Department of Precision and Microsystems Engineering

Shape Correction For 3D Laser Marking

Yu-Xuan Chuang

Report no : 2023.025
Coach : Dr. Martijn Boerkamp
Professor : Dr. Nandini Bhattacharya
Specialisation : Optics for Technology
Type of report : Msc Thesis
Date : 14 April 2023

Shape Correction for 3D Laser Marking

by

Yu-Xuan Chuang

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended on Tuesday, April 26, 2023.

Student number:	5456975	
Project duration:	July 11, 2022 - April 11, 2023	
Thesis committee:	Dr. Nandini Bhattacharya	TU Delft, Supervisor
	Dr. Martijn Boerkamp	inPhocal, Supervisor
	Dr. Alejandro Aragon	TU Delft
	Dr. Andres Hunt	TU Delft

TU Delft inPhocal

Preface

I would like to express my heartfelt thanks to my supervisor Dr. Martijn Boerkamp from inPhocal for the opportunity to work on this project. His supervision and valuable feedback keep me on the right direction to finish this project. I would also like to thank my university supervisor Dr. Nandini Bhattacharya for her help and guidance. Then, I would like to thank Bas Hankel and Sajal Kumardhara for their assistance in using the experimental setup and for the precious discussions when I face problems. Finally, I would like to express my sincere gratitude to all inPhocal colleagues for having me as a part of the inPhocal team.

Yu-Xuan Chuang
Delft, March 2023

Abstract

A startup company, inPhocal, specializing in laser marking systems, develops their own optical module to generate the laser beam with a long depth of focus. Due to the long depth of focus, 3-dimensional curved surfaces can be marked without changing the distance between the target surface and the laser marking system, accomplishing high-speed 3D laser marking.

Nowadays, people use ink and stickers to label information like expiry date, barcode, logo, etc on products, which causes a serious environmental problem. On the other hand, laser marking will not produce extra waste, making it a sustainable way to do the labeling. Consequently, inPhocal tries to replace ink and stickers with their laser marking system. However, laser marking will project 2-dimensional patterns onto the 3-dimensional surfaces orthogonally, and these patterns will be inevitably distorted. In this research, the state-of-the-art method to eliminate this distortion will be introduced and the problem of this method will also be discussed. Then, a new method will be developed in this research.

Currently, to map 2-dimensional patterns onto 3-dimensional surfaces, surface parameterization is used. By finding the 2-dimensional parameterized plane of the target surface and putting a 2-dimensional pattern onto this plane, this pattern can be mapped by reversing the parameterized plane as well as the 2-dimensional pattern to the original 3-dimensional surface. Then, by extracting the x and y coordinates of the mapped pattern, a 2-dimensional corrected pattern is obtained and can be marked on the curved surface without distortion. This method has some issues, making it take too long for correcting patterns for doing laser marking in the production line.

In this research, a new method based on non-uniform rational b-spline (NURBS) is proposed in order to provide a fast and precise method to get the corrected pattern. The full development process and the simulation results will be provided. After that, a comparison of time consumption and performance between this new method and the existing method will be given.

Finally, there are some different scenarios when applying this new shape correction method, and each scenario will take a different time to finish the shape correction for a 2-dimensional pattern, and the correction errors within these scenarios are also different. Therefore, a discussion of these scenarios and how to choose the proper scenario in practice will be presented.

Contents

Preface **I**

Abstract **II**

1 Introduction **1**
1.1 Background and motivation 1
1.2 Problem statement 3
1.2.1 State-of-the-Art: surface parameterization 4
1.2.2 Gap in State-of-the-Art 6
1.3 Research objective 6
1.4 Outline 6

2 Preliminary knowledge **7**
2.1 Structured beams 7
2.2 Non-uniform rational B-spline (NURBS) 9
2.2.1 NURBS curve 9
2.2.2 NURBS surface 12
2.3 Line integral on NURBS 13
2.3.1 First fundamental form 14
2.3.2 Line integral of NURBS curve 14
2.3.3 Line integral of NURBS surface 15
2.3.4 Numerical integration: Composite Simpson's rule 16

3 Shape correction through NURBS **17**
3.1 NURBS surface reconstruction 17
3.2 Pattern mapping 20
3.2.1 Zero padding 21
3.2.2 Mapping domain 23
3.2.3 Pattern mapping 25

4 Methodology **28**
4.1 Experimental setup 28
4.1.1 Shape detecting system 28
4.1.2 Laser marking system 31
4.2 Simulation 32

5 Experimental results **35**
5.1 Marking on the object with given well-designed surface model: computer mouse . . 36
5.2 Marking on the object with unknown shape surface model: mandarin 40

6 Discussion **44**

6.1 Effects of number of subintervals in Simpson's Rule on correction error 44
6.2 Effects of number of points to create mapping domain on correction error 45
6.3 Comparison between shape correction method based on NURBS and surface parameterization 48

7 Conclusion **49**

A Algorithms for NURBS surface **51**
A.1 Data rearrangement algorithm 51
A.2 Data selection algorithm 51
A.3 Chord length parameterization algorithm 52
A.4 Finding knot vector algorithm 52
A.5 Control points algorithm 53
A.6 Basis function algorithm 53
A.7 Rational function algorithm 54
A.8 Get knot vector algorithm 54
A.9 First fundamental form 54
A.10 Line length on NURBS algorithm 55

B BeamConstruct software **56**

C Simulation results of effects of subintervals number in Simpson's Rule on correction error **59**

D Simulation results of effects of mapping domain points number on correction error **65**

E Inspecting shift of the same point on original and orthogonally projected pattern **76**

Chapter 1

Introduction

In this chapter, the background and motivation of this research project will be discussed. Then, the problem statement of the project and the deficiency of the current solution will be discussed. Finally, the goal of this research project and the outline of this thesis will be introduced.

1.1 Background and motivation

Light **A**mplification by **S**timulated **E**mission of **R**adiation (Laser) is electromagnetic radiation emitted by stimulating atoms or molecules. In Atomic theory, electrons orbit randomly around the atomic nucleus and the distances between the atomic nucleus and electrons depend on its energy level. If an electron is stimulated by the external energy, it can be excited to a higher energy level. When this electron falls back to a lower energy level, it will emit radiation with energy equal to the energy difference between this high and low energy level. Figure 1.1 shows the design of a laser, the electrons of the gain medium will get stimulated by the pumping energy, and when the excited electrons fall back to the lower energy level, they will emit photons. However, if the electrons emitted photons through spontaneous emission, the photons will have low coherence. To solve this, the laser uses the metastability of electrons. The electrons in a meta-stable state will be less likely to have spontaneous emission, but more likely to emit photons when other photons pass through them, and the emitted photons will have the same wavelength as the passed-through photons. This process is also known as stimulated emission. Then, these emitted photons will be bounced back and forth between the high reflector(a mirror) and the output coupler (a partially transparent mirror) and make stimulated emission occur again and again to get more photons to form a laser beam and goes out from the output coupler. The light beam coming out of the laser will have high coherence, low divergence, and high power density within a small spot.

