Parametric curve and Partial Differential Equation-based parametric surface reconstruction from point clouds

by

Zaiping Zhu

National Centre for Computer Animation

Faculty of Media & Communication

Bournemouth University

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of *Doctor of Philosophy*

March. 2025

Copyright Statement

Acknowledgements

First of all, I sincerely express my thanks and gratefulness to my supervisors, Prof. Lihua You and Prof. Jianjun Zhang, for their valuable and continuous guidance and support during the past four years of my PhD research. I have learned a lot by communicating with my supervisors, especially Prof. You, from whom I have learned a lot concerning how to do academic research. I would also thank Prof. Iglesias from the University of Cantabria, who also gave me useful advice on my research.

Besides, I would thank my friends for their help and company during my time at Bournemouth University. Even though I worked from home for almost 2 years because of the impact of the Pandemic, I still made a lot of friends in the UK. I would give my thanks to Junheng Fang, Zhiqi Li, Mengqin Huang, Xiaoxiao Liu, Xingyu Ye, Tingting Li and Yingjie Xi, Jiajun Huang for their help. I would also thank Sasha and Kavi for the interesting discussion.

I would also say thanks to my family. Without their support and love, it would be hard for me to do my research during the pandemic. My fiancee Liqi Zhou also supports me concerning my research and choices. By talking to her every day, my stress was relieved to some extent.

Finally, I would like to give my thanks to my country and the China Scholarship Council for their support during the time of pursuing my PhD degree.

Abstract

There are two primary types of representations used to model curves and 3D surfaces in the digital world: explicit and implicit. Explicit representations include parametric representations, polygons, and similar methods, while implicit representations encompass level sets, distance functions, and Constructive Solid Geometry (CSG), among others. Parametric models are especially popular in 3D modelling, design, and manufacturing because they are mathematically defined, making them easy to edit, transmit, and construct. However, real-world data often exists in a discrete form, typically as point sets, rather than in a parametric form. Therefore, it is crucial to reconstruct a parametric representation that closely approximates real-world data.

In the context of parametric curve reconstruction from point sets, two key objectives are to approximate the underlying structure of the point sets as accurately as possible while minimising the number of curves used. Existing methods often struggle to balance these two factors effectively.

To address this gap, we have developed a method that mimics the process of human vectorization of image boundaries. The boundary points are first segmented into multiple segments using corner points. Within each segment, the bisection method is employed to identify the largest subset of points that a single curve can fit. Additional curves are introduced only when the fitting error exceeds a predefined threshold. This process continues until all points in the segment are fitted, thereby minimising the number of Bézier curves required. Additionally, symmetric

shape boundaries within the point sets are detected, further reducing the number of curves needed. My method also allows for the selection of the optimal parameterization method on a case-by-case basis to minimise fitting error, which is a critical step in parametric curve reconstruction. Comparisons with both contemporary and classical methods demonstrate that my approach outperforms existing techniques.

For parametric 3D surface reconstruction from point sets, Bézier, B-spline, and NURBS surfaces have been extensively studied. However, these methods share common drawbacks, such as the need for large data storage and complex geometry processing. Moreover, maintaining good continuity between reconstructed parametric patches is often challenging. In contrast, PDE-based methods for surface reconstruction are advantageous due to their low storage requirement, strong fitting capabilities and the relative ease of achieving good continuity, as most are boundary-based approaches. The primary challenge with PDE-based surface reconstruction methods lies in solving partial differential equations, which is why most studies have focused on implicit PDE-based shape reconstruction, despite its computational expense.

To address these challenges, we propose a novel method that uses an accurate closed-form solution to a fourth-order PDE for reconstructing 3D parametric surfaces from point clouds. This method offers powerful fitting capabilities and is computationally efficient. However, postprocessing is necessary to ensure continuity, as this is not inherently guaranteed in the model. Additionally, the parameterization of point sets in the initial method is not sufficiently effective. To overcome these limitations, my subsequent work integrates linearly blended Coons patches with an analytical solution of a specific fourth-order PDE. This combined approach not only ensures good positional continuity between reconstructed

parametric patches but also uses the Coons patch as an effective tool for accurate parameterization of the point sets. Lastly, tangential continuity is achieved by combining the bicubic Coons patch and a special deformation surface.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Parametric curves and surface reconstruction from point sets are the processes of obtaining the explicit mathematical expression that approximates the underlying structure of the point sets as closely as possible while keeping low storage and good continuity conditions. This important research direction has been widely used in many domains such as reverse engineering, cultural heritage, robotics, biomedical engineering and many others (Varady et al. 1997, Raja and Fernandes 2007, Gomes et al. 2014). To be more specific, the parametric form of the data enables the data to be stored, edited, and used for manufacturing more effectively and efficiently.

The general pipeline of reconstructing parametric curves is shown in Fig. 1.1: The first step is data acquisition, which can be done in many ways, such as 3D scanner, point cloud reconstruction from multi-view images or even a single image. Next, the point sets are preprocessed to remove outliers or add points to complete the data. Then the data is segmented into multiple segments based on the corner points, each of which is reconstructed with a parametric curve or many curves if necessary. Fourthly, the point sets need to be parameterized in a suitable way to obtain the corresponding parameters for each point, and the parameters normally range from 0 to 1. In terms of parameterizing the points, some classical methods include uniform parameterization, chord length method (Farin 2014), centripetal (Piegl and Tiller

2012, Fang and Hung 2013) and hybrid (Shamsuddin and Ahmed 2004), etc. We have to note that each parameterization method has its pros and cons, no method can handle all the problems of curve fitting, and it is not easy to choose a suitable method for the task at hand. Fifthly, fitting of parametric curves or parametric surfaces. In this step, choosing a norm which will be used to measure the fitting errors also matters. Some widely used norms include the L2 norm and L1 norm, other norms have also been investigated, but optimisation with them is relatively difficult. The procedure of recon-

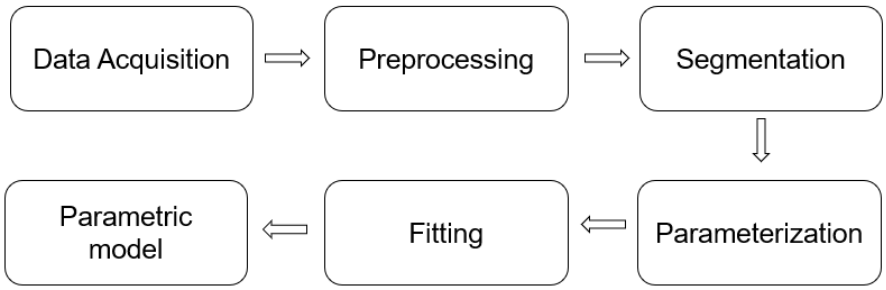


Figure 1.1: General pipeline of parametric curve/surface reconstruction from point clouds.

structing parametric surfaces from point sets is similar to parametric curves reconstruction, but parametric surface reconstruction is a more complex task since parametric surfaces are 2-dimensional objects while parametric curves are 1-dimensional objects, so more consideration should be taken into account. For example, parameterizing the point sets becomes more difficult as there are two parameters associated with each point in the surface case; Furthermore, multiple parametric patches are also required when the point sets are topologically complex, then how to keep good continuity between adjacent patches is also not easy.

The widely used parametric representations for both curves and surfaces are Bézier curves/surfaces, B-spline curves/surfaces, and NURBS curves/surfaces, they are widely used in many areas such as computer-aided design (CAD), computer-aided manufacturing (CAM) and computer-aided engineering (CAE) because of their good properties, like intuitive to edit, powerful modelling capability, etc. Partial differential equation (PDE) based methods have also

been proposed for surface reconstruction because of their advantages, which include fewer design variables, and avoidance of stitching adjacent patches together to achieve required continuity and physics-based nature. Although existing methods based on these representations make great contributions to the area of parametric curves and parametric surface reconstruction, the challenges highlighted in Section 1.2 have not been addressed.

1.2 Main Challenge

For parametric curve reconstruction from point sets, the challenges are related to approximating the underlying structure of the points as closely as possible while using as few design variables as possible; concerning parametric surface reconstruction, its first challenge is similar to that of parametric curve reconstruction. Furthermore, maintaining good continuity between reconstructed adjacent patches of parametric form is not easy; next, effectively and efficiently parametrizing the point sets also requires sophisticated techniques. These challenges will be presented in more detail below.

1. For parametric curve reconstruction, using as few curves as possible while tightly approximating the underlying structure of the point sets is not an easy task, which can both meet the reconstruction accuracy and keep low storage. Besides, the appropriate way of parameterizing the given point sets has a great impact on the final results, thus choosing the suitable way of parameterizing the point sets is also critical, but there is no best way of doing this, and it is case-dependent. Lastly, to further reduce the number of curves required, some other properties may be taken advantage of, but this requires serious consideration. However, existing methods do not behave well considering both keeping tight approximations and using a smaller number of design variables (or curves).

2. For parametric surface reconstruction, it's also desirable to use as few patches as possible(or the same number of design variables has the more

powerful capability for reconstruction); Besides, keeping positional continuity or even high-order continuity is important, as no post-processing is required and such property is essential in some applications. Lastly, parameterizing point sets becomes harder and should be handled with more consideration. Nevertheless, existing methods do not give good results in these respects.

3. Partial differential equation-based shape reconstruction has the following advantages. First, a single PDE surface patch can describe a complicated shape leading to smaller data than other representations such as the NURBS surface. Second, any irregular boundaries can be quickly specified by drawing a closed curve on 3D models, different sculpting forces can be applied to achieve the expected shape, and global shape deformation can also be achieved through shape control parameters. However, the main difficulty for PDE-based shape reconstruction is how to solve them. Due to this difficulty, most studies investigated implicit PDE-based shape reconstruction which involves numerically solving PDEs. Although some research studies investigated explicit PDE-based shape reconstruction by interpolating four curves or satisfying the constraints on two opposite boundaries of a PDE patch. They do not deal with point sets data, which is normally unorganised and harder to process. Furthermore, few studies presented closed-form solutions of a PDE for 4-sided PDE patches, which is very common in 3D modelling and reconstruction.

1.3 Aims and Objectives

This thesis proposes methods for parametric curves and surface reconstruction from point sets, aiming to tackle the aforementioned challenges. Specifically, to reconstruct parametric curves and surfaces from point sets with fewer design variables while keeping a good approximation to the data. Furthermore, suitable ways of parameterizing the point sets should be explored to reduce the fitting error and design variables. Next, analytical solutions

to PDEs should be investigated for parametric surface reconstruction, which can reduce design variables and keep good continuity. Based on our aims, we identify our research questions as follows:

- **How to reduce the number of design variables of parametric curve reconstruction while keeping good approximation?** There are many potential factors which can affect the number of curves required. Finding out ways that can achieve this target requires a lot of consideration, especially when the number of curves is already relatively small, finding methods to further reduce it is not an easy task.

- **How to apply analytical solutions of a PDE for parametric surface reconstruction?** Analytical solutions of a PDE are accurate and computationally efficient, they also have the powerful capability to represent complex shapes. Finding the suitable form of PDE and solving it analytically for parametric surface reconstruction requires considerable thinking as few works applied analytical PDE solutions for this task.

- **How to keep good continuity between the reconstructed adjacent patches of parametric form?** For parametric surface reconstruction from point sets, the point sets are segmented into multiple regions, and each region is reconstructed with a single parametric surface. How to make sure no gap or overlap between adjacent parametric patches, which is a desirable property, is challenging, and besides, effectively parameterizing the points in each region is also a difficult task.

Based on the above questions, this thesis aims to achieve three objectives:

- The first objective is to take advantage of all possible methods for parametric curve reconstruction from point sets, aiming to reduce the number of design variables while keeping a tight approximation to the data, which is presented in Chapter 3.

- The second objective is to propose some PDEs and solve them analytically, and the analytical solutions will be used for parametric surface

reconstruction from point sets, which will be demonstrated in Chapter 4.

- The third objective is to come up with new methods for parametric surface reconstruction from point sets with good continuity between adjacent patches; besides, parameterizing the point sets effectively and efficiently should also be investigated. These objectives will be presented in Chapter 5 and 6.

1.4 Contribution

Aiming to overcome the research challenges and realise the research objectives, this thesis has made the following contributions:

- Bisection and symmetric shape detection techniques are adopted to reduce the number of parametric curves while approximating the underlying structure of the point sets in a tight way.

- Analytical solution to a chosen PDE is obtained and used for parametric surface reconstruction, which shows the great capability of fitting to points data with different levels of complexities; Furthermore, we also extend the parametric solutions to the form with more design variables, thus make it more powerful and flexible concerning surface fitting.

- As no work summarised the parametrization methods used for point sets parametrization, which is a very important step in parametric curve and surface reconstruction from point clouds. We summarise and compare the most used methods for this task, which can be used as a reference for choosing a suitable parametrization method for the task at hand.

- Method of reconstructing parametric surface from point sets is proposed with the property of positional continuity, which is achieved by combining the bilinear blended Coons patch with a specific form solution of a fourth-order PDE. Furthermore, effective and efficient parameterization of the points is also the byproduct of the proposed method.

- Method of reconstructing parametric surface from point sets is proposed with the property of tangential continuity, which is achieved by combining the bicubic blended Coons patch with a specific form of deformation surface.

1.5 List of Publications

- **Zhu, Z.**, Zheng, A., Iglesias, A., Wang, S., Xia, Y., Chaudhry, E., & Zhang, J. (2022). PDE patch-based surface reconstruction from point clouds. Journal of Computational Science, 61, 101647.

- **Zhu, Z.**, Iglesias, A., Zhou, L., You, L., & Zhang, J. (2022). PDE-based 3D surface reconstruction from multi-view 2D images. Mathematics, 10(4), 542.

- **Zhu, Z.**, Iglesias, A., You, L., & Zhang, J. J. (2022, June). A review of 3D point clouds parameterization methods. In International Conference on Computational Science (pp. 690-703). Cham: Springer International Publishing.

- **Zhu, Z.**, You, L., & Zhang, J. J. (2023). Vectorizing binary image boundaries with symmetric shape detection, bisection and optimal parameterization. Computer Animation and Virtual Worlds, 34(3-4), e2191.

- **Zhu, Z.**, You, L., Wang, S., & Zhang, J.J. (2024). Parametric surface reconstruction from 3D point data using Partial differential equation and bilinearly blended Coons. Journal of Computational Physics.

1.6 Outline of Thesis

This thesis is organised as follows:

- Chapter 2 reviews related work on parametric curves and surface reconstruction from point clouds. From points data segmentation, parameterization, and fitting. PDE techniques used in this area are also discussed and the motivation is given for our work.

- Chapter 3 discusses parametric curve reconstruction from point sets by using bisection and symmetric shape detection techniques, various parameterization methods are also used and compared.

- Chapter 4 obtains analytical solutions of a PDE, uses them for parametric surface reconstruction from point sets, and the solutions are also extended to make them more powerful and flexible.

- Chapter 5 presents the proposed method aiming to reconstruct parametric surfaces from point sets with positional continuity, and parameterization of point sets is also incorporated. Followed by this, method to reconstruct parametric surfaces from point sets with C^1 continuity is also presented.

- Chapter 6 makes the conclusion and also gives some future directions that are worth exploring.

Chapter 2

Literature Review

Parametric curve and surface reconstruction from point clouds are generally composed of the following steps: first of all, the points set is preprocessed to remove the outliers or noise. Then, the filtered points are parameterized using some parameterization methods to obtain the corresponding parameters for each point. Next, some parametric form is chosen and fit to the points. Finally, postprocessing may be necessary to remove gaps or overleap. In our work, we focus on the last three steps. First of all, we will discuss the curve reconstruction process, and then methods for parametric surface reconstruction will be discussed.

2.1 Points set parameterization for curve fitting

Fitting parametric curves to points is a process of reverse engineering, which is a very important task in many fields, such as computer-aided design, medical images, 3D reconstruction, data visualisation and many others (Farin 2014, G. Farin and Kim 2002). The curves can be Bézier, B-spline and NURBS curves, etc. The point data can be obtained in many ways, including software programs and technologies like 3D laser scanning and multi-view reconstruction, etc. By fitting curves to point sets optimally, the resulting digital models can not only save a lot of storage capacity but also be easier to transmit, process, and modify than point sets. To fit a curve to a point

set, we have to choose between two techniques, which are approximation and interpolation. The former method makes the fitting curve approximately fit the given points while the latter method requires the fitting curve to pass through the given points. In both cases, parameterization of the points is normally a must and key step as the quality of the final results highly depends on the technique used to parameterize points. The parameter values reflect the distribution of the data points, and a good parameterization should minimise the error between the reconstructed curves and the data sets as much as possible, which is usually measured by the Hausdorff metric as shown in Eq. (2.1). what's more, the number of variables used should also be as small as possible.

$$\underset{\mathbf{i}}{\text{maximize}} \ \underset{\mathbf{t}\in[\mathbf{0},\mathbf{1}]}{\text{minimize}} \, ||P_i - c(t)|| \tag{2.1}$$

where P_i is the i_{th} point, $c(t)$ is the value of the fitting curve corresponding to the i_{th} point, whose parameter is t.

In the parameterization process, given a set of points (N points) $P_i(i = 0, 1, 2, \ldots, N-1)$ in R^n (n=2 or 3), a parametric function f is computed, which is a mapping from the parameter domain (usually [0,1]) to R^n. In the case of interpolation, $f(t_i) = P_i$ and in the case of approximation, $f(t_i) \approx P_i$, where t_i is the associated parametric value of the i_{th} point in the parameter domain. There are many methods proposed to tackle this problem. However, to the best of our knowledge, there are no survey papers about point set parameterization for curve fitting. This section will give a review of these techniques. Overall, the methods of fitting parametric curves to a point set can be classified into two categories. as shown in Fig. 2.1.

The methods in the first category divide the fitting process into two separate steps. In the first step, the point sets are parameterized using some methods belonging to this category, and then the obtained parameters are used for approximating or interpolating the point sets to get the fitting curves. The methods in the second category combine the two steps to treat both the parameters of point sets and other fitting variables as free variables that are to be obtained through simultaneous optimization by applying more complex

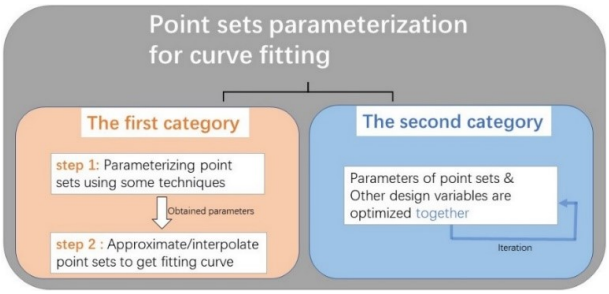


Figure 2.1: Two main types of techniques to fit parametric curves to point sets.

techniques. Table 2.1 and 2.2 list the main methods in the two categories and present a comparison of the pros and cons of the listed methods.

2.1.1 Methods in the first category

The methods in this category parameterize points first, and then the obtained parameters are used for curve fitting. They mainly include uniform parameterization, chord length method, centripetal method, refined centripetal parameterization, dynamical centripetal parameterization, Foley parameterization, universal method, and some other methods. Uniform parameterization (equidistant parameterization) (Farin 2014), as formulated in Eq. (2.2), is characterised by that the travelling time $\Delta t = t_i - t_{i-1} (i = 2, 3, \ldots, N)$ between points is constant if the physical heuristics of driving a car is used to explain it. The method is easy to calculate and mostly applicable to point data that is distributed regularly but usually behaves poorly in cases where points are distributed irregularly.

$$t_i = (i-1)\frac{1}{N-1} \tag{2.2}$$

The chord length method, which is also called the linear method, is shown in Eq. (2.3) in the case of $a = 1$. Its idea is similar to the time calculation of travelling on a curve with a constant speed. With this idea, the parameter t can be regarded as the time of travelling on the curve, and then the travelling time $\Delta t = t_i - t_{i-1}$ is proportional to the arc distance between two adjacent points of the curve. This method is a dominant choice because the

Table 2.1: Methods of parameterizing point sets for curve fitting in the first category

Method	**Pros**	**Cons**
Uniform (Farin 2014)	Simple	Mainly applied to uniform sampled point sets
Chord length (Xu et al. 2022)	Highly related to parametric values	May generate wiggles, Deflections
Centripetal (Piegl and Tiller 2012) (Fang and Hung 2013) (Balta et al. 2019)	Usually outperforms uniform and chord length method	Not affine invariant
Foley (Nielson and Foley 1989)	Affine invariant	Can generate wiggles
Universal (Lim 1998)	Affine invariant	Similar to uniform method in some sense
Hybrid (Shamsuddin and Ahmed 2004)	More accurately	Unstable and more computational capacity required

chord length approximates the arc distance, which is believed to be highly related to parametric values of t. Floater et al. have demonstrated this with many numerical examples in (Floater and Surazhsky 2006) where they show that full approximation order for cubic interpolation can be obtained using the chord length parameterization method. However, to obtain a higher degree of interpolation, more accurate approximations of arc length are required. The chord length method usually outperforms the uniform method in most cases, but one cons of this method is that curves with corners at

Table 2.2: Methods of parameterizing point sets for curve fitting in the second category

Method	**Pros**	**Cons**
Bat (Iglesias et al. 2015)	Nature inspired	Tuning parameters
GAPSO (Gálvez and Iglesias 2013)	Generality, completeness, good performance, high accuracy	Computational time, parameters tuning
Multilayer perceptron (Laube et al. 2018)	First work to apply NN for B-spline curve approximation; Generalize well	Synthetic training data; sub-,super-sampling of points are required
Residual NN (Scholz and Jüttler 2021)	Efficient, easy to implement	Higher polynomial degrees are not explored; synthetic training data set

the data points cannot be reconstructed. Besides, large deflections may be generated if there exist long chords in the data polygon. To alleviate such disadvantages, some other methods have also been proposed to parameterize point sets for curve fitting. For example, Xu et al. present a modified chord length parameterization method (Xu et al. 2022) for B-spline curve interpolation. A series of interpolation arcs are constructed by taking advantage of the relationship between the chord length and chord angle of given points, which replace chord length in calculating the global knot parameters. Their method outperforms classical methods regarding deviation error and smoothness. Another method named centripetal parameterization (Piegl and Tiller 2012) is proposed, whose equation can be expressed as Eq. (2.3) in the case

of $0 < a < 1$.

$$t_0 = 0, t_N = 1, t_i = t_{i-1} + \frac{|P_i - P_{i-1}|^a}{\sum_{i=1}^{N-1} |P_i - P_{i-1}|^a} \tag{2.3}$$

Its idea is also similar to driving with speed unchanged. But when it comes to corner sections where the points are normally dense, we have to slow down. The travelling time is calculated with the equation $\Delta t_i = ||P_i - P_{i-1}||^a$ where $0 < a < 1$ and $\frac{1}{2}$ is recommended. The equation means Δt_i becomes bigger if the distance between two adjacent points is less than one and smaller otherwise, thus achieving slowing speed down around corners. On average, the centripetal method gives better results than the uniform and chord length methods. However, the traditional centripetal method is not very robust in parameterizing point sets with fast changes in chord length as it adopts a fixed power for chord lengths for parameter distribution. To improve the performance of the centripetal method, some methods have been proposed. By creating an osculating circle at each point, Fang and Hung (Fang and Hung 2013) present the refined centripetal parameterization to improve the wiggle deviation of interpolation, especially for abrupt data. The method is more applicable to any data since it considers another important geometric feature, i.e., the angles formed by data sets. Another method named the dynamical centripetal parameterization method has been developed in (Balta et al. 2019). The basic idea of this method is to change the same exponent value a in the centripetal method for long and short chords to different values according to the natural logarithm of the chords. It tackles the problem of the centripetal method in producing an irregular parameter distribution for a data set.

Nevertheless, neither the chord length method nor the centripetal method is invariant under affine transformation, which means that if a non-uniform scaling is applied to the data points, the newly obtained curves using these parameterization methods will not in general be a scaling of the original. To address this issue, Foley parameterization (Nielson and Foley 1989) was proposed by taking into account not only the distance but also the angles between adjacent points to deal with abrupt data distribution, which is demonstrated in Eq. (2.4) and Fig. 2.2. The speed of this method also slows down

around corners other than keeping the same speed as other sections. Even though this method claims to handle point sets of abrupt distribution well, similar wiggles to the chord length method may be generated.

$$\begin{aligned} \Delta t_0 &= d_0[1 + \frac{3\theta_1 d_1}{2(d_0 + d_1)}] \\ \Delta t_i &= d_i[1 + \frac{3}{2}(\theta_i \frac{d_{i-1}}{d_{i-1} + d_i} + \theta_{i+1} \frac{d_{i+1}}{d_i + d_{i+1}})] \\ &\quad (i = 1, 2, ..., N-2) \\ \Delta t_{N-1} &= d_{N-1}[1 + \frac{3\theta_{N-1} d_{N-2}}{2(d_{N-2} + d_{N-1})}] \end{aligned} \tag{2.4}$$

where $d_i = M[P](P_{i-1}, P_i)$ is the Nielson distance between points P_{i-1} and P_i, θ_i is the angle formed by d_{i-1} and d_i shown in Figure 1, and when $\theta_i > \pi/2$, $\pi/2$ is used instead. The universal method (Lim 1998), which is used

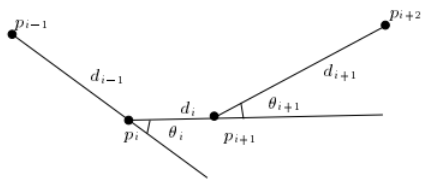


Figure 2.2: Foley parameterization.

to parameterize points for B-spline curve interpolation, was proposed by Choong-Gyoo Lim in 1999. Different from other related methods used to fit B-spline curves to point sets, which usually calculate the knot vector from the parameters, this method selects the maximums of the basic functions as the parameters by starting with a uniformly distributed knot vector. This method is affine invariant and can outcome more natural-looking curves, but unwanted wiggles may also exist. A hybrid parameterization method, which combines the advantages of universal and centripetal methods, was presented in (Shamsuddin and Ahmed 2004). Similarly, the maximum rational B-spline basis functions are taken as the initial values, then the centripetal method is used to obtain the parameters. This method can give better accuracy, but it is not stable and requires more computational effort.

In summary, the chord length and centripetal methods are the preferred ones among the aforementioned techniques, and the uniform method is preferable in computer-aided geometric design applications. Since each method has its advantages and disadvantages, a suitable method should be selected for a specific task.

2.1.2 Methods in the second category

All the methods reviewed in the last subsection divide curve-fitting problems into two steps: point sets parameterization followed by fitting optimisation which uses the parameters obtained from the previous step. However, dividing curve-fitting problems into two steps may not be the most effective and efficient way to tackle such problems. It is because how to choose a suitable way of parameterizing point sets is a non-trivial task and problem-dependent. Besides, it has been demonstrated that better fitting results can be obtained by treating point sets parameters as free variables that are to be optimised (Iglesias et al. 2015). Due to these reasons, some researchers follow this idea and try to optimise the parameters of point sets together with other free variables such as control points of the fitting parametric curves using more advanced optimisation techniques. In doing so, curve fitting becomes a non-linear, multivariate optimisation problem. Metaheuristics and deep learning have been used to tackle the problem. In what follows, we review some representative methods.

2.1.2.1 Metaheuristics-based parameterization

Iglesias Andrés and Gálvez Akemi apply the Bat algorithm (Iglesias et al. 2015), which is a powerful optimisation method inspired by the echolocation behaviour of bats, to obtain an optimal solution for the parameterization problem of fitting Bézier curves to point sets using the least squares technique. Their method can treat the data points with challenging features such as cusps and self-intersections and greatly outperforms the arc-length parameterization.

Another method called IMCH-GAPSO (Gálvez and Iglesias 2013), which consists of genetic algorithms (GA) and particle swarm optimisation (PSO) that both are powerful algorithms for optimisation and global search problems, has been proposed to reconstruct B-spline curves from point sets. The GA and PSO account for parameterizing point data and knots placement respectively, and they are mutually coupled, the process is repeated iteratively until a certain criterion is met. Besides, some classical parameterization can also be incorporated to make convergence faster. Their method is general, accurate and of good performance. However, there are also some limitations to these methods, which treat parameters as free variables. It is not easy to tune the parameters, which are mostly empirical and problem-dependent.

2.1.2.2 Deep learning-based methods

Deep learning has also been adopted in parameterizing point sets for curve approximation. Here, we review two deep learning methods used in parameterizing point sets.

Laube et al. adopted multilayer perceptron (Laube et al. 2018) to parameterize point sets for B-spline curve fitting. The parameters and knots of the B-spline curve are predicted by training an interdependent deep neural network. Their method can reconstruct tight approximation even for not evenly spaced point sets, which are not included in the training data sets. However, there are some limitations to this method. Firstly, segmentation, subsampling and super-sampling are required as a pre-processing step to ensure that the number of points matches the neural network input size, which is fixed. Besides, self-intersecting curves cannot be approximated using their method. In another method (Scholz and Jüttler 2021), a residual deep neural network has been adopted to approximate a function which assigns suitable parameter values to a sequence of points for a certain degree of polynomial curve fitting. The reason why the residual deep neural network is applied is that it can avoid the vanishing gradient problem, which is common in the neural network that consists of many layers. Their method applied 17 layers in total, each point except the two boundary points owns two weights (add to 1), which define the relationship between each point and its neighbours. As

for the loss function, the updated parameters of all points using the weights are plugged into the ground truth curve equation and then subtracted from the ground truth points to form the loss function. Their experiments show better performance than the chord length method and centripetal method in most cases, but the method doesn't outperform some other methods in the first category in some cases. In addition, the method has some other limitations. First of all, the training data set is synthetic, which may hinder it from generalising to data in real life. Furthermore, only a fixed degree of polynomial curves have been reconstructed.

Since there is no publicly available data set that includes a large number of curves of the required degree to be used to train the networks, both methods choose to generate the data set by randomly assigning values to the coefficients of the curves of that degree. Then random parameters are generated to plug into the obtained expression of these curves to get a set of ground truth points on these curves. This is not an ideal solution because the data sets that are closer and related to the data in the real world are better. In spite of this, both methods can serve as a pre-training technique, which can be improved by incorporating more advanced algorithms and additional data.

It should be noted that this thesis does not include all the methods that belong to this category. However, the core idea of the methods in this category lies in optimising appropriate parameter values of data points and other design variables for curve fitting. Each method has its strengths and weaknesses, and no method can be regarded as the best in any situation.

2.2 Points set parameterization for surface fitting

3D point cloud parameterization, also called point cloud mapping, is the process of mapping a 3D point cloud onto a suitable (usually simpler) domain. It has many applications such as object classification, texture mapping and surface reconstruction (Meng et al. 2016, Azariadis 2004, Hormann and Greiner 2000). In many situations, it is computationally expensive or difficult to

work with 3D point clouds directly. Therefore, projecting them onto a lower dimensional space without distorting their shape is necessary. Compared to mesh parameterization, 3D point cloud parameterization is more challenging in general because there is no connectivity information between points, which hinders the direct extension of well-established mesh parameterization algorithms to point cloud parameterization.

Some methods have been proposed to parameterize point clouds. In this paper, we roughly divide them into two main groups according to whether point clouds are organised or not. For each of the two groups, we further divide it into some subgroups based on the property of the mapping process and review each of the methods.

2.2.1 Some concepts

In this section, some concepts related to point clouds will be introduced to help readers understand the problem of point cloud parameterization. Since mesh parameterization has been well investigated in existing work and some ideas of mesh parameterization can be adopted by or adapted to point cloud parameterization, we will also introduce some concepts about mesh parameterization in this section.

1. **Organised and unorganised point clouds:** Generally, point clouds can be divided into organised and unorganised ones. Organised and unorganised point clouds are also called structured and unstructured point clouds, respectively. The division is determined by the way of storing point cloud data. For organised point clouds, the data are stored in a structured manner, while unorganised point cloud data are stored arbitrarily. Specifically, an organized point cloud is similar to a 2D matrix and its data are divided into rows and columns according to the spatial relationships between the points. Accordingly, the spatial layout represented by the xyz-coordinates of the points in a point cloud decides the memory layout of the organised point cloud. Contrary to organized point clouds, unorganised point clouds are just a collection of 3D coordinates, each of which denotes a single point.

2. **Global and local parameterization:** To parameterize point clouds, some methods map the whole point set of an underlying structure to a parameterization domain. In contrast, some other methods split the problem into several subproblems, each of which is called a local parameterization. The choice between global and local parameterization has impacts on mapping processes and results. Globally parameterizing the whole point set can guarantee the reconstructed mesh is a perfect manifold, meaning there are no seams, which may exist if the point cloud is partitioned and locally parameterized. However, processing the whole point cloud at the same time may be computationally expensive, especially for large structures.

3. **Topological shapes:** Topological shapes can be grouped based on the number of holes they own. Shapes with no holes such as spheres and bowls are treated as genus-0 shapes. Similarly, genus-1, genus-2 and genus-3 shapes have one, two and three holes in them, respectively, and so on.

4. **Bijective function:** also called bijection, invertible function, or one-to-one correspondence, pairs each element in one set exactly to one element in the other set, and vice versa.

5. **Isometric, conformal, and equiareal mappings:** Suppose f is a bijective function between a mesh S or a point cloud and a mapping domain S^*, then f is isometric (length preserving) if the length of any arcs on S is preserved on S^*; f is conformal (angle preserving) if the angle of intersection of every pair of intersecting arcs on S is preserved on S^*; f is equiareal (area-preserving) if the area of an area element on S is preserved on S^*. Isometric mappings are equiareal and conformal. Any mappings that are equiareal and conformal are isometric mapping.

