (b). Figure 3.13 (a), Figure 3.14 (a) and Figure 3.15 (a) illustrate the offset generated for valley, peak and saddle segmented areas respectively. For all the figures, the inner offset is shown in red color and the outer one is shown in blue color. The results shown in Figure 3.13 (b), Figure 3.14(b) and Figure 3.15 (b) are the high order surface that were fitted by the 3L-IBS method to the valley, peak and saddle segment areas respectively.

In the results shown in Figure 3.12 (b) to Figure 3.15 (b), it can be observed that some regions are bumpy. Everywhere the surface is bumpier means that there is less points to fit. As mentioned above in Creaform VXelements application, some features can be used to make the surface smoother and remove the spikes available in the surface.

Once a surface patch representing each different segmented regions (ridge, valley, peak, pit, planar, saddle) have been fitted, the next step consist in blending those patches to achieve a single final model. For doing so, the Creaform VXelements application is used.

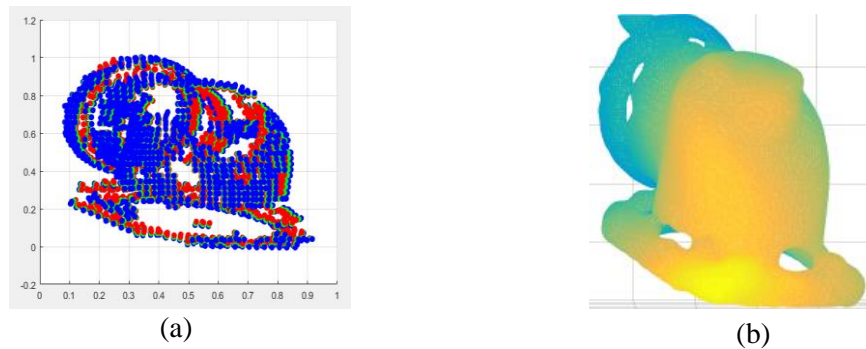


Figure 3.12 : 3L-IBS fitting. (a) Generating inner offset (red) and outer offset (blue) for ridge segment. (b) IBS fitted to the ridge segment area. Voxel size = 5 mm

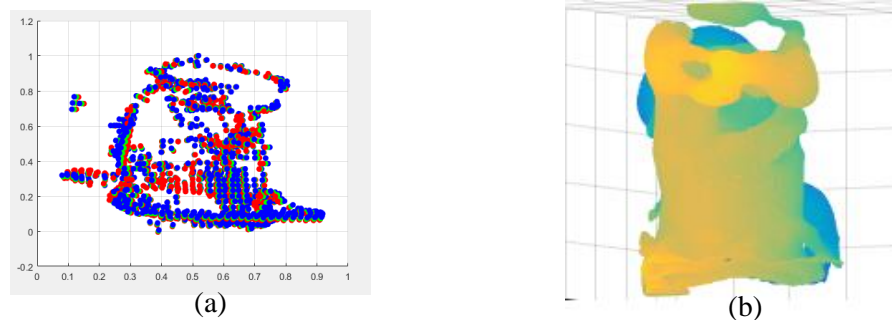
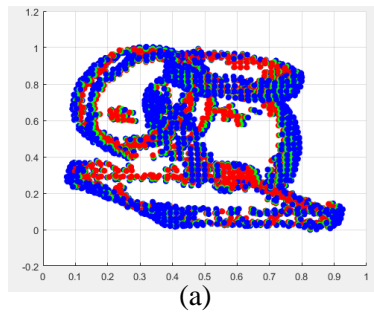
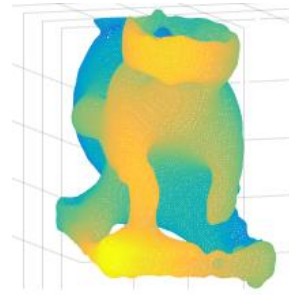


Figure 3.13 : 3L-IBS fitting. (a) Generating inner offset (red) and outer offset (blue) for valley segment. (b) IBS fitted to the valley segment area. Voxel size = 5 mm

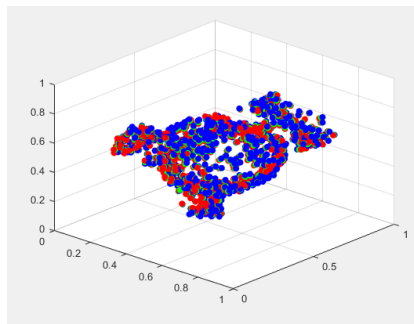


(a)

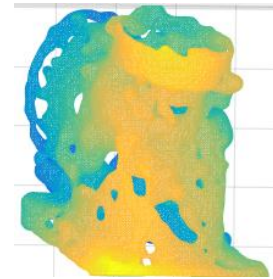


(b)

Figure 3.14 : 3L-IBS fitting. (a) Generating inner offset (red) and outer offset (blue) for peak segment. (b) IBS fitted to the peak segment area. Voxel size = 5 mm



(a)



(b)

Figure 3.15 : 3L-IBS fitting. (a) Generating inner offset (red) and outer offset (blue) for saddle segment. (b) IBS fitted to the saddle segment area. Voxel size = 5 mm

### 3.6. Merging different surface patches to obtain a single 3D mesh

This section presents the last step to achieve a final high-order surface model. The details about the blending approach are given in section 2.6. After IBS fitting on each segment, it is proposed to merge those patches to achieve a final model. To obtain such a model, the VXelements application is used. Figure 3.16 to 3.18 show the process of blending different patches for synthetic data.

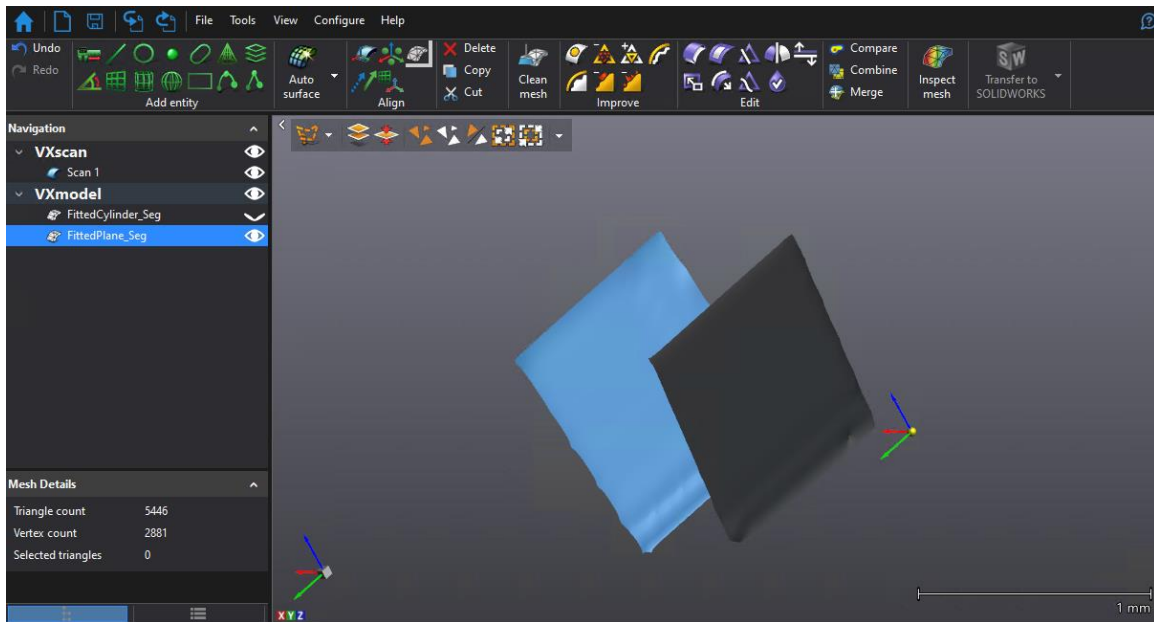


Figure 3.16: Surface representation for planar regions of the synthetic object displayed in VXelements