Due to these characteristics, lasers are widely used in the industrial sector, from leaving visible markings on the materials to even cutting through the materials. Since laser processing is a process to make materials melt or evaporated by absorbing the energy from the laser beam, the absorption rate of materials, which is related to the wavelength of the laser beam, plays an important role in laser processing. However, the wavelength of the laser beam can be manipulated by applying different gain mediums(e.g. CO_2 laser with wavelength 10600 nm can be used in processing organic materials, and Nd:YAG laser with wavelength 1064 nm can be used in processing metals), making laser processing extraordinary flexible and it can be used for almost all materials.

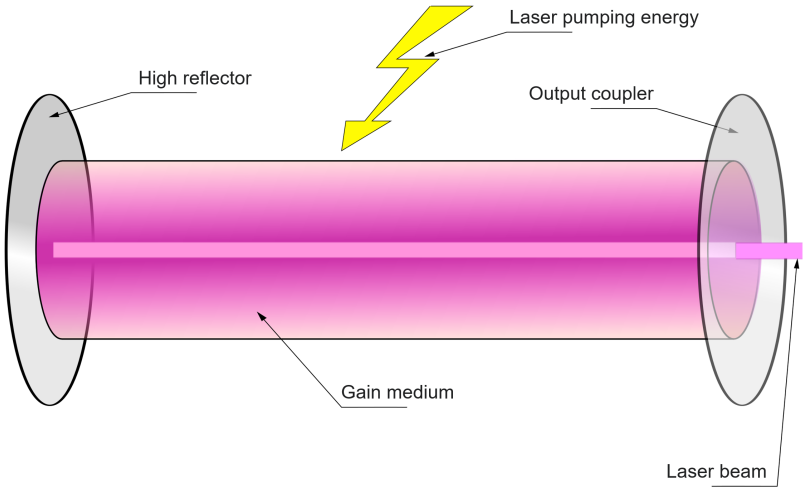
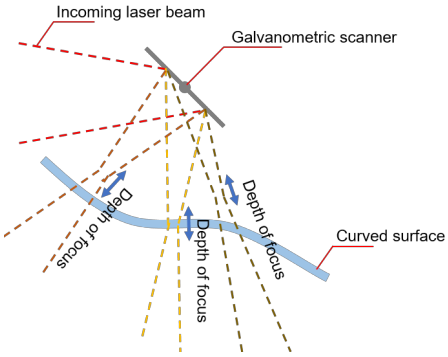


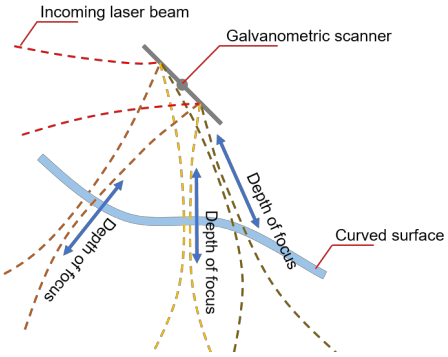
Figure 1.1: Design of laser [1].

Besides, Laser processing is a non-contact machining method, so it gets rid of the cutting tool wear in traditional machining methods, thereby reducing the cost and time-consuming need of replacing cutting tools. Even more, with the laser marking machine, information about products such as expiration dates, bar codes, or logos (this "information" will be represented by "pattern" in the rest part of this thesis), can be printed directly without ink and stickers, which are the dominant way to show such pattern. These, however, pollute the environment by a large degree. As environmental awareness rises, a sustainable way to leave the pattern on products without producing pollutants is demanded, and laser marking seems to be an ideal way to replace inkjet printing and stickers. Nevertheless, since the products usually have 3-dimensional surfaces, there is still a severe problem with traditional laser marking systems. For laser marking, a laser beam should be able to focus on any position where it is expected to leave a mark, which needs to refocus the laser beam again and again when marking on a 3-dimensional surface and slows down the processing speed.

inPhocal, a startup company specializing in high-speed laser marking systems, develops a novel optical module to make the depth of focus of a laser beam much longer than traditional laser beam. Based on CERN's technology [2], inPhocal utilized spherical aberration to transform the incoming laser beam with Gaussian energy distribution into a laser beam with concentric energy distribution. The concentric structure enables a laser beam to stay in focus for a long distance after being converged by a positive lens (see section 2.1). In Figure 1.2, how the depth of focus of the laser beam affects the laser marking is shown. The laser beam with a short depth of focus will be out of focus easily, so the energy with be dissipated into a large area, making the energy density too low to leave marks. On the contrary, the laser beam with a long depth of focus can retain its energy density at a much longer distance than the short depth of focus laser beam can, so it can even leave mark on a curved surface. In traditional laser marking system, the laser beam's depth of focus is usually a few mm, but it can reach 5cm in inPhocal's system, enabling the laser marking system to process a wider range without refocusing the laser beam. With the high-speed laser marking system of inPhocal, the main drawback i.e., the processing speed, of laser marking compared to inkjet printing and stickers is solved. Yet, a distortion will be inevitably introduced when marking a 2-dimensional pattern onto a 3-dimensional surface. Hence, to accomplish leaving marks on 3-dimensional objects by laser marking, a shape correction method for marking distortion-free patterns onto 3-dimensional objects is required.



(a) Laser marking with short depth of focus.



(b) Laser marking with long depth of focus.

Figure 1.2: Effects of laser beam's depth of focus on laser marking.

1.2 Problem statement

A laser marking system is mainly composed of a laser source, two mirrors that can be rotated perpendicularly with respect to each other by galvanometers, and an f-theta lens that can focus the laser beam on the image plane (Figure 1.3).

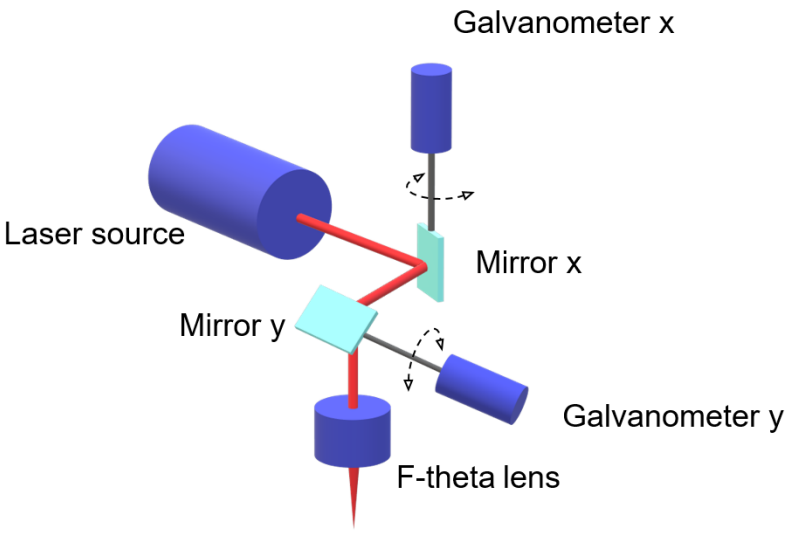


Figure 1.3: Sketch of laser marking.

By giving the x and y coordinates of 2-dimensional pattern that wants to be marked as the control signal of a galvanometer, the laser beam will be directed to this position on the image plane. If the 2-dimensional pattern is marked on a 3-dimensional curved surface, it will be distorted like Figure 1.4. Thus, a method to deform the 2-dimensional pattern beforehand to diminish this distortion to get a proper marking on the 3-dimensional surface is necessary.