2.2.2 Parameterization methods of organised point clouds

To parameterize an organised point cloud, many methods iteratively obtain a topologically identical 2D triangulation from the underlying 3D triangulation

of the point cloud, and the 2D triangulation determines the parameter values of the vertices in the domain plane. Depending on the ways of transforming from 3D to 2D, there are several methods, including Harmonic parameterization (Hadenfeld 1995), Floater's barycentric mappings (Floater 1997) and the most Isometric parameterization (Hormann and Greiner 2000). For Harmonic parameterization, the arc length is regarded as the parameter value of a spline curve, which is used to minimise the integral of the squared curvature with respect to the arc length for fairing the spline curve. With regard to barycentric mappings, a shape-preserving parameterization method is applied for smooth surface fitting; the parameterization that is equivalent to a planar triangulation can be obtained by solving a linear system based on the convex combination. In (Hormann and Greiner 2000), Hormann and Greiner propose a method to parameterize triangulated point clouds globally, the way of parameterizing the inner point set is the same as that of parameterizing the boundary point set. However, they ignore the problem of parameterizing triangulated point clouds with holes.

Energy function has also been defined to minimise the metric distortion in the transformation process from 3D to 2D. The methods described in (Floater 1997, Unther Greiner and Hormann 1996), follow the shared approach, which first parameterizes the boundary points, and then minimises the following edge-based energy function for the parameterization of inner points:

$$E = \frac{1}{2} \sum c_{ij} ||P_i - P_j||^2 \tag{2.5}$$

where c_{ij} is the edge coefficient that can be chosen in various ways, P_i and P_j are two points at the same edge.

In order to reconstruct a tensor product B-spline surface from scattered 3D data with specified topology, choosing a suitable way to parameterize the points is crucial in the reconstruction process. The method adopted by Greiner and Hormann is called the spring model. With this method, the edge of the 3D triangulation is replaced by a spring. Then the boundary points are mapped first onto a plane and stay unchanged. Next, the inner points

are mapped onto this plane by minimising the spring energy. The procedure is repeated to improve the parameters until certain conditions are satisfied.

The above methods are mainly applicable to structured point clouds. They are not efficient when the number of points increases, and are likely to fail when holes and concave sections exist in the point clouds.

2.2.3 Parameterization methods of unorganised point clouds

In comparison with the parameterization of organised point clouds, many more methods have been proposed to parameterize unorganised point clouds. Table 2.3 lists these methods and gives information about the category, parameter domain, local or global parameterization, topology, applications and publication year.

According to the property of the mapping process, we divide the parameterization methods of unorganised point clouds into base surfaces-based methods, meshless parameterization, spherical mapping, methods adapted from mesh parameterization, neural networks-based methods, and other methods. Here, We will not present every method in detail, only relevant methods to this thesis will be reviewed. Please refer to (Zhu et al. 2022a) for a more comprehensive introduction to these methods.

2.2.3.1 Base surfaces-based methods

For the parameterization of unorganised point clouds, base surfaces, which approximate the underlying structure of point clouds, have been widely applied to parameterize point clouds. Base surfaces can be a plane, a Coons patch, or a cylinder (Azariadis 2004). The parameter values of each point in a point cloud can be obtained by projecting the point cloud onto a base surface. The projection direction can either be perpendicular to the surface or based on a determined projection vector. According to (Ma and Kruth 1995), a base surface should own the following properties:

(a) Unique local mapping: The uniqueness implies that any two different points on the underlying surface should be mapped onto two different locations on the mapping domain.

(b) Smoothness and closeness of base surface: This indicates that a base surface should be as smooth and simple as possible, while still approximating the underlying surface as much as possible. The balance between these properties should be carefully considered.

(c) Parameterization of base surface: This implies that how we parameterize a base surface has a direct effect on the parameterization of the fitting surface. We can choose a more suitable way to parameterize a base surface by referring to the underlying structure of the fitting surface.

Table 2.3: methods to parameterize unorganised point clouds

Methods	**Category**	**Parameter domain**	**Local/ global parameterization**	**Topology**	**Applications**
"Simplicial" surface (Hoppe et al. 1992)			/	Arbitrary topology	Surface reconstruction
Manually define (Ma and Kruth 1995)			Global	/	Least square fitting of B-spline curves and surfaces

Continued on next page

Table 2.3 – *Continued from previous page*

Minimising quadratic function (Pottmann et al. 2002)	Base surfaces-based methods	Base surfaces	/	/	B-spline curves and surfaces approximation
Recursive DBS (Azariadis and Sapidis 2005)			Global/ local	Disk	Efficient parameterization
Recursive subdivision technique (Azariadis and Sapidis 2007)			Global/ local	Disk (With hole is ok)	Parameterizing point clouds
Floater meshless parameterization (Floater and Reimers 2001, Floater 2000)			Global	Disk	Surface reconstruction

Continued on next page

Table 2.3 – *Continued from previous page*

Meshless parameterization for spherical topology (Hormann and Reimers 2002)	Meshless parameterization	plane	Local	Genus-0	Surface reconstruction
As-rigid-as possible meshless parameterization (Zhang et al. 2010)			Global	Disk	Denoising and parameterizing point clouds, mesh reconstruction
Meshless quadrangulation by global parameterization (Li et al. 2011b)			Global	Arbitrary genus	Meshless quadrangulation
Spherical embedding (Zwicker and Gotsman 2004)			Global	Genus-0	Mesh reconstruction

Continued on next page

Table 2.3 – *Continued from previous page*

3D point clouds parameterization algorithm (Wang et al. 2008)	Spherical mapping	Sphere	Global	Relatively simple models	Parameterizing point clouds
Spherical conformal parameterization (Choi et al. 2016)			Global	Genus-0	Mesh reconstruction
Discrete one -forms (Tewari et al. 2006)	Adapt from mesh parameterization	Plane	Local	Genus-1	Mesh reconstruction
Periodic global parameterization (Li et al. 2011a)			Global	Arbitrary genus	Direct quad - dominant meshing of point cloud
PDE and SOM (Barhak and Fischer 2001)		Adaptive base surface	Global	Complex sculptured surfaces	Surface reconstruction

Continued on next page

Table 2.3 – *Continued from previous page*

Adaptive sequential learning RBFnetworks (Meng et al. 2013)	Neural networks-based methods	/	Global	Freeform	Point -cloud surface parameterization
Residual neural network (Scholz and Jüttler 2021)		/	Local	Fixed degree curve	Polynomial curve fitting
A new parameterization method (Jung and Kim 2000)	Other	/	/	/	NURBS surface interpolation
Pointshop 3D (Zwicker et al. 2002)		/	/	/	Point -based surface editing
Free -boundary conformal parameterization (Choi et al. 2022)		/	Global/ local	/	Parameterizing point clouds for meshing

To get access to such base surfaces, some approaches have been proposed. For example, Hoppe et al. (Hoppe et al. 1992) propose a method to produce

so-called "simplicial" surfaces. They first define a function to estimate the signed geometric distance to the underlying surface of the point clouds, then a contouring algorithm is applied to approximate the underlying surface by a "simplicial" surface. Their method is capable of reconstructing a surface with or without boundary from an unorganised point set. However, there is no formal guarantee that the reconstructed result is correct and the space required to store the reconstruction is relatively large. In (Ma and Kruth 1995), users can also manually define some section curves and four boundary curves to get a base surface of a point cloud, as some characteristic curves approximating the underlying structure of the point cloud are sufficient in defining a base surface. But it is also necessary to take advantage of the interior characteristic curves when the geometry is complex, even though just four corner points can be used to create a base surface in some cases. A base surface can also be obtained by iteratively minimising a quadratic objective function (Pottmann et al. 2002). With this method, a linear system of equations is solved in each step. To parameterize unstructured point clouds, Dynamic Base Surfaces (DBS) are also proposed by Azariadis (Azariadis 2004). As its name implies, a BDS is gradually improved regarding its approximation to the underlying structure of a point cloud, and the parameter value of each point in the point cloud is obtained by projecting it orthogonally to the DBS. Different from existing methods, no restrictions are required for the density and the homogeneity of point clouds. The limitation of this method is that it is only applicable to point clouds where a closed boundary consisting of four curves exists. Azariadis and Sapidis (Azariadis and Sapidis 2005) present a method to parameterize a point cloud globally and/or locally using recursive dynamic base surfaces. Their method can handle arbitrary point clouds of disk topology. Figure 1 shows the local parameterization of one subset of several point clouds using this method. The same authors (Azariadis and Sapidis 2007) extend the DBS concept and use a recursive subdivision method to improve the accuracy of point clouds parameterization, especially for some small regions of the point clouds, where the approximation error by the DBS is not acceptable. They divide such regions into smaller parts and the points on these parts are approximated by c^0 composite surface based on recursive

DBS subdivision to increase the approximation error, and then to make the point clouds parameterization more accurate.

2.2.3.2 Neural networks-based methods

With the rapid development of neural network techniques, they have been applied to three main tasks of point cloud processing, i.e., 3D shape classification, 3D object detection and tracking, and 3D point cloud segmentation (Guo et al. 2020). Besides their applications in the three main tasks, some researchers have investigated neural network-based point cloud parameterization. For example, Barhak and Fischer (Barhak and Fischer 2001) adopt a self-organising map (SOM) for the parameterization of small sets of clean points with low-frequency spatial variations, which can be used to reconstruct smooth surfaces. There are mainly two steps in the parameterization process: In the first step, Partial Differential Equation (PDE) and SOM are applied where the former technique can yield a parametric grid without self-intersection and the latter one makes sure all the sampled points have an impact on the grid, which guarantees the uniformity and smoothness of the reconstructed surface. In the second step, an adaptively modified 3D base surface is created for point cloud parameterization. Meng et al. (Meng et al. 2013) proposed a method to parameterize larger, noisy and unoriented point clouds by using adaptive sequential learning RBFnetworks. The network adopts a dynamic structure by adaptive learning and the neurons are adjustable regarding their locations, widths and weights, thus making it more powerful compared to other methods that apply RBFs at determined locations and scales. What is more, multi-level parameterization and multiple level-of-details (LODs) can be achieved in two ways. When multiple LODs meshes are required, parameterizing the point clouds with the best resolution and the points and surfaces can be computed at degrading sampling level to get the required LODs. In the second case where only one downgraded LOD is required, downgraded parameterization can be applied to obtain the result.

2.3 Curves and surfaces reconstruction from point clouds

To reconstruct curves and surfaces, there are many proposed methods, this thesis will first present various methods for curves reconstruction, and then in the next subsection, techniques for surfaces will be discussed. Their pros and cons will be demonstrated.

2.3.1 Curve reconstruction

The curves are usually represented in parametric form, which mainly includes Bézier, B-Spline, NURBS, etc. Bézier curve is relatively easy and the most widely used cubic Bézier curve is defined by

$$C(t, \mathbf{p}) = \sum_{k=0}^{3} p_k B_k(t) \tag{2.6}$$

where $p_k \in \mathbf{p}$ (k = 0,1,2,3) are the four control points, $B_k(t)$ are called the Bernstein polynomials. Specifically, $B_0(t) = (1-t)^3$, $B_1(t) = 3 * t * (1-t)^2$, $B_2(t) = 3 * (1-t) * t^2$ and $B_3(t) = t^3$. And t is the parameter variable associated with each point of the segments, $t \in [0, 1]$.

Bézier curves are widely used in Font design, image vectorization, etc. However, when the more complex shape is required by one Bézier curve, higher order degrees of Bézier curves are necessary, but they are computationally expensive and local support is not available. That is why the B-spline curve comes into help, the degree of the B-spline curve does not depend on the number of control points, and it is also locally supported, thus making it very flexible and controllable. A B-spline curve is defined by the following expression:

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_{i,k} \qquad\qquad t \in [0, 1]$$

where $\mathbf{P}_i (i = 0, 1, 2, \ldots, n)$ are the control points, k is the order of the B-spline curve, and $B_{i,k}(t)$ are basis functions, which are defined recursively as

follows:

$$B_{i,1}(t) = \begin{cases} 1 & \text{for } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

and

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(t) \qquad \text{if } k > 1$$

In the above equations, $\{t_i\}_{i=0}^{n+k}$ is the knot vector, a non-decreasing array of numbers from 0 to 1. Note that Bézier curves are special cases of B-spline curves. Specifically, when the order of a B-spline curve matches the number of control points, resulting in an even number of knots, and both halves of the knots are clamped at each end, the B-spline curve reduces to a Bézier curve.

However, B-spline curves cannot represent some types of shapes, such as circles and ellipses, exactly. To obtain a more powerful representation, Non-Uniform Rational B-splines (NURBS) is developed, and this representation has become a standard in more areas such as car design. The NURBS is defined as follows:

$$C(u, \mathbf{p}) = \frac{1}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \sum_{i=0}^{n} N_{i,p}(u) w_i p_i \tag{2.7}$$

where $p_i (i = 0, 1, 2, ..., n)$ are the control points, and the wights $w_i (i = 0, 1, 2, ..., n)$ correspondes to the control points. The basis functions $N_{i,p}(u)$ is defined recursively in a similar way to the B-Spline curve and a knot vector $\{u_0, u_1, ..., u_m\}$ also needs to be defined. Notice that if all weights are equal to 1, a NURBS curve reduces to a B-Spline curve.

Here, we mainly focus on Bézier curves reconstruction, as they are used in our work. To reconstruct Bézier curves from point clouds, many methods have been proposed and most of them adopt the pipeline we demonstrated in the last chapter. From the segmentation on, the corner (salient) points are detected for the given or extracted point clouds, and the corner points are used to segment the points into multiple segments, each of which will be fitted by one or more Bézier curves independently. In the fitting phase,

the points in a segment are first parameterized using some point parameterization techniques and the obtained parameters are used for curve fitting. Meanwhile, break points may be added to further break the segment into multiple subsegments if one curve is not enough to fit all the points in the segment. Specifically, Schneider (Schneider 1990) first locate the corner points by computing the angle between each point and its neighbours, and then for each segment between two adjacent corner points, a cubic Bézier curve is used to fit it. The break points are added if necessary to the place where there exists the maximum error between the fitting curve and the original points. The process is repeated until all the subsets in each segment are fitted successfully. On the contrary, Pavlidis adds the break point at the place with the minimum error (Pavlidis 1983). The middle point of the segment is also used as the break point in (Gonczarowski 1991). But all these methods do not guarantee that the number of curves needed is as few as possible while keeping small fitting errors. To use as few as possible curves, people could try all possible arrangements of the corner and break points and choose the best one. But the time complexity is very high, especially when the number of input points is large. To decrease time complexity, the dynamic programming technique (Plass and Stone 1983) is adopted by Plass and Stone, and it can reduce the time complexity to $O(n^3)$ by dividing the problems into sub-problems recursively, but it's still relatively expensive. Pal et al. (Pal et al. 2007) propose an adaptive method to detect the break point, and then the initial approximated Bézier control points are obtained using the interpolation technique, from which the control points of the best fitting curves are found by applying two-dimensional logarithmic and an evolutionary search algorithm. Hoshyari et al. (Hoshyari et al. 2018) adopt machine learning techniques to reconstruct curves from points of semi-structured boundaries, which are usually distinctly coloured and piecewise continuous. The supervised learning technique is used to detect the corner points, which are combined with global cues that both consider simplicity and continuity to output results that comply with human perception of points of semi-structured boundaries. Their framework is computationally expensive when the size of the input is large and other machine-learning techniques

may give better results. He et al. (He et al. 2023) present a method to reconstruct curves from boundary points of binary shapes using affine scale-space, which consists of three steps: firstly, at the sub-pixel level, the curvature extrema of the boundary are computed, then the control points are detected as the curvature extrema in the affine scale space, and finally the points between adjacent control points are fitted with cubic Bézier curve using the least square method under the condition that the fitting error is less than a predefined accuracy.

Concerning the optimisation methods, many techniques have been proposed. For example, the least square is a widely adopted method for minimising an error function because of its effectiveness, efficiency and simplicity. To further reduce the fitting error, a method named reparameterization (Plass and Stone 1983, Schneider 1990) is adopted to find a better distribution of parameters for each point iteratively. This method works in some cases but may fail in some other cases because the solution obtained using the Newton-Raphson method may only find the local optimum other than the global optimum solution (Chang and Yan 1998), as the newly updated parameters may not satisfy $0 = t_0 \leq t_1 \leq \cdots \leq t_n = 1$. These optimisation methods take the determined parameters of the points as input and optimise the error function to obtain the optimal positions of the control points of the fitting curve, which means the output results also depend on the parameter distributions of the points. There are many techniques to choose to parameterize the points, but it's difficult to decide which one works better for a certain case. So, some other methods treat the parameters of the points as free variables to be optimised rather than calculate them in one shot using a chosen parameterization method. It has also been shown that better results can be obtained by doing so (Iglesias et al. 2015, Gálvez and Iglesias 2013). However, these methods that treat point parameters as free variables have some limitations. Firstly, it's not easy to tune the design variables such as the search range and the number of iterations, which are mostly empirical and problem-dependent. What's more, there is no guarantee that the process would converge.

Overall, all the aforementioned methods have both pros and cons, and they do not guarantee the number of Bézier curves is as few as possible while keeping good fitting quality.

2.3.2 Surface reconstruction

For surface reconstruction from 3D point data, there are numerous surface representations to choose from. Broadly, these representations can be categorised into two primary types: explicit and implicit. Each type has its own advantages and disadvantages. For a more comprehensive and detailed introduction to various methods of surface reconstruction from point clouds, please refer to the review papers (Sulzer et al. 2023, Berger et al. 2017, Lim and Haron 2014, Berger et al. 2014, Shen et al. 2023).

Methods in the first category include parametric surfaces, triangular surfaces, etc. Triangular surfaces are typically reconstructed using Delaunay triangulations and Voronoi diagrams (Edelsbrunner 1998, Amenta et al. 1998). These methods use numerous flat triangles to approximate curved shapes, which can require very high storage. A parametric surface, on the other hand, is defined by a mathematical expression, so it is a very compact representation. Parametric surfaces also have the advantages of easily generating points on the surface and intuitively altering the surface shape by adjusting the position of control points or other parameters in the mathematical expression. However, determining whether a point lies on which side of a parametric surface is not straightforward. Examples of parametric surfaces include Coons surfaces, Bézier surfaces (Bołtuć and Zieniuk 2021, Monterde and Ugail 2006), B-spline surfaces (Yuwen et al. 2006, Gordon and Riesenfeld 1974), NURBS surfaces (Gálvez and Iglesias 2012, Piegl and Tiller 2012), and PDE surfaces (Zhu et al. 2022b), etc. Specifically, Bézier surfaces have been widely adopted as a representation for surface reconstruction, because they are easy to control and relatively computationally efficient for lower-order degrees. However, to represent more complicated 3D shapes, we should either use multiple patches or increase the degree of the Bézier surfaces. For the former option, how to achieve good continuity between adjacent patches

is challenging for surface reconstruction from point clouds. For the latter option, increasing the degree is computationally expensive and there is no local support. To represent more complex 3D shapes while avoiding the need to increase the degree of the surfaces, the B-spine surface comes to the rescue. However, one challenge of B-spline surface reconstruction is that sharp features tend to be smoothed out. Lee et al. proposed an approach to preserve sharp features for B-spline surface reconstruction from point cloud (Lee et al. 2020). They took advantage of the curvature information of the B-spline patch and identified segments of sharp features, which could be preserved by adding more points to those regions. However, their method was only applied to a relatively simple shape of point clouds. The B-spline approach also contains some limitations. For instance, the rigid structure of the B-spline recurrence formula and its constraints on the order of the parameters and the basis functions can lead to models that have significantly larger numbers of degrees of freedom than it should be actually required. On the other hand, it is well known that B-splines are not well suited to model T-junctions, and hence, it is difficult to apply them to structural modelling involving such features. NURBS surfaces have been proposed to overcome the shortcomings of B-spline surfaces, as they are a generalisation of the classical polynomial B-splines. Dimitrov et al. presented a new approach for NURBS surfaces fitting to unorganised point clouds (Dimitrov and Golparvar-Fard 2014). They used intermediate B-spline surfaces to parameterize the points in a point cloud and the reconstructed NURBS surface was refined. A main challenge of fitting NURBS surfaces to point clouds lies in the proper parameterization of the input points. Several methods have been proposed to overcome such a difficulty. Bo et al. (Bo et al. 2012) fitted a parametric surface to a point cloud by minimising the squared orthogonal distance from the surface to point cloud, thus it becomes a nonlinear least-squares minimisation problem. They use an initial surface to approximate the point cloud, and for each point, they compute the closest point on the surface. They kept updating the surface by minimising a quadratic function until the fitting error was smaller than a certain threshold. For parametric surface reconstruction from

3D point data, it is often necessary to use multiple patches, each defined by a mathematical expression, to construct a complex shape.

Implicit representation has become a very popular choice, it is powerful because it can theoretically represent arbitrarily complex shapes and is highly compatible with deep learning techniques for surface reconstruction. Compared to explicit representation, it is much easier to determine whether a point lies inside or outside the surface. However, generating a point from an implicit representation and altering the shape of surfaces is not straightforward. Additionally, implicit surfaces need to be converted into explicit representations like polygon surfaces or voxels for display in the digital world. Typical examples of implicit representation include level sets (Fuhrmann and Goesele 2014, Zhao et al. 2001), distance functions (Boissonnat and Cazals 2000, Calakli and Taubin 2011), algebraic surfaces (Yavartanoo et al. 2021, Beauville 1996), and Constructive Solid Geometry (Xiao and Furukawa 2014, Laidlaw et al. 1986). Among distance function representations, Poisson surface reconstruction (Kazhdan et al. 2006) is a classic method, which solves for an approximate indicator function of the inferred solid, whose gradient best matches the normal of the point set. The reconstructed surface is obtained by extracting an appropriate isosurface of the indicator function using adaptations of the Marching Cubes algorithm. Implicit B-spline surfaces have also been investigated and used for reconstruction (Rouhani et al. 2014).

In addition to the aforementioned methods for reconstructing 3D surfaces from 3D data points, PDE surfaces have also been adopted using both explicit and implicit representations. For example, Ugail and Kirmani employed an elliptic PDE equation and analytically solved it by utilising a set of curves as the boundary condition, resulting in a highly efficient approach (Ugail and Kirmani 2006). Rodrigues et al. explicitly resolved a Laplace equation and used it for compressing and reconstructing 3D data (Rodrigues et al. 2013). In the context of 3D surface reconstruction using implicit PDE models, many approaches have also been proposed (Duan et al. 2004, Linz et al. 2006, Franchini et al. 2010). To be more specific, Duan et al. presented a PDE-based deformable surface to evolve its shape to reconstruct 3D surfaces, where the input data can be either volumetric data or unorganised point clouds and

multi-view 2D images. Franchini et al. proposed a method to reconstruct a 3D shape from an unorganised point set by adopting a PDE-based deformable surface. Their method can also be applied to Boolean operations between various data. Linz et al. developed a technique to reconstruct 3D shapes from implicit PDE definition. Please also refer to (Othman et al. 2019) for a more comprehensive and detailed discussion on surface reconstruction using PDE.

In recent years, different machine learning methods, mostly based on neural networks and deep learning (Gu and Yan 1995, Sharma et al. 2021), metaheuristic techniques such as genetic algorithms and nature-inspired optimisation algorithms (Gálvez et al. 2012, Gálvez and Iglesias 2012), or combinations of both (Xiyu et al. 2003) have also been used for 3D reconstruction from point clouds. In this part, we will briefly review some of the recent works.

For parametric surfaces, Sharma et al. presented a parametric surface network for 3D point clouds and it was an end-to-end trainable model (Sharma et al. 2020). They firstly decomposed a 3D point cloud into several basic geometric primitives and B-spline patches, and the number of patches was automatically determined. Then they fit each segmented portion of the point cloud with a parametric patch. Post-processing was also necessary to better fit the point cloud, but producing seamless boundaries was still a challenge. At the object level, many deep learning-based methods learnt priors from the dataset (Kanazawa et al. 2018, Kato et al. 2018, Mescheder et al. 2019). However, these methods do not generalise well to unseen objects during training since they learn priors at the object level. To make this technique more general, Badki et al. proposed to learn the local shape (patch-level) prior of objects for mesh reconstruction (Badki et al. 2020). The learnt local shape features serve as a dictionary of local features, and they can be used to reconstruct 3D shapes even from unseen categories. Williams used a similar method to learn geometric prior for surface reconstruction (Williams et al. 2019). They fit many neural networks in parallel while enforcing global consistency to compute an atlas of multiple mappings, which can reconstruct complex objects very well. However, gap or jagged areas may exist because

of inconsistent mapping between different patches. Based on this observation, Deng et al. proposed a method to achieve global consistency between local mapping by incorporating consistency of local surface normal and minimising a prescribed stitching error (Deng et al. 2020). They also obtained adjacent patches that nearly coincide, a clear indication of good stitching.

Chapter 3

Vectorizing binary image boundaries with symmetric shape detection, bisection and optimal parameteterization

Binary image boundary vectorization refers to the process of converting raster images represented by two-dimensional pixels to vector images represented by a sequence of reconstructed mathematical curves such as widely used Bézier curves, which approximate the underlying structure of the pixel images. The latter representation is more compact as less storage is required and it can also be scaled up or down to any scale level without introducing aliasing or losing information compared to the former one (Joshi 2014). Moreover, vectorization representation has many other applications, such as remoting sensing (Kirsanov et al. 2010), feature recognition (Nadal et al. 1990) and many others (Zou and Yan 2001, Chiang et al. 1998, Lopes et al. 2019, Dominici et al. 2020, Bhunia et al. 2021). The pixel images can also be reconstructed from the vector images without losing significant information if needed. Two main factors in reconstructing mathematical curves for image boundary vectorization are to approximate the underlying structure of the image boundaries as much as possible while using as few numbers of mathematical curves as possible (Plass and Stone 1983).

There are some varieties in the vectorization pipelines. The pipeline used

in our method is based on (Sarfraz and Khan 2004) where cubic Bézier curves are used to fit the boundary of binary images. It consists of the following steps: preprocessing, boundaries (contours) detection for reconstructing the Bézier curves, and corner (salient) points detection, which are used to segment each boundary point into multiple segments, each of which will be fitted by one or more Bézier curves independently. In the fitting phase, the boundary points in a segment are first parameterized using some point parameterization techniques that are discussed in detail in Chapter 2, and the obtained parameters are used for curve fitting. Meanwhile, break points may be added to further break the segment into multiple subsegments if one curve is not enough to fit all the points in the segment.

Concerning the first three steps for boundary and corner point detection, many methods have been proposed, and we can select a suitable one from them for our task. For the last three steps, which include parameterization of segment points, fitting and breaking (if necessary), many techniques have also been presented. Each parameterization method has its pros and cons. No method can handle all the problems of curve fitting. Therefore, it is recommended to adopt these methods based on specific problems and requirements. However, most proposed techniques for vectorizing images just choose one of the methods (normally chord length). Besides, break points may be necessary when one curve is not accurate enough to fit a segment. Thus, it is critical to effectively choose a minimal number of break points while tightly approximating the underlying structure of the segment points. For example, Schneider (Schneider 1990) treats the point with maximum error as the break point. On the contrary, the point with zero error is chosen as the break point by Pavlidis (Pavlidis 1983) as he finds "splitting the interval at points where the error is zero is preferable to splitting at points where the error estimate is maximum". The middle point of the segment is also used as the break point in (Gonczarowski 1991). However, all those methods do not guarantee that each curve fits as many points as possible, thus may result in more curves needed to fit a segment.

To further reduce the number of curves required, a method to fit a sequence of cubic Bézier curves to image boundaries using symmetric detec-

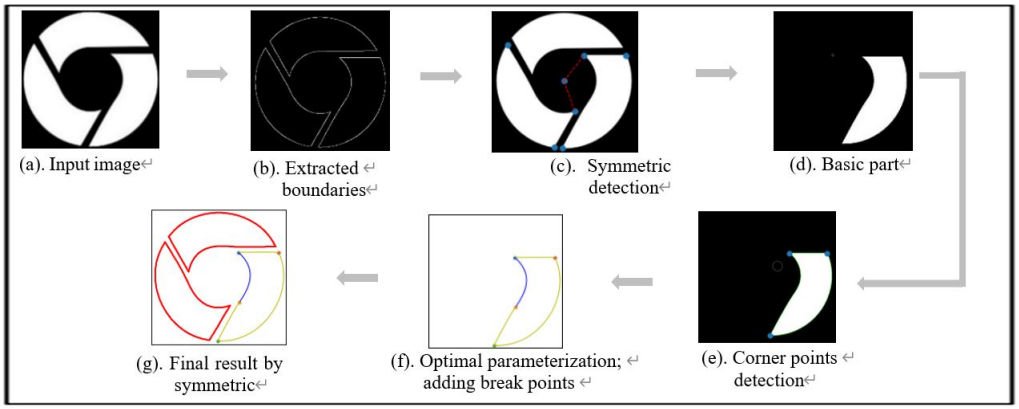


Figure 3.1: The pipeline of our method for vectorizing binary images.

tion, bisection and optimal parameterization is proposed. For the symmetric detection, because Bézier curves are invariant under affine transformation, which means for axis and point-symmetric picture boundaries, we just need to vectorize a part of the boundaries and the remaining parts can be vectorized by transforming the already reconstructed Bézier curves. Furthermore, the bisection method we adopt works similarly to the bisection algorithm in matching an element to the same element in an ordered list or finding the roots of a polynomial equation (McNamee and Pan 2013). It is simple but very efficient and can be adopted to determine the best break point quickly. By doing so, we can further minimise the number of Bézier curves to represent the input image boundaries. Finally, we investigate the impact of different parameterization methods on fitting the same segment and propose to use different parameterization methods for different segments to reduce the fitting error or even the number of fitting curves, which is named optimal parameterization in this chapter. The general pipeline of our method is shown in Fig. 3.1.

3.1 Symmetric axis and point detection

In order to determine whether the boundary of a given image is symmetric about an axis or a point, we propose the following idea. First, we determine whether the boundary is symmetric about an axis. If not, we further test

whether it is symmetric about a point. If the boundary of the image is not symmetric about both an axis and a point, the image is treated as a regular one.

To determine whether the image boundary is symmetric about an axis or not, the boundaries are first detected using the findContours function in the OpenCV library (Culjak et al. 2012). Next, the principal component analysis (PCA) technique is adopted to find the two main axes of the boundaries. Since only the directions of the two axes are determined and they are free in their perpendicular direction, and we need to find one point they should go through to fix both their position and direction. We notice that a symmetric axis should go through the centric point of the boundary points, which can be calculated by adding all the coordinates of the boundary points and dividing the sum by the number of boundary points. After obtaining the two principal axes that go through the centric point of the image shape, as shown in Fig. 3.2(a), we then test whether the shape is symmetric about any of these two axes. To do so, for each axis, we flip one side of the shape around the axis and use it to compute the intersection and union with another side of the shape. If the ratio between the intersection and union is close to one, it means that the shape is symmetric about this axis, as demonstrated in Figs. 3.2(b) and 3.2(c), the intersection and union shapes are almost the same. For another axis, the ratio between the corresponding intersection and union is much smaller than one, which means the shape is not symmetric about this axis, as shown in Figs. 3.2(d) and 3.2(e). By setting a threshold, we can decide whether a shape is symmetric about an axis or not.

Given the image shape is symmetric about an axis whose equation can be expressed as $ay + bx + c = 0$, in which a, b, c are known constants. In such a case, only half of the image boundaries that lie on one side of the symmetric axis need to be vectorized. Supposing half of the image boundaries have been fitted with one cubic Bézier curve defined by four control points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ whose coordinates are $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$, respectively in the same coordinate system as the symmetric axis. Then the boundary points lying on the other side of the symmetric axis can be vectorized by another cubic Bézier curve, whose control can be obtained by mirroring $\mathbf{p}_0$,

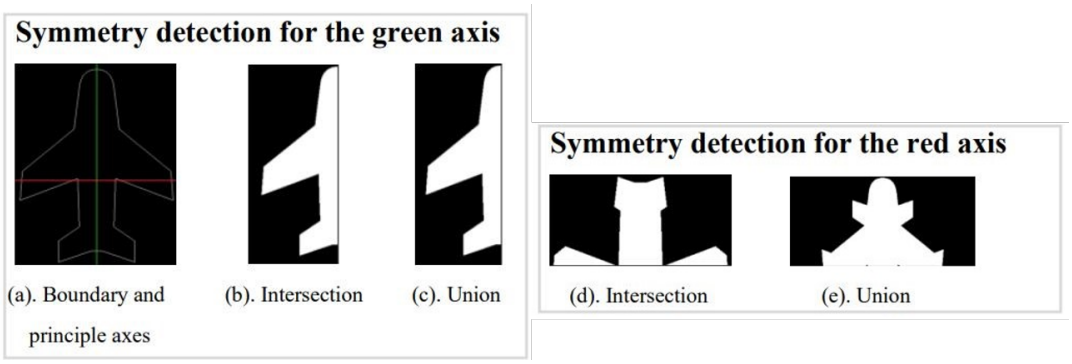


Figure 3.2: Symmetric axes detection

$\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ around the symmetric axis using the following equation: given a point coordinate (p, q) and a symmetric axis equation $ay + bx + c = 0$, the symmetric point (p_s, q_s) of the given point about the symmetric axis is shown in Eq. 3.1.

$$p_s = \frac{p * (a^2 - b^2) - 2 * b * (a * q + c)}{a^2 + b^2}, \quad q_s = \frac{q * (b^2 - a^2) - 2 * a * (b * q + c)}{a^2 + b^2} \tag{3.1}$$

As demonstrated in Fig. 3.3, for a symmetric shape, only half of the shape needs to be vectorized. So, for an axis-symmetric image shape, we just need to keep half of the control points and three variables which represent the symmetric line. By doing so, the number of curves can be further reduced.