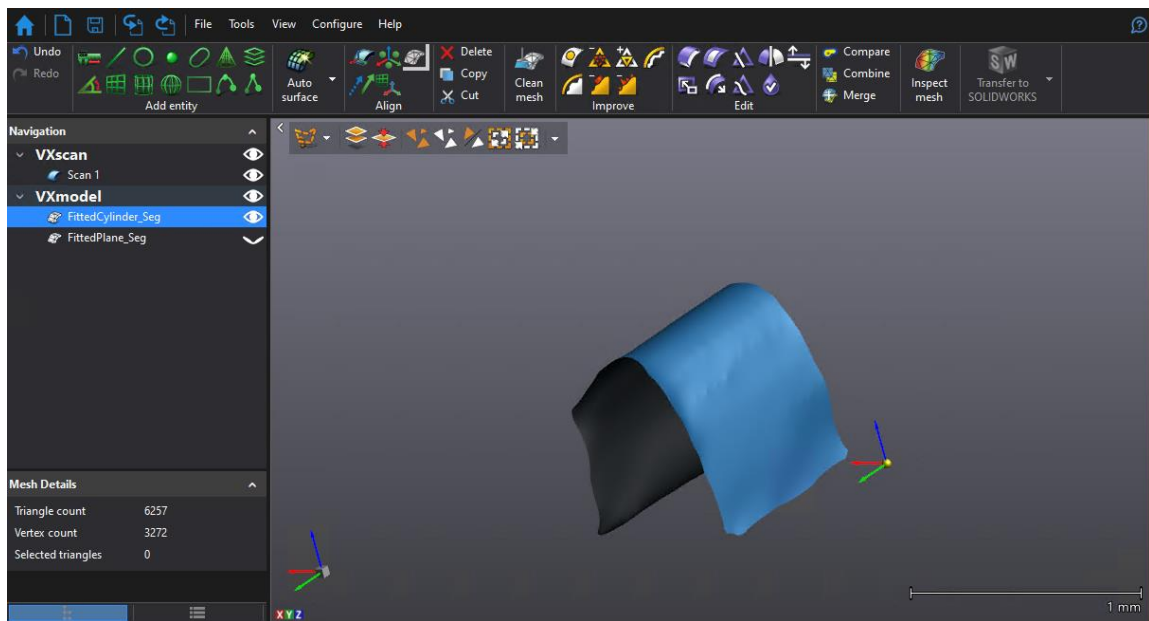


Figure 3.17: Surface representation for the cylindrical region of the synthetic object displayed in VXelements

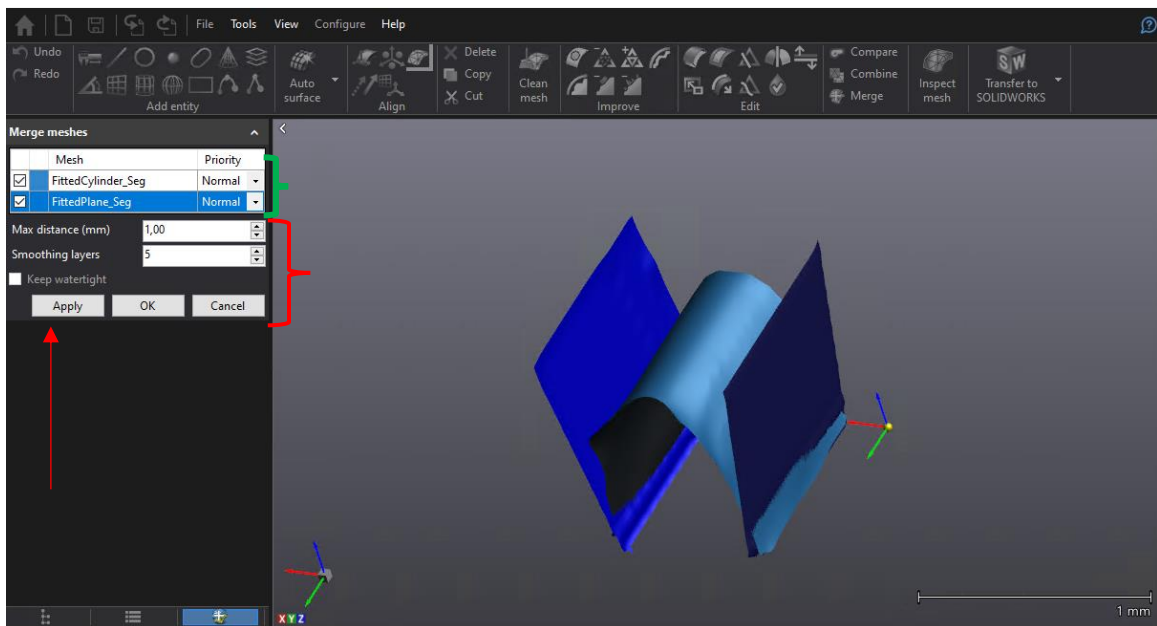


Figure 3.18: Merging the ridge and planar area obtained from the synthetic object

As indicated by the green braces in Figure 3.18, different patches are selected and then as pointed by the red braces, maximum distance and the smoothing layer for merging the patches is selected. Finally, by using the “Apply” button the different selected patches are merged together.

The “Max distance” indicates the maximum distance between the different patches that is needed for the merging model. The “Smoothing layer” is the parameter that could smooth the boundary’s vertices of each mesh.

In this example, the Max distance is chosen as to 5 mm and the smoothing layer is chosen as “2”. The spike level has been set to “25” and the smooth weight has been also set to “25”. Then these parameters are applied to the merging patches. These values may vary depending of the changes that need to be make in each mesh.

The final result of the merging patches in order to achieve a single model for the synthetic data generated in MATLAB is shown in Figure 3.19.

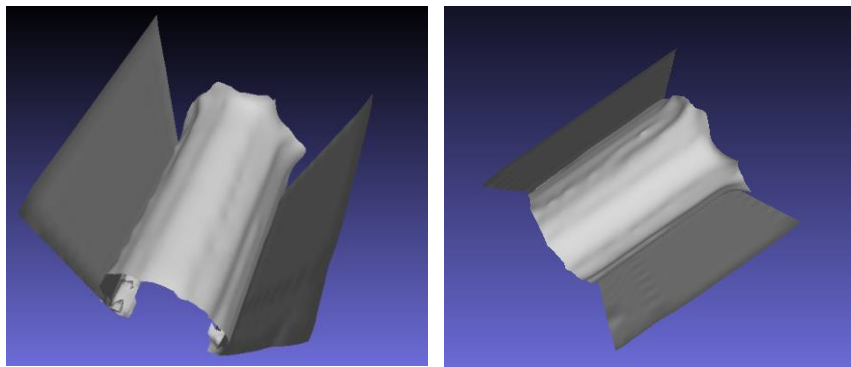


Figure 3.19 : The result of merging the different patches for the synthetic data.