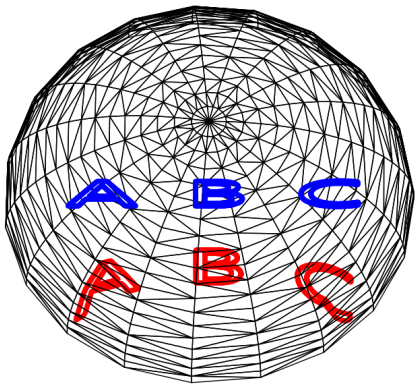


Figure 1.4: 2D information(blue) and distorted mark after marking onto 3D surface(red).

1.2.1 State-of-the-Art: surface parameterization

In the current state of the art, the pre-deformed 2-dimensional pattern for laser marking without distortion can be obtained by finding a 2-dimensional parameterized plane of the 3-dimensional surface, mapping the 2-dimensional pattern onto this plane, and reversing the 2-dimensional parameterized plane as well as the mapped 2-dimensional pattern back to the original 3-dimensional surface. Then, by removing the original 3-dimensional surface, the mapped pattern without distortion is left, and the laser marking process can be finished by giving the x and y coordinates as the control signal of the laser marking system. The 2-dimensional parameterized plane can be obtained by triangulating the original 3-dimensional surface first, and by following two different criteria: keep the edge-lengths [3] or keep the angles [4] of triangles of the triangulated surface, the triangulated surface can be flattened into a 2-dimensional plane. Because some surfaces, such as spheres, are topologically non-developable, they will lose the accuracy in angle when they are flattened by following the edge-lengths criterion and vice versa. In some cases, the surfaces will be flattened into a plane with a complex shape (Figure 1.5), so the 2-dimensional pattern needs to be pre-processed in order to fit this shape, making the mapping process very complicated. Therefore, there is another way to get a parameterized plane. Based on graph theory [5], Michael S Floater [6] proposed a method to parameterize surfaces into a predefined convex polygon. In this method, the boundary of the triangulated surface will be mapped into a convex polygon like a unit square or a unit circle (Figure 1.6). Then, the inner vertices of triangles will be determined by solving a linear system based on convex combinations. In this method, the area of the parameterized plane will not be the same as that of the original 3-dimensional surface, so the 2-dimensional pattern still needs to be resized before being mapped onto the parameterized plane. For both methods, the relationship between a 3-dimensional surface and its parameterized plane is a unique affine mapping (Figure 1.7):

$$P(u) = \frac{A(u, u_2, u_3)p_1 + A(u_1, u, u_3)p_2 + A(u1, u_2, u)p_3}{A(u_1, u_2, u_3)p_2} \tag{1.1}$$

where $\{p_1, p_2, p_3\}$ are the vertices of a triangle of the 3-dimensional triangulated surface and $\{u_1, u_2, u_3\}$ are the vertices of its corresponding triangle on the parameterized plane. To map the 2-dimensional pattern onto the 3-dimensional surface, each point of the 2-dimensional pattern should be located in one triangle on the parameterized plane, and the corresponding location on the 3-dimensional surface can be found through affine mapping.

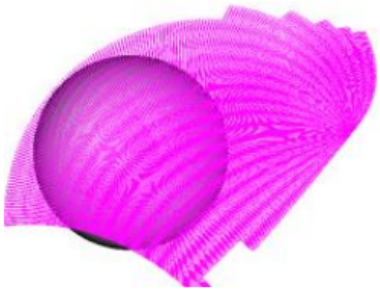
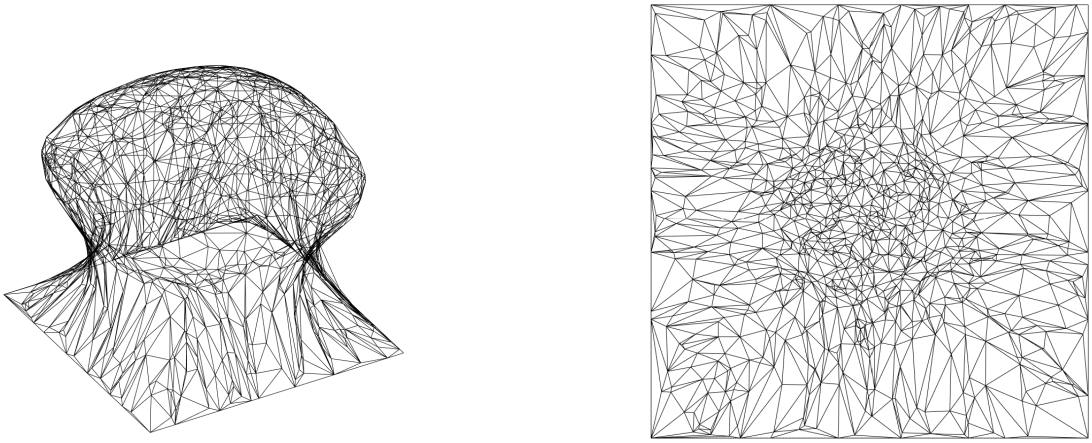


Figure 1.5: Flattened sphere with complex shape [7].



(a) Triangulated salt dome.

(b) Parameterization of salt dome.

Figure 1.6: Surface parameterization through Floater's method [6].

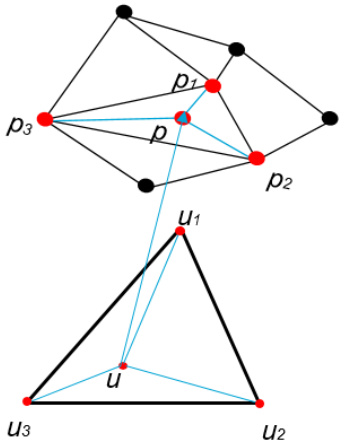


Figure 1.7: Affine mapping.

1.2.2 Gap in State-of-the-Art

As mentioned in subsection 1.2.1., the 2-dimensional pattern needs to be either pre-processed or resized before being mapped, and the process of locating points of the 2-dimensional pattern in the triangles on the parameterized plane is time-consuming. When applying laser marking to the production line, it is desired to mark products as fast as possible to maximize productivity. In other words, a time-consuming pattern mapping method that will slow the speed of production line down is not suitable for the fast production line. Although it is possible to use a powerful computer to speed up the calculation, the cost will also be dramatically increased since the expense of the powerful computer itself and the energy consumption of operating it can be extremely high. Due to these drawbacks, surface parameterization is not an ideal way for generating pre-deformed 2-dimensional patterns for laser marking, and it can be said that there is still lacking a fast and easy way to generate pre-deformed 2-dimensional patterns.

1.3 Research objective

With the unique optical module developed by inPhocal, the laser beam is able to maintain its focus entirely over a 3-dimensionally shaped object. Therefore, some companies are interested in employing this novel system in their production line. For example, with the increasing demand for personalized computer hardware, such as mouses and keyboards, Logitech plans to use laser marking to decorate these products with unique patterns. This system can also be used in labeling the expiry date, barcode, or logo on the products themselves, so the usage of ink and sticker would be reduced. For this application, the objective of this project is to provide an algorithm that can deform the 2-dimensional pattern properly beforehand so that it can be marked onto a 3-dimensional surface without losing its profile. For Logitech case, the surface of the object is determined, but for the other cases, e.g. marking on a fruit, the surface of the object needs to be acquired by a shape detection device. Hence, the algorithm proposed in this project should be able to deal with correcting the shape of the 2-dimensional pattern after marking onto both an object with a pre-determined model and an object with an unknown shape.