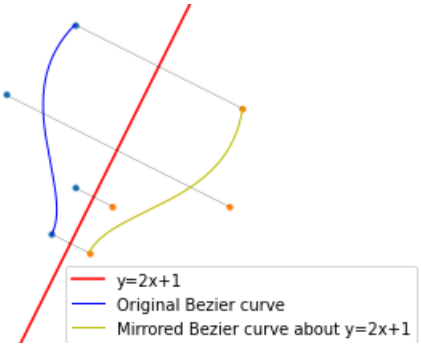


Figure 3.3: A cubic Bézier curve and its mirrored counterpart about an axis

To determine whether an image shape is symmetric around a point, if so, it can be observed that the centric point of the image boundaries should be

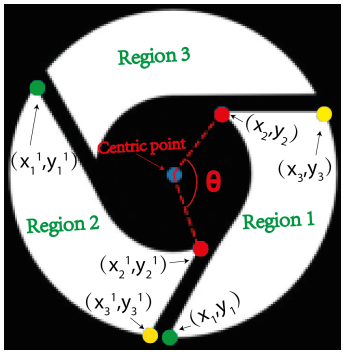


Figure 3.4: Symmetric point detection

the symmetric point. In such cases, we then use the Connected Component Labelling technique (He et al. 2017) to segment the images into multiple regions (number= r). Because the corner points (number= n) detected are ordered based on the quality level, there are $t = \frac{n}{r}$ corner points correspondences for each region with other regions, as shown in Fig. 3.4, the corner points $(x_1, y_1), (x_2, y_2)$, and (x_3, y_3) in Region 1 correspond to the corner points $(x'_1, y'_1), (x'_2, y'_2)$, and (x'_3, y'_3) in Region 2, respectively. With these corresponding points, we can estimate the potential rotational angle (θ) of one region with respect to its adjacent region around the symmetric point using the following equation:

$$\begin{bmatrix} x_1 & -y_1 \\ y_1 & x_1 \\ x_2 & -y_2 \\ y_2 & x_2 \\ \vdots & \vdots \\ x_t & -y_t \\ y_t & x_t \end{bmatrix} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_t \\ y'_t \end{bmatrix} \tag{3.2}$$

where $(x_1, y_1), (x'_1, y'_1), \ldots, (x_t, y_t), (x'_t, y'_t)$ are corresponding corner points for the two regions. This is normally an overdetermined linear system of equations. To find the best solution, we use the pseudoinverse method. After obtaining the potential rotating angle θ, we can rotate one region about the centric point by θ and calculate the ratio between the intersection and union.

By doing so, we can also reduce the number of curves needed. Lastly, if an image shape is neither symmetric about an axis nor about a point, then it is treated as a regular image, which has to be vectorized fully.

3.2 Fitting

A cubic Bézier curve is defined by Eq. (2.6). Given N points $(P_1, P_2, \ldots, P_n)$ of a segment, the parameter t_i associated with each point can be obtained using one of the points parameterization methods that we discussed in Section 2.1, then a cubic Bézier curve will be used to fit to those points. So, the following error function should be minimised:

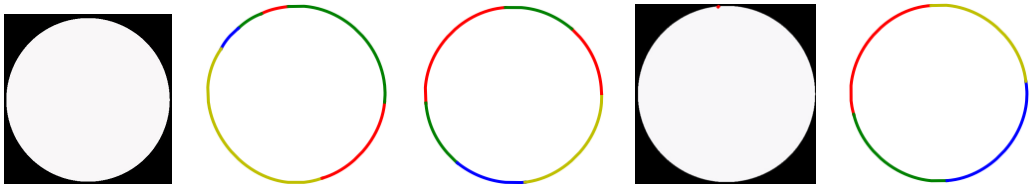
$$min \sum_{i=1}^{N} \left(C(t_i, \mathbf{p}) - P_i\right)^2 \tag{3.3}$$

As a cubic Bézier curve would go through its two end control points, it means that the two end control points p_0 and p_3 are fixed at the first and last point of the segment. So, $p_0 = P_1, p_3 = P_N$ only p_1 and p_2 are to be optimised. To minimise the error function, we take its derivative with respect to p_1 and p_2, and set the derived equations to zeros, which can be used to derive p_1 and p_2. Specifically, let

$$\begin{aligned} f(p_1, p_2) = \sum_{i=1}^{N} (C(t_i, p) - P_i)^2 = \\ \sum_{i=1}^{N} [(C(t_i, p) - P_i)]^T [C(t_i, p) - P_i] \end{aligned} \tag{3.4}$$

We also define $a_i = 3 * t_i * (1 - t_i)^2$, $b_i = 3 * (1 - t_i) * t_i^2$ and $c_i = (1 - t_i)^3 * p_0 + t_i^3 * p_3 - P_i$, which are known once given the points of a segment, then Eq. (3.4) becomes:

$$\begin{aligned} f(p_1, p_2) = \sum_{i=1}^{N} [(a_i * p_1 + b_i * p_2 + c_i)]^T \\ [(a_i * p_1 + b_i * p_2 + c_i)] \end{aligned} \tag{3.5}$$



(a) (b) 6 curves (c) 6 curves (d) error (e) 4 curves

Figure 3.5: (a) Input images; (b) adding break points at positions with maximum error (Schneider 1990); (c) positions at middle point (Gonczarowski 1991); (d) at positions with minimum error (Pavlidis 1983); (e) our bisection method.

Taking the derivate of $f(p_1, p_2)$ with respect to p_1, p_2, we obtain

$$\frac{\partial f}{\partial p_1} = \sum_{i=2}^{N-1}[2 * a_i^2 * p_1^T + 2 * a_i * b_i * p_2^T + 2 * a_i * c_i] \tag{3.6}$$

$$\frac{\partial f}{\partial p_2} = \sum_{i=2}^{N-1}[2 * a_i^2 * p_1^T + 2 * a_i * b_i * p_2^T + 2 * a_i * c_i] \tag{3.7}$$

We then define $a_1 = \sum_{i=2}^{N-1} 2 * a_i^2$, $b_1 = \sum_{i=2}^{N-1} 2 * a_i * b_i$, $b_1 = \sum_{i=2}^{N-1} 2 * a_i * c_i$, $b_1 = \sum_{i=2}^{N-1} 2 * b_i^2$, $b_1 = \sum_{i=2}^{N-1} 2 * b_i * c_i$, which are all known given points of a segment. Then equation (3.6) and (3.7) becomes:

$$a_1 * p_1^T + b_1 * p_2^T = -c_1 \tag{3.8}$$

$$b_1 * p_1^T + a_1 * p_2^T = -c_2 \tag{3.9}$$

Let matrix $A = \begin{bmatrix} a_1 & b_1 \\ b_1 & a_2 \end{bmatrix}$, $b = \begin{bmatrix} -c_1 \\ -c_2 \end{bmatrix}$. Then we can obtain our solution $\begin{bmatrix} p_1^T \\ p_2^T \end{bmatrix} = A^{-1} * b$.

3.3 Bisection method

As we mentioned in the introduction section, the bisection method is applied when one cubic Bézier curve cannot fit all the points of a segment well.

Algorithm 1 Bisection method to determine the break point(**input:**: points in a segment):

```
//comments: In a segment, l is the index of the left point, r is the index
of right point, and m is the index of the middle point.
l_0 = 0, r_0 = len(points)-1, l = l_0, r = r_0, m = (l + r)/2
if a curve can fit the points between l_0 and r_0 then return
    // no break point needed
else
    while (m changes) do
        m = (l + r)/2
        if a curve can fit points between l_0 and m then
            l = m
        else
            r = m
        end if
    end while
return m as break point index
end if
```

The bisection method works similarly to the bisection algorithm in matching an element to an ordered list. The pseudocode algorithm using the bisection method to find breaking points is presented in Algorithm 1. In the algorithm, we use l_0 to indicate the index of the left point and r_0 to indicate the index of the right point. If l_0 and r_0 are two adjacent corner points and a cubic Bézier curve can fit all the points from l_0 to r_0, no breaking points are added between the two adjacent corner points l_0 and r_0. Otherwise, we find a middle point m so that all the points from l_0 to m can be fitted by a cubic Bézier curve with the following iteration algorithm below. First, we set $l = l_0$ and $r = r_0$. Then we calculate the middle point $m = (l + r)/2$. If a cubic Bézier curve can be used to fit all the points from l_0 to the middle point m, we set $l = m$. Otherwise, we set $r = m$. Then we calculate the new middle point $m = (l + r)/2$ and check whether a cubic Bézier curve can be used to fit all the points from l_0 to the new middle point m. A breaking point m is found by repeating the iteration until the middle point m does not change. After the middle point m becomes a breaking point, we set $l_0 = m$ and check whether all the points from l_0 to r_0 can be fitted by a cubic Bézier curve. If

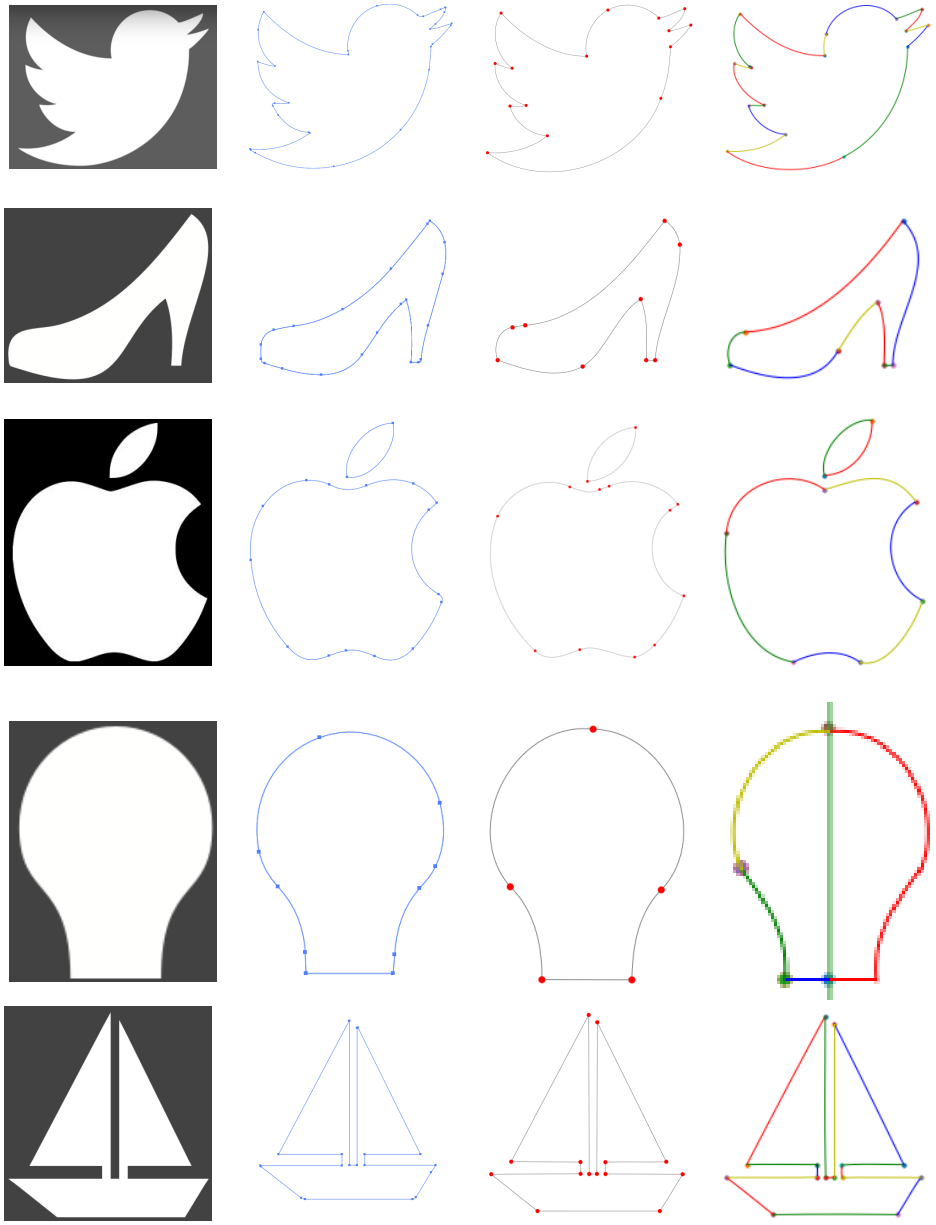
yes, no breaking points are added between l_0 and r_0. Otherwise, the above iteration is used to find another braking point m. This process is repeated until all the points from l_0 to r_0 can be fitted with cubic Bézier curves.

To demonstrate that the bisection method outperforms traditional methods in determining the suitable positions of break points, we compare it with three other methods, which choose the break point at the middle point, the point with the largest error, and the smallest error respectively. We set the threshold the same for all the cases, as we can see from Fig. 3.5(b) and 3.5(c), more curves will be used if we add the break points at the positions with the maximum error or at the middle point position. Besides, putting break point at the position with the minimum error may cause a failure because the chosen position is just one point away from its near endpoint, resulting in a segment with just three points as shown in Fig. 3.5(d), with which a Bézier curve cannot be determined. However, using the bisection method, the number of curves is further minimised as shown in Fig. 3.5(e).

3.4 Results and comparison

In this section, we present the results of vectorizing image boundaries using our method and compare it with the classical and new methods. We set the error tolerance the same for our method and the new method to make the comparison fair. For the classic method, setting the error tolerance is not available, so we keep the default setting.

First, we use some less complex image shapes to compare our method with two existing methods, i.e., the classical Image tracer method and the new Affine scale-space method. The fitted boundary curves are shown in Fig. 3.6 and the number of curves needed for each of the input images shown in Fig. 3.6 is given in Table 3.1. As we can see from Figure 3.6 and Table 3.1, our proposed method outperforms the Image tracer method and the Affine scale-space method in terms of using a minimal number of cubic Bézier curves while achieving a similar good approximation.



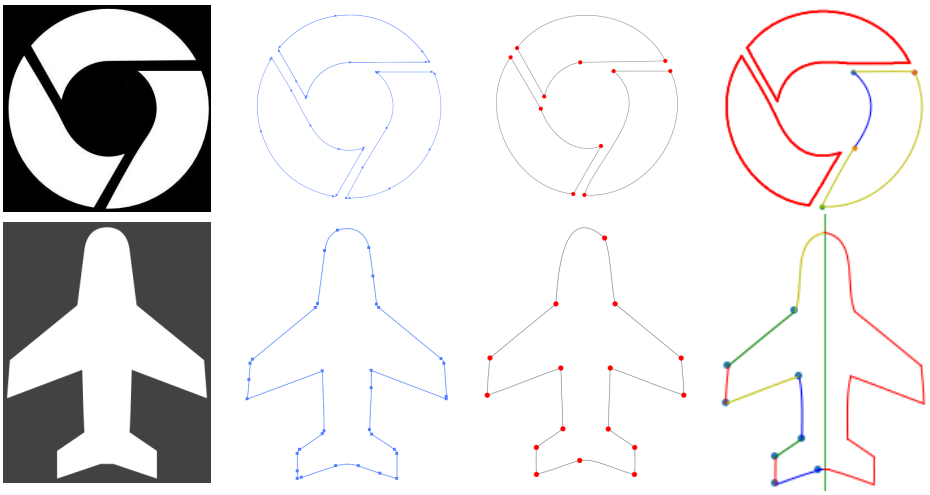


Figure 3.6: (a) Input images (S.L. 2025); (b) Image tracer; (c) Affine scale space; (d) Our method

Table 3.1: Number of curves required for our method and other methods for less complex image shapes.

Input	Image tracer	Affine scale-space	Our method
bird	>20	15	15
heel	>20	9	**7**
apple	17	14	**8**
bulb	12	5	**3**
ship	17	14	14
circle like	>10	10	**5**
airplane	>18	not good	**10**

To further demonstrate the effectiveness and advantages of our proposed method, we compare our method with the Image tracer method and the Affine scale-space method on more complex and diverse image shapes and present the comparison results in Fig. 3.7 and Table 3.2. Once again, Fig. 3.7 and Table 3.2 show our proposed method outperforms the Image tracer method and the Affine scale-space method on these complex and diverse

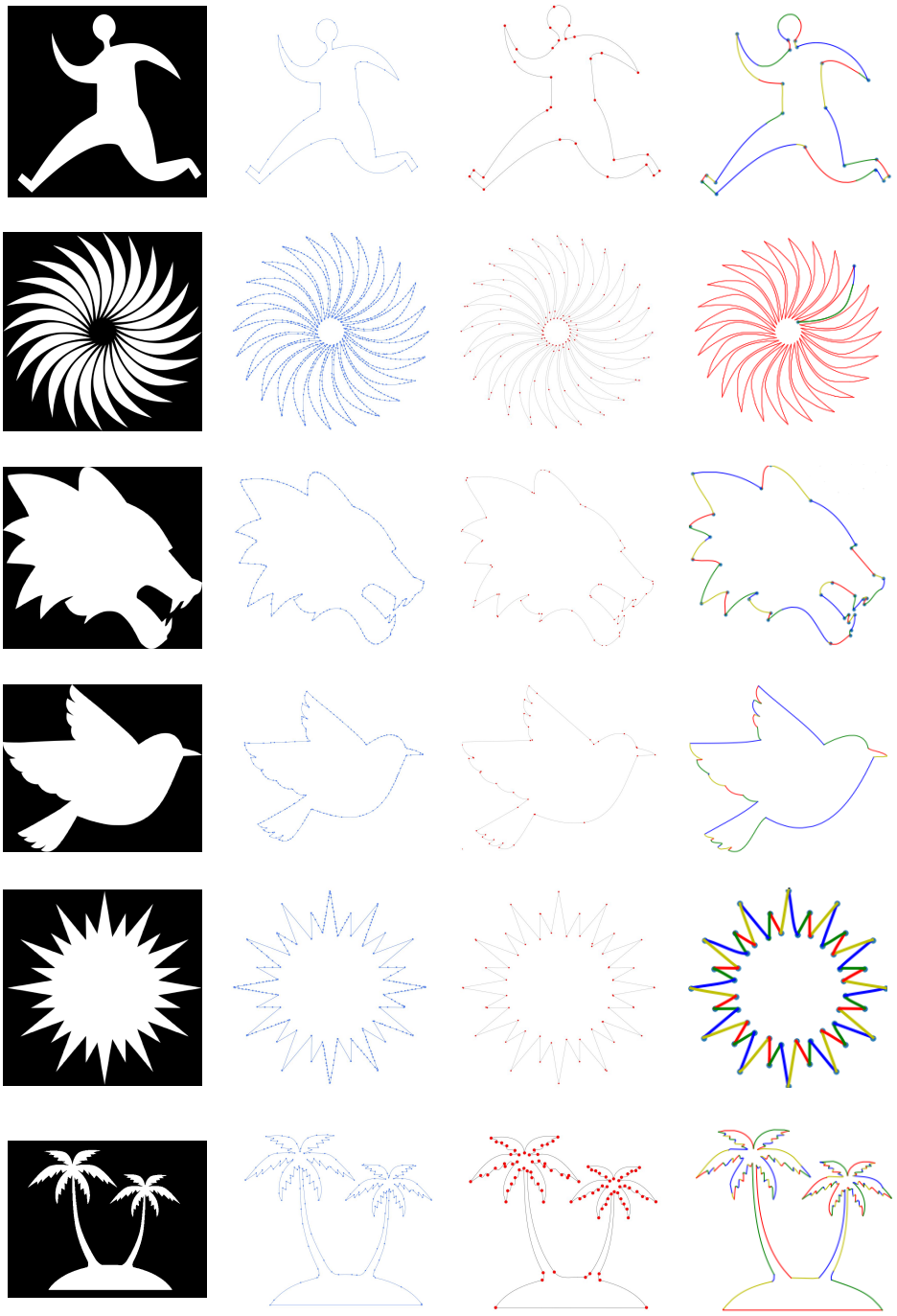


Figure 3.7: (a) Input images (S.L. 2025); (b) Image tracer; (c) Affine scale space; (d) Our method

image shapes.

Table 3.2: Number of curves required for our method and other methods for more complex image shapes.

Input	Image tracer	Affine scale-space	Our method
man	44	29	**28**
flower	>126	126	**6**
wolves	>55	55	**40**
new bird	>36	36	**26**
star	>52	52	**48**
tree	>142	not good (too smooth)	**142**

The image shapes shown in Fig. 3.6 and Fig. 3.7 are noise-free. For such noise-free image shapes, the above comparisons demonstrate the advantages of our proposed method over the existing methods. We have also used our proposed method, the Image tracer method, and the Affine scale-space method to vectorize the boundary of a noisy image shape shown in Fig. 3.8(a) and present the results in Figs. 3.8(b), 3.8(c) and 3.8(d) where Figs. 3.8(b) and 3.8(c) are obtained with the Image tracer method and the Affine scale-space method, respectively, and Fig. 3.8(d) is obtained with our proposed method. The fitted curves in these figures clearly indicate our proposed method uses fewer curves and achieves better quality, which indicates that our proposed method also outperforms the Image tracer method and the Affine scale-space method in vectorizing the boundary of the noisy image shape.

Lastly, we investigate the impact of different parameterization methods, as shown in Figs. 3.9(b) and 3.9(c), each coloured curve represents a cubic Bézier curve fitted to a segment, and there are eight of them. Even though two vectorized images using two approaches look similar, their fitting error is different, which is demonstrated in Table 3.3. As we can see, the optimal parameterization should be chosen for each segment to reduce the fitting

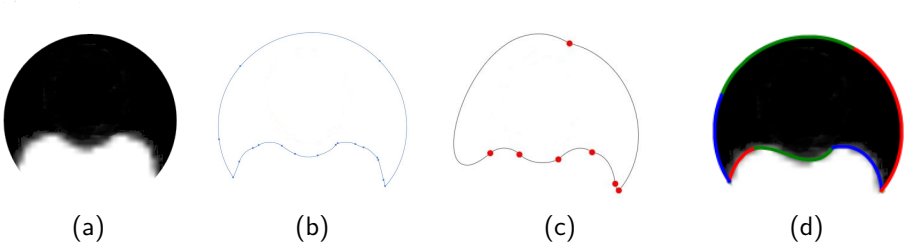


Figure 3.8: (a) Input noise images; (b) Image tracer; (c) Affine scale-space; (d) Our method.

Figure 3.9: (a)Input images, (b) traditional parameterization method (chord length method), (c) our optimal parameterization method.

Table 3.3: Comparison between chord length parameterization and optimal parameterization methods.

Segment NO.	Traditional	Our optimal
1	1.168	1.168 (chord)
2	1.144	**1.054**(uniform)
3	1.484	1.484 (chord)
4	1.468	**1.410**(uniform)
5	1.482	1.482 (chord)
6	1.416	1.416 (chord)
7	1.456	1.456 (chord)
8	1.337	**1.222** (centri)

error further rather than just adopting one method for all cases as most papers did.

Using a small number of curves while keeping tight approximation i. e., high accuracy in converting binary image boundaries into vector representations has many impacts. It has been acknowledged that presenting images as a list of mathematical curves can compress data to provide a more compact representation and reduce data storage, increase data transmission speed, and achieve faster processing to raise computational efficiency (Joshi 2014). The proposed method obviously reduces the number of cubic Bézier curves to further maximise its impact on these applications. Fitting quality quantified by fitting errors plays an important role in representing original image boundaries accurately. Due to its importance, existing vectorization methods mainly focus on accuracy during the conversion from raster images to vector images (Xiong et al. 2017). The proposed method has smaller fitting errors than existing methods, demonstrating its advantage in accurately representing original image boundaries.

3.5 Summary

This work proposes a method which incorporates symmetric image shape detection, bisection, and optimal parameterization to vectorize image boundaries, aiming to minimise the number of Bézier curves needed while keeping good approximation. We take advantage of the affine transformation invariant property of Bézier curves and present a method for detecting a symmetric axis or point for an image shape if it exists, which can further reduce the number of curves required. Besides, we apply the bisection method to add suitable break points in an efficient way. Lastly, we also find different parameterization methods for each segment that can be adopted to further reduce the fitting error or even the number of curves.

Chapter 4

Parametric surface reconstruction using closed-form solution of a fourth-order PDE

Compared to other types of surface representations, PDE-based representation has many advantages such as smaller data storage, physics-based and good continuity under certain conditions. However, a main difficulty for PDE-based shape manipulation is how to solve partial differential equations. Due to this difficulty, most studies investigated implicit PDE-based shape reconstruction which involves numerically solving partial differential equations. Although some research studies investigated explicit PDE-based shape reconstruction by interpolating four curves or satisfying the constraints on two opposite boundaries of a PDE patch, few studies presented closed-form solutions of partial differential equations for 4-sided PDE patches. In this chapter, we will propose a mathematical model, derive its closed-form solutions, and use one of the closed-form solutions to achieve shape reconstruction from point clouds.

We will first derive the closed-form solution to a fourth-order PDE, then the analytical form will be used to reconstruct a parametric surface from one single patch of points. Furthermore, more complex point sets will be used to validate the powerful capability of our method in fitting. Lastly,

the closed-form solution will be extended to an analytical form with more design variables, so it will have a more powerful fitting capability, which also showcases the flexibility of our method.

4.1 Mathematical model and closed-form solution

Partial differential equation-based shape reconstruction can be roughly divided into two categories: one uses implicit solutions of partial differential equations and the other uses explicit solutions of partial differential equations. Shape reconstruction using implicit solutions of partial differential equations involves a lot of numerical calculations, causing slow shape reconstruction which is not suitable for many applications requiring real-time performance. In contrast, shape reconstruction using explicit solutions of partial differential equations is based on accurate analytical or approximate analytical solutions of partial differential equations, which involves fewer calculations and is more efficient than shape construction using implicit solutions of partial differential equations. However, a main problem for shape reconstruction using explicit solutions of partial differential equations is how to obtain accurate analytical or approximate analytical solutions of partial differential equations. Since solving partial differential equations analytically is not an easy task, the current explicit solutions of partial differential equations used for PDE-based geometric modelling and shape reconstruction mainly deal with two boundaries of a PDE surface patch, i.e., accurately satisfy partial differential equations and continuity constraints at two opposite boundaries of a PDE surface patch. How to obtain accurate analytical solutions of partial differential equations which exactly satisfy partial differential equations and continuity constraints on four boundaries of a PDE surface patch is an important topic. A vector-valued partial differential equation used to describe a 3D surface patch involves two parametric variables u and v. The four boundaries of the 3D surface patch are defined by u=0, u=1, v=0, and v=1. In order to satisfy positional continuities, four unknowns should be included in a closed-form solution to a vector-valued partial differential

equation to satisfy four positional functions, i.e. boundary curves at the four boundaries of a 3D surface patch. Similarly, to satisfy up to tangential continuities, eight unknowns should be involved in a closed-form solution of a vector-valued partial differential equation to satisfy four positional functions and four tangential functions at the four boundaries of a 3D surface patch. From the theory of partial differential equations, the closed-form solution to a second-order partial differential equation of parametric variables u and v has four unknowns, and the closed-form solution to a fourth-order partial differential equation has eight unknowns. Tangential continuities are most popularly used to create smooth 3D models. Taking all of these factors and a closed-form solution into account, we propose to use the following vector-valued fourth-order partial differential equation for shape reconstruction:

$$\mathbf{a}_1 \frac{\partial^4 \mathbf{X}(u,v)}{\partial u^4} + \mathbf{a}_2 \frac{\partial^4 \mathbf{X}(u,v)}{\partial v^4} = \mathbf{F}(u,v) \tag{4.1}$$

where $\mathbf{a}_1$ and $\mathbf{a}_2$ are three-dimentional vectors, and $\mathbf{X}(u,v)$, and $\mathbf{F}(u,v)$ are three-dimensional vector-valued functions. Each of them has three components. Note that in this chapter vectors and vector-valued functions are denoted in bold.

4.1.1 Closed-form solution derivation

To simplify the notation in the chapter, the following mathematical operations are defined:

$$\begin{array}{ll} \mathbf{f}''''(u) = \frac{d^4\mathbf{f}(u)}{du^4} & \mathbf{g}''''(v) = \frac{d^4\mathbf{g}(v)}{dv^4} \\ e^{\mathbf{a}_1} = [e^{a_{1x}}, e^{a_{1y}}, e^{a_{1z}}] & \mathbf{a}_1\mathbf{a}_2 = [a_{1x}a_{2x}, a_{1y}a_{2y}, a_{1z}a_{2z}]^T \\ \frac{\mathbf{a}_1}{\mathbf{a}_2} = \left[\frac{a_{1x}}{a_{2x}}, \frac{a_{1y}}{a_{2y}}, \frac{a_{1z}}{a_{2z}}\right]^T & \sqrt[n]{\frac{\mathbf{a}_1}{\mathbf{a}_2}} = \left[\sqrt[n]{\frac{a_{1x}}{a_{2x}}}, \sqrt[n]{\frac{a_{1y}}{a_{2y}}}, \sqrt[n]{\frac{a_{1z}}{a_{2z}}}\right]^T \end{array} \tag{4.2}$$

In the future, we will investigate how to use a general expression for $\mathbf{F}(u,v)$ in order to make PDE patch-based surface reconstruction more powerful. In this chapter, we set $\mathbf{F}(u,v)$ to 0, which makes Eq. (4.1) homogeneous. In that case, we can use the method of separation of variables to

derive its four closed-form solutions. Assuming that the variables u and v in Eq. (4.1) are separable, $\mathbf{X}(u, v)$ can be expressed as follows:

$$\mathbf{X}(u, v) = \mathbf{f}(u)\mathbf{g}(v) \tag{4.3}$$

Substituting Eq. (4.3) back to Eq. (4.1), we get:

$$\begin{aligned} &\mathbf{a}_1\mathbf{g}(v)\tfrac{d^4\mathbf{f}(u)}{du^4} + \mathbf{a}_2\mathbf{f}(u)\tfrac{d^4\mathbf{g}(v)}{dv^4} = 0 \Rightarrow \\ &\mathbf{a}_1\mathbf{f}''''(u)\tfrac{1}{\mathbf{f}(u)} = -\mathbf{a}_2\mathbf{g}''''(v)\tfrac{1}{\mathbf{g}(v)} \end{aligned} \tag{4.4}$$

By setting both sides in Eq. (4.4) to $\mathbf{c}_0$, which is a vector-valued constant, we get Eq. (4.5):

$$\mathbf{a}_1\mathbf{f}''''(u)\frac{1}{\mathbf{f}(u)} = -\mathbf{a}_2\mathbf{g}''''(v)\frac{1}{\mathbf{g}(v)} = \mathbf{c}_0 \tag{4.5}$$

With the above treatment, the partial differential equation in Eq. (4.1) is transformed into two ordinary differential equations given in Eq. (4.5). The first ordinary differential equation is:

$$\mathbf{a}_1\mathbf{f}''''(u) = \mathbf{c}_0\mathbf{f}(u) \tag{4.6}$$

From Eq. (4.6), we know that $\mathbf{f}(u)$ can be taken to be:

$$\mathbf{f}(u) = e^{\mathbf{r}u} \tag{4.7}$$

From the above equation, we obtain the fourth-order derivative of $\mathbf{f}(u)$ as:

$$\mathbf{f}''''(u) = \mathbf{r}^4 e^{\mathbf{r}u}$$

Substituting the expressions of $\mathbf{f}(u)$ and $\mathbf{f}''''(u)$ into Eq. (4.6), we obtain:

$$\mathbf{a}_1\mathbf{r}^4 e^{\mathbf{r}u} = \mathbf{c}_0 e^{ru} \tag{4.8}$$

From Eq. (4.8), we obtain:

$$\mathbf{r}^4 = \frac{\mathbf{c}_0}{\mathbf{a}_1} \tag{4.9}$$

To solve Eq. (4.9) for $\mathbf{r}$ two cases should be considered. The first case is $\dfrac{\mathbf{c}_0}{\mathbf{a}_1} > 0$ and the second case is $\dfrac{\mathbf{c}_0}{\mathbf{a}_1} < 0$, where the inequalities must be understood as component-wise.

For the first case, $\frac{\mathbf{c}_0}{\mathbf{a}_1} > 0$, we have $\frac{\mathbf{c}_0}{\mathbf{a}_1} = \left|\frac{\mathbf{c}_0}{\mathbf{a}_1}\right| > 0$. To simplify mathematical notations, we let:

$$\mathbf{q}_1 = \sqrt[4]{\frac{\mathbf{c}_0}{\mathbf{a}_1}} = \sqrt[4]{\left|\frac{\mathbf{c}_0}{\mathbf{a}_1}\right|}$$

Now we can obtain the four roots of Eq. (4.9) as follows:

$$\mathbf{r}_1 = \mathbf{q}_1, \qquad \mathbf{r}_2 = -\mathbf{q}_1, \qquad \mathbf{r}_3 = i\mathbf{q}_1, \qquad \mathbf{r}_4 = -i\mathbf{q}_1 \tag{4.10}$$

where $i^2 = -1$. Substituting Eq. (4.10) back to Eq. (4.7), we obtain $\mathbf{f}(u)$ as follows:

$$\mathbf{f}(u) = \mathbf{c}_1 e^{\mathbf{q}_1 u} + \mathbf{c}_2 e^{-\mathbf{q}_1 u} + \mathbf{c}_3 cos(\mathbf{q}_1 u) + \mathbf{c}_4 sin(\mathbf{q}_1 u) \tag{4.11}$$

where $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4$, are vector-valued constants.