The final model in Figure 3.19 is as expected. To achieve a smoother mesh, the smooth weight parameter could be increased.

Figure 3.20 (a) illustrates the results of merging four different patches to achieve a single 3D mesh of the object scanned by the HandyScan and Figure 3.20 (b) and 3.20 (c) show the single mesh achieved after merging the patches. As shown in Figure 3.12 (b) to Figure 3.15

(b), some part the regions fitted by high order surface representation are bumpy, which can be removed in the VXelements application.

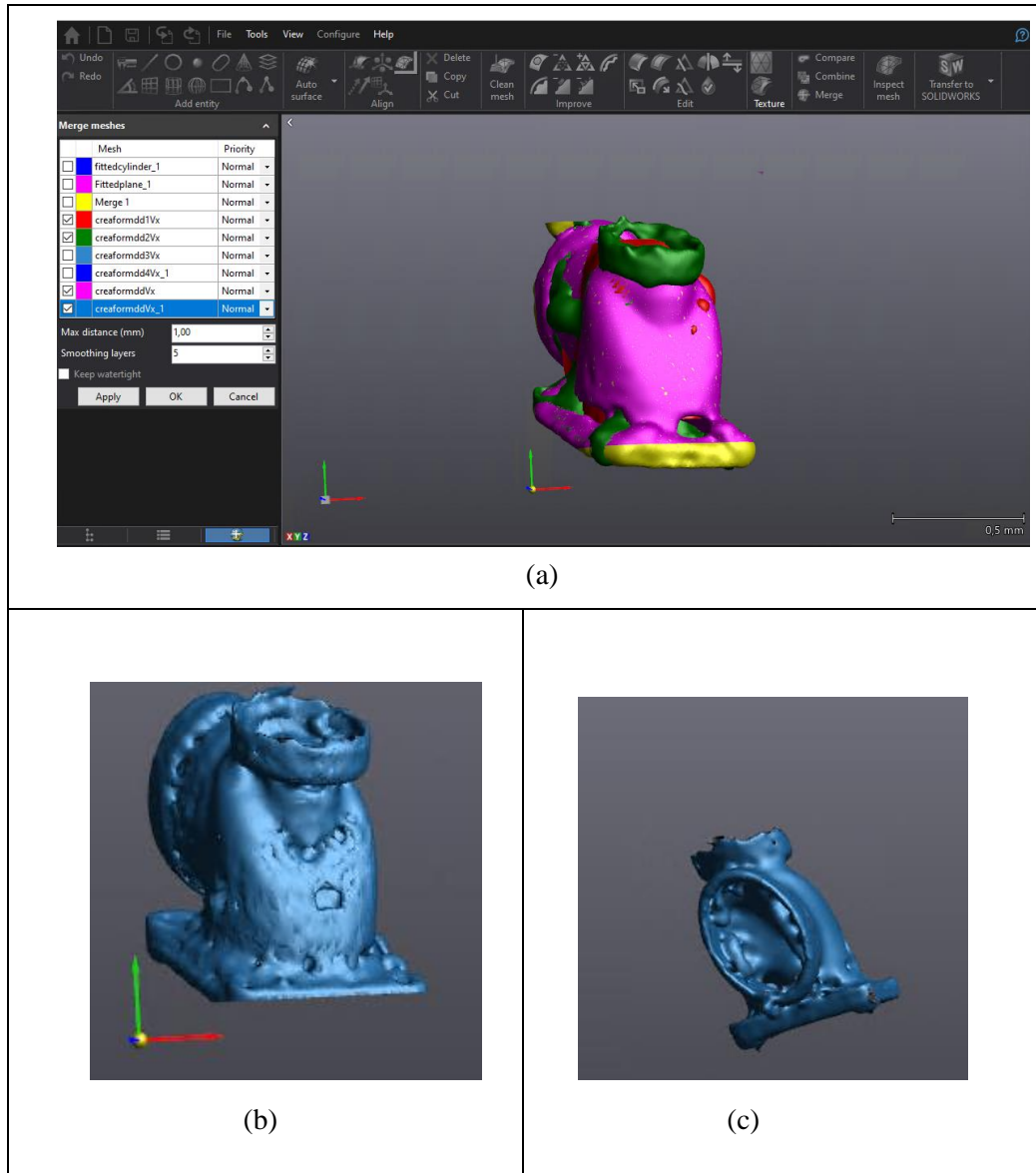


Figure 3.20: Merging patches for the 3D point collected by the HandyScan Creaform scanner. (a) Merging four different patches (b) 3D mesh achieved by merging (c) different view of the 3D mesh

### 3.7. Experimental results achieved of the implementation in C++

Due to the limitation of the memory in MATLAB, using large objects to show the results of our proposed approach was a problem. To remove the shortcomings of this problem, Christophe Bolduc has implemented our approach during a summer internship in C++. Figure 3.21 to 3.23 illustrates our approach by this C++ implementation for different large objects.

The results shown in Figure 3.21 is for a cube. It is clear that the faces of the cube are planar. The result shown in Figure 3.21 (b) is demonstrate our expectation to see the planar surface type in each face of the cubic shape. While the edges are mix of peak and ridge surface types. The resolution for the vector field is consider  $40 \times 40 \times 40$  for this object. The final model of the cube is shown in Figure 3.21 (c).