1.4 Outline

This thesis has introduced the basic knowledge of the laser and how inPhocal's laser marking system can possibly replace ink and stickers on labeling products in chapter 1. In chapter 2, the detail of the long depth of focus laser beam, NURBS, which is the background knowledge of the pattern's shape correction in this research, and how to calculate the curve length on NURBS surfaces will be presented. The algorithm of the new shape correction method proposed in this research will be developed in chapter 3. chapter 4 shows the shape-detecting sensor set-up for getting the surface information of objects without the given surface model and the inPhocal's laser marking system set-up. The simulation results of mapping distortion-free patterns by using python for both objects with and without the given surface model are shown in this chapter as well. For chapter 5, the experimental results of the NURBS-based shape correction method proposed in this research will be shown, and the simulation results of the state-of-the-art shape correction (surface parameterization-based) also will be given for comparison. In chapter 6, the factors that will affect the performance of the shape correction will be discussed, and the optimal scenario will be provided. Besides, the advantages of the new shape correction method will also be discussed. Finally, the conclusion will be made in chapter 7.

Chapter 2

Preliminary knowledge

In this chapter, the way to produce a long depth-of-focus laser beam will be discussed to give an insight into how inPhocal's laser marking system can process a 3-dimensional object without losing focus. Then, Non-uniform rational B-spline (NURBS) will be introduced to reconstruct 3-dimensional object surfaces for deforming the 2-dimensional patterns beforehand. In order to map a distortion-free 2-dimensional pattern onto a 3-dimensional surface, the length between two given points of the 2-dimensional pattern should keep consistent after mapping onto the 3-dimensional surface, so the line integral on NURBS will also be introduced.

2.1 Structured beams

In laser marking, the laser beam should be focused on the point that should be processed to make the material absorb enough energy to induce chemical reaction, charring, melting, or evaporating. A laser beam with a short depth of focus will quickly lose its focus when processing a surface with height variation, making it only applicable to flat surfaces. On the other hand, a laser beam with a long depth of focus and high energy intensity can make the beam spot contain enough energy for a long distance, which allows it to process curved surfaces and make it an ideal source for laser marking. Traditionally, the light beam that comes out from the laser is a Gaussian beam, whose depth of focus is short, and the energy distribution within the beam's spot is a Gaussian distribution, making it far from an ideal source for laser marking. In contrast, the Bessel beam is a type of light beam that has a concentric structure that can be used to make a depth of focus that is much longer than what can be obtained with a Gaussian beam. A Bessel beam can be produced by making a Gaussian beam pass through an axicon lens. (Figure 2.1). Because of the conical structure of an axicon, a Gaussian beam with a planar wavefront will be bent in different progressing directions. Then, these bent wavefronts will interact with each other to form a light beam with intensity distribution on the cross-section plane being represented by the Bessel function, which is known as a Bessel beam. When a Bessel beam is converged by a positive lens, the central spot will be focused at the focal point. With the propagation of the beam, this central spot will diverge after the focal point, but the surrounding rings will also be converged to form the new central spot, making the long depth of focus of the Bessel beam. There are still some other properties of the Bessel beam to make it supremely qualified for serving as the source for laser marking: non-diffracting, self-healing, low divergence of the central spot, and small size of the central spot.

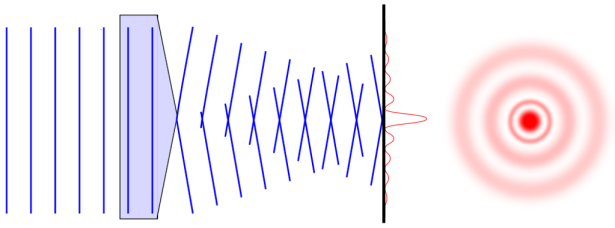


Figure 2.1: A Bessel beam created by an axicon. [8]

Theoretically, the central spot of the Bessel beam is surrounded by infinity concentric rings, and the energy within a Bessel beam is evenly distributed into the central spot and these rings. That is to say, the Bessel beam will contain infinite energy, so the ideal Bessel beam can never be generated. Besides that, since only the central spot is used in laser marking, the energy within the rings would be wasted. To solve this problem, CERN proposed several optical systems for producing structured beams (SBs), which share the same properties as Bessel beams(i.e., long depth-of-focus, low divergence, etc.) but don't have infinity concentric rings and most of the energy is in the central spot. When a collimated light beam passes through a convex lens with a high refractive index, the different radial segments of the incoming beam will follow different paths with different phase delays due to the spherical surface of the convex lens, which is also known as spherical aberration. After emerging from the convex lens, the rays with different phases will interfere with each other and form aSB. Because of the high refractive index of the convex lens, the emerging beam would diverge quickly, so putting a focusing lens behind the convex lens is needed to capture the SB. Figure 2.2 shows one of the optical systems proposed by CERN to generate SBs, the convex lens used in this system is a ball lens, and the focusing lens used in capturing the SB is called an expander lens.

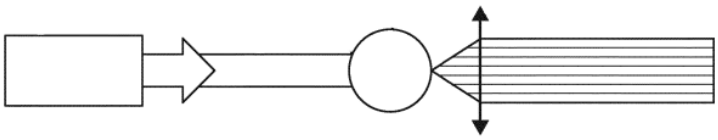
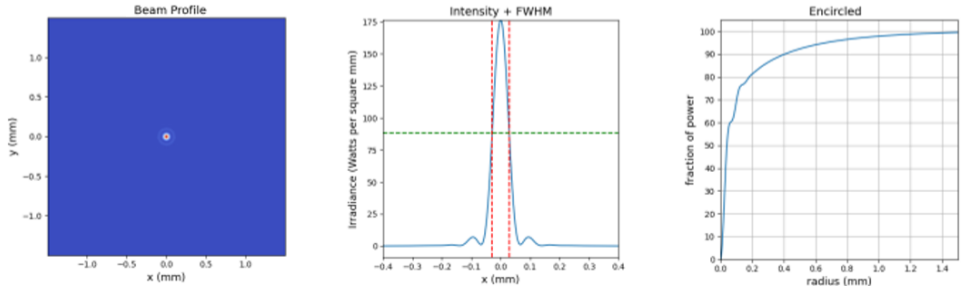


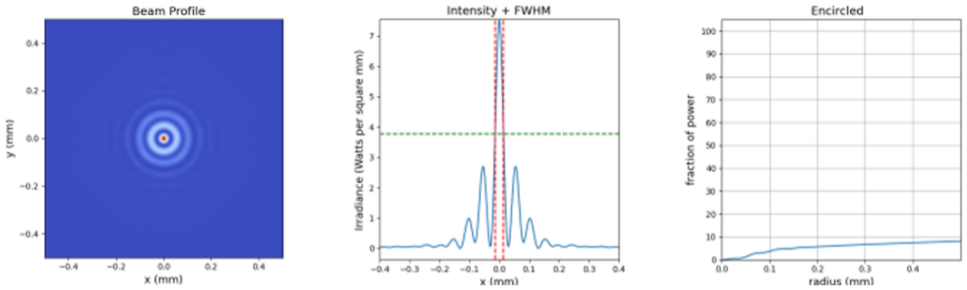
Figure 2.2: An optical system with a ball lens and an expander lens to generate structure beams [2].