For the second case, $\frac{\mathbf{c}_0}{\mathbf{a}_1} < 0$,we let:

$$\mathbf{q}_2 = \frac{\sqrt{2}}{2}\sqrt[4]{\left|\frac{\mathbf{c}_0}{\mathbf{a}_1}\right|} = \frac{\sqrt{2}}{2} q_1$$

We can also obtain the four roots of Eq. (4.8) as follows:

$$\mathbf{r}_1 = \mathbf{q}_2(1+i), \quad \mathbf{r}_2 = -\mathbf{q}_2(1+i), \quad \mathbf{r}_3 = \mathbf{q}_2(1-i), \quad \mathbf{r}_4 = -\mathbf{q}_2(1-i)$$

In such a case, $\mathbf{f}(u)$ can be expressed as follows:

$$\begin{aligned} \mathbf{f}(u) = \quad & e^{\mathbf{q}_2 u}\left[\mathbf{c}_1 cos(\mathbf{q}_2 u) + \mathbf{c}_2 sin(\mathbf{q}_2 u)\right] + \\ & e^{-\mathbf{q}_2 u}\left[\mathbf{c}_3 cos(\mathbf{q}_2 u) + \mathbf{c}_4 sin(\mathbf{q}_2 u)\right] \end{aligned} \tag{4.12}$$

The second ordinary differential equation given in Eq. (4.5) can be written as:

$$-\mathbf{a}_2 \mathbf{g}''''(v) = \mathbf{c}_0 \mathbf{g}(v) \tag{4.13}$$

We can use the same method as solving $\mathbf{f}(u)$ to obtain the solution of $\mathbf{g}(v)$ for the two cases: $\frac{\mathbf{c}_0}{\mathbf{a}_2} > 0$ and $\frac{\mathbf{c}_0}{\mathbf{a}_2} < 0$.

For the first case, $\frac{\mathbf{c}_0}{\mathbf{a}_2} > 0$, we let:

$$\mathbf{q}_3 = \sqrt[4]{\left|\frac{\mathbf{c}_0}{\mathbf{a}_2}\right|}$$

We can get the solution of $\mathbf{g}(v)$ below:

$$\mathbf{g}(v) = \mathbf{c}_5 e^{\mathbf{q}_3 v} + \mathbf{c}_6 e^{-\mathbf{q}_3 v} + \mathbf{c}_7 cos(\mathbf{q}_3 v) + \mathbf{c}_8 sin(\mathbf{q}_3 v) \tag{4.14}$$

where $\mathbf{c}_5, \mathbf{c}_6, \mathbf{c}_7, \mathbf{c}_8$ are vector-valued constants.

For the second case, $\dfrac{\mathbf{c}_0}{\mathbf{a}_2} < 0$, we define:

$$\mathbf{q}_4 = \frac{\sqrt{2}}{2} \sqrt[4]{\left|\frac{\mathbf{c}_0}{\mathbf{a}_2}\right|} = \frac{\sqrt{2}}{2}\mathbf{q}_3$$

Under such a case, we obtain the expression of $\mathbf{g}(v)$ as follows:

$$\begin{aligned} \mathbf{g}(v) = \quad & e^{\mathbf{q}_4 v}\left[\mathbf{c}_5 cos(\mathbf{q}_4 v) + \mathbf{c}_6 sin(\mathbf{q}_4 v)\right] + \\ & e^{-\mathbf{q}_4 v}\left[\mathbf{c}_7 cos(\mathbf{q}_4 v) + \mathbf{c}_8 sin(q\mathbf{q}_4 v)\right] \end{aligned} \tag{4.15}$$

Since $\mathbf{f}(u)$ and $\mathbf{g}(v)$ both have two forms, which are Eq. (4.11) and Eq. (4.12) for $\mathbf{f}(u)$ and Eq. (4.14) and Eq. (4.15) for $\mathbf{g}(v)$, they can be substituted into Eq. (4.3) to obtain four solutions of $\mathbf{X}(u, v) = \mathbf{f}(u)\mathbf{g}(v)$. We use $\mathbf{X}_1(u, v)$, $\mathbf{X}_2(u, v)$, $\mathbf{X}_3(u, v)$, and $\mathbf{X}_4(u, v)$, to denote the four solutions, which are obtained below. Multiplying Eq. (4.11) with Eq. (4.14), we get $\mathbf{X}_1(u, v)$ below:

$$\begin{aligned} \mathbf{X}_1(u, v) = \quad & \left[\mathbf{c}_1 e^{\mathbf{q}_1 u} + \mathbf{c}_2 e^{-\mathbf{q}_1 u} + \mathbf{c}_3 cos(\mathbf{q}_1 u) + \mathbf{c}_4 sin(\mathbf{q}_1 u)\right] \\ & \left[\mathbf{c}_5 e^{\mathbf{q}_3 v} + \mathbf{c}_6 e^{-\mathbf{q}_3 v} + \mathbf{c}_7 cos(\mathbf{q}_3 v) + \mathbf{c}_8 sin(\mathbf{q}_3 v)\right] \end{aligned} \tag{4.16}$$

Multiplying Eq. (4.11) with Eq. (4.15), we get $\mathbf{X}_2(u, v)$ below:

$$\begin{aligned} \mathbf{X}_2(u, v) = \quad & \left[\mathbf{c}_1 e^{\mathbf{q}_1 u} + \mathbf{c}_2 e^{-\mathbf{q}_1 u} + \mathbf{c}_3 cos(\mathbf{q}_1 u) + \mathbf{c}_4 sin(\mathbf{q}_1 u)\right] \\ & (e^{\mathbf{q}_2 u}\left[\mathbf{c}_1 cos(\mathbf{q}_2 u) + \mathbf{c}_2 sin(\mathbf{q}_2 u)\right] + \\ & e^{-\mathbf{q}_2 u}\left[\mathbf{c}_3 cos(\mathbf{q}_2 u) + \mathbf{c}_4 sin(\mathbf{q}_2 u)\right]) \end{aligned} \tag{4.17}$$

Multiplying Eq. (4.12) with Eq. (4.14), we get $\mathbf{X}_3(u, v)$ below:

$$\begin{aligned} \mathbf{X}_3(u, v) = \quad & (e^{\mathbf{q}_2 u}\left[\mathbf{c}_1 cos(\mathbf{q}_2 u) + \mathbf{c}_2 sin(\mathbf{q}_2 u)\right] + \\ & e^{-\mathbf{q}_2 u}\left[\mathbf{c}_3 cos(\mathbf{q}_2 u) + \mathbf{c}_4 sin(\mathbf{q}_2 u)\right]) \\ & \left[\mathbf{c}_5 e^{\mathbf{q}_3 v} + \mathbf{c}_6 e^{-\mathbf{q}_3 v} + \mathbf{c}_7 cos(\mathbf{q}_3 v) + \mathbf{c}_8 sin(\mathbf{q}_3 v)\right] \end{aligned} \tag{4.18}$$

Multiplying Eq. (4.12) with Eq. (4.15), we get $\mathbf{X}_4(u,v)$ below:

$$\begin{aligned}\mathbf{X}_4(u,v) = \quad & (e^{\mathbf{q}_2 u}\left[\mathbf{c}_1 cos(\mathbf{q}_2 u) + \mathbf{c}_2 sin(\mathbf{q}_2 u)\right] + \\ & e^{-\mathbf{q}_2 u}\left[\mathbf{c}_3 cos(\mathbf{q}_2 u) + \mathbf{c}_4 sin(\mathbf{q}_2 u)\right]) \\ & (e^{\mathbf{q}_4 v}\left[\mathbf{c}_5 cos(\mathbf{q}_4 v) + \mathbf{c}_6 sin(\mathbf{q}_4 v)\right] + \\ & e^{-\mathbf{q}_4 v}\left[\mathbf{c}_7 cos(\mathbf{q}_4 v) + \mathbf{c}_8 sin(\mathbf{q}_4 v)\right])\end{aligned} \tag{4.19}$$

Each of the above four solutions can be used to define 4-sided PDE patches for surface reconstruction. In this paper, $\mathbf{X}_4(u,v)$ is adopted to reconstruct 3D surfaces from point clouds. The reason why we select $\mathbf{X}_4(u,v)$ over $\mathbf{X}_1(u,v)$, $\mathbf{X}_2(u,v)$, and $\mathbf{X}_3(u,v)$ is that it exhibits a slightly superior fitting capability, which will be demonstrated in Section 4.2

Conducting the multiplication operation in Eq. (4.19) and letting:

$$\begin{aligned}\mathbf{f}_1(u,v) &= e^{\mathbf{q}_2 u} e^{\mathbf{q}_4 v} cos(\mathbf{q}_2 u) cos(\mathbf{q}_4 v) \\ \mathbf{f}_2(u,v) &= e^{\mathbf{q}_2 u} e^{\mathbf{q}_4 v} cos(\mathbf{q}_2 u) sin(\mathbf{q}_4 v) \\ \mathbf{f}_3(u,v) &= e^{\mathbf{q}_2 u} e^{\mathbf{q}_4 v} sin(\mathbf{q}_2 u) cos(\mathbf{q}_4 v) \\ \mathbf{f}_4(u,v) &= e^{\mathbf{q}_2 u} e^{\mathbf{q}_4 v} sin(\mathbf{q}_2 u) sin(\mathbf{q}_4 v) \\ \mathbf{f}_5(u,v) &= e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} cos(\mathbf{q}_2 u) cos(\mathbf{q}_4 v) \\ \mathbf{f}_6(u,v) &= e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} cos(\mathbf{q}_2 u) sin(\mathbf{q}_4 v) \\ \mathbf{f}_7(u,v) &= e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} sin(\mathbf{q}_2 u) cos(\mathbf{q}_4 v) \\ \mathbf{f}_8(u,v) &= e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} sin(\mathbf{q}_2 u) sin(\mathbf{q}_4 v) \\ \mathbf{f}_9(u,v) &= e^{-\mathbf{q}_2 u} e^{\mathbf{q}_4 v} cos(\mathbf{q}_2 u) cos(\mathbf{q}_4 v) \\ \mathbf{f}_{10}(u,v) &= e^{-\mathbf{q}_2 u} e^{\mathbf{q}_4 v} cos(\mathbf{q}_2 u) sin(\mathbf{q}_4 v) \\ \mathbf{f}_{11}(u,v) &= e^{-\mathbf{q}_2 u} e^{\mathbf{q}_4 v} sin(\mathbf{q}_2 u) cos(\mathbf{q}_4 v) \\ \mathbf{f}_{12}(u,v) &= e^{-\mathbf{q}_2 u} e^{\mathbf{q}_4 v} sin(\mathbf{q}_2 u) sin(\mathbf{q}_4 v) \\ \mathbf{f}_{13}(u,v) &= e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} cos(\mathbf{q}_2 u) cos(\mathbf{q}_4 v) \\ \mathbf{f}_{14}(u,v) &= e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} cos(\mathbf{q}_2 u) sin(\mathbf{q}_4 v) \\ \mathbf{f}_{15}(u,v) &= e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} sin(\mathbf{q}_2 u) cos(\mathbf{q}_4 v) \\ \mathbf{f}_{16}(u,v) &= e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} sin(\mathbf{q}_2 u) sin(\mathbf{q}_4 v)\end{aligned} \tag{4.20}$$

Eq. (4.19) can be transformed into:

$$\mathbf{X}(u,v) = \sum_{j=1}^{16} \mathbf{d}_j \mathbf{f}_j(u,v) \tag{4.21}$$

Table 4.1: 16 points used to define the surface.

(x, y, z)	(x, y, z)	(x, y, z)	(x, y, z)
$(0.05, -0.07, -1.12)$	$(0.07, -0.07, -1.11)$	$(0.10, -0.07, -1.11)$	$(0.13, -0.07, -1.12)$
$(0.04, -0.10, -1.10)$	$(0.07, -0.10, -1.09)$	$(0.10, -0.10, -1.10)$	$(0.13, -0.10, -1.10)$
$(0.04, -0.13, -1.08)$	$(0.07, -0.13, -1.06)$	$(0.10, -0.13, -1.07)$	$(0.13, -0.13, -1.09)$
$(0.04, -0.16, -1.05)$	$(0.07, -0.16, -1.04)$	$(0.10, -0.16, -1.04)$	$(0.13, -0.16, -1.06)$

where $\mathbf{d}_j$, $(j = 1, \ldots, 16)$ are the vector-valued unknowns. Note that $\mathbf{X}(u, v)$ in Eq. (4.21) is a parametric surface, which defines a 4-sided PDE patch. The unknowns $\mathbf{d}_j$, $(j = 1, \ldots, 16)$ can be calculated by using the linear least square method.

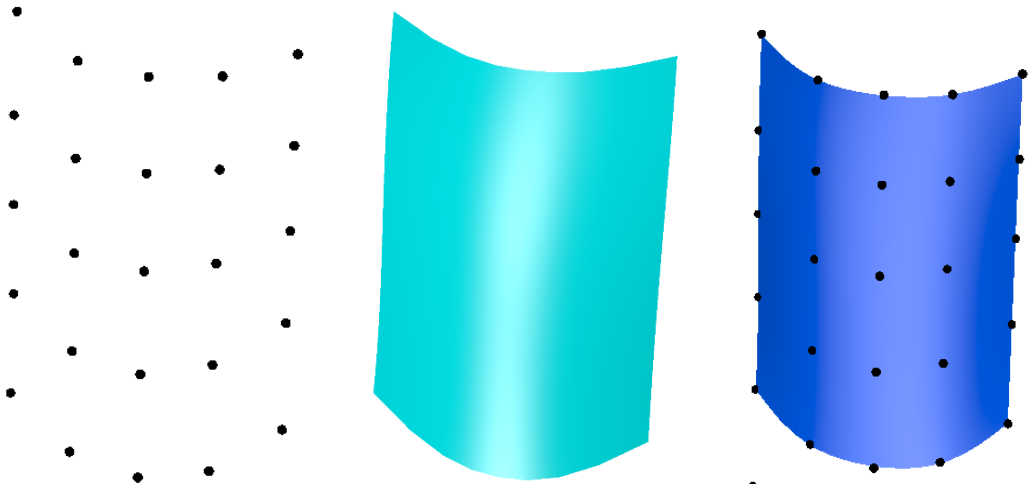
4.2 Reconstruction from a single patch of points

Firstly, we consider reconstructing a surface from 16 points given in Table 4.1. Since a PDE surface patch (defined by Eq. (4.21)) involves 16 unknowns, the PDE surface patch should pass through the 16 points if the interpolation operation is used to determine the 16 unknowns. In this chapter, the fitting operation, not the interpolation operation, is used to determine the 16 unknowns. Although the interpolation operation is not used, it is expected that the fitting operation should give high accuracy if not passing the 16 points. The original surface defined by the 16 points and the reconstructed PDE-based parametric surface is shown in Fig. 4.1. Comparing the two surfaces, we could not find any differences, indicating the reconstructed PDE surface is the same as the original surface defined by the 16 points. This observation is also supported by the maximum error and average error given in Table 4.2. The maximum error between the two surfaces is 5.52×10^{-5} and the average error between the two surfaces is 2.31×10^{-5}. Both errors are very small, which indicates high accuracy of the fitting operation.

Similarly, we consider reconstructing a surface from 25, 36, 49, 64 and 81 points respectively, their original surfaces defined by those points and the reconstructed PDE-based parametric surfaces are shown in Fig. 4.2 to

(a) (b) (c)

Figure 4.1: (a) Input 16 points. (b) surface defined by 16 original points. (c) the reconstructed PDE Surface.

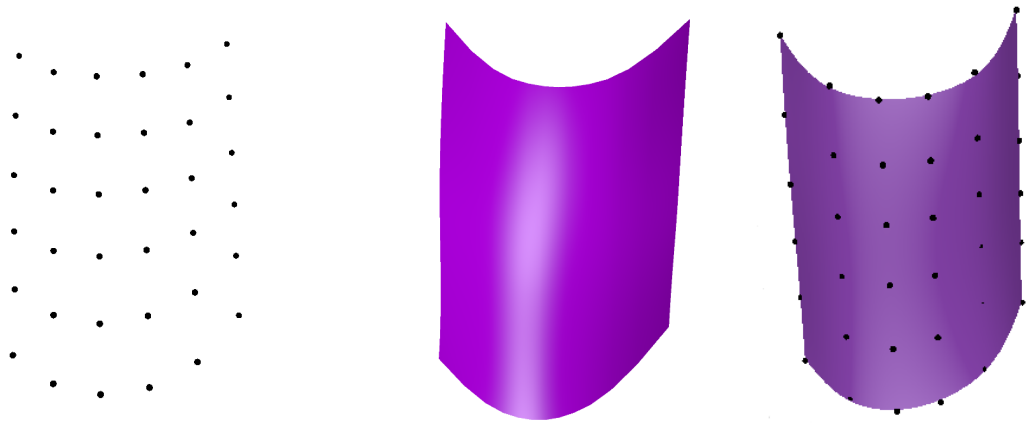


(a) (b) (c)

Figure 4.2: (a) Input 25 points. (b) surface defined by 25 original points. (c) the reconstructed PDE Surface.

Table 4.2: Maximum errors and average errors between the two surfaces.

N	ErrM	ErrA
16	5.52×10^{-5}	2.31×10^{-5}
25	1.56×10^{-3}	4.47×10^{-4}
36	2.96×10^{-3}	1.33×10^{-3}
49	9.68×10^{-3}	3.80×10^{-3}
64	1.40×10^{-2}	5.76×10^{-3}
81	2.02×10^{-2}	8.16×10^{-3}

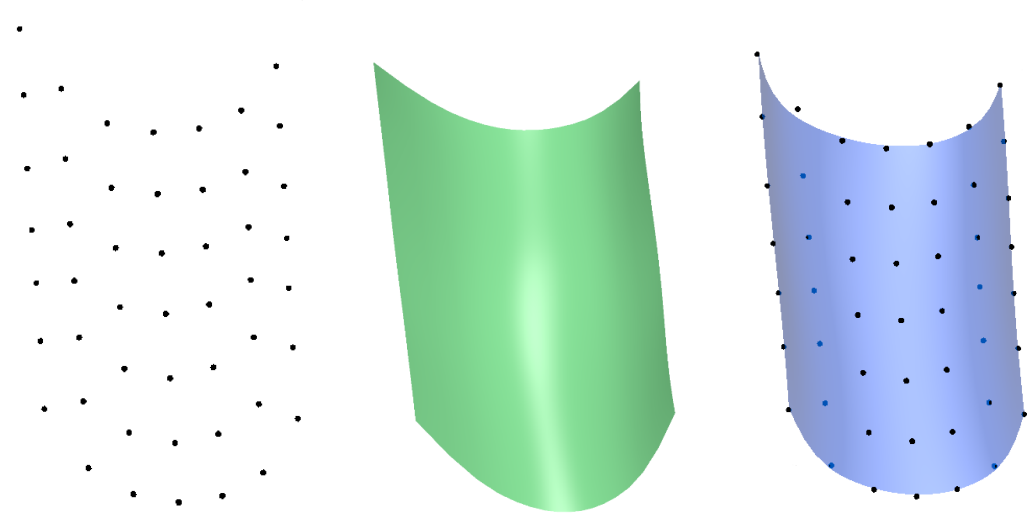


(a) (b) (c)

Figure 4.3: (a) Input 36 points. (b) surface defined by 36 original points. (c) the reconstructed PDE Surface.

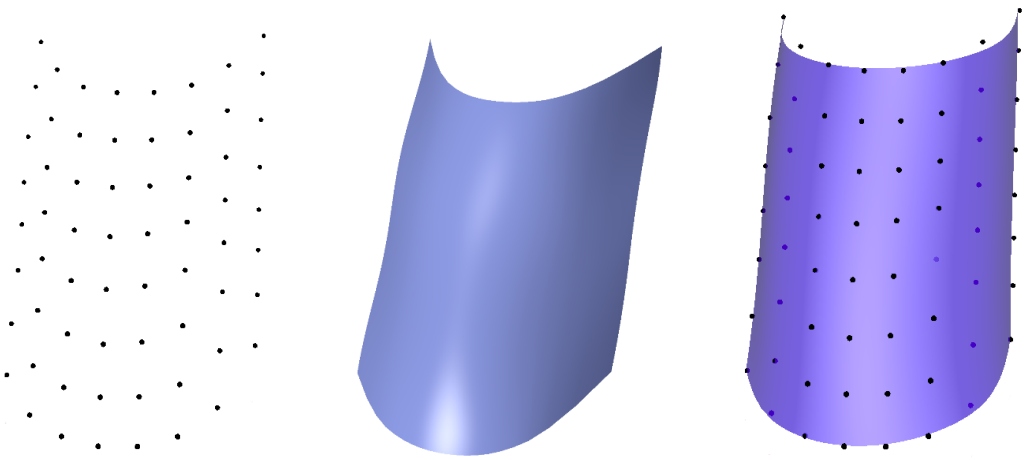
Fig. 4.6. The maximum error and the average error between the two surfaces are given in Table 4.2, and we can see the fitting errors in all cases are small, indicating the powerful capability in parametric surface reconstruction of our proposed method.

Note that the data points used in this study are sampled from a nose model, which is defined by the 81 points shown in Fig. 4.7 (a). The surfaces in the middle from Fig. 4.1 to Fig. 4.6 are generated by lofting the curves defined by these points, as demonstrated in Fig. 4.7 (b). The 16, 25, 36, 49, 64 and 81 points are extracted from the full set of 81 points in Fig. 4.7 (a),



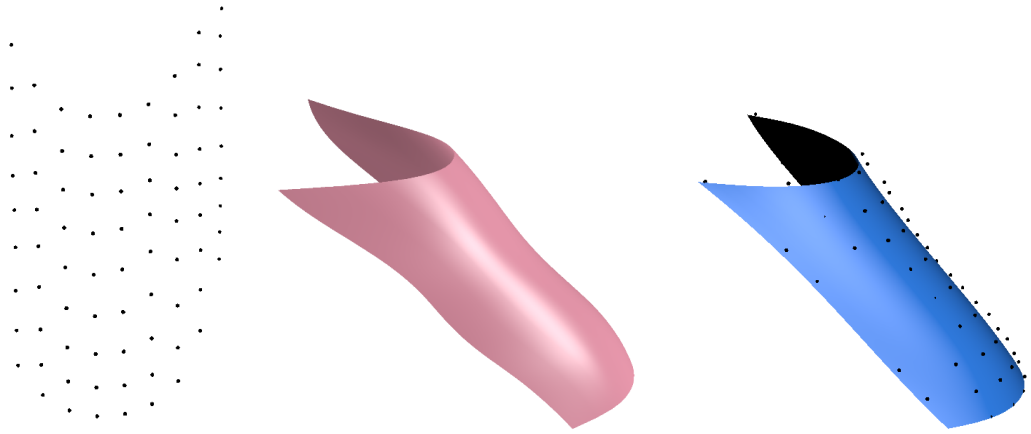
(a) (b) (c)

Figure 4.4: (a) Input 49 points. (b) surface defined by 49 original points. (c) the reconstructed PDE Surface.



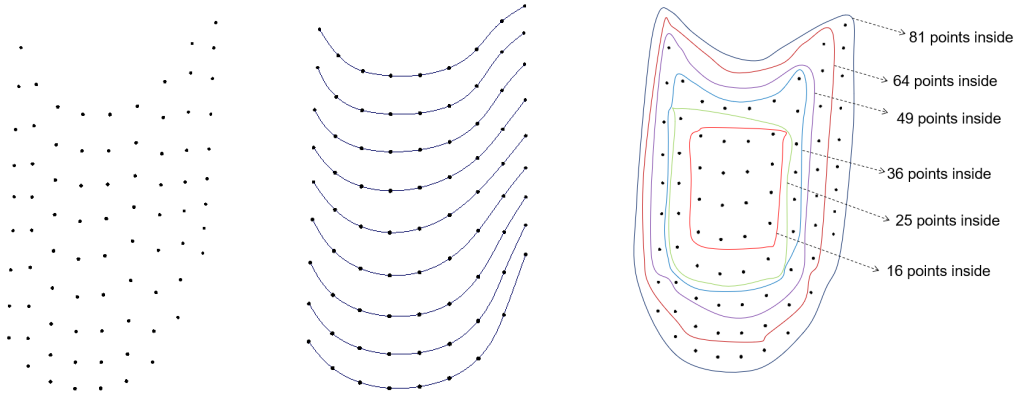
(a) (b) (c)

Figure 4.5: (a) Input 64 points. (b) surface defined by 64 original points. (c) the reconstructed PDE Surface.



(a) (b) (c)

Figure 4.6: (a) Input 81 points. (b) surface defined by 81 original points. (c) the reconstructed PDE Surface.



(a) (b) (c)

Figure 4.7: (a) whole 81 points from a nose model. (b) curves defined by the input points. (c) 16, 25, 36, 49, 64 and 81 are sampled from the 81 points.

with the extraction process visualised in Fig. 4.7 (c). As more points are included, the shapes defined by the points become more complex, enabling a thorough evaluation of our method's fitting capability.

As discussed in Section 4.1, $\mathbf{X}_4(u, v)$ is chosen for reconstruction instead of $\mathbf{X}_1(u, v)$, $\mathbf{X}_2(u, v)$, and $\mathbf{X}_3(u, v)$ because it offers slightly enhanced fitting performance. To validate this, we apply all four models - $\mathbf{X}_1(u, v)$, $\mathbf{X}_2(u, v)$, $\mathbf{X}_3(u, v)$ and $\mathbf{X}_4(u, v)$ - to fit the data points from Fig. 4.1 to Fig. 4.6. The mean square errors (MSE) are computed and compared, given that MSE serves as the criterion in the linear least squares method used for this fitting process. A smaller MSE indicates a more effective model. As shown in Table 4.3, $\mathbf{X}_4(u, v)$ consistently produces the lowest fitting errors across all datasets, supporting its selection.

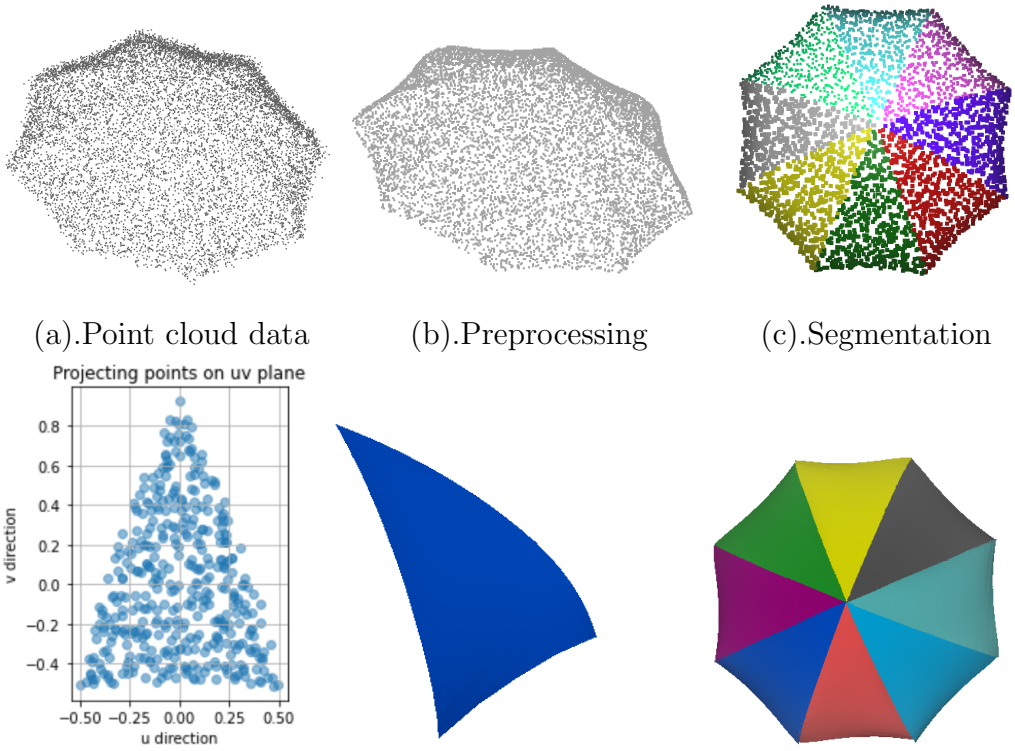
Table 4.3: Mean fitting errors of the four solutions on various input data

	$\mathbf{X}_1(u, v)$	$\mathbf{X}_2(u, v)$	$\mathbf{X}_3(u, v)$	$\mathbf{X}_4(u, v)$
16	4.9378×10^{-4}	4.1927×10^{-5}	1.2499×10^{-4}	$\mathbf{2.31 \times 10^{-5}}$
25	2.3544×10^{-3}	4.4776×10^{-4}	5.2367×10^{-4}	$\mathbf{4.4668 \times 10^{-4}}$
36	2.4141×10^{-3}	1.3359×10^{-3}	1.3428×10^{-3}	$\mathbf{1.3331 \times 10^{-3}}$
49	4.0697×10^{-3}	3.8148×10^{-3}	3.8288×10^{-3}	$\mathbf{3.7957 \times 10^{-3}}$
64	6.3903×10^{-3}	5.8645×10^{-3}	5.8476×10^{-3}	$\mathbf{5.76 \times 10^{-3}}$
81	8.3199×10^{-3}	8.1952×10^{-3}	8.2388×10^{-3}	$\mathbf{8.16 \times 10^{-3}}$

4.3 Reconstruction from multiple patches of points

Since it is difficult to reconstruct a complicated 3D shape with only a single PDE patch, combining multiple PDE patches enables any complicated 3D shape to be reconstructed. In this section, complex 3D shapes will be reconstructed using the proposed PDE method.

The general pipeline of our proposed surface reconstruction method consists of six steps as shown in Fig. 4.8. For an input point cloud shown



(a).Point cloud data (b).Preprocessing (c).Segmentation

(d).Parameterization (e).Patch fitting (f).Reconstructed surface

Figure 4.8: The pipeline of surface reconstruction from point clouds.

in Fig. 4.8(a), preprocessing is carried out to change the input point cloud into the one shown in Fig. 4.8(b). Then, the preprocessed point cloud in Fig. 4.8(b) is segmented into some subsets depicted in Fig. 4.8(c). Next, the points in each of the subsets are parameterized to obtain their parametric values in Fig. 4.8(d). After that, one 4-sided PDE patch in Fig. 4.8(e) is used to fit the points in each of the subsets (note that the 3-sided patch in that figure is a particular case of 4-sided patches). After fitting the points in all the subsets, a reconstructed 3D shape shown in Fig. 4.8(f) consisting of multiple 4-sided PDE patches is obtained.

4.3.1 Segmentation of point clouds

Like mesh segmentation, there are also a lot of works focusing on the segmentation of point clouds. The survey paper in (Nguyen and Le 2013) classified

the existing methods into five main groups: *edge-based methods* (which detect the boundaries of several regions to obtain segmented regions), *region-based methods* (which group nearby points with similar properties while finding dissimilarities for different regions), *attributes-based methods* (which rely on clustering attributes to segment the point clouds), *model-based methods* (which use geometric primitive shapes for assigning points to regions), and *graph-based methods* (which describe the point clouds in terms of graphs and apply standard techniques for graph clustering). As remarked by the authors, all these methods can roughly grouped into two main approaches: a first (mostly mathematical) approach based on geometric processing techniques combined with linear or nonlinear fitting models, and a second (mostly computational) approach based on 3D features extraction and machine learning techniques to learn different classes of objects subsequently used to classify the data points using the extracted features. The first approach provides reasonable computation times and good results for simple 3D scenes, but is sensitive to noise, and does not work well for complex 3D scenes. In such cases, machine learning techniques are usually preferred, as they provide better results although they tend to be slower and rely on the quality of the feature extraction process.

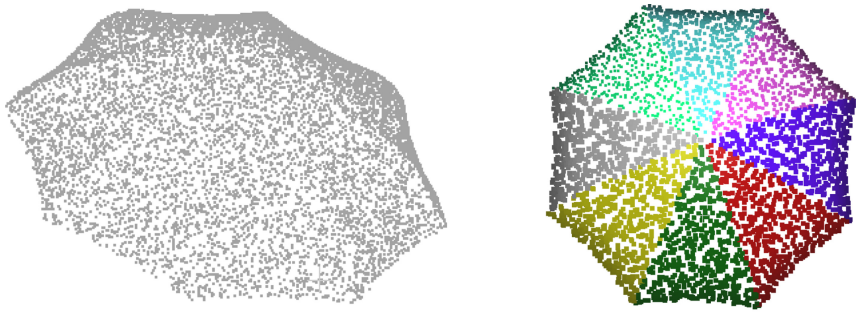
Classical segmentation methods are RANSAC (RANdom SAmple Consensus) (Schnabel et al. 2007), often combined with clustering methods (e.g. K-means, DBSCAN) (Shi et al. 2011) and Hough transform (Illingworth and Kittler 1988). An example of the application of RANSAC combined with K-means will be discussed later for the PDE-based shape reconstruction of a table.

Recently, it has been shown that deep learning techniques can be valuable tools to address these issues. As a consequence, deep learning has been increasingly introduced into point cloud segmentation. A nice survey on this topic can be found in (Guo et al. 2020), where they classified the existing 3D point cloud segmentation methods into three categories: semantic segmentation, instance segmentation, and part segmentation. One of the most commonly adopted methods is applying different neural networks to part-segmentation of point clouds. For example, PointNet is a very popular

neural network architecture, which applies a specialised deep neural network to point clouds for several tasks, including object classification and part segmentation (Qi et al. 2017). Later work also proposed many methods to improve the performance of part segmentation of point clouds. The graph convolution is one of the most used neural network architectures. Please refer to (He et al. 2021) and references within for a comprehensive review of deep learning-based 3D segmentation.

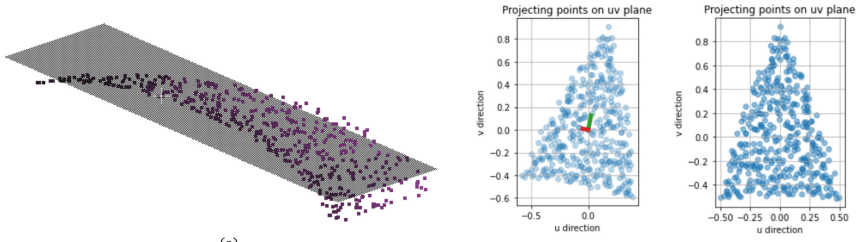
The above methods have a problem when they are used to segment a point cloud into some subsets. The problem is that the boundaries between the two reconstructed patches are not very smooth because the points in the regions around the boundaries are sparse and irregular. To tackle such a problem, edge-based methods for segmentation can be an effective tool. For instance, Bazazian proposed a weakly supervised learning approach to detect the edges of point clouds (Bazazian and Parés 2021).