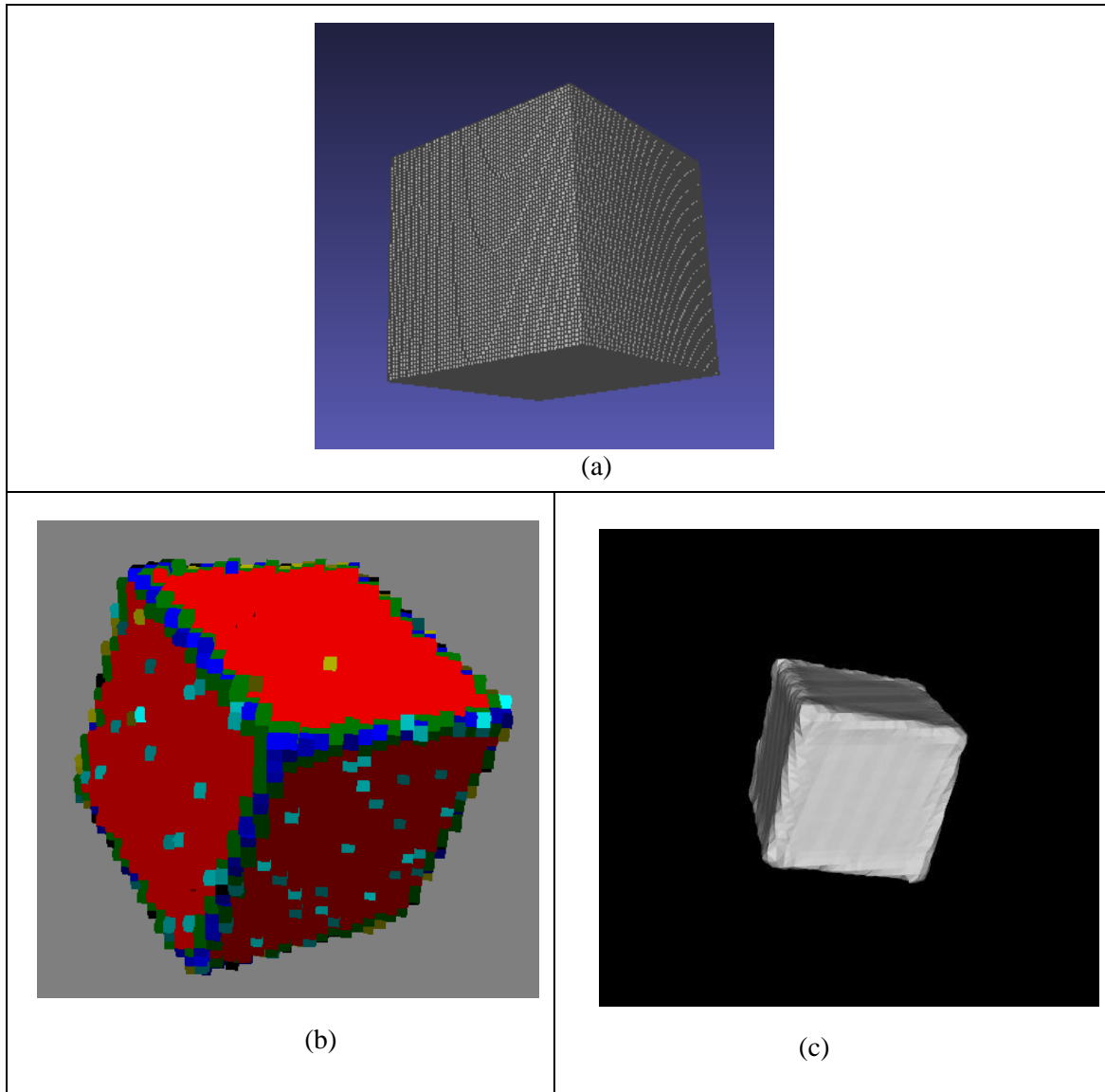


Figure 3.21: Results of the proposed methods on a cube object. (a) Object mesh data (b) Identifying different surface type (c) constructed model with  $40 \times 40 \times 40$  resolution in vector field.

The results shown in Figure 3.22 are for an electrical outlet. The area indicated by red arrows are planar and the region indicated by blue arrows are spherical. The results shown in Figure 3.22 (b) and 3.22 (d) demonstrate our approach and the results are obtained as expected. The final model of the electrical outlet with  $50 \times 50 \times 50$  resolution for vector field is shown in Figure 3.22 (e).

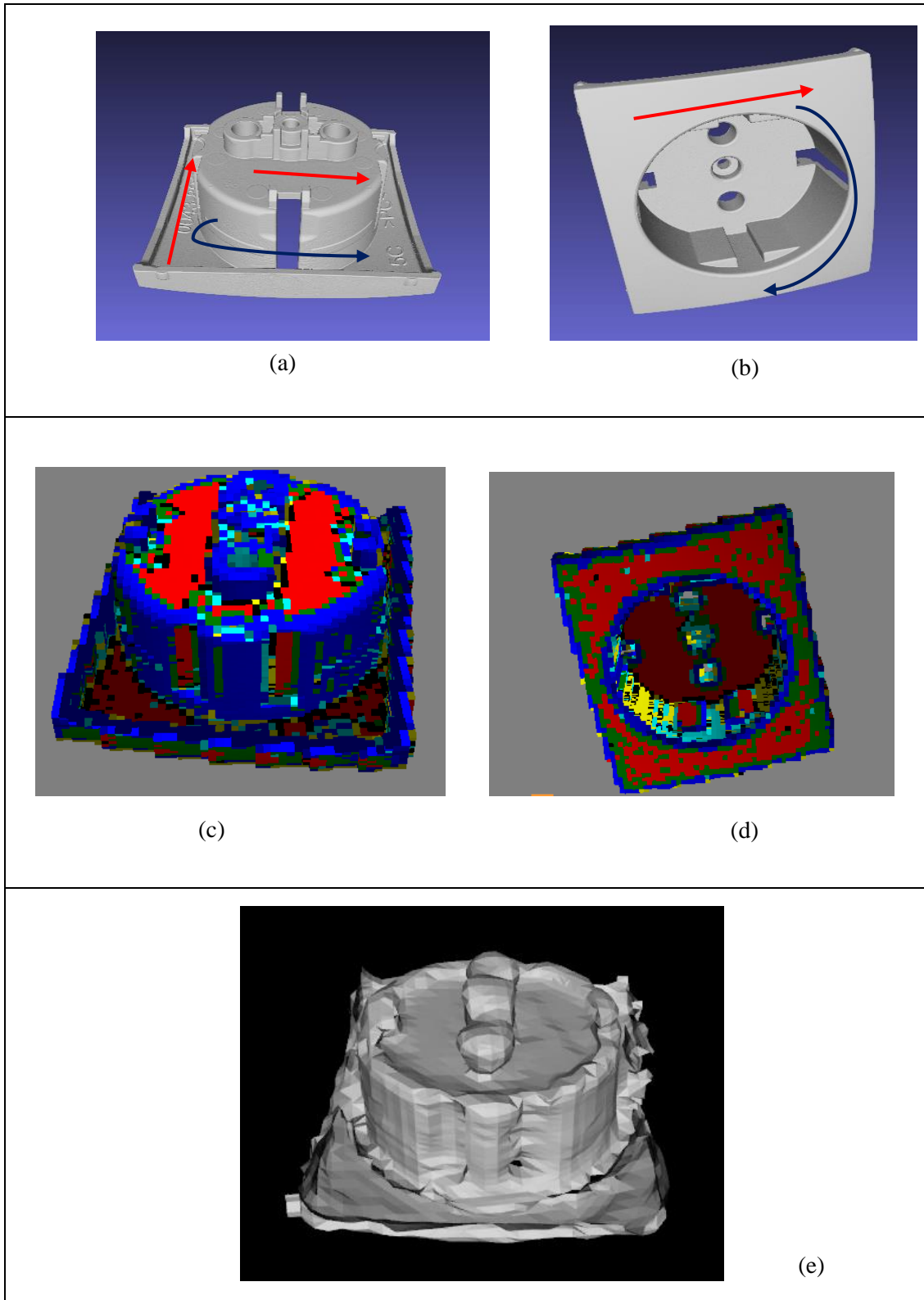


Figure 3.22: Results of the proposed methods for an electrical outlet. (a) , (b) Object mesh data (c) , (d) Identifying different surface types. (e) Constructed model with 50×50×50 resolution in vector field