There is still a downside to CERN's technology. The SB that comes out from the ball lens will diverge so rapidly that the expander lens cannot capture all this beam, making a lot of energy not applicable in laser marking. A possible solution is to put the expander lens as close to the ball lens as possible. In this way, an extremely short focal length expander lens might be required, which has limitations since that kind of lens might be difficult to manufacture. Based on the technology from CERN, inPhocal developed their own SBs [9] [10] [11], which solve this disadvantage. In inPhocal's laser system, a spherical mirror or a spherical lens is exploited to induce spherical aberration. In this case, the outgoing beam will not diverge fast as the ball lens system do. Then, by carefully choosing the shape, refractive index, and position of the expander lens, SBs have a longer focus, a larger depth of focus, and a high power ratio within the central spot can be generated. Figure 2.3 shows the comparison of beam profile, intensity, and energy ratio encircled

within a certain radius. It can be found that radius of the central spot (distance between the peak of the central spot and the peak of the first ring) of inPhocal's SB is around 0.1mm, and the energy ratio encircled within this radius is around 70%. On the contrary, the central spot of CERN's SB only has less than 10% energy ratio.



(a) Profile (left), intensity distribution and full width at half maximum (middle, full width at half maximum(FWHM) is denoted in red lines), and the energy ratio encircled within a given radius (right) on the cross-section plane of the SB of inPhocal. [11]



(b) Profile (left), intensity distribution and full width at half maximum (middle, full width at half maximum(FWHM) is denoted in red lines), and the energy ratio encircled within a given radius (right) on the cross-section plane of the SB of CERN. [11]

Figure 2.3: Beam profile and properties of SB generated by (a) inPhocal's system and (b) CERN's system.

2.2 Non-uniform rational B-spline (NURBS)

Non-**U**niform **R**ational **B**-**S**pline (NURBS) [12] is a common way to construct a curve or a surface in 3-dimensional space in computer-aided design software. In this section, a simple case to reconstruct a given curve will be discussed in order to give an insight into applying NURBS first, and the way to create NURBS surface will be introduced after that.

2.2.1 NURBS curve

A curve can be regarded as the continuous trajectory of a moving point, so we can only choose some sample points on a curve to represent it in the discrete system. That is to say, the value between two sample points should be determined in other ways. One of these ways is an interpolation. By choosing the proper weight of two adjacent sample points, the value in any position of the curve can be approximated by the adjacent sample points and their weights. In fact, NURBS is also an interpolation method, whose interpolated value consists of a number of polynomials. These polynomials is defined by basis function $N_{i,p}(u)$:

$$N_{i,0}(u) = \begin{cases} 1, & u_i \le u \le u_{i+1} \\ 0, & otherwise \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.1}$$

where u_i is given by knot vector $\mathbf{U}$, $i = 0, ..., m$ with $m+1$ is the number of knots in $\mathbf{U}$, and p is the order of polynomials used in approximating the curve. In general, a curve will be approximated by second-order $p = 2$ polynomials. Figure 2.4 shows the plot of basis function with order $p = 0, 1, 2$ and knot vector $\mathbf{U} = \{0, 1, 2, 3, 4, 5, ...\}$, it can be seen that a smooth curve can be approximated with at least second order($p = 2$) polynomials.

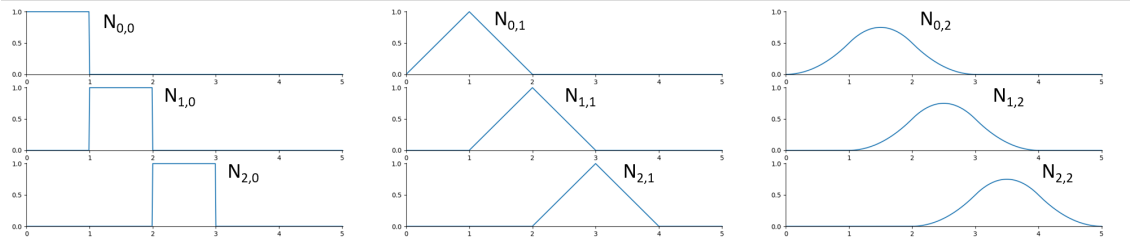


Figure 2.4: Visualization of Basis function with order $p = 0, 1, 2$ and knot vector $\mathbf{U} = \{0, 1, 2, 3, 4, 5, ...\}$.

After having the basis function $N_{i,p}(u)$, a NURBS curve can be represented by Equation 2.2. The value at a specific position on the target curve $C(u)$ can be obtained by summing up the product of all basis function $N_{i,p}(u)$ with knot vector $\mathbf{U} = \{u_0, ..., u_i, ..., u_{m-1}\}$ and the control points $\mathbf{P}_i$ as well as the weight w_i and normalized by dividing the summation of product $N_{i,p}(u)$ and w_i. Figure 2.5 shows a sinusoidal curve constructed by NURBS with given control points and knot vector.

$$C(u) = \frac{\sum_{i=0}^{m} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{m} N_{i,p}(u) w_i} \tag{2.2}$$

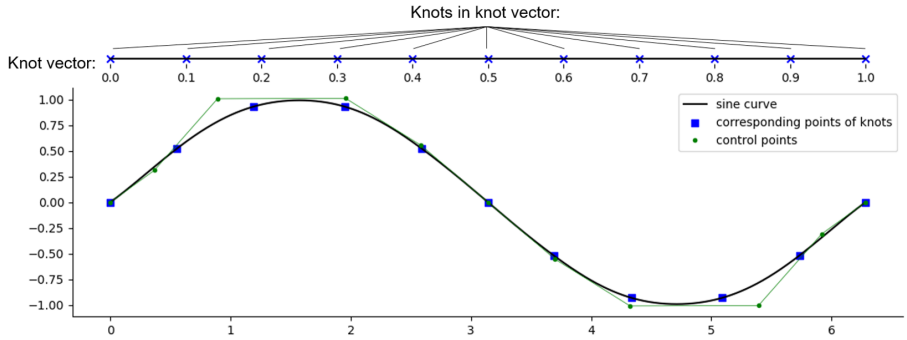


Figure 2.5: Sinusoidal curve (black curve) constructed by NURBS with control points (green dots), knot vector $\mathbf{U} = \{0.0, 0.1, 0.2, ..., 1.0\}$ (blue cross) and corresponding point of each knot (blue square).