We will reconstruct several objects using our proposed PDE-based method, including a sphere, a cylinder, a table, an umbrella and a car. The point clouds of these objects can be segmented automatically or manually into some subsets. Specifically, the point sets of the sphere and the cylinder can be segmented into several subsets automatically based on the coordinates of the point sets. As for the point set of the table, which is composed of planes, an automatic method can also be applied to obtain several subsets (planes), which will be introduced in detail in the experimental section. Lastly, we perform segmentation of the point sets of the umbrella and the car examples using software called *CloudCompare* by following a region-based approach. We remark that it is also possible to perform automatic segmentation of these point sets using more sophisticated techniques, such as point-based part segmentation, as described above. Figure 4.9 shows the original point cloud and the segmented result of an umbrella that we will reconstruct later. The points in each of the subsets will be parameterized, and our proposed PDE-based reconstruction method will be applied to reconstruct a PDE patch by fitting these points. The parameterization of point clouds and the fitting process will be described in detail in the following subsections.



(a) (b)

Figure 4.9: Segmentation of a point cloud of the umbrella example: (a) Original point set; (b) segmented point subsets.

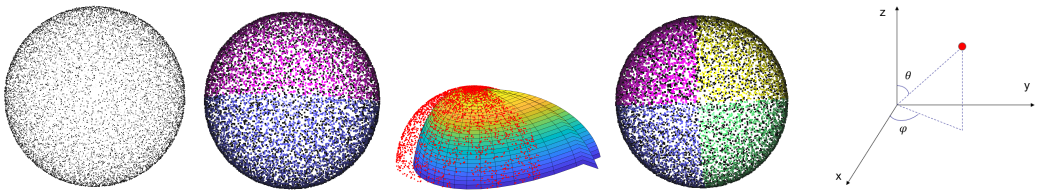


(a) (b) (c)

Figure 4.10: Parameterization of points in a subset: (a) Fitting plane. (b) Projecting points to a plane. (c) Aligned projected points with u and v direction.

4.3.2 Point cloud parameterization

Proper parameterization of point clouds has a big impact on the final reconstruction quality. Various parameterization methods have been proposed in the literature, as discussed in Chapter 2. When the point cloud is scattered, the approach, based on projecting the data points on a base surface may be a better choice. An illustrative example of this process is shown in Fig. 4.10 where the base surface is a simple plane. In this Chapter, we apply different parameterization methods to different types of models. For relatively simple geometry primitives such as a sphere and a cylinder, we follow a model-based approach: since they have an analytical representation, we use their parametric mathematical equations to achieve their parameterization. For example, a sphere can be parameterized with two angles in a spherical coordinate system, which can be normalised to get the parametric values



(a) (b) (c) (d) (e)

Figure 4.11: (a) Point cloud of a sphere. (b) Segment the point clouds into 2 equal subsets. (c) Reconstructed shape with 2 PDE patches. (d) Segment the point clouds into 4 equal subsets. (e) The spherical coordinate system.

of parametric variables u and v on the unit interval for the points on the sphere. Since the points on a whole sphere cannot be reconstructed with a single PDE patch, the point cloud representing a sphere is initially segmented into two equal subsets, as shown in Fig. 4.11(b), which can readily be done automatically. However, in our experiments, we found that the points in the two subsets still cannot be well reconstructed by two PDE patches, as Fig. 4.11(c) demonstrates, the reconstructed surface does not fit the upper subset of points well. Consequently, we further divide the sphere into four equal subsets, as depicted in Fig. 4.11(d) or eight equal subsets. We find that the four-subset approach yields better reconstruction quality while also requiring half the number of design variables compared to the eight-subset approach. Finally, the points in each of the four subsets are parameterised using a spherical coordinate system, as shown in Fig. 4.11(e), to obtain their corresponding u and v parametric values.

A cylinder also has a parametric mathematical expression. Similar to the parameterization of a sphere, we segment the point cloud defining a cylinder into 2 subsets. For the points in each of the two subsets, we obtain the parametric values of two parametric variables u and v from the two variables angle and height defining the points in each of the two subsets.

The third model we reconstruct is a table that is completely composed of planes. Segmenting the table into planes can also be done automatically. The details of the automatic segmentation of the table will be given in the next section. For each segmented plane, it can be regarded as a $u - v$ plane.

Normalising the coordinates of the points on every plane would give us the parametric values of the parametric variables u and v for each point on the segmented plane. Then, the points on each of the planes are used to reconstruct a PDE patch. Combining all the reconstructed PDE patches generates the final reconstructed 3D shape of the table.

Next, an umbrella is also reconstructed. The point set is firstly segmented into eight equal subsets, as shown in Fig. 4.9. For the points in each of the segmented subsets, we find a plane that best fits the point set, as shown in Fig. 4.10(a). Then the 3D points in the subset are projected onto the plane, which can also be regarded as a $u - v$ plane. Since the principal component analysis (PCA) axes of the projected points on the $u - v$ plane do not align with the u and v directions as shown in Fig. 4.10(b), we apply a rotation transformation to the projected points to make their PCA axes aligning with the u and v directions as shown in Fig. 4.10(c). Finally, normalising operations are applied to the points on the $u - v$ plane to obtain the parametric values of the points in each of the segmented subsets. A similar approach has also been applied to the car example. We omit the details here to avoid unnecessary duplication of material.

In the case of very complicated 3D shapes, the previous parameterization methods cannot be ensured to work properly. In such cases, a segmentation procedure is first required to segment the whole point cloud into an appropriate number of multiple subsets, each of which is approximated by a PDE surface patch. Then, the point cloud of each particular patch is projected onto a base surface for coarse parameterization. The simplest solution is to select a fitting plane that reflects well the distribution of points in the 3D space (e.g., the PCA plane). However, this approach fails in case data points cannot be projected in an unambiguous way. A possible solution is to segment the subsets even further to remove the ambiguities of the projecting plane. The counterpart of this approach is that the complexity of the model is increased. An alternative solution is to consider other potential base surfaces, such as cylinders or Coons patches. A third way is to apply evolutionary methods to compute a suitable parameterization of the point cloud. This approach has already been successfully applied to the cases of B-splines and

NURBS surfaces (Gálvez et al. 2012, Gálvez and Iglesias 2012) and could be generalised to the case of PDE patches. The drawback is that such methods are generally slow and might require high computational resources. Finally, deep learning can also be applied to point cloud parameterization. For instance, the ParSeNet deep network in (Sharma et al. 2020) applies a neural architecture of edge convolution layers performing graph convolution combined with max pooling to extract a global representation of the point cloud. Then, a mean-shift clustering method is combined with segment classification to decompose a 3D point cloud into parametric surface patches (fitted with a SplineNet network) and basic geometric primitives (determined through least-squares fitting). The resulting patches can be used as suitable base surfaces to address data parameterization of the segmented subsets with PDE patches. Furthermore, the ParSeNet is also capable of generating robust and reliable parameterizations, which could also be used directly in some cases. Although this approach is powerful and can handle complex shapes, it relies on the ability of the network to learn complex shapes through intensive training on a large dataset of object instances. It also requires post-processing optimisation for high accuracy.

To summarise, point cloud parameterization of complex shapes is a challenging issue and there is no universal solution so far for this problem. Note, however, that our method is general and does not preclude any parameterization method to be readily embedded into our approach.

4.3.3 Fitting

After parameterizing the points in each of the segmented subsets, we obtained their parametric values u_n and v_n for each point $\mathbf{X}_n$ in the subsets. Then we fit the PDE patch to the points. As discussed above, applying our developed PDE patch to surface reconstruction from point clouds requires finding the 16 vector-valued unknowns $\mathbf{d}_j$, $(j = 1, 2, 3, \ldots, 16)$ so that the PDE patch $\mathbf{X}(u, v)$ will best approximate the points in the subset.

If there are N points $\mathbf{X}_n$ $(n = 1, 2, 3, \ldots, N)$ in a subset to be reconstructed by one PDE patch $\mathbf{X}(u, v)$, the squared sum of the errors between

the known points $\mathbf{X}_n$ and the unknown points $\mathbf{X}(u_n, v_n)$ can be determined with the following equation:

$$\mathbf{E} = \sum_{n=1}^{N} [\mathbf{X}(u_n, v_n) - \mathbf{X}_n]^2 = \sum_{n=1}^{N} \left[\sum_{j=1}^{16} \mathbf{d}_j \mathbf{f}_j(u_n, v_n) - \mathbf{X}_n \right]^2 \tag{4.22}$$

To minimise the error $\mathbf{E}$ and find the 16 vector-valued unknowns, we apply the method of least squares, given by the following equation:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{d}_k} = 0 \qquad (k = 1, 2, 3, \ldots, 16) \tag{4.23}$$

Substituting Eq. (4.22) into Eq. (4.23), the following system of equations is obtained:

$$\sum_{j=1}^{16} \mathbf{d}_j \sum_{n=1}^{N} \mathbf{f}_j(u_n, v_n) \mathbf{f}_k(u_n, v_n) = \sum_{n=1}^{N} \mathbf{X}_n \mathbf{f}_k(u_n, v_n) \tag{4.24}$$

for $k = 1, 2, 3, \ldots, 16$. Therefore, there are 16 equations in Eq. (4.24) that must be solved to find the 16 vector-valued unknowns $\mathbf{d}_k$ $(k = 1, \ldots, 16)$.

Note that in Eq. (4.24), $\mathbf{f}_j(u_n, v_n)$ and $\mathbf{f}_k(u_n, v_n)$ depend on the constants $\mathbf{q}_2$ and $\mathbf{q}_4$, as indicated in Eq. (4.20). These constants can be considered as design variables and optimised to obtain the optimal PDE patch that best fits the points $\mathbf{X}_n$ $(n = 1, \ldots, N)$. However, treating $\mathbf{q}_2$ and $\mathbf{q}_4$ as design variables would make the minimisation of Eq. (4.22) a nonlinear problem, thereby increasing the complexity of solving for the 16 vector-valued unknowns $\mathbf{d}_k$. To address this challenge, we simplify the approach in this paper by treating $\mathbf{q}_2$ and $\mathbf{q}_4$ as constants and assigning $\mathbf{q}_2 = \mathbf{q}_4 = \mathbf{0.1}$, which has proven effective across all our experiments.

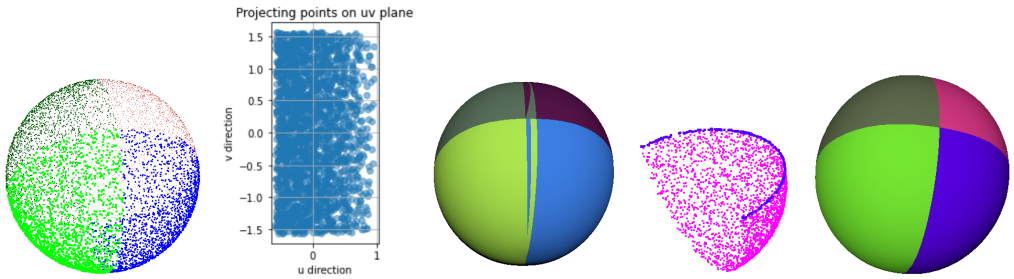
4.3.4 Experiments and results

For the points in each of the subsets obtained from segmenting a point cloud, we use Eq. 4.21 to reconstruct a PDE patch $\mathbf{X}(u, v)$. After the points in all the subsets have been used to reconstruct PDE patches, the reconstructed 3D shape consisting of the reconstructed PDE patches is obtained. In what follows, we present some examples to illustrate this process.

The first example is to reconstruct a sphere from its point cloud. As discussed above, the point cloud describing a sphere is segmented into four subsets as shown in Fig. 4.12(a). For the points in each of the four subsets, Eq. (4.21) is used to reconstruct one PDE patch from the points. To do this, the points in the subset are projected to a $u - v$ plane as shown in Fig. 4.12(b). After the four PDE patches have been reconstructed, they are automatically connected together to represent the reconstructed shape from the point cloud. However, we found that the reconstructed patches have small overlaps between two PDE patches such as those between the grey and red patches and between the green and blue patches as shown in Fig. 4.12(c). To address the problem, we increase the density of points around the boundaries. This is done by first upsampling the original point cloud, as illustrated in Fig. 4.13(b), then keeping the points around the boundaries using a boundary detection algorithm based on the coordinates of the point cloud, and finally combining the extracted boundary points with the original point cloud, which are demonstrated in Fig. 4.13(c) and (d). Increasing the number of points near the boundary effectively increases their weight in the fitting process, leading to better continuity between adjacent surfaces. As shown in Fig. 4.12(e), this adjustment successfully eliminates small overlaps and improves reconstruction quality.

The second example is to reconstruct a cylinder from its point cloud. As discussed in Subsection 4.3.2, the point cloud has been segmented into two equal subsets as shown in Fig. 4.14(a). Then, the points in each of the subsets are projected to a $u-v$ plane shown in Fig. 4.14(b). After using Eq. (4.21) to reconstruct the two PDE patches shown in Fig. 4.14(c), small overlaps occur again. With the same treatment as discussed above, we add more points in the regions around the boundaries between the two PDE patches as shown in Fig. 4.14(d). By adding new points, the small overlaps disappear as shown by the two reconstructed PDE patches depicted in Fig. 4.14(e).

The third example is to reconstruct a table from its point cloud. The points in the point cloud of a table are on some planes. This example is used to demonstrate that our proposed PDE-based method can be used to reconstruct not only 3D shapes consisting of curved surfaces but also 3D



(a) (b) (c) (d) (e)

Figure 4.12: PDE-based reconstruction from the point cloud of a sphere: (a) Segmented point cloud of a sphere. (b) Projecting the points in a subset to a u-v plane. (c) Reconstructed shape with small overlaps consisting of four PDE patches without adding points in the regions around boundaries. (d) The points in a subset after adding points to the regions around boundaries. (e) The final result without overlaps is obtained by adding points to the regions around boundaries.

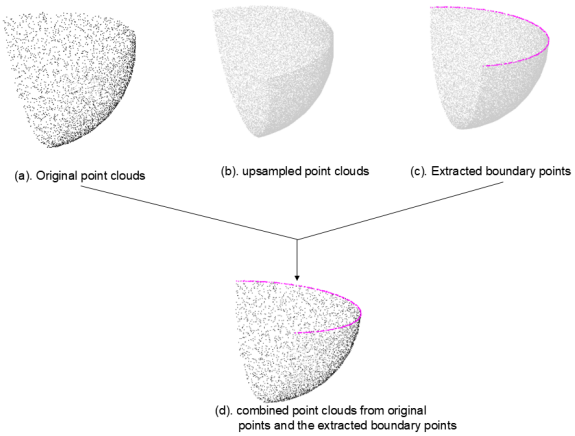
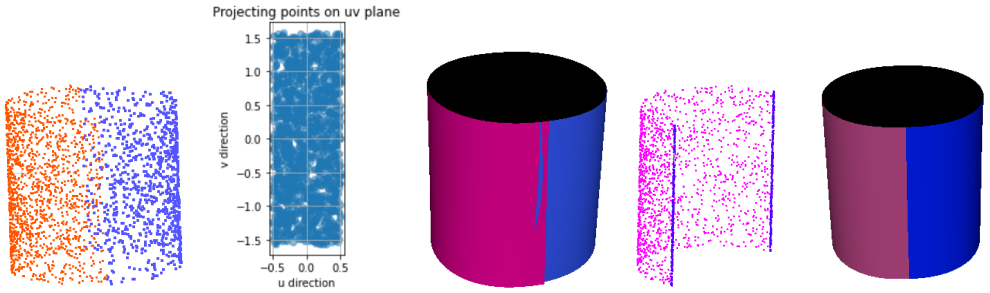


Figure 4.13: Adding more points around the boundary of the original point clouds: (a) original point cloud; (b) upsampled point clouds; (c) extracted boundary points of the upsampled point clouds; (d) combined point clouds.



(a) (b) (c) (d) (e)

Figure 4.14: PDE-based reconstruction from the point cloud of a cylinder: (a) Segmented point cloud of a cylinder. (b) Projecting the points in a subset to a uv plane. (c) Reconstructed shape with small overlaps consisting of two PDE patches without adding points in the regions around boundaries. (d) The points in a subset after adding points to the regions around boundaries. (e) The final result without overlaps obtained by adding points to the regions around boundaries.

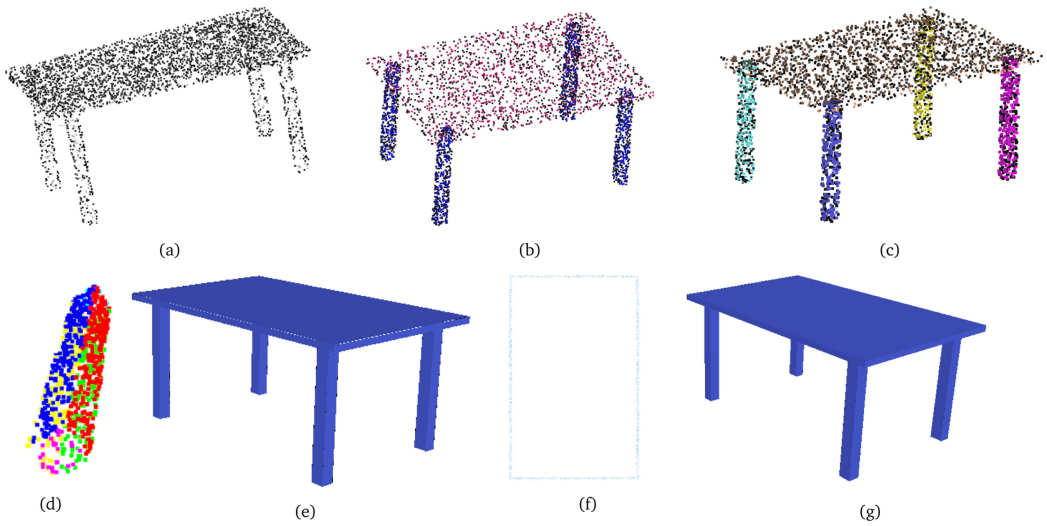


Figure 4.15: PDE-based reconstruction from the point cloud of a table (left–right, top–bottom): (a) Point cloud of a table. (b) Segmenting the point cloud of the table into a top part and a bottom part. (c) Using the K-means clustering algorithm to segment the bottom part into four subparts. (d) Using the RANSAC algorithm to segment each of the subparts into six subsets. (e) Reconstructed shape with small gaps between two adjacent PDE patches. (f) Points in a subset after adding points to the regions around boundaries. (g) The final result without gaps obtained by adding points to the regions around boundaries.

(a) (b)

Figure 4.16: (a) Reconstructed umbrella; (b) Final result after post-processing.

shapes consisting of flat planes or a combination of curved surfaces and flat planes. Even though a plane seems to be a simple primitive, plane detection and reconstruction are important in many fields such as robotic perception and image processing. For example, robots need to detect ground where it is safe to walk, as well as the walls and other artificial (and mostly linear and flattened) areas for collision avoidance. For a point cloud of this type, it usually takes three steps to segment a point cloud. In the first step, we segment the point cloud in Fig. 4.15(a) of the table into the top part and leg part shown in Fig. 4.15(b). Then in the second step, we use a K-means clustering algorithm for the leg part and change it into four subparts shown in Fig. 4.15(c). Each subpart represents a leg. Finally, in the third step, we use a plane detection algorithm called RANSAC, which is a state-of-the-art plane detector, from Point Cloud Library (PCL) to segment each of the four subparts into six subsets, which are six planes of a leg. With this treatment, the points in each of the subparts are segmented into those in the six subsets (planes) of a leg as shown in Fig. 4.15(d). For the points in each of the segmented subsets, we reconstruct a PDE patch by applying our developed PDE-based reconstruction method. Fig. 4.15(e) shows the reconstructed PDE patches from the point cloud of the table. It can be observed that there are very small gaps between reconstructed PDE patches. To tackle this problem, we add more points in the regions around the boundaries.

Fig. 4.15(f) shows the result of adding more points in the regions around the boundaries of a plane. The final result after adding more points is shown in Fig. 4.15(g), which removes the gaps and is of good quality.

The fourth example is to reconstruct an umbrella from its point cloud. The point cloud of the umbrella is segmented into eight subsets. The points in each of the eight subsets are projected to a $u - v$ plane. Since the projected points do not fill the top left region and the top right region as shown in Fig. 4.10(c), trimming the reconstructed PDE patch corresponding to the two regions is necessary. Fig. 4.16(a) shows the reconstructed umbrella after trimming the reconstructed PDE patches. As we can see, there are some very small gaps. This problem can be addressed by sampling more points in the regions around the boundaries of subsets. In this example, we manually eliminate the gaps and obtain the result shown in Fig. 4.16(b).

Finally, a car model is reconstructed to further demonstrate that our method can also be applied to reconstruct more complex 3D shapes. Note that the car model is symmetric, so only half of the model is to be reconstructed as the other half can readily be obtained by mirroring the reconstructed result. Fig. 4.17(a) shows the point clouds of a car, while Fig. 4.17(b) shows the segmented result of the point clouds. Fig. 4.17(c) shows the result when we consider the point clouds in the back part of the car as a whole subset, which is not visually good. To improve the quality, we keep segmenting the point clouds in this part into three smaller subsets. The reconstructed result is shown in Fig. 4.17(d). As the reader can see, more details are captured compared to Fig. 4.17(c). As expected, the reconstructed accuracy usually improves as the number of subsets increases. So, for more complicated 3D shapes, we can proceed by increasing the number of subsets until the final result meets the desired accuracy. However, there are still some artefacts in Fig. 4.17(d) as some small gaps exist between some adjacent reconstructed patches. To remove them, we apply post-processing to the reconstructed result by filling the gaps. Figs. 4.17(e) and (f) show the final results from both the rear view and the front view, respectively.

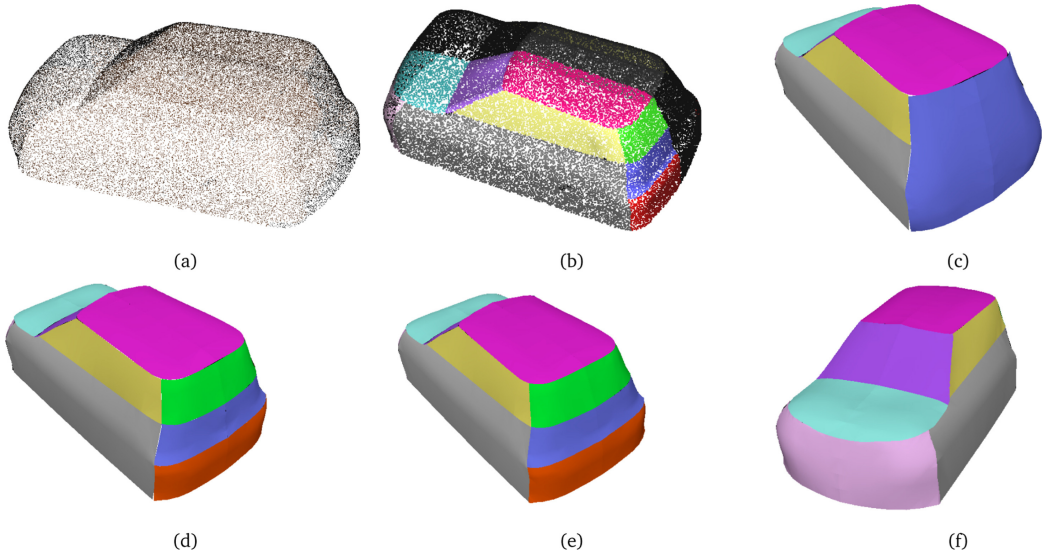


Figure 4.17: PDE-based reconstruction from the point cloud of a car (left–right, top–bottom): (a) original point cloud; (b) segmented point cloud; (c) rear part as a single subset; (d) segmentation refinement of the rear part; (e) final reconstructed rear part after post-processing; (f) final reconstructed front part after post-processing.

4.3.5 Comparison with implicit PDE method

To illustrate the advantages of our proposed PDE-based surface reconstruction technique, we also reconstruct polygon surfaces from the same point clouds used in this paper by applying a widely adopted method called Poisson surface reconstruction (Kazhdan et al. 2006). The main aspect we consider is the number of variables needed to reconstruct the same 3D shape while keeping the reconstructed surface with high quality. The errors between the surface defined by the original points set and the reconstructed surfaces, including the PDE surface and the polygon surface, are calculated to demonstrate the quality of the result. Specifically, we calculate the error between the surface defined by the original point set and the reconstructed PDE surface through the following equations:

$$\begin{aligned} Err_{Mean} &= \frac{1}{N}\sum_{n=1}^{N}|\mathbf{X}_n - \mathbf{X}(u_n, v_n)| \\ Err_{Max} &= \max_{n=1,2,\ldots,N} |\mathbf{X}_n - \mathbf{X}(u_n, v_n)| \end{aligned} \tag{4.25}$$

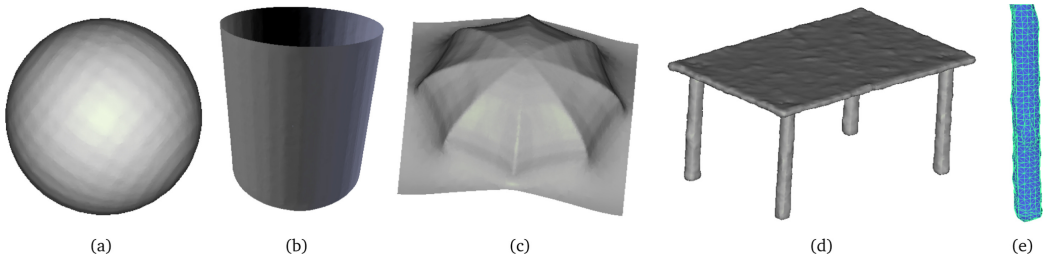


Figure 4.18: 3D shapes reconstructed using the Poisson surface reconstruction technique: (a) Reconstructed sphere; (b) Reconstructed cylinder; (c) Reconstructed umbrella; (d) Reconstructed table; (e) Close view of a leg of the reconstructed table.

where Err_{Mean} indicates the average error between the two surfaces, Err_{Max} stands for the maximum error between the two surfaces, and $|.|$ indicates the Euclidean distance between two points (vectors). For the error between the surface and the reconstructed polygon surface, we take advantage of a tool in a point cloud processing software called CloudCompare and the tool is designed to calculate the difference between two surfaces.

Fig. 4.18 shows the polygon surfaces reconstructed from the point cloud of the sphere, cylinder, table and umbrella, which we used to reconstruct PDE patches in the above subsection. Comparing the reconstructed shapes shown in Fig. 4.18 obtained from the Poisson surface reconstruction algorithm with those obtained with our proposed PDE-based method, it is clear that our proposed PDE-based method has a better quality of reconstructed 3D shapes.

Table 4.4 gives the number of the design variables required by the polygon-based method and PDE-based method to reconstruct different 3D shapes. We can see that for the point clouds defining 3D shapes with curved surfaces, our proposed PDE-based surface reconstruction method requires much fewer design variables than the polygon-based method.

To demonstrate the quality of the reconstructed surface, we use the implicit PDE method to reconstruct the 3D shapes in Table 4.4 to make their vertex number roughly the same as the number of variables needed in the corresponding PDE-based surface, thus the number of variables needed to represent the same 3D shape would be roughly the same. Table 4.5 shows the number of variables required to represent various 3D shapes for both the

polygon-based method and our proposed PDE-based method after simplifying the reconstructed polygon mesh. Then we calculated the average error

Table 4.4: Number of the design variables needed by our proposed PDE-based method and the implicit PDE method to reconstruct 3D shapes from different point clouds.

	Sphere	Cylinder	Table	Umbrella
Implicit PDE	726	192	45,279	86,613
Our method	192	96	1080	384

Table 4.5: Number of variables required to represent different 3D shapes for implicit PDE method and proposed PDE-based method after simplification of the reconstructed polygon mesh.

	Sphere	Cylinder	Table	Umbrella
Implicit PDE	222	96	1131	552
Our method	192	96	1080	384

Table 4.6: The mean error and its deviation after simplification between the surface defined by different point clouds and the reconstructed PDE surface and the polygon surface respectively (a/b: a refers to the mean error, b refers to the maximum error.)

	Sphere	Cylinder	Table	Umbrella
Implicit PDE	0.0091/	0.0045/	0.1083/	0.0120/
	0.0388	0.0224	3.5127	0.5446
Our method	0.0026/	0.0029/	9.0×10^{-6}/	0.0045/
	0.0129	0.0084	5.6×10^{-5}	0.0159

and the maximum error between the surface defined by the original point set and the reconstructed surfaces. Table 4.6 shows the results and we can see that both the mean error and the maximum error between the PDE surface

and the original point set is smaller than that between the polygon surface and the original point set. In conclusion, our proposed PDE-based surface reconstruction method from point clouds outperforms a classical implicit PDE-based surface reconstruction technique from point clouds regarding the number of variables and the quality of the output 3D shape.