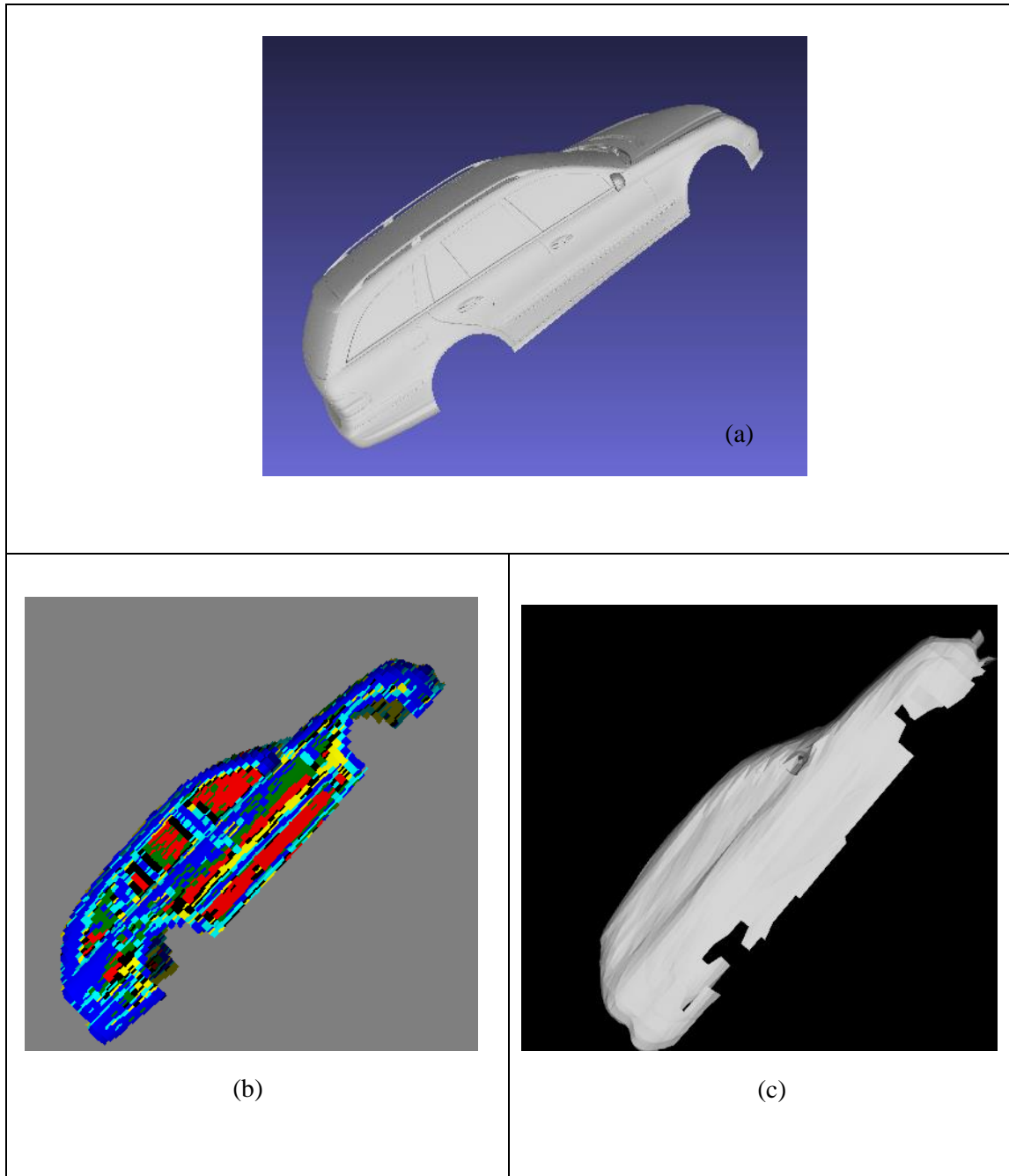


Figure 3.23: Results of the proposed methods for a car body. (a) Object mesh data (b) Identifying different surface types. (c) Constructed model with  $50 \times 50 \times 50$  resolution in vector field

Figure 3.23 illustrate the results for car body. It is clear that the car side windows, and some areas close to the car doors are basically planar and the body specially around the wheels is spherical.

Figure 3.23 (b) validate the identification surface types as expected. Planar region are shown in red, peak region are shown in blue. The final model achieved with  $50 \times 50 \times 50$  resolution in vector field is shown in Figure 3.23 (c).

### 3.8. More results for different sample objects obtained in MATLAB

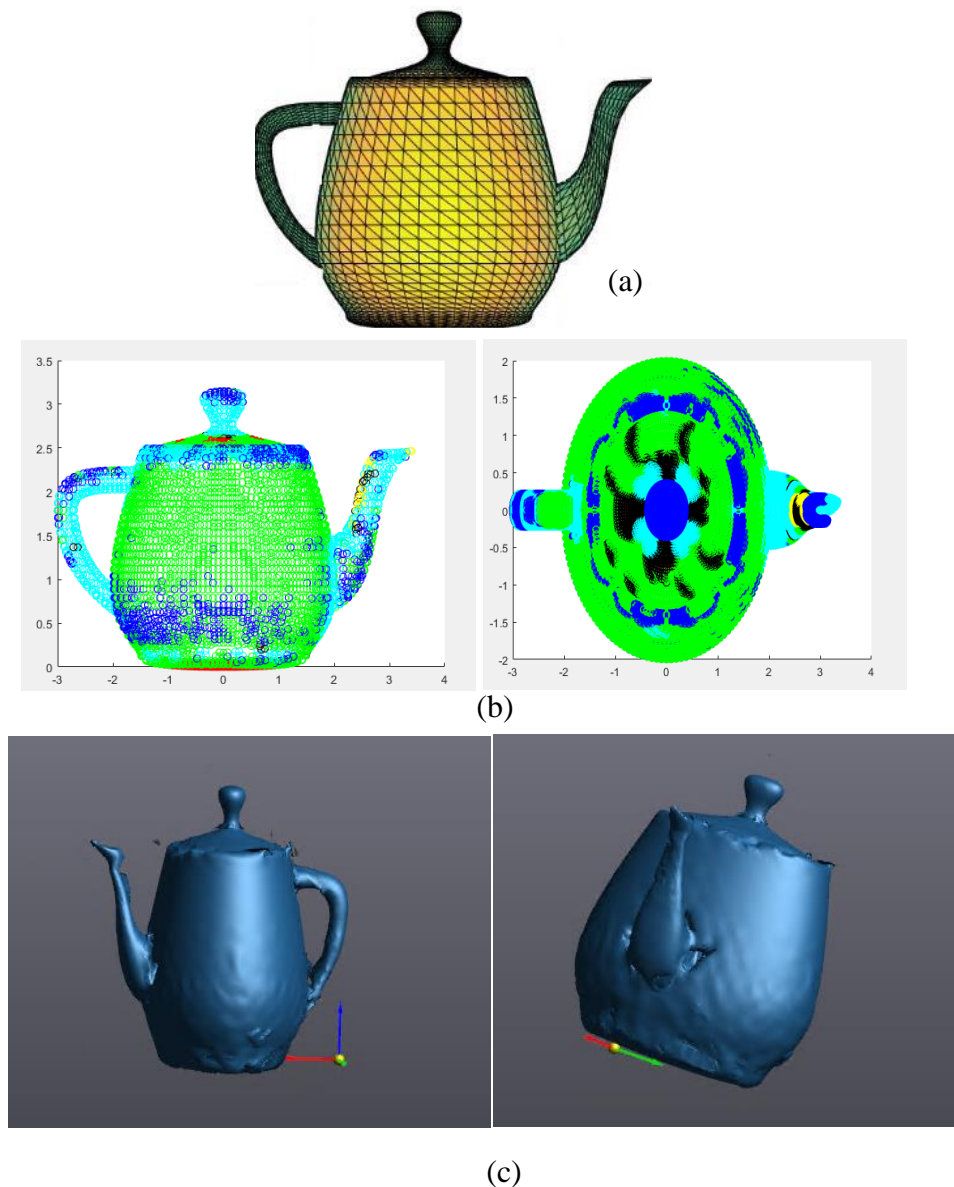
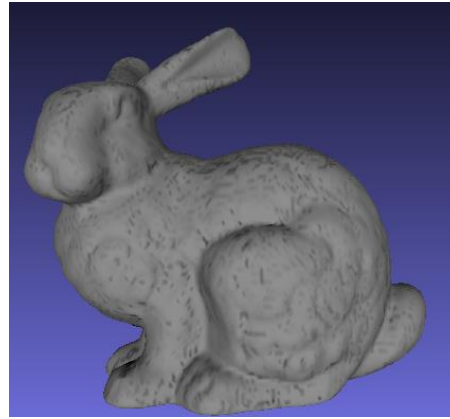
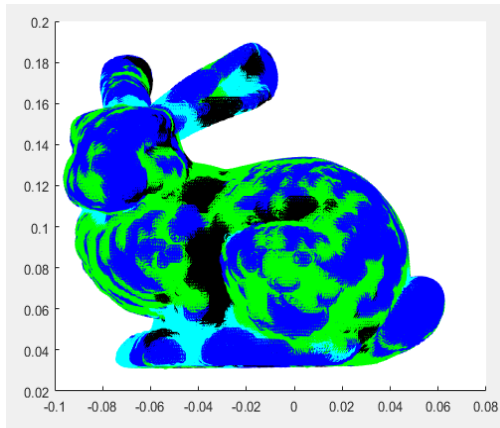