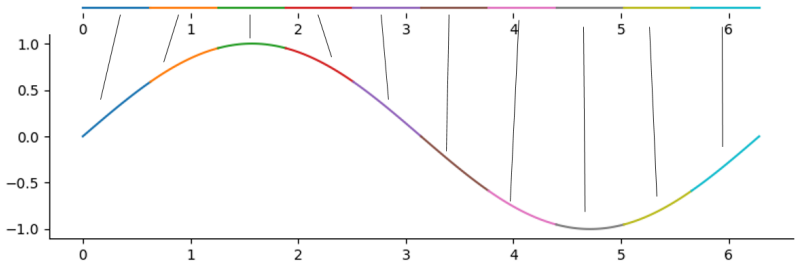
It is not always that the control points and knot vector are given. More commonly, the curve should be reconstructed by using a set of sample points. In this situation, the knot vector $\mathbf{U}$ can be obtained through this approach:

$$\mathbf{U} = \{0, 0, ..., 0, u_{p+1}, ..., u_m, 1, 1, ..., 1\}$$
$$u_{j+p} = \frac{1}{p} \sum_{i=j}^{j+p-1} \bar{u}_i, \quad \text{for} \quad j = 1, ..., m-p \tag{2.3}$$

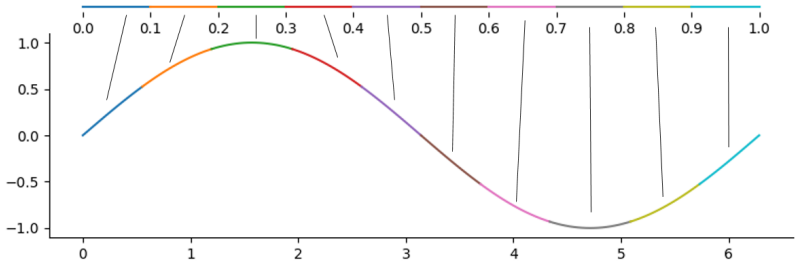
where the $\bar{u}_i$ is obtained by sample points $\mathbf{Q} = \{Q_0, ..., Q_j, ...Q_m\}$ through chord length parameterization:

$$\bar{u}_0 = 0, \quad \bar{u}_m = 1$$
$$\bar{u}_i = \bar{u}_{i-1} + \frac{|Q_i - Q_{i-1}|}{\sum_{j=0}^{m} |Q_j - Q_{j-1}|}, \quad \text{for} \quad i = 1, ..., m \tag{2.4}$$

Since the interval of the knot vector is determined by the chord length between sample points of the target curve, if line segments with the same length in the knot vector domain are mapped onto the target curve, they will still retain the same length. This is the reason why NUBRS can be used in eliminating distortion of pattern mapping. After having the knot vector, the sample points can be treated as the points $C(u)$ and the u_i calculated from sample points are the corresponding u for the input to calculate basis function $N_{i,p}(u)$. Then, by giving value to weights w_i, the control points $\mathbf{P}_i$ can be determined. Figure 2.6 shows mapping ten straight line segments of the same length onto a sinusoidal curve by projecting them orthogonally and through NURBS. In Figure 2.6a, the line segments projected onto the parts with larger curvature of the sinusoidal curve will be elongated more than those projected onto smaller curvature parts. That is to say, each line segment is in different length after mapping. In Figure 2.6b, the line segments mapped through NURBS will still be the same length because of the chord length parameterization used in creating a knot vector.



(a) Mapping line segments with the same length on a sinusoidal curve through orthogonal projection



(b) Mapping line segments with the same length on a sinusoidal curve through NURBS

Figure 2.6: Pattern mapping through orthogonal projection to show the distortion in (a) and through NURBS with $p = 2$ and $w_i = 1$ for all i to eliminate this distortion in (b).

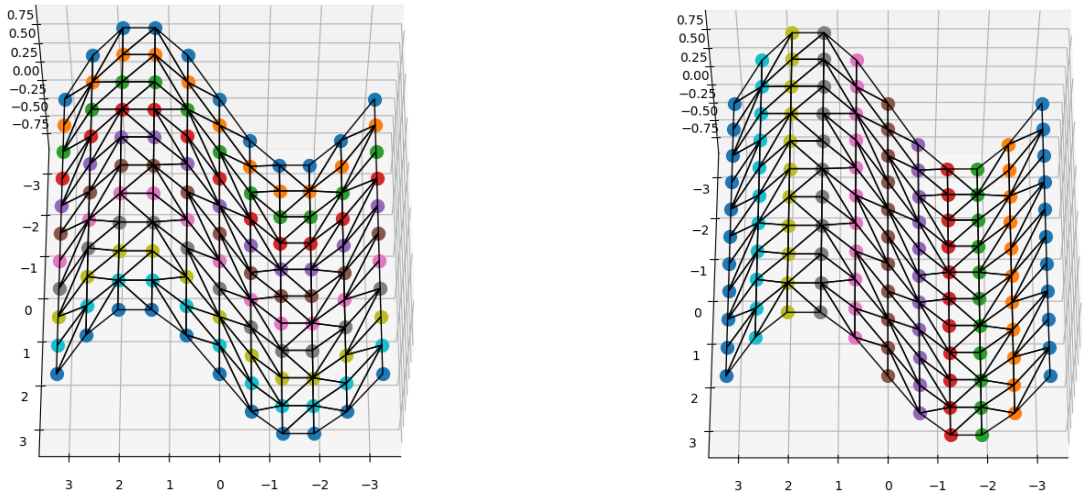
2.2.2 NURBS surface

The process of constructing a NURBS surface is very similar to that of NURBS curve. The difference between them is the span created by the knot vector. The value on a curve can be obtained by only considering one parameter, but there are two parameters that should be taken into consideration to determine the value on a surface. Hence, another knot vector should be added to expand the span from a 1-dimensional line to a 2-dimensional plane. For the same reason, two basis functions are required to determine the value of certain point $S(u, v)$ on the surface:

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \tag{2.5}$$

To calculate the knot vectors $\mathbf{U}$ and $\mathbf{V}$, the sample points of the target surface $\mathbf{Q} = \{Q_{i,j}\}$ for $i = 0, ..., m$ and $j = 0, ..., n$ should be sliced in two different directions along i and j respectively (Figure 2.7). Then, by exploiting Equation 2.3 and Equation 2.4, knot vectors $\mathbf{U}$ and $\mathbf{V}$ can be obtained. With the same process as reconstructing a NURBS curve, basis functions $N_{i,p}(u)$ and $N_{j,q}(v)$ and by giving value to weights $w_{i,j}$, the control points $\mathbf{P}_{i,j}$ can be determined. Usually, third-order polynomials are used in constructing a NURBS surface. Figure 2.8 shows a sinusoidal surface reconstructed through NURBS with 11×11 sample points in i and j direction, knot vectors $\mathbf{U} = \{0, 0, 0, 0, 0.2044, 0.2956, 0.3927, 0.5, 0.6072, 0.7044, 0.7956, 1, 1, 1, 1\}$, $\mathbf{V} = \{0, 0, 0, 0, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 1, 1, 1, 1\}$, order $p, q = 3$ and $w_{i,j} = 1$ for all i, j.

It should be noted that, since NURBS is an interpolation method, the number of sample points will largely affect the reconstructed surface. If the sample points fail to include some features of the original surface, these features will not be shown in the reconstructed surface. Figure 2.9 shows sinusoidal surfaces reconstructed through NURBS with a different number of sample points. The sinusoidal surface in Figure 2.9a is reconstructed with 9×9 sample points and the sinusoidal surface in Figure 2.9b is reconstructed with 4×4 sample points. It is clear that the two peaks of the sinusoidal surface in Figure 2.9b are flatter than those of the sinusoidal surface in Figure 2.9a. It is caused by an insufficient number of sample points. That is to say, the sampling rate is too low so some peak features of a sinusoidal surface might not be sampled and the sinusoidal surface will not be approximated by NURBS properly.