4.4 Extended closed-form solution

There are 16 vector-valued unknowns in Eq. (4.20), which can be used to reconstruct a single surface patch. However, complex shapes can rarely be represented accurately through a single patch. Generally, multiple patches are required for complicated shapes, each defined by Eq. (4.20). Unfortunately, this solution may not be adequate in some instances. For example, it is not easy to segment complicated 3D point clouds into an appropriate number of subsets automatically. This problem can be addressed by enlarging the number of unknown variables in Eq. (4.20), thus providing extra degrees of freedom to the problem. To achieve this, we can obtain more complex and powerful solutions by combining the solutions to the fourth-order partial differential equation under different conditions. Then, $X(u, v)$ can be extended to $X^*(u, v)$ as following:

$$\begin{aligned}
\mathbf{X}_4(u,v) = \; & [e^{\mathbf{q}_2 u}\left(\mathbf{c}_1 cos(\mathbf{q}_2 u) + \mathbf{c}_2 sin(\mathbf{q}_2 u)\right) + \\
& e^{-\mathbf{q}_2 u}\left(\mathbf{c}_3 cos(\mathbf{q}_2 u) + \mathbf{c}_4 sin(\mathbf{q}_2 u)\right) + \\
& e^{\mathbf{q}_5 u}\left(\mathbf{c}_5 cos(\mathbf{q}_5 u) + \mathbf{c}_6 sin(\mathbf{q}_5 u)\right) + \\
& e^{-\mathbf{q}_5 u}\left(\mathbf{c}_7 cos(\mathbf{q}_5 u) + \mathbf{c}_8 sin(\mathbf{q}_5 u)\right)] \\
& [e^{\mathbf{q}_4 v}\left(\mathbf{c}_9 cos(\mathbf{q}_4 v) + \mathbf{c}_{10} sin(\mathbf{q}_4 v)\right) + \\
& e^{-\mathbf{q}_4 v}\left(\mathbf{c}_{11} cos(\mathbf{q}_4 v) + \mathbf{c}_{12} sin(\mathbf{q}_4 u)\right) + \\
& e^{\mathbf{q}_6 v}\left(\mathbf{c}_{13} cos(\mathbf{q}_6 v) + \mathbf{c}_{14} sin(\mathbf{q}_6 v)\right) + \\
& e^{-\mathbf{q}_6 v}\left(\mathbf{c}_{15} cos(\mathbf{q}_6 v) + \mathbf{c}_{16} sin(\mathbf{q}_6 v)\right)]
\end{aligned} \tag{4.26}$$

Conducting the multiplication operation in the above equation and letting

$$
\begin{aligned}
&\mathbf{f}_1(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_4 v} \cos(\mathbf{q}_2 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_2(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_4 v} \cos(\mathbf{q}_2 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_3(u,v) = e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} \cos(\mathbf{q}_2 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_4(u,v) = e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} \cos(\mathbf{q}_2 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_5(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_6 v} \cos(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_6(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_6 v} \cos(\mathbf{q}_2 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_7(u,v) = e^{\mathbf{q}_2 u} e^{-\mathbf{q}_6 v} \cos(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_8(u,v) = e^{\mathbf{q}_2 u} e^{-\mathbf{q}_6 v} \cos(\mathbf{q}_2 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_9(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_4 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{10}(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_4 v} \sin(\mathbf{q}_2 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{11}(u,v) = e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{12}(u,v) = e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} \sin(\mathbf{q}_2 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{13}(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_6 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{14}(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_6 v} \sin(\mathbf{q}_2 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_{15}(u,v) = e^{\mathbf{q}_2 u} e^{-\mathbf{q}_6 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{16}(u,v) = e^{\mathbf{q}_2 u} e^{\mathbf{q}_6 v} \sin(\mathbf{q}_2 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_{17}(u,v) = e^{-\mathbf{q}_2 u} e^{\mathbf{q}_4 v} \cos(\mathbf{q}_2 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{18}(u,v) = e^{-\mathbf{q}_2 u} e^{\mathbf{q}_4 v} \cos(\mathbf{q}_2 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{19}(u,v) = e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} \cos(\mathbf{q}_2 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{20}(u,v) = e^{\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} \cos(\mathbf{q}_2 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{21}(u,v) = e^{-\mathbf{q}_2 u} e^{\mathbf{q}_6 v} \cos(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{22}(u,v) = e^{-\mathbf{q}_2 u} e^{\mathbf{q}_6 v} \cos(\mathbf{q}_2 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_{23}(u,v) = e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_6 v} \cos(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{24}(u,v) = e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_6 v} \cos(\mathbf{q}_2 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_{25}(u,v) = e^{-\mathbf{q}_2 u} e^{\mathbf{q}_4 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{26}(u,v) = e^{-\mathbf{q}_2 u} e^{\mathbf{q}_4 v} \sin(\mathbf{q}_2 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{27}(u,v) = e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{28}(u,v) = e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_4 v} \sin(\mathbf{q}_2 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{29}(u,v) = e^{-\mathbf{q}_2 u} e^{\mathbf{q}_6 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{30}(u,v) = e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_6 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v) \\
&\mathbf{f}_{31}(u,v) = e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_6 v} \sin(\mathbf{q}_2 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{32}(u,v) = e^{-\mathbf{q}_2 u} e^{-\mathbf{q}_6 v} \sin(\mathbf{q}_2 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_{33}(u,v) = e^{\mathbf{q}_5 u} e^{\mathbf{q}_4 v} \cos(\mathbf{q}_5 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{34}(u,v) = e^{\mathbf{q}_5 u} e^{\mathbf{q}_4 v} \cos(\mathbf{q}_5 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{35}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_4 v} \cos(\mathbf{q}_5 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{36}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_4 v} \cos(\mathbf{q}_5 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{37}(u,v) = e^{\mathbf{q}_5 u} e^{\mathbf{q}_6 v} \cos(\mathbf{q}_5 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{38}(u,v) = e^{\mathbf{q}_5 u} e^{\mathbf{q}_6 v} \cos(\mathbf{q}_5 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_{39}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_6 v} \cos(\mathbf{q}_5 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{40}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_6 v} \cos(\mathbf{q}_5 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_{41}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_4 v} \sin(\mathbf{q}_5 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{42}(u,v) = e^{\mathbf{q}_5 u} e^{\mathbf{q}_4 v} \sin(\mathbf{q}_5 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{43}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_4 v} \sin(\mathbf{q}_5 u) \cos(\mathbf{q}_4 v), \quad \mathbf{f}_{44}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_4 v} \sin(\mathbf{q}_5 u) \sin(\mathbf{q}_4 v) \\
&\mathbf{f}_{45}(u,v) = e^{\mathbf{q}_5 u} e^{\mathbf{q}_6 v} \sin(\mathbf{q}_5 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{46}(u,v) = e^{\mathbf{q}_5 u} e^{\mathbf{q}_6 v} \sin(\mathbf{q}_5 u) \sin(\mathbf{q}_6 v) \\
&\mathbf{f}_{47}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_6 v} \sin(\mathbf{q}_5 u) \cos(\mathbf{q}_6 v), \quad \mathbf{f}_{48}(u,v) = e^{\mathbf{q}_5 u} e^{-\mathbf{q}_6 v} \sin(\mathbf{q}_5 u) \sin(\mathbf{q}_6 v)
\end{aligned}
\tag{4.27}
$$

$$\mathbf{f}_{49}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_4 v}\cos(\mathbf{q}_5 u)\cos(\mathbf{q}_4 v), \quad \mathbf{f}_{50}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_4 v}\cos(\mathbf{q}_5 u)\sin(\mathbf{q}_4 v)$$
$$\mathbf{f}_{51}(u,v) = e^{-\mathbf{q}_5 u}e^{-\mathbf{q}_4 v}\cos(\mathbf{q}_5 u)\cos(\mathbf{q}_4 v), \quad \mathbf{f}_{52}(u,v) = e^{-\mathbf{q}_5 u}e^{-\mathbf{q}_4 v}\cos(\mathbf{q}_5 u)\sin(\mathbf{q}_4 v)$$
$$\mathbf{f}_{53}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_6 v}\cos(\mathbf{q}_5 u)\cos(\mathbf{q}_6 v), \quad \mathbf{f}_{54}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_6 v}\cos(\mathbf{q}_5 u)\sin(\mathbf{q}_6 v)$$
$$\mathbf{f}_{55}(u,v) = e^{-\mathbf{q}_5 u}e^{-\mathbf{q}_6 v}\cos(\mathbf{q}_5 u)\cos(\mathbf{q}_6 v), \quad \mathbf{f}_{56}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_4 v}\cos(\mathbf{q}_5 u)\sin(\mathbf{q}_6 v)$$
$$\mathbf{f}_{57}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_4 v}\sin(\mathbf{q}_5 u)\cos(\mathbf{q}_4 v), \quad \mathbf{f}_{58}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_4 v}\sin(\mathbf{q}_5 u)\sin(\mathbf{q}_4 v)$$
$$\mathbf{f}_{59}(u,v) = e^{-\mathbf{q}_5 u}e^{-\mathbf{q}_4 v}\sin(\mathbf{q}_5 u)\cos(\mathbf{q}_4 v), \quad \mathbf{f}_{60}(u,v) = e^{-\mathbf{q}_5 u}e^{-\mathbf{q}_4 v}\sin(\mathbf{q}_5 u)\sin(\mathbf{q}_4 v)$$
$$\mathbf{f}_{61}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_6 v}\sin(\mathbf{q}_5 u)\cos(\mathbf{q}_6 v), \quad \mathbf{f}_{62}(u,v) = e^{-\mathbf{q}_5 u}e^{\mathbf{q}_6 v}\sin(\mathbf{q}_5 u)\sin(\mathbf{q}_6 v)$$
$$\mathbf{f}_{63}(u,v) = e^{-\mathbf{q}_5 u}e^{-\mathbf{q}_6 v}\sin(\mathbf{q}_5 u)\cos(\mathbf{q}_6 v), \quad \mathbf{f}_{64}(u,v) = e^{-\mathbf{q}_5 u}e^{-\mathbf{q}_6 v}\sin(\mathbf{q}_5 u)\sin(\mathbf{q}_6 v) \tag{4.27}$$

Then Eq. (4.26) can be expressed as follows:

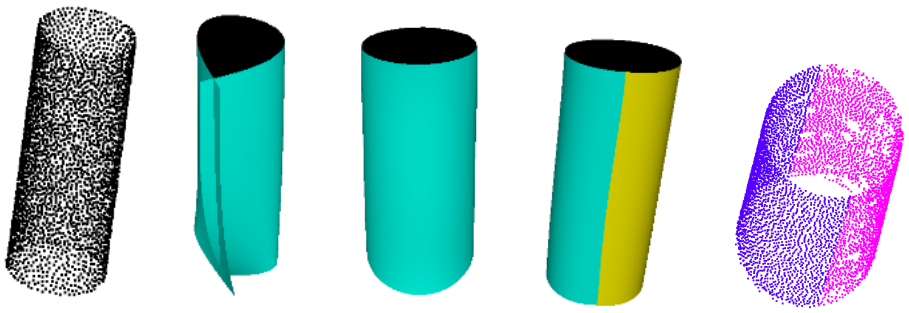
$$\mathbf{X}^*(u,v) = \sum_{j=1}^{64} \mathbf{d}_j \mathbf{f}_j(u,v) \tag{4.28}$$

Similarly, the least square method can be used to apply Eq. (4.28) for parametric surface reconstruction from point clouds. We will also compare the results with the previous equation of 16 variables.

4.5 Results and comparison

In this section, we use our proposed method to reconstruct the PDE surfaces from the obtained point clouds. Because the capability of reconstructing 3D surfaces for PDE with 16 variables and 64 variables is different, it is necessary to choose a suitable approach for a certain type of object. To illustrate this choice process, we also conduct a comparison between two PDE approaches regarding reconstructing chosen 3D objects.

As a first example, we reconstruct the PDE surface from the point set of the cylinder shape. To demonstrate their capability of reconstructing the 3D surface for these two PDE models, we just use one PDE surface patch to reconstruct the cylinder shape in both cases. Fig. 4.19 shows the reconstructed result with the two proposed PDE models. We can see from the reconstructed result in both cases that our proposed PDE model with



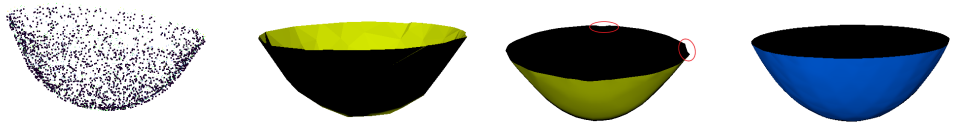
(a) (b) (c) (d) (e)

Figure 4.19: (a) Reconstructed 3D point cloud of a cylinder shape from multi-view 2D images; (b) reconstructed PDE surface using a single PDE model with 16 variables; (c) reconstructed PDE surface using a single PDE model with 64 variables; (d) reconstructed PDE surface using two PDE models with 16 variables; (e) segmented point cloud.

64 variables, shown in Fig. 4.19(c), is more powerful in reconstructing 3D surfaces than the PDE model with 16 variables, shown in Fig. 4.19(b). To reconstruct the cylinder using the PDE model with 16 variables, we can firstly segment the cylinder into equal two parts, each of which will be reconstructed using one PDE surface patch defined by the 16-variable PDE model. The reconstructed result in this way is shown in Fig. 4.19(d).

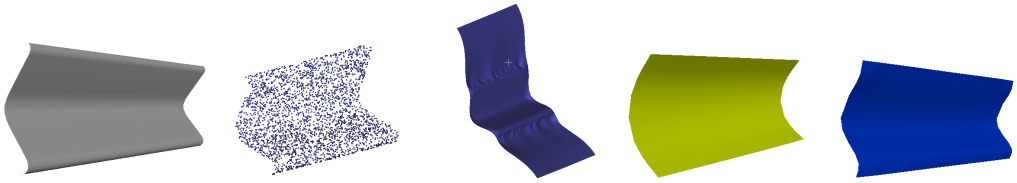
Secondly, we choose to reconstruct a bowl and make a comparison with the implicit PDE method. The PDE-based surface is reconstructed from the point cloud with both the 16-variable PDE model and the 64-variable PDE model to demonstrate their capability to reconstruct more complex 3D shapes. In this example, we use just one PDE patch under both conditions. We also reconstruct a polygon surface from the obtained point cloud using the Poisson surface reconstruction method. Fig. 4.20 shows the reconstructed results using these methods. As we can see, the PDE model with 64 variables outperforms the PDE model with 16 variables. Note also that some areas are missing when reconstructing the 3D surface using the 16-variable PDE, which are marked by red-coloured circles in Fig. 4.20(c).

Next, a bench and a slide model will be reconstructed, as shown in Fig. 4.21 and 4.22 respectively, the PDE model with 64 variables gives bet-



(a) (b) (c) (d)

Figure 4.20: (a) Point set of a bowl; (b) surface reconstructed using Poisson; (c) PDE-based surface using single 16-variables PDE model; (d) PDE-based surface using single 64-variables PDE model.



(a) (b) (c) (d) (e)

Figure 4.21: (a) The ground truth of a bench surface; (b) point set of a bench surface; (c) surface reconstructed using Poisson; (d) PDE-based surface using a single 16-variables PDE model; and (e) PDE-based surface using a single 64-variables PDE mode.

ter results compared to the PDE model with 16 variables and the Poisson method in both two cases. To better demonstrate the effects of different segments on reconstructed shapes and the applicability of our proposed method in reconstructing complicated 3D shapes, we choose to reconstruct a hat and a car model. Between them, the hat model is used to demonstrate the effects of different segments on reconstructed shapes, and the car model is used to demonstrate the applicability of our proposed method in reconstructing complicated 3D shapes.

In order to show how different numbers of segments affect the reconstruction quality, we segment the top of the hat into two different segments: two subsets only as shown in Fig. 4.23 (b), and three subsets as shown in Fig. 4.23(d). The reconstructed models are shown in Figs. 4.23(c) and (e), respectively. Comparing the shapes in Figs. 4.23(c) and (e), we can find that when the top of the hat has one segment, the top part of the reconstructed

(a) (b) (c) (d) (e)

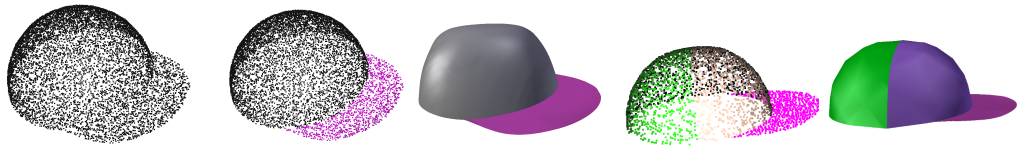
Figure 4.22: (a) The ground truth of a slide surface; (b) point set of a slide surface; (c) surface reconstructured using Poisson after segmentation; (d) PDE-based surface using a single 16-variable PDE model; (e) PDE-based surface using a single 64-variable PDE model.

hat model is flat, which is different from the round shape of the corresponding point set. In contrast, when the top of the hat is segmented into two subsets, the top part of the reconstructed hat model becomes round, which is closer to the shape of the corresponding point sets.

For the car model, we segment its point cloud shown in Fig. 4.24(a) into 10 subsets shown in Fig. 4.24(b). For each of the segmented subsets, a PDE patch is reconstructed. The reconstructed car model consisting of 10 PDE patches is shown in Fig. 4.24(c). This reconstruction example indicates that for any complicated models, their point cloud can be segmented into subsets with each of the segmented subsets having a less complicated shape, and our proposed method can be used to reconstruct the shape from the points in the subset and obtain the reconstructed shape of complicated models from their point clouds.

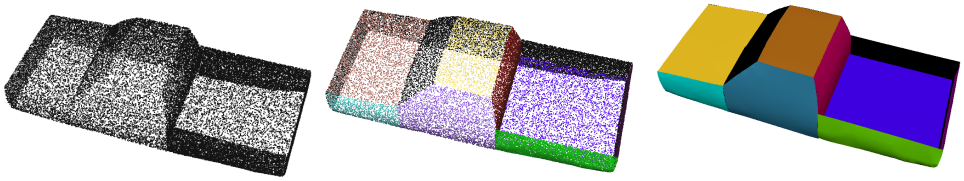
4.6 Summary

In this Chapter, I develop a new method to reconstruct 3D shapes from point clouds with multiple PDE patches. Each PDE patch is based on an explicit closed-form solution to a vector-valued fourth-order partial differential equation, which is efficient and accurate due to the feature of analytical



(a) (b) (c) (d) (e)

Figure 4.23: (a) The point cloud of a hat; (b) segmented 2 subsets; (c) reconstructed PDE-based surface using 2 PDE patches defined by the 64-variables PDE model; (d) segmented 3 subsets; (e) reconstructed PDE-based surface using 3 PDE patches defined by the 64-variable PDE mode.



(a) (b) (c)

Figure 4.24: (a) The point cloud of a truck; (b) segmented subsets; (c) reconstructed PDE-based surface.

closed-form solutions of the obtained mathematical expressions. In comparison with the implicit PDE method, our proposed PDE-based surface reconstruction method involves much fewer design variables. Since multiple PDE patches are used in reconstructing 3D shapes from point clouds, many complex shapes can be reconstructed with our proposed PDE-based method.

Furthermore, the explicit PDE model is expanded from 16 vector variables to 64 vector variables, as it's challenging to segment a given point cloud data into an appropriate number of subsets. Experimental results show that the PDE model with 64 vector-valued unknowns is more powerful and accurate than the PDE model with 16 variables. So, in some cases, using multiple 16-variables PDE surface patches can be replaced by applying a single 64-variables PDE surface patch, like in the cylinder example.

Chapter 5

Parametric surface reconstruction from 3D point data using partial differential equation with positional and tangential continuous patches

5.1 Background

Existing methods of parametric surface reconstruction from 3D point data normally segment the points into several subsets, each fitted with a parametric surface patch. These methods have two problems. First, they fail to achieve positional and tangential continuity between adjacent patches as gaps or overlaps occur between them, or they require high storage. Second, for the data in each subset, parameterization of them is a challenging task. In this chapter, we address these two problems by proposing a new surface reconstruction method based on Partial differential equation(PDE) based deformation surfaces and a specific form of deformation surface with bilinearly and bicubic blended Coons patches. First, we extract four boundaries for each of the subsets. Then, we generate a bilinearly or bicubic blended Coons patch from the four extracted boundaries depending on the required order of continuity. The errors between the points in the subset and the corresponding points on the bilineraly or bicubic blended Coons patch are removed or

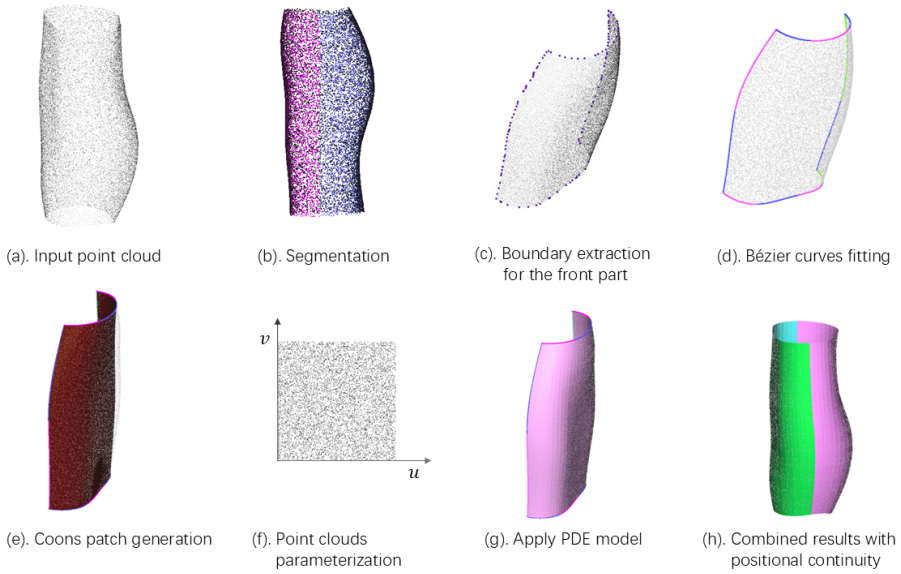


Figure 5.1: Pipeline of our proposed method.

minimised by a PDE deformation surface or a specific deformation surface, which involves as many unknown constants as required in a lateral force to achieve the aim. The proposed method has the following advantages. First, it guarantees good positional or tangential continuity between adjacent patches since the same boundary or tangent is shared by the patches. Second, the reconstruction errors can be easily controlled by tuning the hyper-parameters in the equation used to generate the PDE deformation surface, which changes the number of unknown constants accordingly. Third, the problem of point parameterization in each subset is solved easily by means of the Coons patch. We test our proposed method on various data sets of different complexities and shapes, and the results validate the effectiveness and advantages of our proposed method.

5.2 Method Pipeline

The pipeline of our proposed method is illustrated in Fig. 5.1. Specifically, given a point cloud data shown in Fig. 5.1(a), we first segment it into multiple subsets as shown in Fig. 5.1(b). For each subset, we then extract its bound-

aries and corner points using an existing method, as depicted in Fig. 5.1(c). Each extracted boundary is fitted using Bézier curves, following the approach outlined in Chapter 3. The results, displayed in Fig. 5.1(d), reveal that multiple Bézier curves are necessary for each boundary of this data, as a single curve does not provide sufficient accuracy. Next, we use these boundary curves to generate a bilinearly or bicubic blended Coons patch, illustrated in Fig. 5.1(e). This Coons patch serves as an excellent base surface for the parameterization of the points within the subset, allowing us to obtain their (u, v) parameters, as depicted in Fig. 5.1(f). Our proposed PDE patch or a specific deformation surface is then applied to deform the interior part of the Coons patch, ensuring that the combined parametric surface fits the points in the subset well without altering the boundaries of the Coons patch, as shown in Fig. 5.1(g). Since adjacent parametric surface patches share the same boundary or boundary tangent, they are seamlessly connected with positional or tangential continuity. The final result, shown in Fig. 5.1(h), demonstrates that the adjacent patches are seamlessly connected, eliminating the need for postprocessing required by other methods.

5.2.1 Segmentation and boundary extraction of 3D point data

Various approaches exist for segmenting 3D point data, including methods using attributes, 3D models, and edges, etc. For a comprehensive and detailed introduction to these methods, please refer to (Nguyen and Le 2013). For simplicity, this paper adopts a geometric primitive (3D models) method to segment an input point cloud into an appropriate number of subsets. To detect the boundary points of a given point set, many techniques have been proposed (Chen et al. 2022, Mineo et al. 2019, Edelsbrunner 2011). Specifically, Chen et al. (Chen et al. 2022) proposed an improved Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to detect point cloud boundaries, but their method mainly focuses on planar points. Mineo proposed a novel algorithm for point cloud boundary detection by calculating the local resolution of the point cloud with the aid of the K-nearest

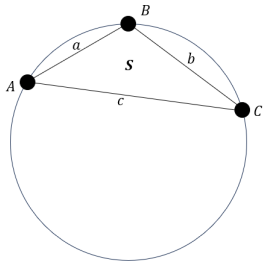


Figure 5.2: Curvature calculation of points on the boundary.

neighbour method (Mineo et al. 2019). Alpha-shape (Edelsbrunner 2011), a classical method for detecting the convex or concave regions of a given point set, can also be used for point cloud detection. Here, we adopt the technique in (Edelsbrunner 2011) as it meets our needs in most cases. In cases where the extracted boundary is not satisfactory, we will fine-tune the parameters of the boundary detection method to achieve better results. Specifically, if the extracted boundary points contain outliers or too few points, we will adjust the parameters of the alpha-shape method, as this method is dependent on both the parameters and the specific input. For the detected boundary points, we also need to identify the corner points, which will divide the boundary points into four or three segments (curves). To accomplish this, we first calculate the curvature of each point using the following formulation (Belyaev 1999):

$$k = (4 * S)/(a * b * c)$$

where S is the area formed by the point of interest (point B in Fig. 5.2) with its two adjacent points (points A and C), and a, b, c are the length of the AB, BC and AC, respectively, as shown in Fig. 5.2. The top four or three curvature points will be treated as the corner points.

5.2.2 Bézier curves fitting

To fit the points along each boundary, we use a parametric curve that ensures the first and last points are included. A Bézier curve is well-suited for this task, though for more complex shapes, multiple Bézier curves may be required. We adopt the method introduced in Chapter 3.

As illustrated in Fig. 5.3(a), a single Bézier curve does not adequately fit the boundary points, necessitating multiple curves. The final result, shown in Fig. 5.3(b), is obtained using the method proposed in Chapter 3 and multiple Bézier curves are used for the points on each boundary.

(a) (b)

Figure 5.3: (a).One Bézier curve can not fit the boundary points well. (b).Multiple Bézier curves are used for the points on each boundary.

5.2.3 Point cloud parameterization for 3D surface fitting

For unstructured 3D point data, the parameters associated with each point are unknown and not easily obtained. The process of obtaining suitable parameters for each point is called point cloud parameterization. Numerous techniques have been proposed to achieve this. One of the most widely used methods involves using a base surface, which can be a plane, a sphere, or a surface patch that approximates the shape of the 3D point data, as discussed in Chapter 2. In our approach, a bilinearly blended Coons patch serves as the base surface. Given the base surface and a point in a 3D point data, we need to find the closest point on the base surface and use this point to determine the parameter values u and v of the original point, which is a nonlinear problem. Many methods have been proposed to tackle this nonlinear problem, which can be roughly divided into five categories: Newton-Raphson method (Mortenson 1997), subdividing method (Johnson and Cohen 2005),

solver methods (Elber and Kim 2001), clipping method (Chen et al. 2008), and geometric method (Li et al. 2019). Each method has its advantages and disadvantages. After testing some methods from each category, we found that the geometric method best fits our needs due to its effectiveness and efficiency. We adopt the geometric method proposed in (Li et al. 2019) to find the closest point and its associated parameters on a parametric surface to a given point. Specifically, given a parametric surface and a test point whose parameters are to be obtained, a normal curvature sphere of the surface is constructed. The radius and centre of this sphere, along with the initially guessed parameters, are specified. Next, the intersection point between the line segment defined by the test point and the centre of the normal curvature sphere is found. Lastly, to iteratively update the parameters of the test point, the iterative formula is derived using Taylor's expansion of the parametric surface. This method is independent of the initial parameter values and converges quickly. Since there may be tens of thousands of points in a subset, using the geometric method point by point is inefficient. To tackle this issue, we use parallel computing to improve the efficiency of point cloud parameterization, as parameterizing each point using the geometric method is independent of the others.

5.3 3D shape reconstruction from point clouds with positional and tangential continuous patches

5.3.1 Based surface

To achieve positional continuity, the bilinearly blended Coons patch is adopted, which is a parametric surface defined by four boundary curves, and it passes through these boundaries. Specifically, given four boundary curves $\mathbf{P}(u, 0)$, $\mathbf{P}(u, 1)$, $\mathbf{P}(0, v)$ and $\mathbf{P}(1, v)$ which are all parametric curves with the parametric variables u and v normally defined in the range $[0, 1]$, the corresponding equation of the bilinearly blended Coons patch is given by the following

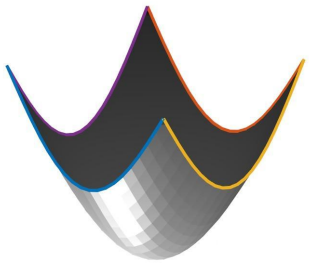


Figure 5.4: Bilinearly blended Coons patch.

equation (Salomon 2007):

$$\mathbf{S}_c(u,v) = \begin{pmatrix} 1-u & u & 1 \end{pmatrix} \begin{pmatrix} -\mathbf{P}_{00} & -\mathbf{P}_{01} & \mathbf{P}(0,v) \\ -\mathbf{P}_{10} & -\mathbf{P}_{11} & \mathbf{P}(1,v) \\ \mathbf{P}(u,0) & \mathbf{P}(u,1) & (0,0,0) \end{pmatrix} \begin{pmatrix} 1-v \\ v \\ 1 \end{pmatrix} \tag{5.1}$$

where $\mathbf{P}_{00}$ is the intersecting point of curves $\mathbf{P}(u,0)$ and $\mathbf{P}(0,v)$ when u takes 0 and v takes 0, respectively. Similarly, $\mathbf{P}_{01}$, $\mathbf{P}_{10}$ and $\mathbf{P}_{11}$ are the remaining three corner points intersected by the other three pairs of adjacent boundary curves. This is demonstrated in Fig. 5.4.

To achieve tangential continuity, a bicubic Coons patch can be used, which exactly satisfies the positional and tangential continuity requirements on the four boundaries of the patch and the twist continuity requirements at the four corner points of the patch. It can be used as a base patch.

A bicubic Coons patch $\mathbf{S}(u,v)$ can be mathematically written as (Piegl 1988):

$$\mathbf{S}(u,v) = \mathbf{S}_1(u,v) + \mathbf{S}_2(u,v) - \mathbf{S}_3(u,v) \tag{5.2}$$

In the above equation, the function $\mathbf{S}_1(u,v)$ is defined by:

$$\mathbf{S}_1(u,v) = \begin{pmatrix} \mathbf{S}(u,0) & \mathbf{S}(u,1) & \mathbf{S}_v(u,0) & \mathbf{S}_v(u,1) \end{pmatrix} H \begin{pmatrix} v^3 \\ v^2 \\ v \\ 1 \end{pmatrix} \tag{5.3}$$

where $\mathbf{S}_v(u,v)$ is $\frac{\partial \mathbf{S}(u,v)}{\partial v}$, and

$$H = \begin{pmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix} \tag{5.4}$$

The function $\mathbf{S}_2(u,v)$ is defined by:

$$\mathbf{S}_2(u,v) = \begin{pmatrix} \mathbf{S}(0,v) & \mathbf{S}(1,v) & \mathbf{S}_u(0,v) & \mathbf{S}_u(1,v) \end{pmatrix} H \begin{pmatrix} u^3 \\ u^2 \\ u \\ 1 \end{pmatrix} \tag{5.5}$$

where $\mathbf{S}_u(u,v) = \frac{\partial \mathbf{S}(u,v)}{\partial u}$, and the function $\mathbf{S}_3(u,v)$ is defined by:

$$\mathbf{S}_3(u,v) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} H^T \begin{pmatrix} \bar{\mathbf{S}} & \bar{\mathbf{S}}_v \\ \bar{\mathbf{S}}_u & \bar{\mathbf{S}}_{uv} \end{pmatrix} H \begin{pmatrix} v^3 \\ v^2 \\ v \\ 1 \end{pmatrix} \tag{5.6}$$

where $\bar{\mathbf{S}}$ is a corner matrix, $\bar{\mathbf{S}}_u$ and $\bar{\mathbf{S}}_v$ are tangent matrices, and $\bar{\mathbf{S}}_{uv}$ is a twist matrix, which has the forms of:

$$\begin{aligned} \bar{\mathbf{S}} &= \begin{pmatrix} \mathbf{S}(0,0) & \mathbf{S}(0,1) \\ \mathbf{S}(1,0) & \mathbf{S}(1,1) \end{pmatrix}, \bar{\mathbf{S}}_u = \begin{pmatrix} \mathbf{S}_u(0,0) & \mathbf{S}_u(0,1) \\ \mathbf{S}_u(1,0) & \mathbf{S}_u(1,1) \end{pmatrix}, \\ \bar{\mathbf{S}}_v &= \begin{pmatrix} \mathbf{S}_v(0,0) & \mathbf{S}_v(0,1) \\ \mathbf{S}_v(1,0) & \mathbf{S}_v(1,1) \end{pmatrix}, \bar{\mathbf{S}}_{uv} = \begin{pmatrix} \mathbf{S}_{uv}(0,0) & \mathbf{S}_{uv}(0,1) \\ \mathbf{S}_{uv}(1,0) & \mathbf{S}_{uv}(1,1) \end{pmatrix} \end{aligned} \tag{5.7}$$

and the subscript u, v indicates the second partial derivatives of the patch with respect to the parametric variables u and v, $\mathbf{S}(0,0)$, $\mathbf{S}(0,1)$, $\mathbf{S}(1,0)$, and $\mathbf{S}(1,1)$ are corner vectors consisting of the coordinate values of the four corner vertices, $\mathbf{S}_u(0,0)$, $\mathbf{S}_u(0,1)$, $\mathbf{S}_u(1,0)$, $\mathbf{S}_u(1,1)$, $\mathbf{S}_v(0,0)$, $\mathbf{S}_v(0,1)$, $\mathbf{S}_v(1,0)$, $\mathbf{S}_v(1,1)$ are eight tangent vectors consisting of the values of the first partial derivatives at the four corner vertices, and $\mathbf{S}_{uv}(0,0)$, $\mathbf{S}_{uv}(0,1)$, $\mathbf{S}_{uv}(1,0)$,

$\mathbf{S}_{uv}(1,1)$ are twist vectors consisting of the values of the second partial derivatives at the four corner vertices.

From the points in a segmented sub-region, we can obtain the four boundary curves $\mathbf{S}(u,0)$, $\mathbf{S}(u,1)$, $\mathbf{S}(0,v)$, $\mathbf{S}(1,v)$ and the first partial derivative functions curves $\mathbf{S}_v(u,0)$, $\mathbf{S}_v(u,1)$, $\mathbf{S}_u(0,v)$, $\mathbf{S}_u(1,v)$, on the four boundary curves. From the four boundary curves, we can determine the four corner vectors. From the first partial derivatives on the four boundary curves, we can determine the eight tangent vectors. Finally, from the second partial derivatives of the first partial derivatives with respect to u or v respectively, we can determine the four twist vectors. Substituting the above-obtained quantities into Eqs. (5.3), (5.5), (5.6), and (5.7), we obtain $\mathbf{S}_1(u,v)$, $\mathbf{S}_2(u,v)$ and $\mathbf{S}_3(u,v)$. Substituting them into Eq. (5.2), we obtain a base patch $\mathbf{S}(u,v)$. Among two adjacent base patches constructed with the above methods, positional continuity, tangential continuity, and twist continuity between the two adjacent base patches are achieved.

5.3.2 Deformation surface

To achieve positional continuity, PDE deformation surfaces are crucial in removing or minimising the errors between the points in a segmented subset and the corresponding points on the bilinearly blended Coons patch. These surfaces can be derived from the particular solution to the PDE proposed below. The bending equation of an elastic thin plate is:

$$D(\frac{\partial^4 \mathbf{w}}{\partial x^4} + 2\frac{\partial^4 \mathbf{w}}{\partial x^2 \partial y^2} + \frac{\partial^4 \mathbf{w}}{\partial y^4}) = \mathbf{q_w} \tag{5.8}$$

where x and y are the position variables,$\mathbf{w}$ is the lateral displacement,$\mathbf{q_w}$ is the lateral force, and

$$D = \frac{Eh^3}{12(1-\mu^2)} \tag{5.9}$$

is called the flexural rigidity, which is determined by Young's modulus E, Poisson's ratio μ, and the thickness h of the elastic plate. If we replace the position variables x and y with the parametric parameters (u and v),

and replace the lateral displacement with the position displacements x, y, z, Eq. (5.8) is changed into the following partial differential equations:

$$D(\frac{\partial^4 w}{\partial u^4} + 2\frac{\partial^4 w}{\partial u^2 \partial v^2} + \frac{\partial^4 w}{\partial v^4}) = q_w \qquad (w = x, y, z) \tag{5.10}$$

For simplicity, we will not use Young's modulus E, Poisson's ratio μ, and the thickness h to determine the flexural rigidity D. Instead, we set its value to 1, i.e., D=1.

Since the purpose of using Eq. (5.10) is to add deformations to the bilinearly blended Coons patch to remove or minimise the errors between the point clouds and the Coons patch, only the particular solution to Eq. (5.10) is necessary. Although the deformations may be complex, they can be mathematically represented with a Fourier series. To ensure positional continuity between reconstructed patches, the Fourier series can be taken to be the following *sine* series. Taking these factors into account, the mathematical expression of the lateral force $\mathbf{q_w}$ are taken to be the following forms:

$$q_w = \sum_{m=1}^{M}\sum_{n=1}^{N} q_{wmn} sin(m\pi u) sin(n\pi v) \qquad (w = x, y, z) \tag{5.11}$$

where M and N are hyper-parameters, which can be tuned to control surface reconstruction errors. For example, when the shape of a point cloud is complicated and the reconstruction error exceeds the specified error, we can increase the value of M and N to make the disparity between the reconstructed surface and the point data no more than the specified error, which will be demonstrated in Section 5.4.