Figure 3.24 : Results of the proposed method on a teapot. (a) Object mesh data. (b) Identifying different surface types. (c) Different views of the constructed model after applying the post processing methods. Voxel size = 6mm

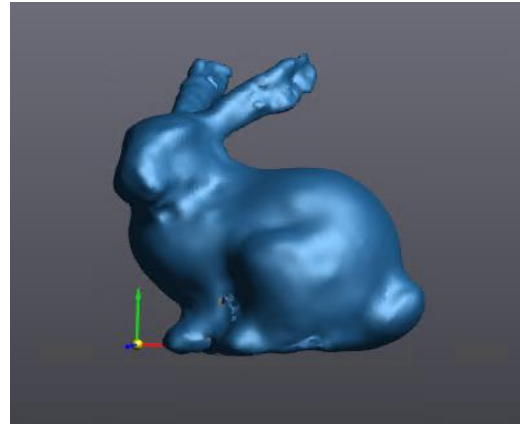
It is apparent that for the teapot that the main body is basically cylindrical. And it is also clear that the knob of the teapot is peak and the spout is a saddle surface. Figure 3.24 (a), (b) show the main body in green color, which is a ridge surface, and the knob in blue colour, which demonstrate that the region is peak as expected. Different views of the constructed model of the teapot after applying the post processing methods with 6 mm resolution is illustrated in Figure 3.24 (c).



(a)



(b)



(c)

Figure 3.25: Results of the proposed method on a Bunny. (a) Object mesh data. (b) Identifying different surface types. (c) Constructed model after applying the post processing methods. Voxel size = 6mm

It can be observed on the model data shown in Figure 3.25 (a) that the regions around the muzzle, nose, belly and tail are peak and the regions inside the ears and between the legs are saddle. The region between the leg and belly are valley. So as expected, Figure 3.25 (b) shows the region around expected peak region in blue and voxels around the expected valley region in black, and expected saddle surface in cyan. The constructed model after applying the post processing methods with the 6 mm resolution in the vector field is shown in Figure 3.25 (c).

Figure 3.26 shows the result of our approach on a pure synthetic saddle shape in voxel size equal to 6 mm. Figure 3.26 (a), illustrate identification of surface type. It can be seen that just a small amount of voxels around the edge belonging to the spherical surface types, the rest are saddle. Figure 3.26 (b) show the final model after applying the post processing approach.

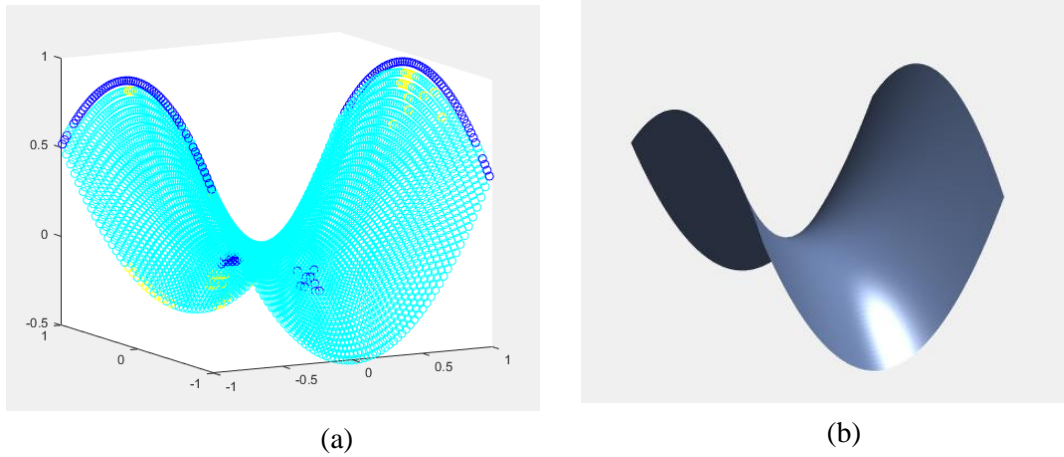


Figure 3.26 : Results of the proposed method on a synthetic saddle data. (b) Identifying different surface types. (c) Constructed model after applying the post processing methods. Voxel size = 6 mm

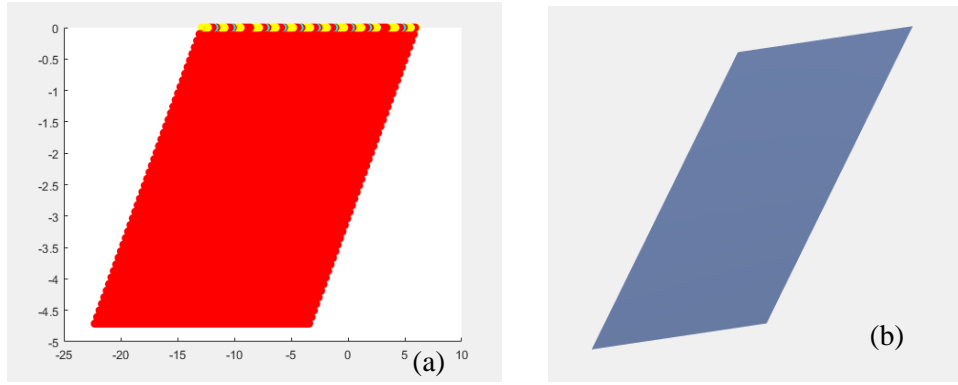


Figure 3.27 : Results of the proposed method on a synthetic plane data. (b) Identifying different surface types. (c) Constructed model. Voxel size = 6 mm

Figure 3.27 shows the result of our approach on a pure synthetic planar shape in voxel size equal to 6 mm. Figure 3.27 (a), illustrate identification of surface type. Figure 3.27 (b) shows the final model after applying the post processing approach.

# Conclusion

The recent availability of powerful 3D scanning devices has made it possible to capture very large sets of points at the surface of objects of various sizes such as man-made objects up to car-sized objects. These scanners provided 3D point cloud data for application such as computer-aided design, reverse engineering, metrology, industrial inspection and virtual reality to name a few. Transforming an unstructured set of 3D points captured by these devices into a meaningful digital representation of the scanned object is called 3D surface reconstruction.

Tubic et al. [5] proposed a novel real-time implicit representation technique called the vector field framework, in which registration, integration and visualization can be achieved in real time. A major advantage of the vector field representation is that it encodes the surface normal as well as information on the nearest neighbors in each voxel, thus enabling nearest neighbor search in linear time complexity. However, if an accurate model needs to be built, the voxel size must be small which may lead to huge memory requirements.