(a) Slices in along i direction

(b) Slices in along j direction

Figure 2.7: Slicing sample points of a sinusoidal surface along (a) i direction and (b) j direction

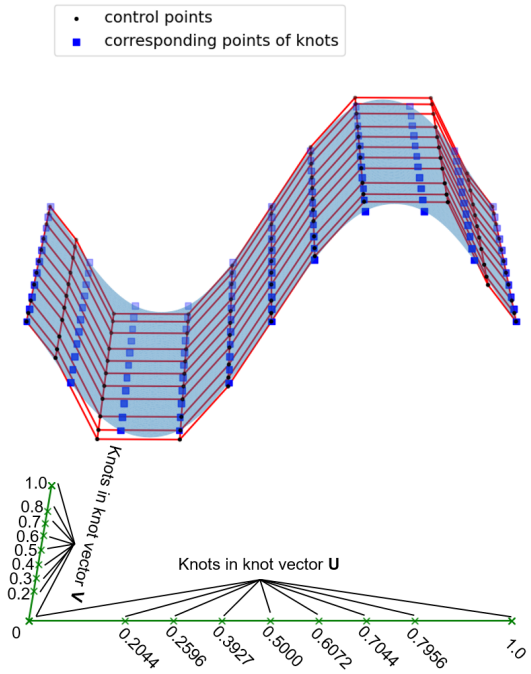
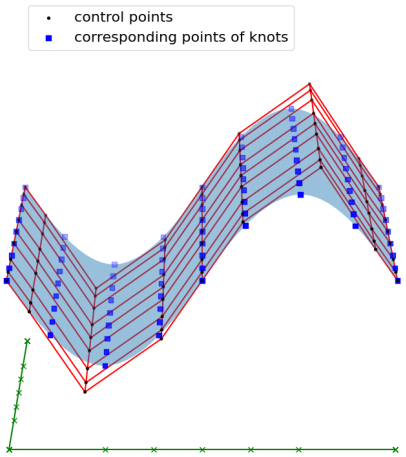
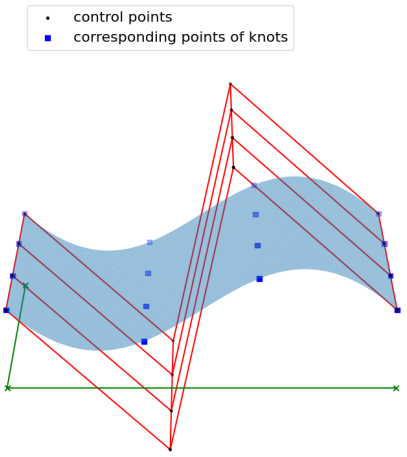


Figure 2.8: Sinusoidal surface reconstructed through NURBS with control points (black dots), knot vectors **U** and **V** (green cross), corresponding point of knots (blue square), order $p, q = 3$ and $w_{i,j} = 1$ for all i, j.



(a) Sinusoidal surface reconstructed through NURBS with 9×9 sample points



(b) Sinusoidalsurface reconstructed through NURBS with 4×4 sample points

Figure 2.9: Effects on number of sample points

2.3 Line integral on NURBS

A mapped pattern without distortion means the distance between every pair of selected points of the 2-dimensional pattern will remain the same after being mapped onto a 3-dimensional surface. Thus, a method to calculate the distance between two points on a NURBS surface is needed. In this section, the first fundamental form, a way to calculate the length of curves in space, will be

introduced. Then, how to apply the first fundamental form of NURBS will be elaborated by first applying to the simple case: NURBS curve again, then, applying to the NURBS surface after that. Since it is extremely complicated to use the Newton-Leibniz formula in calculating the integral of NURBS, a numerical integration method: Simpson's rule will be adopted to calculate curve length.

2.3.1 First fundamental form

Unlike the distance between two points in the points can be easily obtained by calculating the length of the straight line segment connected by these two points, the distance between two points lies on a surface should be calculated by considering the curve connected by these two points on the surface. Imagining there is a bug crawling on a surface from one point to another, then the path of this crawly bug is the curve $\boldsymbol{\gamma}$ and the distance between these two points is the length of this path that can be calculated by Equation 2.6 [13].

$$\int \parallel \dot{\boldsymbol{\gamma}}(t) \parallel dt \tag{2.6}$$

Suppose the surface is represented by $\boldsymbol{\sigma}(u, v)$. Since curve $\boldsymbol{\gamma}$ lies on this surface, it can be written in $\boldsymbol{\gamma}(t) = \boldsymbol{\sigma}(u(t), v(t))$ and the differential form of $\boldsymbol{\gamma}$ can be written in:

$$\dot{\boldsymbol{\gamma}}(t) = \frac{\partial \boldsymbol{\sigma}}{\partial u}\frac{du}{dt} + \frac{\partial \boldsymbol{\sigma}}{\partial v}\frac{dv}{dt} \tag{2.7}$$

Let $\frac{\partial \boldsymbol{\sigma}}{\partial u} = \boldsymbol{\sigma}_u$, $\frac{\partial \boldsymbol{\sigma}}{\partial v} = \boldsymbol{\sigma}_v$, then the dot product of $\dot{\boldsymbol{\gamma}}$ and $\dot{\boldsymbol{\gamma}}$ can be writen in :

$$\parallel \dot{\boldsymbol{\gamma}}(t) \parallel = \dot{\boldsymbol{\gamma}} \cdot \dot{\boldsymbol{\gamma}} = \|\boldsymbol{\sigma}_u\|^2 + 2\boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_v (\frac{du}{dt})^2 (\frac{dv}{dt})^2 + \|\boldsymbol{\sigma}_u\|^2 \tag{2.8}$$

Let $\|\boldsymbol{\sigma}_u\|^2 = E$, $\boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_v = F$, $\|\boldsymbol{\sigma}_u\|^2 = G$, the first fundamental form is given in:

$$\text{first fundamental form} = Edu^2 + 2Fdudv + Gdv^2 \tag{2.9}$$

and the length of curve $\boldsymbol{\gamma}$ in Equation 2.6 can be rewritten in :

$$\int \parallel \dot{\boldsymbol{\gamma}}(t) \parallel dt = \int (E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2)^{\frac{1}{2}} dt \tag{2.10}$$

where $\dot{u} = \frac{du}{dt}$ and $\dot{v} = \frac{dv}{dt}$. With the first fundamental form of the surface, the length of curves lying on it can be calculated.

2.3.2 Line integral of NURBS curve

To apply the first fundamental form to a NURBS curve, the derivative of the NURBS curve [14] should be pointed out first:

$$\begin{aligned}
\frac{dC(u)}{du} &= C(u)' \\
&= \left(\frac{\sum_{i=0}^{m} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{m} N_{i,p}(u) w_i} \right)' \\
&= \frac{\sum_{i=0}^{m} N_{i,p}(u)' w_i \mathbf{P}_i \sum_{i=0}^{m} N_{i,p}(u) w_i}{(\sum_{i=0}^{m} N_{i,p}(u) w_i)^2} \\
&- \frac{\sum_{i=0}^{m} N_{i,p}(u) w_i \mathbf{P}_i \sum_{i=0}^{m} N_{i,p}(u)' w_i}{(\sum_{i=0}^{m} N_{i,p}(u) w_i)^2}
\end{aligned} \tag{2.11}$$

where the term $N_{i,p}(u)'$ equals to:

$$N_{i,p}(u)' = \frac{p}{u_{i+p} - u_i} N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.12}$$