According to Eq. (5.11), the displacement w can be taken to be

$$w = \sum_{m=1}^{M}\sum_{n=1}^{N} w_{mn} sin(m\pi u) sin(n\pi v) \qquad (w = x, y, z) \tag{5.12}$$

From Eq. (5.12), we obtain the fourth partial derivatives of w with respect to u, u and v, and v, respectively. They have the forms of:

$$\begin{aligned}
\frac{\partial^4 w}{\partial u^4} &= \pi^4 \sum_{m=1}^{M}\sum_{n=1}^{N} m^4 w_{mn} sin(m\pi u) sin(n\pi v) \\
\frac{\partial^4 w}{\partial u^2 \partial v^2} &= \pi^4 \sum_{m=1}^{M}\sum_{n=1}^{N} m^2 n^2 w_{mn} sin(m\pi u) sin(n\pi v) \\
\frac{\partial^4 w}{\partial v^4} &= \pi^4 \sum_{m=1}^{M}\sum_{n=1}^{N} n^4 w_{mn} sin(m\pi u) sin(n\pi v)
\end{aligned} \tag{5.13}$$

Substituting the above Eq. (5.11) and Eq. (5.13) into Eq. (5.10), we obtain

$$\begin{aligned}
&D[\pi^4 \sum_{m=1}^{M}\sum_{n=1}^{N} m^4 w_{mn} sin(m\pi u) sin(n\pi v) + 2\pi^4 \sum_{m=1}^{M}\sum_{n=1}^{N} m^2 n^2 w_{mn} sin(m\pi u) sin(n\pi v) \\
&+ \pi^4 \sum_{m=1}^{M}\sum_{n=1}^{N} n^4 w_{mn} sin(m\pi u) sin(n\pi v)] = \sum_{m=1}^{M}\sum_{n=1}^{N} q_{wmn} sin(m\pi u) sin(n\pi v)
\end{aligned} \tag{5.14}$$

The above equation can be simplified as:

$$\sum_{m=1}^{M}\sum_{n=1}^{N} [D\pi^4 (m^4 + 2m^2 n^2 + n^4) w_{mn} - q_{wnm}] sin(m\pi u) sin(n\pi v) = 0 \tag{5.15}$$

From Eq. 5.15, we obtain the following w_{mn}:

$$\begin{aligned}
w_{mn} &= \frac{q_{wmn}}{D\pi^4 (m^4 + 2m^2 n^2 + n^4)} \\
(w = x, y, z; m &= 1, 2, 3, ..., M; n = 1, 2, 3, ..., N)
\end{aligned} \tag{5.16}$$

By Substituting Eq. 5.16 into Eq. 5.12, the particular solution shown below can be obtained

$$\begin{aligned}
w = \frac{1}{D\pi^4} \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{q_{wmn}}{(m^4 + 2m^2 n^2 + n^4)} sin(m\pi u) sin(n\pi v) \\
(w = x, y, z)
\end{aligned} \tag{5.17}$$

Suppose the number of points in a given point cloud subset is I, then at the points (u_i, v_i) $(i = 1, 2, 3, \ldots, I)$, the errors between the points in

the point cloud and the corresponding points obtained from the bilinearly blended Coons surface $S_c(u_i, v_i)$ are $\overline{w}_i$. To fit a parametric surface to the interior points, we minimise the following error:

$$E_w = \sum_{i=1}^{I} [\overline{w}_i - \frac{1}{D\pi^4} \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{q_{wmn}}{(m^4 + 2m^2n^2 + n^4)} sin(m\pi u_i) sin(n\pi v_i)]^2 \tag{5.18}$$

which can be solved using the least square method below:

$$\begin{aligned} \frac{\partial E_w}{\partial q_{wkl}} = 2 \sum_{i=1}^{I} [\overline{w}_i - \frac{1}{D\pi^4} \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{q_{wmn}}{m^4 + 2m^2n^2 + n^4} sin(m\pi u_i) sin(n\pi v_i)][\\ - \frac{1}{D\pi^4} \frac{1}{(k^4 + 2k^2l^2 + l^4)} sin(k\pi u_i) sin(l\pi v_i)] = 0 \\ (w = x, y, z; k = 1, 2, 3, \ldots, M; l = 1, 2, 3, \ldots, N) \end{aligned} \tag{5.19}$$

Solving the above $M \times N$ equations, we obtain $q_{wmn}(w = x, y, z; m = 1, 2, 3, \ldots, M; n = 1, 2, 3, \ldots, N)$. Substituting them into Eq. (5.17), we obtain the PDE patch $\mathbf{S}_{PDE}(u, v)$, which has the form $\mathbf{S}_{PDE}(u, v) = [x(u, v), y(u, v), z(u, v)]^T$. Finally, the reconstructed parametric surface $\mathbf{S}(u, v)$ is:

$$\mathbf{S}(u, v) = \mathbf{S}_c(u, v) + \mathbf{S}_{PDE}(u, v) \tag{5.20}$$

To achieve tangential continuity, the deformation surface should not deform the boundary of the Bicubic Coons patch, which includes the four boundary positions, tangents and the four twist vectors. Besides, the deformation surface also needs to own powerful capability to add deformation to the interior part of the Coons patch to make it fit to the interior points. To achieve this aim, we are inspired by the previous deformation surface defined by the PDE and propose the following deformation surface:

$$\begin{aligned} w = \sum_{m=1}^{M} \sum_{n=1}^{N} q_{wmn} \left(\sin(m\pi u)\right)^2 * \left(\sin(n\pi v)\right)^2 \\ (w = x, y, z) \end{aligned} \tag{5.21}$$

To assess the reconstruction accuracy of our proposed approach, we compute the average error and maximum error between the reconstructed parametric surface and the original I points $(\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_I)$ in the point cloud

subset using the following expression:

$$\begin{aligned} Err_{Mean} &= \frac{1}{I}\sum_{i=1}^{I} |\mathbf{P}_i - \mathbf{S}(u_i, v_i)| \\ Err_{Max} &= \max_{i=1,2,\ldots,I} |\mathbf{P}_i - \mathbf{S}(u_i, v_i)| \end{aligned} \tag{5.22}$$

Notice that when the product $M \times N$ of the hyper-parameters M and N is equal to the number I of the points in the point cloud subset, the reconstruction error is removed, i.e., the reconstruction error is zero.

5.4 Results

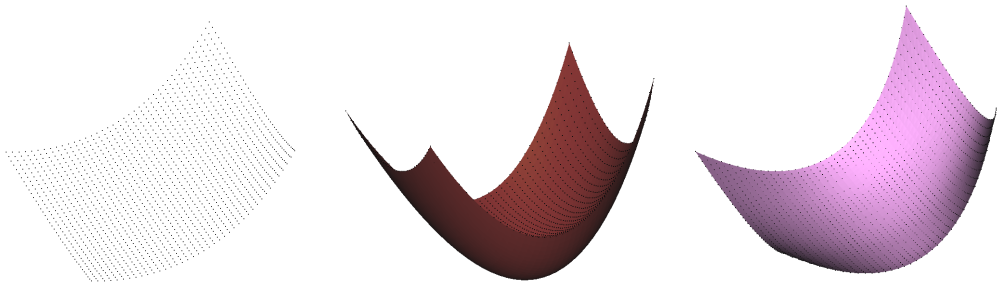
To validate the effectiveness of our proposed method, we tested it on various datasets, including structured point clouds, unstructured point clouds of varying complexities, and data with different levels of noise.

5.4.1 Surface reconstruction from structured point clouds

There are mainly two types of point clouds: structured and unstructured. In structured point clouds, the relationship between points is known, whereas in unstructured point clouds, this relationship is unknown and more complex. To obtain a structured point cloud, we uniformly sample points on a parametric Bézier surface, knowing the position of each point and its corresponding parameters (u_i, v_i). The point cloud is shown in Fig. 5.5(a), with fewer points sampled for clarity.

Notice the Bézier surface from which the points are sampled is not cubic; however, since we fit cubic Bézier curves to the boundary points following the method in Chapter 3, the parametric equations of the four boundaries cannot be derived simply by fixing $u = 0, 1$ and $v = 0, 1$ in the parametric equation of the Bézier surface. Instead, we follow the proposed method in Chapter 3 to fit cubic Bézier curves to the boundary points. From these four boundaries, we construct a bilinearly blended Coons patch based on Eq. (5.1), which is shown in Fig. 5.5(b). The figure illustrates that while the bilinearly blended Coons patch passes through the four boundaries, it does

not fit the interior points well, with a mean reconstruction error of 0.7120. To reduce this error, we apply the particular solution of Eq. (5.17) to the PDE in Eq. (5.10), adding deformations to the bilinearly blended Coons patch. With hyper-parameters $M = 5$ and $N = 5$, the resulting reconstruction surface, which combines the PDE patch and the bilinearly blended Coons patch, is shown in Fig. 5.5(c). As illustrated, the combined parametric surface fits all the points very well, significantly reducing the mean reconstruction error to 0.0035.



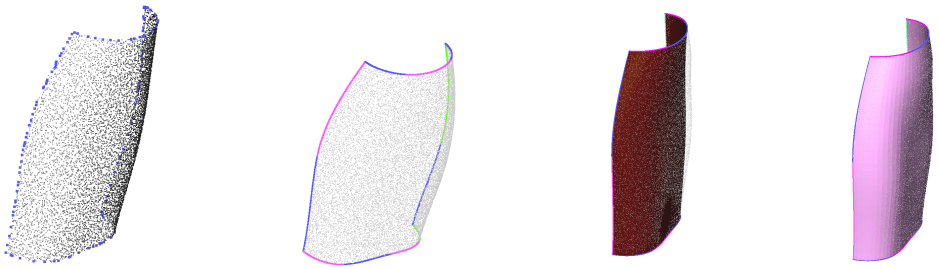
(a) (b) (c)

Figure 5.5: Reconstruction of the surface from structured point clouds: (a).Input point clouds. (b).Generated Coons patch with an average error of 0.7120. (c).Final parametric surface with an average error of 0.0035.

5.4.2 Surface reconstruction from unstructured point clouds

We first test our method with an unstructured point cloud of one patch with four sides. The point cloud is shown in black in Fig. 5.6(a), representing the front part of a skirt model. To obtain the Coons patch, we first identify the four boundaries of the point cloud, ensuring that one endpoint of adjacent boundaries is the same point, serving as the corner point of the Coons patch. The detected boundary points are shown in blue in Fig. 5.6(a) and the fitting Bézier curves are shown in Fig. 5.6(b). After obtaining the Bézier curves equations for the four boundaries, the bilinearly blended Coons patch is generated from these four boundaries using Eq. (5.1), as shown in Fig. 5.6(c). The Coons patch does not fit the point cloud well. To improve the fitting,

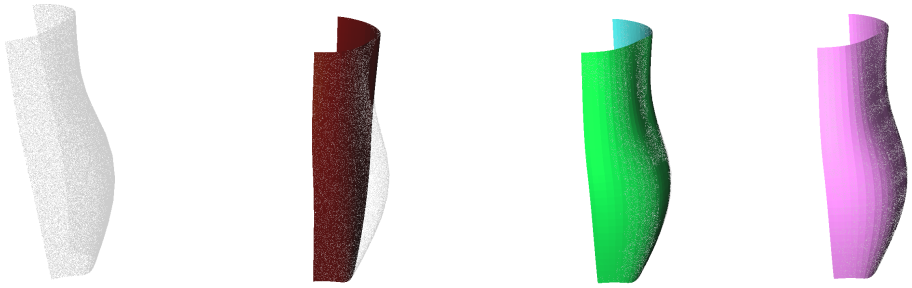
we calculate the distances between the points in the point cloud and the corresponding points on the Coons patch. The PDE deformation surface from Eq. (5.17) is then used to compensate for this discrepancy. The final result is shown in Fig. 5.6(d), demonstrating that the reconstructed surface fits the point cloud very well.



(a) (b) (c) (d)

Figure 5.6: Reconstruction of the surface from unconstructed point clouds of the front part of a skirt model:(a)Input point clouds and extracted boundaries. (b)Reconstructed Bézier curves. (c)Generated Coons patch. (d)Final parametric surface.

To further demonstrate the effectiveness and controllability of our method, we apply it to reconstruct the back part of the skirt model using the same procedure. Figs. 5.7(a), 5.7(b) and 5.7(c) show the original input point cloud, the reconstructed bilinearly blended Coons patch and the final parametric surface, respectively. As expected, the Coons patch does not fit the point cloud well initially, but by adding our proposed PDE deformation surface, the final reconstruction result is satisfactory. Furthermore, the reconstruction error from our proposed method is controllable. This can be achieved by adjusting the values of M and N. The reconstruction surface shown in Fig. 5.7(c) uses $M = 5, N = 5$, with corresponding mean error and maximum errors of 0.0052 and 0.0216, respectively. To reduce the reconstruction error, we can increase the value of M and N. Fig. 5.7(d) shows the result with $M = 10, N = 10$, where the mean and maximum errors are 0.0015 and 0.0077, respectively.



(a) (b) (c) (d)

Figure 5.7: Reconstruction of the surface from unconstructed point clouds of the back part of a skirt model:(a)Input point clouds. (b)Generated Coons patch. (c)Reconstructed parametric surface with $M = 5, N = 5$. (d)Reconstructed parametric surface with $M = 10, N = 10$.

The point clouds shown in Fig. 5.6(a) and Fig. 5.7(a) are the front and back parts of the point clouds shown in Fig. 5.8(a), respectively. Their reconstructed surfaces are shown in Fig. 5.8(b). As depicted, the reconstruction surfaces fit the point cloud very well. Moreover, the two reconstructed surfaces are seamlessly connected, requiring no post-processing, unlike the PDE-based reconstruction method presented in (Zhu et al. 2022b).

(a) (b)

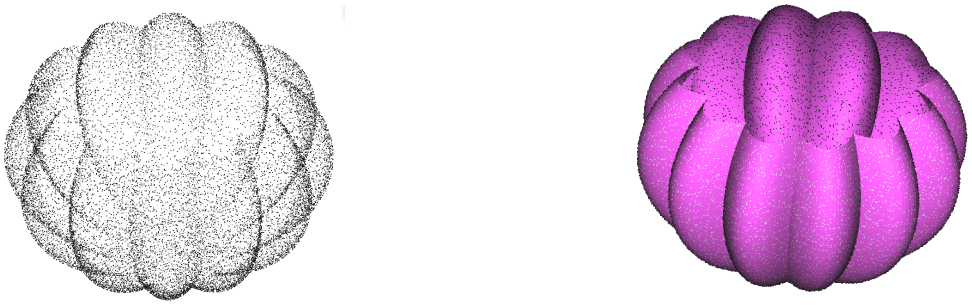
Figure 5.8: Reconstruction of the surface from unconstructed point clouds of a skirt model:(a)Input point clouds. (b)Reconstructed parametric surface after combination.

We further test our method with the examples shown in Fig. 5.9 and Fig. 5.10. In Figure 5.9, we use one patch to reconstruct the flag shape from

the point cloud shown in Fig. 5.9(a), with the reconstructed surface shown in Fig. 5.9(b). In Fig. 5.10, we segment the point cloud in Fig. 5.10(a) into 12 subsets, using a single parametric surface to approximate the 3D points in each subset. These reconstructed surfaces are seamlessly connected to form the pot shape in Fig. 5.10(b). These examples further illustrate the capability of our method to effectively generate parametric surfaces that closely match various datasets.

(a) (b)

Figure 5.9: Reconstruction of the surface from unstructured point data of a flag model:(a)Input point clouds. (b)Reconstructed parametric surface.

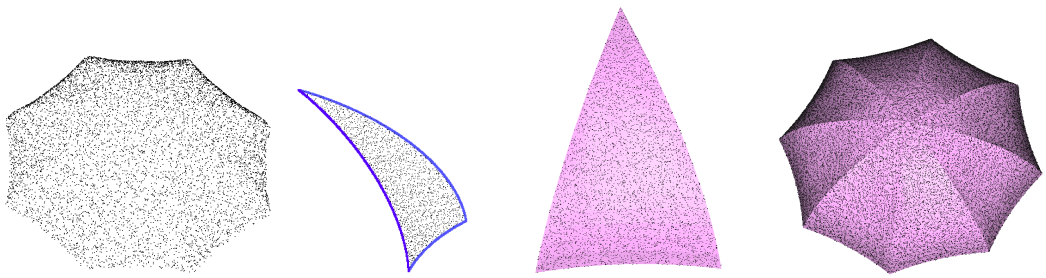


(a) (b)

Figure 5.10: Reconstruction of the surface from unstructured point data of a pot model:(a)Input point clouds. (b)Reconstructed parametric surface.

The above examples demonstrate our method's effectiveness in reconstructing four-sided patches from 3D point data. However, not all point clouds have four boundaries or can be segmented into subsets with four

boundaries. Some point clouds may have three boundaries. In such cases, one boundary of the reconstructed patch degenerates to a single point, and our proposed method can still handle such datasets. Fig. 5.11(a) shows such a point cloud, segmented into eight subsets, each with three boundaries, as shown in Fig. 5.11(b). Notice that since the underlying structures of the three boundaries are relatively simple, our experiments indicate that a single Bézier curve is sufficient to fit the points on each boundary, as shown in Fig. 5.11(b). Our approach reconstructs the shape from the points within each subset. The result for one subset is depicted in Fig. 5.11(c), while combining the reconstructions from all subsets yields the final result, as shown in Fig. 5.11(d). This demonstrates that the reconstructed surfaces accurately align with the original point cloud.



(a) (b) (c) (d)

Figure 5.11: Surface reconstruction from unstructured point clouds of an umbrella model:(a)Input point clouds. (b)Points on the three boundaries can be fitted with just one Bézier curve. (c)Reconstructed parametric surface for a subset with three boundaries. (c) Final parametric surface after combination.

5.4.3 Surface reconstruction from complicated point clouds

It is difficult or impossible to reconstruct a point cloud with a complicated shape by using a single parametric patch. In such a situation, we first segment a complicated point cloud into some sub-regions. After that, we use one patch to fit the points in one sub-region and assemble all the patches to obtain 3D shape reconstruction from complicated point clouds.

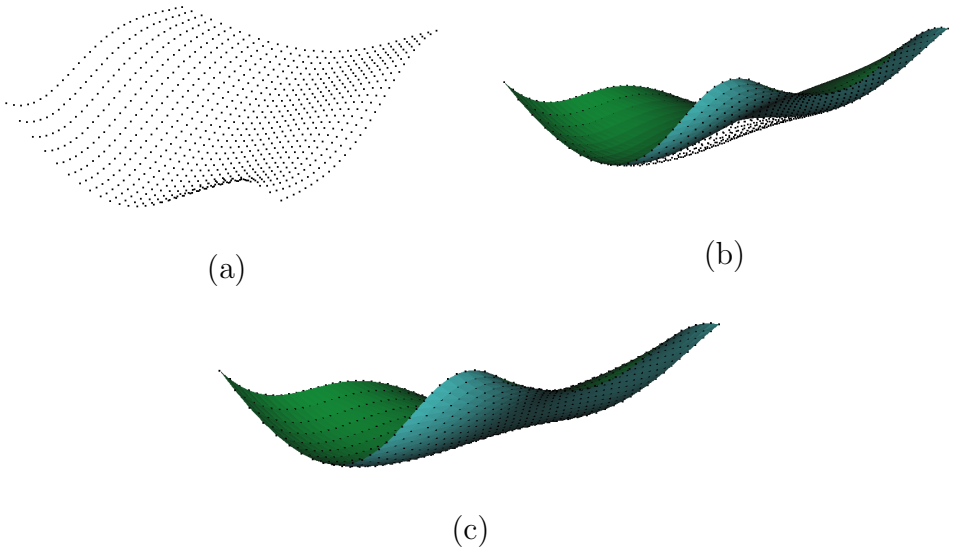


Figure 5.12: Reconstruction of the surface from constructed point data. (a). Input point clouds; (b). Generated bicubic Coons patch; (c). Final results after applying the deformation surface

To achieve both positional and tangential continuities between two patches reconstructed from two adjacent sub-regions, we treat a patch as the sum of a base patch and a deformation patch. The base patch exactly satisfies the positional and tangential continuity requirements on the four boundaries of the patch and the twisting continuity requirements at the four corner points of the patch. The deformation patch does not change the positional, tangential, and twisting requirements.

The first point clouds data is sampled from a fourth-order Bézier surface, as shown in Fig. 5.12(a), and a bicubic Coons patch is generated based on Eq. (5.2). We can see that the bicubic Coons patch does not fit the point clouds very well, specifically, the mean and maximum fitting errors are 0.1830 and 0.5968, respectively. Then we apply the deformation surface defined by Eq. (5.21) to the bicubic Coons patch to obtain the final parametric surface, as shown in Fig. 5.12(c), and it fits the point clouds very well with mean and maximum errors reduced to 1.9707×10^{-4} and 5.6038×10^{-4}, respectively.

Next, we consider a more complicated object. The input point cloud

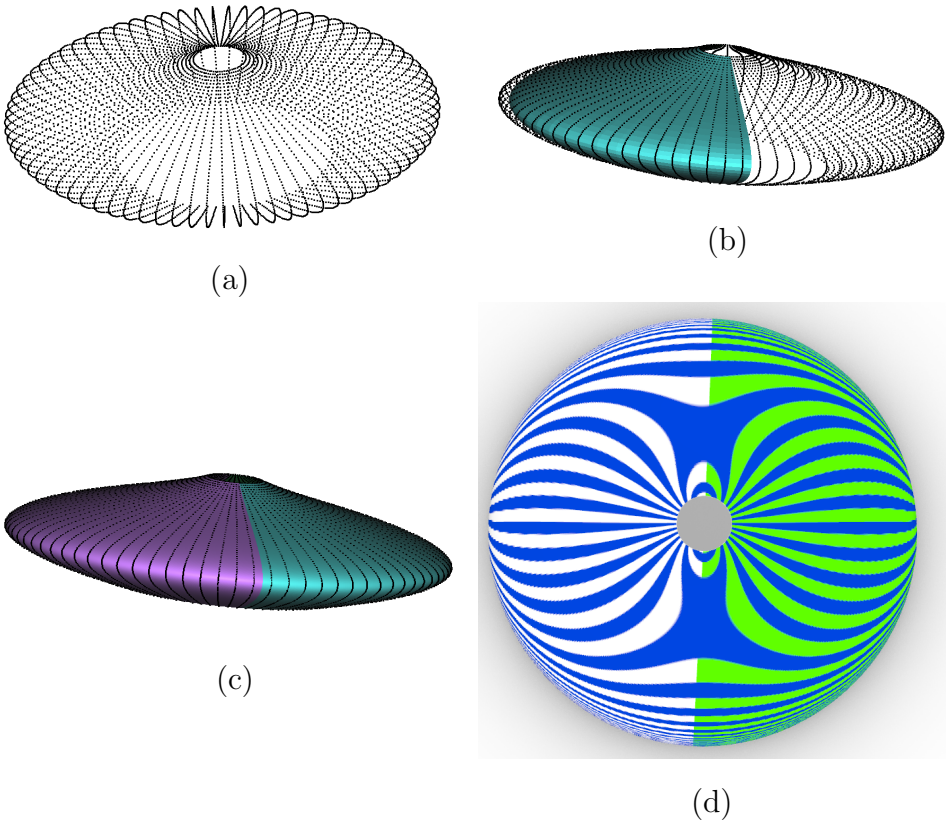


Figure 5.13: Reconstruction of the surface from structured point data. (a). Input point clouds; (b). Generated bicubic Coons patch of left side point clouds; (c). Final results after applying the deformation surface; (d). The Zebra analysis of the two reconstructed parametric surfaces

data is shown in Fig. 5.13, as it is difficult to fit the points with just one parametric surface patch, we segment it into two subsets. Similarly, the bicubic Coons patch for one subset is generated based on Eq. (5.2), and there exists a disparity between it and the point clouds. Specifically, the mean and maximum fitting errors are 0.0248 and 0.0836 respectively. Then we apply the deformation surface and the final parametric surface is shown in Fig. 5.13(c), it can be seen that the final parametric surface fit the point cloud very well, with mean and maximum fitting errors decreased to 3.1×10^{-3} and 7.5×10^{-3}, respectively. To maintain C1 continuity between the two

parametric surface patches, we make sure the second bicubic Coons patch shares the required boundary condition as the first bicubic Coons patch, which includes the 2 boundary curves, the tangent at the 2 boundary curves and the twist at the endpoints of the 2 shared boundary curves. Since our deformation would not change these conditions and only deforms the interior part of the bicubic Coons patch to fit the points, the C1 continuity is kept. To see this, we use the Zebra analysis tool in Rhino software to analyse the continuity between the two reconstructed parametric surfaces. We can see the zebra crossing runs smoothly from one surface patch to another one, which means C1 continuity is obtained.

5.4.4 The impact of hyper-parameters

To further investigate how different values of the hyper-parameters M and N affect reconstruction errors, we set $M = N = 5$ and $M = N = 10$ for surface reconstruction from the point clouds shown in Figs. 5.5, 5.6, 5.7, 5.9, 5.10 and 5.11. The mean errors and maximum errors are presented in Table 5.1 for $M = N = 5$ and in Table 5.2 for $M = N = 10$. Comparing the mean errors and the maximum errors given in the tables, we can conclude that both the mean error and maximum error can be reduced by increasing the values of M and N, which further demonstrates the effectiveness and controllability of our suggested approach. To provide a clearer comparison, we also plot the mean and maximum errors for different degrees of freedom ($M \times N$) across all datasets. Fig. 5.14 illustrates the mean fitting errors relative to the degrees of freedom, showing a consistent decrease in mean errors as the degrees of freedom increase in all cases. For the maximum errors, the trend is less clear when plotted together due to varying scales, so we present them in three separate subplots in Fig. 5.15. In all cases, the maximum errors decrease as the degrees of freedom increase. These figures demonstrate the flexibility and controllability of our method.

In Section 5.3, we take the value of D as 1 for simplicity and use it in Eq. (5.19) for linear least square fitting. Here, we investigate the impact of its values on the fitting results. It is important to note that the value

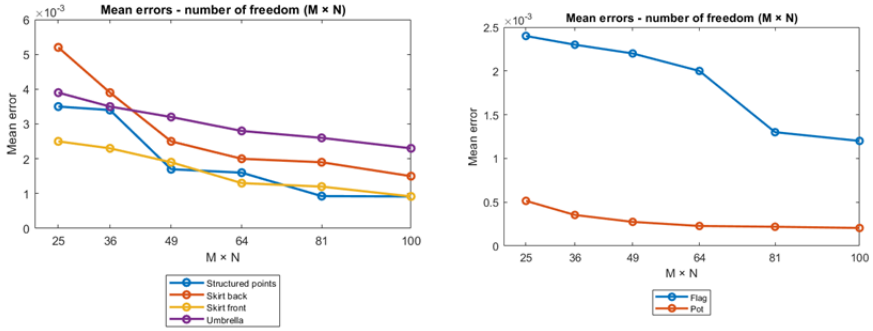


Figure 5.14: Mean errors respect to $M \times N$ for all the datasets

of D does not affect the reconstructed results or the fitting errors. This is because in Eq. (5.19), D can be incorporated into the expression within the sum, allowing q_{wmn}/D to be treated as a new variable. Therefore, altering D would change q_{wmn} correspondingly, leaving their ratio unchanged. To illustrate this, we varied the value of D in Eq. (5.19) for the least squares fitting multiple times, selecting random values between 10^{-3} and 10^{3}. Each time, both the average error and the maximum error remained constant, and the value of q_{wmn}/D were unchanged.

	Figure 5.5: Structured points	Figure 5.6: Front part of skirt	Figure 5.7: Back part of skirt	Figure 5.9: Flag	Figure 5.10: Pot	Figure 5.11: Umbrella
Mean error	0.0035	0.0025	0.0052	0.0024	5.1492×10^{-4}	0.0039
Max error	0.0131	0.0096	0.0216	0.0053	0.0021	0.0206

Table 5.1: Reconstruction errors when $M = N = 5$.

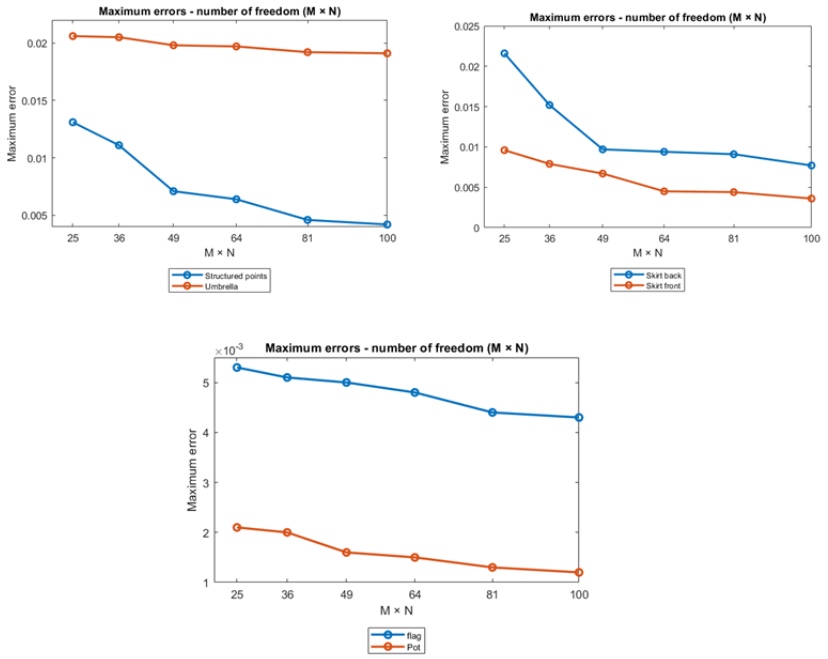


Figure 5.15: Max errors respect to $M \times N$ for all the datasets

	Figure 5.5: Structured points	Figure 5.6: Front part of skirt	Figure 5.7: Back part of skirt	Figure 5.9: Flag	Figure 5.10: Pot	Figure 5.11: Umbrella
Mean error	9.1745×10^{-4}	9.1820×10^{-4}	0.0015	0.0012	2.0462×10^{-4}	0.0023
Max error	0.0042	0.0036	0.0077	0.0043	0.0012	0.0191

Table 5.2: Reconstruction errors when $M = N = 10$.

5.4.5 Surface reconstruction from point clouds with various levels of noise

To investigate the robustness of our method to various levels of noise (defined as l), we begin with the constructed point sets sampled from a Bézier surface that is used as the structured point cloud in Section 5.4.1, which is noise-

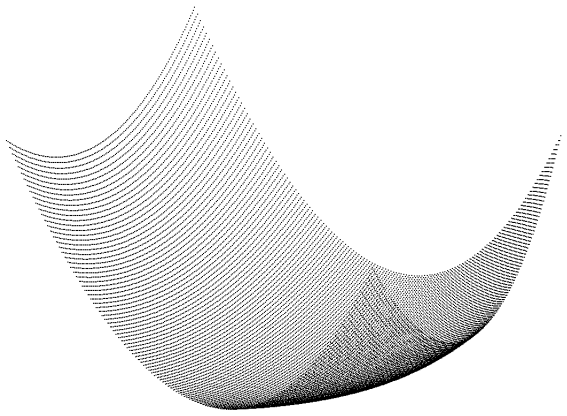
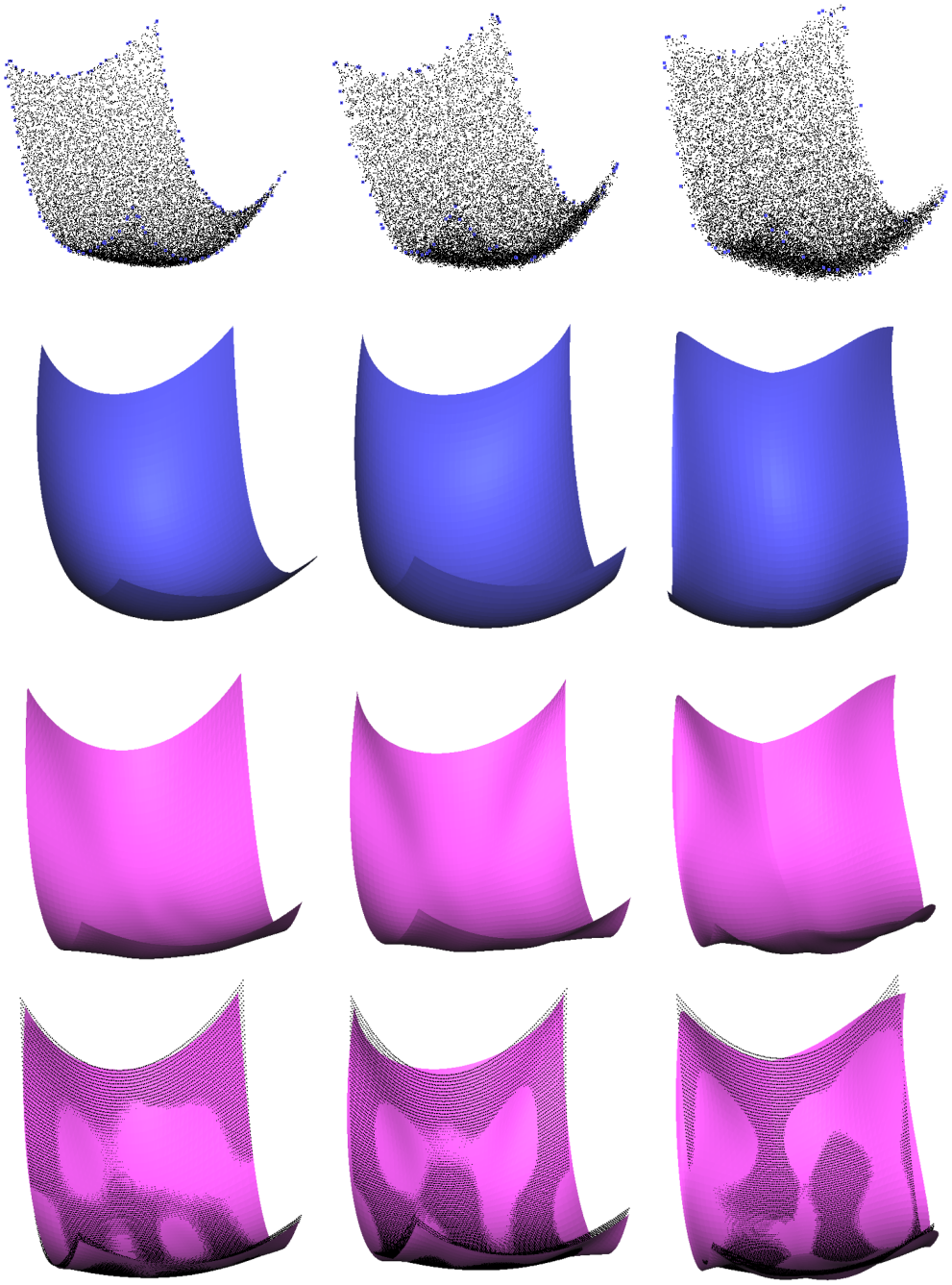


Figure 5.16: Noise free point cloud

free and can be used as the ground truth for calculating the fitting errors. Similar to other methods to test the performance on point clouds with various levels of noise (Yuwen et al. 2006, Khameneifar and Ghorbani 2019), which reconstruct B-Spline surfaces for sharp feature preservation and curvature estimation respectively, we define the noise level as follows: We first compute the smallest bounding box that encapsulates the noise-free point cloud, and the diagonal length of the bounding box is calculated and defined as d. Then each point is displaced in a random direction with a random magnitude, and the magnitude is drawn from a Gaussian distribution with zero mean and a certain standard deviation, which is calculated by $(l * d)\%$.