In this thesis, a set of approaches has been presented for the accurate reconstruction of the model of an object while keeping the memory requirements as low as possible.

The contributions of this thesis are:

- 1) An approach of the identification of six different surface types using covariance-based differential geometry applied to the vector field implicit representation.
- 2) A surface variation approach in the vector field representation to select an optimal voxel size.
- 3) A segmentation algorithm to connect the neighboring voxels belonging to the same surface type.
- 4) An implicit surface fitting approach to represent each segment by a high order surface representation



- 5) An approach to blend the different surface patches and to produce a non-overlapping mesh as a final model.

Overall, we demonstrate that it is possible to combine covariance-based differential geometry and implicit surface representation methods to perform the identification of different surface types of an unorganized point cloud (and not just a range map) into six surface types. The advantages of combining covariance-based differential geometry and implicit surface representation are that the initial segmentation does not require surface fitting, parametrization and complex calculations nor does it require that all points be processed, thus reducing computational complexity. The key point in our approach is that instead of working with points, we are working with voxels and their neighbors, which clearly reduces the computational complexity. Since the number of voxels is much smaller than the number of points which can amount to several millions.

Choosing a random voxel size in 3D volumetric representation is not optimal. The surface variation method represented in the vector field enables the objective selection of the optimal voxels size while keeping the memory consumption as low as possible and still achieve an accurate model of the surface.

The 3L-Implicit B-Splines method is adapted to represent each segment of the different surface types by a high order model. At the end, different surface patches need to be blended to achieve a single model as our final goal.

To blend the patches, VElements is adapted to finalize the blending of these patches and the construction of the final model.

By comparing our approach with an alternative approach such as quadratic fitting, differential geometry approach [31], it can be observed that since we are working with voxels instead of working with each point the computational performance is much better than the alternative approaches. Our approach can manage memory requirements by choosing an optimal voxel size for the vector field and reduces the computational complexity because it can segment an unorganized set of points without any surface fitting and the estimation of derivatives. The vector field approach combined with the marching cubes requires very small voxels since the

planar approximation must be satisfied. In contrast, since a higher-order surface fitting is used, there is no need for satisfying the planar approximation everywhere on the surface.

The acquisition of the point cloud data has been achieved with handheld scanners used in metrology applications. Our approach has been validated on synthetic data as well as point clouds borrowed from common datasets. Scans obtained from commercial metrologic handheld 3D sensors are also used for validation.

For future work, to improve the algorithm for identifying different surface types, an approach to find the eight different surface types instead of six different surface types without requiring parametrization and surface fitting could be implemented. By using the eight different surface types the accuracy of the reconstructed model could be improved. Moreover, the quality of the fitting could also be improved.

# Bibliography

- [1] N. Amenta, M. Bern and M. Kamvysselis, "A new Voronoi-based surface reconstruction algorithm," in *In Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998.
- [2] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE transactions on visualization and computer graphics*, vol. 5, no. 4, pp. 349-359, 1999.
- [3] R. Mencl, "A graph-based approach to surface reconstruction," *In Computer Graphics Forum*, vol. 14, no. 3, pp. 445-456, 1995.
- [4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Surface reconstruction from unorganized points," *ACM*, vol. 26, no. 2, pp. 71-78, 1992.
- [5] D. Tubic, P. Hebert and D. Laurendeau, "A volumetric approach for interactive 3d modeling," *In Proceedings of First International Symposium on 3D data Processing Visualization and Transmission (3DPVT)*, vol. 1, pp. 150-158, 2002.
- [6] P. Turaga, A. Sirastava, S. Joshi and B. Ben, "Differential Geometry in Computer Vision and Machine Learning," [Online]. Available: <https://www.frontiersin.org/research-topics/17080/differential-geometry-in-computer-vision-and-machine-learning>.
- [7] Chaudhuri, "Digital geometry processing," [https://www.cse.iitb.ac.in/~cs749/spr2016/handouts/vas\\_surf2.pdf](https://www.cse.iitb.ac.in/~cs749/spr2016/handouts/vas_surf2.pdf), 2016.
- [8] P. Seidel, "Differential geometry," 2008. [Online]. Available: [https://ocw.mit.edu/courses/mathematics/18-950-differential-geometry-fall-2008/lecture-notes/ch2\\_revised.pdf](https://ocw.mit.edu/courses/mathematics/18-950-differential-geometry-fall-2008/lecture-notes/ch2_revised.pdf).
- [9] M. Najafkhah, *Applied differential geometry*, Tehran: Tehran: Iran University of science and technology, 2013.
- [10] M. Powell, "Introduction to differentiable geometry," 2014. [Online]. Available: <http://www.math.uqam.ca/~powell/M435-chapter-4-2nd-FF-curvature.pdf>.
- [11] P. Besl and R. Jain, "Invariant surface characteristics for 3D object recognition in range images," *Computer vision, graphics, and image processing*, no. 33(1), pp. 33-80, 1986.
- [12] S. Verpoort, *The geometry of the second fundamental form: curvature properties and variational aspects*, 2008.

- [13] Rusinkiewicz, "Estimating curvatures and their derivatives on triangle meshes.," *3DPTV* , vol. Proceedings. 2nd International Symposium , pp. 486-493, 2004.
- [14] Manfredo , P. Do Carmo, Differential geometry of curves and surfaces, 1976.
- [15] E. W. Weisstein, "Principal Curvatures," [Online]. Available: From MathWorld--A Wolfram Web Resource. <https://mathworld.wolfram.com/PrincipalCurvatures.html>.
- [16] R. Lisle, N. Toimil, J. Aller and A. Bobillo, "The hinge lines of non-cylindrical folds," *Journal of Structural Geology*, vol. 32, pp. 166-171, 2010.
- [17] E. W. Weisstein, "Gaussian Curvature," From MathWorld--A Wolfram Web Resource. <https://mathworld.wolfram.com/GaussianCurvature.html>.
- [18] O. Sorkine, "Smoothing parametrization remeshing," 2013. [Online]. Available: <http://slideplayer.com/slide/3362221/>.
- [19] B. Bulca and K. Arslan, "Surfaces given with the Monge patch in  $E^4$ ," in *arXiv preprint arXiv:1305.4373*., 2013.
- [20] A. Khatamian and H. Arabnia, "Survey on 3D Surface Reconstruction," *Journal of Information Processing Systems*, no. 12(3), 2016.
- [21] M. Botsch, M. Pauly, C. Ross, S. Bischoff and L. Kobbelt, "Geometric modeling based on triangle meshes," in *ACM SIGGRAPH 2006 Courses (p. 1)*. ACM., 2006.
- [22] C. Shene, "Introduction to Computing with Geometry Notes," Michigan Technological University..
- [23] S. ILIC, "Implicit meshes: Unifying implicit and explicit surface representation for 3D reconstruction and tracking(Doctoral dissertation, PhD Thesis)," ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2005.
- [24] D. Tubic, "On Surface Representation in 3D Modelling: a framework for interactive 3D real-time modelling," PhD thesis, Laval University, 2006.
- [25] T. Brochu, "Dynamic explicit surface meshes and applications," Doctoral dissertation, PhD Thesis. University of British Columbia, 2012.
- [26] T. Masuda, "Object shape modelling from multiple range images by matching signed distance fields," In *3D Data Processing Visualization and Transmission, First International Symposium*, pp. 439-448, 2002.
- [27] J. Salvi, C. Matabosch, D. Fofi and J. Forest, "A review of recent range image registration methods with accuracy evaluation," *Image and Vision computing*, no. 25(5), pp. 578-596, 2007.
- [28] S. Hesabi, "3D modeling of large elongated structures for Non-Destructive Testing and inspection," *Doctoral dissertation, Université Laval*, 2017.