Given the noise-free point set, as shown in Fig. 5.16, we set $l = 0.5$. The obtained noisy data using the aforementioned steps with noise levels $l = 0.5$, $l = 1.0$, and $l = 1.5$, together with the detected boundaries, are shown in the first row of Fig. 5.17. The base surfaces from the detected boundaries are shown in the second row. The last two rows show the reconstructed results and those superimposed on the structured point cloud that is free of noise. Notice we set $M = N = 5$ in this section. As we can see, our method gives a relatively good result when l is no more than 1.0. As l gets larger, the quality of the result decreases. However, it is important to note that the first step of the general pipeline for parametric surface reconstruction from noisy data involves preprocessing, which filters noise and outliers or adds points to make the data complete. Since this section aims to investigate



(a) $l = 0.5$ (b) $l = 1.0$ (c) $l = 1.5$

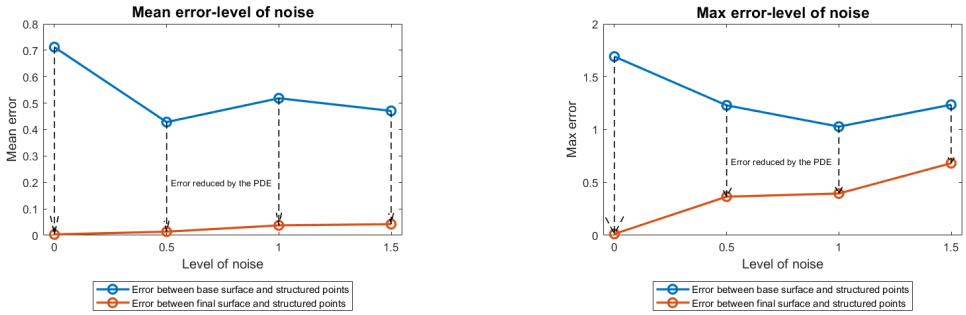
Figure 5.17: Reconstruction of the surface from unstructured point data with $l = 0.5$(first column), 1.0(second column), 1.5(third column).
Row 1:Noisy point clouds; Row 2: Base surface; Row 3: Reconstructed parametric surface; Row 4: Reconstructed parametric surface with structured point cloud.

the robustness of our method to various levels of noise, preprocessing is not carried out to denoise the data. As the level of noise increases, the detected boundary degrades, and the generated base surface also deteriorates, as seen in the third row when $l = 1.5$. This affects the parameterization of points and the subsequent fitting process, thus impacting the final result. To illustrate, we use the boundary of the original structured point cloud to generate the base surface shown in Fig. 5.18(a), which is used for the noisy data ($l = 1.5$) parameterization, and the final result is shown in Fig. 5.18(b). We can see a better result can be obtained if a better base surface is generated, which can be achieved when the data is processed to filter the noise.

Finally, we plot the fitting errors along the way in all cases, as shown in Fig. 5.19. Our PDE model consistently reduces the error between the base surface (Coons patch) and the structured points in all cases.

(a) (b)

Figure 5.18: Case $l = 1.5$: (a).Generated base surface from the boundary of the structured points; (b).Reconstructed parametric surface



(a) (b)

Figure 5.19: Errors between Coons patch with reconstructed surface and the structured point clouds (a).Mean errors; (b).Maximum errors

5.5 Summary

In this chapter, I developed a new physics-based method using PDE deformation surfaces and bilinearly blended Coons patches for parametric surface reconstruction from point clouds. The proposed approach modifies the governing equation for elastic bending of thin plates to derive a PDE incorporating several unknown constants within the lateral force term, allowing for minimisation or elimination of reconstruction errors. The particular solution of this PDE is derived and used to generate a PDE deformation surface, which is then combined with a bilinearly blended Coons surface—constructed by interpolating the four boundaries of a point cloud or its subsets—to obtain the final reconstruction surface. Experimental results validate the effectiveness of this approach, demonstrating its advantages: seamless continuity between reconstructed patches, easily controllable reconstruction errors, and efficient parameterisation of point data.

Meanwhile, inspired by the PDE-based deformation surface, a similar methodology is applied, but a new base surface and a new deformation surface are introduced to achieve both positional and tangential continuity between adjacent reconstructed patches derived from point clouds.

Chapter 6

Conclusion and Future Work

In this chapter, we conclude the thesis by summarising our work and outlining potential future research, which are addressed in Sections 6.1 and 6.2, respectively.

6.1 Conclusion

In various fields, such as computer graphics and manufacturing, parametric forms play a crucial role due to their numerous advantages, including high precision, low storage requirements, compactness, and ease of editing. In particular, some products can only be produced accurately when a parametric (mathematical) representation is available. Often, we are presented with discrete point cloud data representing certain objects, and the challenge is to find a good approximation using parametric forms while minimising storage requirements (number of design variables).

This thesis explores the reconstruction of both parametric curves and surfaces from point clouds. In the context of parametric curve reconstruction, several key factors are considered to achieve optimal results. Firstly, by leveraging the affine transform invariant property of Bézier curves, only a portion of the entire point cloud needs to be reconstructed for data with certain symmetries. Additionally, since the parameterization of point clouds significantly influences the final outcomes, selecting the appropriate parameterization method based on the specific data is crucial to further reducing

fitting errors. Lastly, when dealing with complex point cloud segments that require multiple curves, we employ an efficient algorithm called bisection to determine the optimal break points, dividing the segment into an appropriate number of sub-segments. This approach ensures a tight approximation while minimising the number of curves used.

For parametric surface reconstruction from point clouds, various representations exist, such as Bézier surfaces, B-spline surfaces, and NURBS surfaces. However, PDE-based parametric surfaces offer distinct advantages, including low storage requirements, excellent continuity, and physics-based properties. A significant challenge with PDE-based representations, however, lies in solving the partial differential equations, as most existing methods rely on implicit PDE representations or numerical solutions, which can be computationally expensive and prone to accuracy loss. To address this issue, we first introduce closed-form solutions for partial differential equations applied to 4-sided PDE patches, utilising them as a parametric representation for surface reconstruction from point clouds. We validate the effectiveness of our method using datasets of varying complexity. Furthermore, we extend the closed-form solution to include additional variables, enhancing its capability to fit more complex point cloud data.

Although PDE-based representations in 3D modelling can achieve higher-order continuity by specifying boundary conditions, including positional, tangential, or higher-order derivative conditions, applying this approach to PDE-based parametric surface reconstruction from point clouds presents challenges. Specifically, it is difficult to satisfy the boundary conditions while accurately approximating the interior points. To address this issue, we proposed a PDE-based deformation surface, combined with the bilinearly Coons patch to ensure that the boundary positional conditions of the point cloud are met while also approximating the interior points. The Coons patch also serves as a better base surface compared to a plane to parameterize the point clouds more effectively. Our method results in accurate outcomes without the need for postprocessing.

In some cases, achieving G^1 or C^1 continuity is essential, meaning that tangential information, in addition to positional conditions, must be consid-

ered when constructing the initial surfaces. To accomplish this, we employ a bicubic Coons patch as the initial base surface and propose a parametric deformation surface that adjusts the base surface to fit the interior points while preserving both its positional and tangential conditions at the boundaries. This approach ensures that C^1 continuity is maintained.

Although this research has focused on parametric curve and surface reconstruction from point clouds, aiming to use as few variables as possible while maintaining a good approximation, certain limitations remain. For instance, the corner point detection algorithm used in the parametric curve reconstruction is case-dependent, and developing a more general and robust detection method would be a valuable area for further exploration. Additionally, detecting other types of affine transformations, such as shear transformations, presents another challenge worth investigating.

Regarding parametric surface reconstruction, while we address gaps in PDE-based representations for surface reconstruction from point clouds, a significant challenge remains in ensuring that the analytical solution satisfies various boundary conditions to achieve higher-order continuity. Further exploration is needed to develop methods for using PDE-based representations to consistently achieve higher-order continuity.

6.2 Future Work

This research explores parametric curve and PDE-based parametric surface reconstruction from point clouds, and several areas for future work are identified.

Corner points detection. In the parametric curve reconstruction section, this thesis identifies corner points, segmenting the point set into multiple segments using existing methods. However, these methods are case-dependent. Future research will aim to develop a more general and robust corner point detection approach. Furthermore, treating the parameters for each point as free variables to be optimised might eliminate the need for parameterizing these points and could reduce the fitting error. However, this

approach leads to a high-dimensional, non-linear problem that is challenging to optimise. We believe that deep learning techniques have significant potential to address these tasks.

Higher order continuity. In the area of 3D modelling, higher-order continuity (C^2 or even higher) is desirable and can be achieved with PDE representations (Wang 2021). Specifically, PDEs ensure continuity by satisfying boundary conditions between adjacent patches. However, maintaining higher-order continuity while considering both boundary conditions and interior points remains challenging. Improving point cloud segmentation could enhance reconstruction results. With the rapid advancement of deep learning techniques, integrating them into the reconstruction pipeline could help achieve this goal. For example, Deng et al. (Deng et al. 2020) proposed using deep learning techniques for parametric surface reconstruction, aiming to improve the stitching between patches. However, they did not succeed in achieving higher-order continuity between patches. To address this, new architectures or loss functions could be developed. Furthermore, Physics-Informed Neural Networks (PINNs) have garnered significant attention, as they allow for solving complex PDEs (particularly nonlinear ones) subject to various boundary conditions using neural networks to some extent (Raissi et al. 2019). In future work, we will explore the application of this technique to PDE-based parametric surface reconstruction from point clouds.

Dynamic shape reconstruction. In some cases, the objects to be reconstructed may be in motion, making dynamic object reconstruction a challenging task in computer graphics and computer vision. By incorporating the variable t (representing time) into the PDE, dynamic object reconstruction will require more complex PDE forms and advanced solvers. Future research will explore this direction.

Bibliography

Amenta, N., Bern, M. and Kamvysselis, M., 1998. A new voronoi-based surface reconstruction algorithm. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 415–421.

Azariadis, P. and Sapidis, N., 2005. Efficient parameterization of 3d point-sets using recursive dynamic base surfaces. *Panhellenic Conference on Informatics*, Springer, 296–306.

Azariadis, P. and Sapidis, N., 2007. Product design using point-cloud surfaces: A recursive subdivision technique for point parameterization. *Computers in Industry*, 58 (8-9), 832–843.

Azariadis, P. N., 2004. Parameterization of clouds of unorganized points using dynamic base surfaces. *Computer-Aided Design*, 36 (7), 607–623.

Badki, A., Gallo, O., Kautz, J. and Sen, P., 2020. Meshlet priors for 3d mesh reconstruction. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2849–2858.

Balta, C., Öztürk, S., Kuncan, M. and Kandilli, I., 2019. Dynamic centripetal parameterization method for b-spline curve interpolation. *IEEE Access*, 8, 589–598.

Barhak, J. and Fischer, A., 2001. Parameterization and reconstruction from 3d scattered points based on neural network and pde techniques. *IEEE Transactions on visualization and computer graphics*, 7 (1), 1–16.

Bazazian, D. and Parés, M. E., 2021. Edc-net: Edge detection capsule network for 3d point clouds. *Applied Sciences*, 11 (4), 1833.

Beauville, A., 1996. *Complex algebraic surfaces*. 34, Cambridge University Press.

Belyaev, A. G., 1999. A note on invariant three-point curvature approximations (singularity theory and differential equations). , 1111, 157–164.

Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., Sharf, A. and Silva, C. T., 2017. A survey of surface reconstruction from point clouds. *Computer graphics forum*, Wiley Online Library, volume 36, 301–329.

Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Levine, J. A., Sharf, A. and Silva, C. T., 2014. State of the art in surface reconstruction from point clouds. *35th Annual Conference of the European Association for Computer Graphics, Eurographics 2014-State of the Art Reports*, The Eurographics Association.

Bhunia, A. K., Chowdhury, P. N., Yang, Y., Hospedales, T. M., Xiang, T. and Song, Y.-Z., 2021. Vectorization and rasterization: Self-supervised learning for sketch and handwriting. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5672–5681.

Bo, P., Ling, R. and Wang, W., 2012. A revisit to fitting parametric surfaces to point clouds. *Computers & Graphics*, 36 (5), 534–540.

Boissonnat, J.-D. and Cazals, F., 2000. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Proceedings of the sixteenth annual symposium on Computational geometry*, 223–232.

Bołtuć, A. and Zieniuk, E., 2021. Parametric integral equation system (pies) for solving problems with inclusions and non-homogeneous domains using bézier surfaces. *Journal of Computational Science*, 51, 101343.

Calakli, F. and Taubin, G., 2011. Ssd: Smooth signed distance surface reconstruction. *Computer Graphics Forum*, Wiley Online Library, volume 30, 1993–2002.

Chang, H.-H. and Yan, H., 1998. Vectorization of hand-drawn image using piecewise cubic bezier curves fitting. *Pattern recognition*, 31 (11), 1747–1755.

Chen, H., Liang, M., Liu, W., Wang, W. and Liu, P. X., 2022. An approach to boundary detection for 3d point clouds based on dbscan clustering. *Pattern Recognition*, 124, 108431.

Chen, X.-D., Yong, J.-H., Wang, G., Paul, J.-C. and Xu, G., 2008. Computing the minimum distance between a point and a nurbs curve. *Computer-Aided Design*, 40 (10-11), 1051–1054.

Chiang, J. Y., Tue, S. and Leu, Y., 1998. A new algorithm for line image vectorization. *Pattern recognition*, 31 (10), 1541–1549.

Choi, G. P., Liu, Y. and Lui, L. M., 2022. Free-boundary conformal parameterization of point clouds. *Journal of Scientific Computing*, 90, 1–26.

Choi, G. P.-T., Ho, K. T. and Lui, L. M., 2016. Spherical conformal parameterization of genus-0 point clouds for meshing. *SIAM Journal on Imaging Sciences*, 9 (4), 1582–1618.

Culjak, I., Abram, D., Pribanic, T., Dzapo, H. and Cifrek, M., 2012. A brief introduction to opencv. *2012 proceedings of the 35th international convention MIPRO*, IEEE, 1725–1730.

Deng, Z., Bednařík, J., Salzmann, M. and Fua, P., 2020. Better patch stitching for parametric surface reconstruction. *2020 International Conference on 3D Vision (3DV)*, IEEE, 593–602.

Dimitrov, A. and Golparvar-Fard, M., 2014. Robust nurbs surface fitting from unorganized 3d point clouds for infrastructure as-built modeling. *Computing in civil and building engineering (2014)*, 81–88.

Dominici, E. A., Schertler, N., Griffin, J., Hoshyari, S., Sigal, L. and Sheffer, A., 2020. Polyfit: Perception-aligned vectorization of raster clip-art via intermediate polygonal fitting. *ACM Transactions on Graphics (TOG)*, 39 (4), 77–1.

Duan, Y., Yang, L., Qin, H. and Samaras, D., 2004. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III 8*, Springer, 238–251.

Edelsbrunner, H., 1998. Shape reconstruction with delaunay complex. *Latin American Symposium on Theoretical Informatics*, Springer, 119–132.

Edelsbrunner, H., 2011. Alpha shapes-a survey. *Tessellations in the sciences: Virtues, techniques and applications of geometric tilings*.

Elber, G. and Kim, M.-S., 2001. Geometric constraint solver using multivariate rational spline functions. *Proceedings of the sixth ACM symposium on Solid modeling and applications*, 1–10.

Fang, J.-J. and Hung, C.-L., 2013. An improved parameterization method for b-spline curve and surface interpolation. *Computer-aided design*, 45 (6), 1005–1028.

Farin, G., 2014. *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier.

Floater, M. S., 1997. Parametrization and smooth approximation of surface triangulations. *Computer aided geometric design*, 14 (3), 231–250.

Floater, M. S., 2000. Meshless parameterization and b-spline surface approximation. *The Mathematics of Surfaces IX: Proceedings of the Ninth IMA Conference on the Mathematics of Surfaces*, Springer, 1–18.

Floater, M. S. and Reimers, M., 2001. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design*, 18 (2), 77–92.

Floater, M. S. and Surazhsky, T., 2006. Parameterization for curve interpolation. *Studies in Computational Mathematics*, Elsevier, volume 12, 39–54.

Franchini, E., Morigi, S., Sgallari, F. et al., 2010. Implicit shape reconstruction of unorganized points using pde-based deformable 3d manifolds. *Numerical Mathematics: Theory, Methods and Applications*, 3 (4), 405–430.

Fuhrmann, S. and Goesele, M., 2014. Floating scale surface reconstruction. *ACM Transactions on Graphics (ToG)*, 33 (4), 1–11.

G. Farin, J. H. and Kim, M.-S., 2002. *Handbook of computer aided geometric design*. Elsevier.

Gálvez, A. and Iglesias, A., 2012. Particle swarm optimization for non-uniform rational b-spline surface reconstruction from clouds of 3d data points. *Information Sciences*, 192, 174–192.

Gálvez, A. and Iglesias, A., 2013. A new iterative mutually coupled hybrid ga–pso approach for curve fitting in manufacturing. *Applied Soft Computing*, 13 (3), 1491–1504.

Gálvez, A., Iglesias, A. and Puig-Pey, J., 2012. Iterative two-step genetic-algorithm-based method for efficient polynomial b-spline surface reconstruction. *Information Sciences*, 182 (1), 56–76.

Gomes, L., Bellon, O. R. P. and Silva, L., 2014. 3d reconstruction methods for digital preservation of cultural heritage: A survey. *Pattern Recognition Letters*, 50, 3–14.

Gonczarowski, J., 1991. A fast approach to auto-tracing (with parametric cubics). *Raster imaging and digital typography*, volume 91, 1–15.

Gordon, W. J. and Riesenfeld, R. F., 1974. B-spline curves and surfaces. *Computer aided geometric design*, Elsevier, 95–126.

Gu, P. and Yan, X., 1995. Neural network approach to the reconstruction of freeform surfaces for reverse engineering. *Computer-Aided Design*, 27 (1), 59–64.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. and Bennamoun, M., 2020. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43 (12), 4338–4364.

Hadenfeld, J., 1995. Local energy fairing of b-spline surfaces.

He, L., Ren, X., Gao, Q., Zhao, X., Yao, B. and Chao, Y., 2017. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70, 25–43.

He, Y., Kang, S. H. and Morel, J.-M., 2023. Binary shape vectorization by affine scale-space. *Image Processing On Line*, 13, 22–37.

He, Y., Yu, H., Liu, X., Yang, Z., Sun, W. and Mian, A., 2021. Deep learning based 3d segmentation: A survey. *arXiv preprint arXiv:2103.05423*.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W., 1992. Surface reconstruction from unorganized points. *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, 71–78.

Hormann, K. and Greiner, G., 2000. Mips: An efficient global parametrization method. *Curve and Surface Design: Saint-Malo 1999*, 153–162.

Hormann, K. and Reimers, M., 2002. Triangulating point clouds with spherical topology. *Curve and Surface Design: Saint-Malo*, 215–224.

Hoshyari, S., Dominici, E. A., Sheffer, A., Carr, N., Wang, Z., Ceylan, D. and Shen, I.-C., 2018. Perception-driven semi-structured boundary vectorization. *ACM Transactions on Graphics (TOG)*, 37 (4), 1–14.

Iglesias, A., Gálvez, A. and Collantes, M., 2015. Bat algorithm for curve parameterization in data fitting with polynomial bézier curves. *2015 International Conference on Cyberworlds (CW)*, IEEE, 107–114.

Illingworth, J. and Kittler, J., 1988. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44 (1), 87–116.

Johnson, D. E. and Cohen, E., 2005. Distance extrema for spline models using tangent cones. *Proceedings of Graphics Interface 2005*, 169–175.

Joshi, P., 2014. Image vectorization and significant point detection. AIAA, Proceedings of International Conference on Advances in Engineering and Technology, 74–78.

Jung, H. and Kim, K., 2000. A new parameterisation method for nurbs surface interpolation. *The International Journal of Advanced Manufacturing Technology*, 16 (11), 784–790.

Kanazawa, A., Tulsiani, S., Efros, A. A. and Malik, J., 2018. Learning category-specific mesh reconstruction from image collections. *Proceedings of the European Conference on Computer Vision (ECCV)*, 371–386.

Kato, H., Ushiku, Y. and Harada, T., 2018. Neural 3d mesh renderer. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3907–3916.

Kazhdan, M., Bolitho, M. and Hoppe, H., 2006. Poisson surface reconstruction. *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7.

Khameneifar, F. and Ghorbani, H., 2019. On the curvature estimation for noisy point cloud data via local quadric surface fitting. *Comput.-Aided Des. Appl.*, 16 (1), 140–149.

Kirsanov, A., Vavilin, A. and Jo, K., 2010. Contour-based algorithm for vectorization of satellite images. *International Forum on Strategic Technology 2010*, IEEE, 241–245.

Laidlaw, D. H., Trumbore, W. B. and Hughes, J. F., 1986. Constructive solid geometry for polyhedral objects. *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 161–170.

Laube, P., Franz, M. O. and Umlauf, G., 2018. Deep learning parametrization for b-spline curve approximation. *2018 International Conference on 3D Vision (3DV)*, IEEE, 691–699.

Lee, D., Quan, I., Wu, C., Wu, J., Tamir, D. and Rishe, N., 2020. Optimizing b-spline surface reconstruction for sharp feature preservation. *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 0359–0364.

Li, E., Che, W., Zhang, X., Zhang, Y.-K. and Xu, B., 2011a. Direct quad-dominant meshing of point cloud via global parameterization. *Computers & Graphics*, 35 (3), 452–460.

Li, E., Lévy, B., Zhang, X., Che, W., Dong, W. and Paul, J.-C., 2011b. Meshless quadrangulation by global parameterization. *Computers & Graphics*, 35 (5), 992–1000.

Li, X., Wu, Z., Pan, F., Liang, J., Zhang, J. and Hou, L., 2019. A geometric strategy algorithm for orthogonal projection onto a parametric surface. *Journal of Computer Science and Technology*, 34, 1279–1293.

Lim, C.-G., 1998. *A universal parametrization in B-spline curve and surface interpolation and its performance evaluation*. Louisiana State University and Agricultural & Mechanical College.

Lim, S. P. and Haron, H., 2014. Surface reconstruction techniques: a review. *Artificial Intelligence Review*, 42, 59–78.

Linz, C., Goldlücke, B. and Magnor, M., 2006. A point-based approach to pde-based surface reconstruction. *Pattern Recognition: 28th DAGM Symposium, Berlin, Germany, September 12-14, 2006. Proceedings 28*, Springer, 729–738.

Lopes, R. G., Ha, D., Eck, D. and Shlens, J., 2019. A learned representation for scalable vector graphics. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7930–7939.

Ma, W. and Kruth, J.-P., 1995. Parameterization of randomly measured points for least squares fitting of b-spline curves and surfaces. *Computer-Aided Design*, 27 (9), 663–675.

McNamee, J. M. and Pan, V., 2013. *Numerical Methods for Roots of Polynomials-Part II*. Newnes.

Meng, Q., Li, B., Holstein, H. and Liu, Y., 2013. Parameterization of point-cloud freeform surfaces using adaptive sequential learning rbfnetworks. *Pattern Recognition*, 46 (8), 2361–2375.

Meng, T. W., Choi, G. P.-T. and Lui, L. M., 2016. Tempo: feature-endowed teichmuller extremal mappings of point clouds. *SIAM Journal on Imaging Sciences*, 9 (4), 1922–1962.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S. and Geiger, A., 2019. Occupancy networks: Learning 3d reconstruction in function space. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4460–4470.

Mineo, C., Pierce, S. G. and Summan, R., 2019. Novel algorithms for 3d surface point cloud boundary detection and edge reconstruction. *Journal of Computational Design and Engineering*, 6 (1), 81–91.

Monterde, J. and Ugail, H., 2006. A general 4th-order pde method to generate bézier surfaces from the boundary. *Computer Aided Geometric Design*, 23 (2), 208–225.

Mortenson, M. E., 1997. *Geometric modeling*. John Wiley & Sons, Inc.

Nadal, C., Legault, R. and Suen, C. Y., 1990. Complementary algorithms for the recognition of totally unconstrained handwritten numerals. *[1990] Proceedings. 10th International Conference on Pattern Recognition*, IEEE, volume 1, 443–449.

Nguyen, A. and Le, B., 2013. ª3d point cloud segmentation: A survey, º in 2013 6th ieee conference on robotics, automation and mechatronics (ram).

Nielson, G. M. and Foley, T. A., 1989. A survey of applications of an affine invariant norm. *Mathematical methods in computer aided geometric design*, Elsevier, 445–467.

Othman, M., Yusoff, Y., Haron, H. and You, L., 2019. An overview of surface reconstruction using partial differential equation (pde). *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, volume 551, 012054.

Pal, S., Ganguly, P. and Biswas, P., 2007. Cubic bézier approximation of a digitized curve. *Pattern recognition*, 40 (10), 2730–2741.

Pavlidis, T., 1983. Curve fitting with conic splines. *ACM Transactions on Graphics (TOG)*, 2 (1), 1–31.

Piegl, L., 1988. Coons-type patches. *Computers & graphics*, 12 (2), 221–228.

Piegl, L. and Tiller, W., 2012. *The NURBS book*. Springer Science & Business Media.

Plass, M. and Stone, M., 1983. Curve-fitting with piecewise parametric cubics. *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, 229–239.

Pottmann, H., Leopoldseder, S. and Hofer, M., 2002. Approximation with active b-spline curves and surfaces. *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.*, IEEE, 8–25.

Qi, C. R., Su, H., Mo, K. and Guibas, L. J., 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Raissi, M., Perdikaris, P. and Karniadakis, G. E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378, 686–707.

Raja, V. and Fernandes, K. J., 2007. *Reverse engineering: an industrial perspective*. Springer Science & Business Media.

Rodrigues, M., Osman, A. and Robinson, A., 2013. Partial differential equations for 3d data compression and reconstruction. *ADSA Advances in Dynamical Systems and Applications*, 8 (2), 303–315.

Rouhani, M., Sappa, A. D. and Boyer, E., 2014. Implicit b-spline surface reconstruction. *IEEE transactions on image processing*, 24 (1), 22–32.

Salomon, D., 2007. *Curves and surfaces for computer graphics*. Springer Science & Business Media.

Sarfraz, M. and Khan, M., 2004. An automatic algorithm for approximating boundary of bitmap characters. *Future Generation Computer Systems*, 20 (8), 1327–1336.

Schnabel, R., Wahl, R. and Klein, R., 2007. Efficient ransac for point-cloud shape detection. *Computer graphics forum*, Wiley Online Library, volume 26, 214–226.

Schneider, P. J., 1990. An algorithm for automatically fitting digitized curves. *Graphics gems*, 612–626.

Scholz, F. and Jüttler, B., 2021. Parameterization for polynomial curve approximation via residual deep neural networks. *Computer Aided Geometric Design*, 85, 101977.

Shamsuddin, S. M. H. and Ahmed, M. A., 2004. A hybrid parameterization method for nurbs. *Proceedings. International Conference on Computer Graphics, Imaging and Visualization, 2004. CGIV 2004.*, IEEE, 15–20.

Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S. and Měch, R., 2020. Parsenet: A parametric surface fitting network for 3d point clouds. *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, Springer, 261–276.

Sharma, R., Schwandt, T., Kunert, C., Urban, S. and Broll, W., 2021. Point cloud upsampling and normal estimation using deep learning for robust surface reconstruction. *arXiv preprint arXiv:2102.13391*.

Shen, Y., Ren, J., Huang, N., Zhang, Y., Zhang, X. and Zhu, L., 2023. Surface form inspection with contact coordinate measurement: a review. *International Journal of Extreme Manufacturing*, 5 (2), 022006.

Shi, B.-Q., Liang, J. and Liu, Q., 2011. Adaptive simplification of point cloud using k-means clustering. *Computer-Aided Design*, 43 (8), 910–922.

S.L., F. C., 2025. *FREEPik*. URL `https://www.freepik.com`, [Accessed 5 February 2023].

Sulzer, R., Marlet, R., Vallet, B. and Landrieu, L., 2023. A survey and benchmark of automatic surface reconstruction from point clouds. *arXiv preprint arXiv:2301.13656*.

Tewari, G., Gotsman, C. and Gortler, S. J., 2006. Meshing genus-1 point clouds using discrete one-forms. *Computers & Graphics*, 30 (6), 917–926.

Ugail, H. and Kirmani, S., 2006. Method of surface reconstruction using partial differential equations. *Proceedings of the 10th WSEAS International Conference on Computers, Athens, Greece*, 13–15.

Unther Greiner, G. and Hormann, K., 1996. Interpolating and approximating scattered 3d-data with hierarchical tensor product b-splines. *Proceedings of Chamonix*, volume 1.

Varady, T., Martin, R. R. and Cox, J., 1997. Reverse engineering of geometric models—an introduction. *Computer-aided design*, 29 (4), 255–268.

Wang, L., Yuan, B. and Miao, Z., 2008. 3d point clouds parameterization alogrithm. *2008 9th International Conference on Signal Processing*, IEEE, 1410–1413.

Wang, S., 2021. *Partial differential equation-based surface modelling and applications.*. Ph.D. thesis, Bournemouth University.

Williams, F., Schneider, T., Silva, C., Zorin, D., Bruna, J. and Panozzo, D., 2019. Deep geometric prior for surface reconstruction. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10130–10139.

Xiao, J. and Furukawa, Y., 2014. Reconstructing the world's museums. *International journal of computer vision*, 110, 243–258.

Xiong, X., Feng, J. and Zhou, B., 2017. Real-time contour image vectorization on gpu. *Computer Vision, Imaging and Computer Graphics Theory and Applications: 11th International Joint Conference, VISIGRAPP 2016, Rome, Italy, February 27–29, 2016, Revised Selected Papers 11*, Springer, 35–50.

Xiyu, L., Mingxi, T. and Hamilton Frazer, J., 2003. Shape reconstruction by genetic algorithms and artificial neural networks. *Engineering Computations*, 20 (2), 129–151.

Xu, H., Zhao, G., Liu, Y. and Ye, N., 2022. An improved parameterized interpolation method based on modified chord length. *Journal of Computing and Information Science in Engineering*, 22 (6), 061001.

Yavartanoo, M., Chung, J., Neshatavar, R. and Lee, K. M., 2021. 3dias: 3d shape reconstruction with implicit algebraic surfaces. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12446–12455.

Yuwen, S., Dongming, G., Zhenyuan, J. and Weijun, L., 2006. B-spline surface reconstruction and direct slicing from point clouds. *The International Journal of Advanced Manufacturing Technology*, 27, 918–924.

Zhang, L., Liu, L., Gotsman, C. and Huang, H., 2010. Mesh reconstruction by meshless denoising and parameterization. *Computers & Graphics*, 34 (3), 198–208.

Zhao, H.-K., Osher, S. and Fedkiw, R., 2001. Fast surface reconstruction using the level set method. *Proceedings IEEE workshop on variational and level set methods in computer vision*, IEEE, 194–201.

Zhu, Z., Iglesias, A., You, L. and Zhang, J. J., 2022a. A review of 3d point clouds parameterization methods. *International Conference on Computational Science*, Springer, 690–703.

Zhu, Z., Zheng, A., Iglesias, A., Wang, S., Xia, Y., Chaudhry, E., You, L. and Zhang, J., 2022b. Pde patch-based surface reconstruction from point clouds. *Journal of Computational Science*, 61, 101647.

Zou, J. J. and Yan, H., 2001. Cartoon image vectorization based on shape subdivision. *Proceedings. Computer Graphics International 2001*, IEEE, 225–231.

Zwicker, M. and Gotsman, C., 2004. Meshing point clouds using spherical parameterization. *PBG*, Citeseer, 173–180.

Zwicker, M., Pauly, M., Knoll, O. and Gross, M., 2002. Pointshop 3d: An interactive system for point-based surface editing. *ACM Transactions on Graphics (TOG)*, 21 (3), 322–329.