- [29] V. T. Nguyen, "Exploitation of the Vector Field Representation for Pose Estimation, View Integration and 3D Modeling Using Natural Features," Doctoral dissertation, Université Laval, 2015.
- [30] M. Eskandari and D. Laurendeau, "Covariance Based Differential Geometry Segmentation Techniques for Surface Representation Using Vector Field Framework," in WSCG, 2020.
- [31] P. Besl and R. Jain, "Segmentation Through Variable-Order Surface Fitting," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, no. 2, pp. 167-192, 1988.
- [32] A. Presley, "Elementary Differential Geometry," p. 468, 2010.
- [33] J. Berkman and T. Caelli, "Computation of surface geometry and segmentation using covariance techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 16(11), pp. 1114-1116, 1994.
- [34] P. Liang and J. Todhunter, "Representation and recognition of surface shapes in range images," *Computer Vision, Graphics and Image Processing*, vol. 52, no. 10, pp. 78-109, 1990.
- [35] J. Digne and J. Morel, "Numerical Analysis of Differential Operators on Raw Point Clouds," *Numerische Mathematik*, pp. 127:255-289, 2014.
- [36] Y. Zhang, J. Cao, Z. Chen and X. Zeng, "B-spline surface fitting with knot position optimization," *Computers & Graphics*, vol. 58, pp. pp.73-83, 2016.
- [37] V. Weiss, L. Andor, G. Renner and T. Varady, "Advanced surface fitting techniques," *Computer Aided Geometric Design*, vol. 19, no. 1, pp. 19-42, 2002.
- [38] L. Piegl and W. Tiller, *The NURBS book*, Berlin: Heidelberg:SpringerScience&Business Media, 2012.
- [39] J. Carr, R. Beaton, J. Cherrie, T. Mirchell, W. Fright, B. McCallum and T. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.
- [40] Y. He and H. Qin, "Surface reconstruction with triangular B-splines," in *In:Proceedings ofgeometric modeling and processing, IEEE*, 2004.
- [41] Y. Tan, J. Hua and H. Qin, "Physically based modeling and simulation with dynamic spherical volumetric simplex splines," 2010.
- [42] M. Rouhani and A. Sappa, "Implicit B-spline fitting using the 3L algorithm," in *In 2011 18th IEEE International Conference on Image Processing (pp. 893-896). IEEE.*, 2011.
- [43] M. Blane, Z. Lei, H. Civi and D. Cooper, "The 3L algorithm for fitting implicit polynomial curves and surfaces to data," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

- [44] M. Rouhani, A. Sappa and E. Boyer, "Implicit B-spline surface reconstruction," in *IEEE Transactions on Image Processing*, 24(1), pp.22-32, 2014.
- [45] M. Blane, Z. Lei and D. Cooper, "The 3L Algorithm for Fitting Implicit Polynomial Curves and Surfaces to Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (under review), 1996.
- [46] M. Rouhani, "Shape representation and registration using implicit functions," Universitat Autònoma de Barcelona, 2014.
- [47] C. Hoffmann and J. Hopcroft, "The geometry of projective blending surfaces," *Artificial Intelligence*, Vols. 37(1-3), pp. 357-376, 1988.
- [48] D. DeCarlo and D. Metaxas, "Blended deformable models.," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 443-448, 1996.
- [49] S. Bela and M. Szilvási-Nagy, "Merging B-Sspline Curves or Surfaces Using Matrix Representation," *Journal of Computer Graphics Techniques* , vol. 6, no. 3, 2017.
- [50] H. Pungorta, G. Knopf and R. Canas, "Merging multiple B-spline surface patches in a virtual reality environment," *Computer-Aided Design*, vol. 42, no. 10, pp. 847-859, 2010.
- [51] C. Tai, S. Hu and Q. Huang, "Approximate merging of B-spline curves via knot adjustment and constrained optimization," *Computer-Aided Design*, vol. 35, no. 10, pp. 893-899, 2003.
- [52] J. Chen and G. Wang, "Approximate merging of B-spline curves and surfaces," *Applied Mathematics-A Journal of Chinese Universities*, vol. 25, no. 4, pp. 429-436, 2010.
- [53] D. Cohen-Steiner and F. Da, "A greedy Delaunay-based surface reconstruction algorithm," *The visual computer*, vol. 20, no. 1, pp. 4-16, 2004.
- [54] G. Turk and M. Levoy, "Zippered polygon meshes from range images," *In Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 311-318, 1994.
- [55] M. Daneshmand, A. Helmi, E. Avots, F. Noroozi, F. Alisanoglu, H. Sait Arslan, J. Gorbova, R. Haamer, C. Ozcinar and G. Anbarjafari, "3D Scanning: A Comprehensive Survey," *arXiv:1801.08863v1*, 2018.
- [56] S. Cheng, T. Dey and J. Shewchuk, "Delauney Mesh Generation," *Chapman & Hall / CRC computer and information science series*, p. 386, 2012.
- [57] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303-312, 1996.
- [58] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *SIGGRAPH '87 Conference Proceedings*, vol. 21, no. 4, pp. 163-169, 1987.
- [59] Patrikalakis-Maekawa, "Second fundamental form," MIT, 2009.

- [60] M. Pauly, M. Gross and L. Kobbelt, "Efficient simplification of point-sampled surfaces," *In IEEE Visualization.*, pp. 163-170, 2002.
- [61] C. Zhang and B. Mao, "3D building models segmentation based on K-means++ cluster analysis," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 57, 2016.
- [62] R. Gove, "Using the elbow method to determine the optimal number of clusters for k-means clustering," <https://bl.ocks.org/rpgove/0060ff3b656618e9136b>, 2017.
- [63] "Yellowbrick," Yellowbrick: Machine Learning Visualization, 2019. [Online]. Available: <https://www.scikit-yb.org/en/latest/api/cluster/elbow.html>.
- [64] C. Sahin, R. Kouskouridas and T. Kim, "Iterative hough forest with histogram of control points for 6 dof object registration from depth images," in *In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.
- [65] Z. Yang, J. Deng and F. Chen, "Fitting unorganized point clouds with active implicit b-spline curves," *The Visual Computer*, vol. 21, no. 8, pp. 831-839, 2005.
- [66] C. Sahin, R. Kouskouridas and T. Kim, "A learning-based variable size part extraction architecture for 6D object pose recovery in depth," arXiv preprint arXiv:1701.02166., 2017.
- [67] A. Hilton and J. Illingworth, "Geometric fusion for a hand-held 3d sensor," *Machine vision and applications*, vol. 12, no. 1, pp. 44-51, 2000.
- [68] A. Hilton, A. Stoddart, J. Illingworth and T. Winder, "Reliable surface reconstruction from multiple range images. In European conference on computer vision," *Springer*, pp. 117-126, 2006.
- [69] E. W. Weisstein, "Mean curvature," From MathWorld--A Wolfram Web Resource. <https://mathworld.wolfram.com/MeanCurvature.html>, n.d..
- [70] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on visualization and computer graphics*, vol. 9, no. 1, pp. 3-15, 2003.