In this case, the crawly bug moves on the NURBS curve $C(u)$ with variable u, so the path length from the start point a to the endpoint b of the bug can be written in:

$$\int_a^b \parallel C(u)' \parallel du = \int_a^b (Eu'^2 + 2Fu'v' + Gv'^2)^{\frac{1}{2}} du \tag{2.13}$$

where $E = \|C_u(u)\|^2 = \|C(u)'\|^2$, $F = C_u(u) \cdot C_v(u) = 0$, $G = \|C_v(u)\|^2 = 0$, $u' = \frac{du}{du} = 1$, $v' = \frac{dv}{du} = 0$, and $a, b \in [0, 1]$. In fact, this form can be regarded as the integral of the tangent of NURBS curve $C(u)'$ times the infinity small step du within the interval $[a, b]$ and can be rewritten in:

$$\int_a^b C(u)' \, du \tag{2.14}$$

2.3.3 Line integral of NURBS surface

For the first fundamental form of the NURBS surface, the partial derivative of two different directions u and v should be calculated respectively [14]:

$$\begin{aligned}
\frac{\partial S(u,v)}{\partial u} &= S_u(u,v) \\
&= \left(\frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \right)_u \\
&= \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)'N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j} \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}}{(\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j})^2} \\
&- \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j} \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)'N_{j,q}(v)w_{i,j}}{(\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j})^2}
\end{aligned} \tag{2.15}$$

$$\begin{aligned}
\frac{\partial S(u,v)}{\partial v} &= S_v(u,v) \\
&= \left(\frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \right)_v \\
&= \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)'w_{i,j}\mathbf{P}_{i,j} \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}}{(\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j})^2} \\
&- \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j} \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)'w_{i,j}}{(\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j})^2}
\end{aligned} \tag{2.16}$$

then the length of the curve lying on this NURBS surface can be written in:

$$\int_a^b \parallel \dot{S}(u,v) \parallel dt = \int_a^b (E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2)^{\frac{1}{2}} dt \tag{2.17}$$

where $E = \|S_u(u,v)\|^2$, $F = S_u(u,v) \cdot S_v(u,v)$, $G = \|S_v(u,v)\|^2$, $\dot{u} = \frac{du}{dt}$, and $\dot{v} = \frac{dv}{dt}$. Here, $S_u(u,v)$ and $S_v(u,v)$ is given in Equation 2.15 and Equation 2.16. $N_{i,p}(u)'$, $N_{j,q}(v)'$ can be calculated by using Equation 2.12. To find $\dot{u}$ and $\dot{v}$, a line segment $s(u(t), v(t))$ in knot vectors U, V with start point (u_{start}, v_{start}) and endpoint (u_{end}, v_{end}) should be taken into account. With the changing of variable t from 0 to 1, u and v are also moved from the start point to the endpoint. That is to say, given a specific t and give it a small change dt, the $u(t)$ and $v(t)$ in this specific point will also be changed to $u(t + dt)$ and $v(t + dt)$. Then $\dot{u}$ and $\dot{v}$ is calculated by $\frac{u(t+dt)-u(t)}{t+dt-t} = \frac{du}{dt}$ and $\frac{v(t+dt)-v(t)}{t+dt-t} = \frac{dv}{dt}$. And Equation 2.17 can be rewritten in:

$$\int_0^1 \parallel \dot{S}(u(t), v(t)) \parallel dt = \int_0^1 (E\dot{u}^2 + 2F\dot{u}\dot{v} + G\dot{v}^2)^{\frac{1}{2}} dt \tag{2.18}$$

Figure 2.10 shows an example that a line segment mapped onto a sinusoidal surface with start point $(0.25, 0.25)$ and endpoint $(0.75, 0.75)$ in U, V knot vectors span. Since it is a straight-line segment, the changing rate $\dot{u}$ and $\dot{v}$ of u and v with respect to t is a constant and can be obtained by the variation between the start point and endpoint of u, v, and t: $\frac{u_{end}-u_{start}}{t_{end}-t_{start}} = \frac{0.75-0.25}{1-0} = 0.5$ and $\frac{v_{end}-v_{start}}{t_{end}-t_{start}} = \frac{0.75-0.25}{1-0} = 0.5$.

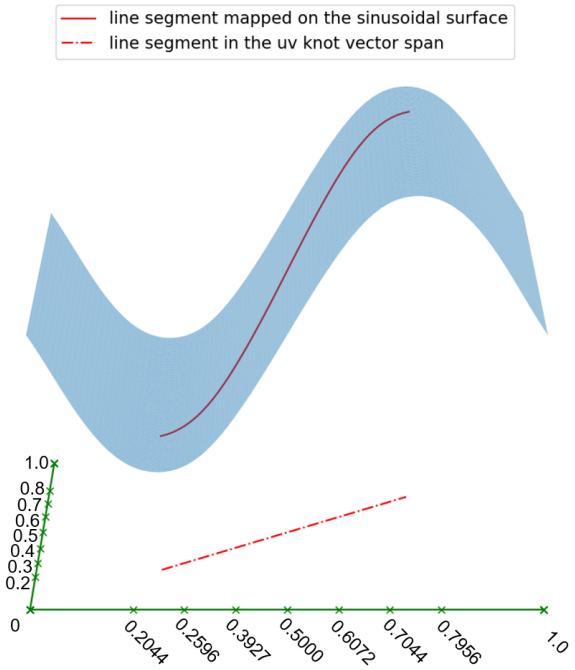


Figure 2.10: Line segment $s(u(t), v(t))$ mapped onto a sinusoidal surface with start point $(0.25, 0.25)$ and endpoint $(0.75, 0.75)$ in U, V knot vectors span.

2.3.4 Numerical integration: Composite Simpson's rule

When referring to the line length integral of NURBS curve and surface, it is obvious that the denominator is very complicated so the Newton-Leibniz formula can hardly be used in calculating this integration. Hence, the assistance of a numerical integration method is required. Simpson's rules are several approximations for definite integrals being used in numerical integration[15]. One of these rules is called Composite Simpson's 1/3 rule, or just called Simpson's rule, given in Equation 2.19 and will be used in this research to calculate curve length on NURBS surfaces.

$$\int_a^b f(x)\,dx \approx \frac{b-a}{3n}\left[f(x_0) + 4\sum_{i=1}^{n/2} f(x_{2i-1}) + 2\sum_{i=1}^{n/2-1} f(x_{2i}) + f(x_n)\right] \tag{2.19}$$

The error between the actual interaction value and the value approximated by Composite Simpson's rule is $-\frac{1}{180}\frac{(b-a)^4}{n^4}(b-a)f^{(4)}(\xi)$, where ξ is some number between a and b. It is clear that the approximation will be more accurate when n is larger, but it will also be more demanded on computation power.

To apply Composite Simpson's rule to calculate curve length on NURBS surface, the square root of the first fundamental form should be calculated first and this will be served as the function $f(t)$ in Composite Simpson's rule. Then the definite integral will be taken from $a = 0$ to $b = 1$.

Chapter 3

Shape correction through NURBS

In this chapter, an algorithm to reconstruct a freeform surface with given sample points through NURBS will be introduced. After having the reconstructed NURBS surface, the algorithm to map patterns onto this NURBS surface without distortion will be developed.

3.1 NURBS surface reconstruction

To start reconstructing a NURBS surface, a set of points on the target surface should be sampled first. Here a human face (Figure 3.1) will serve as the target surface.

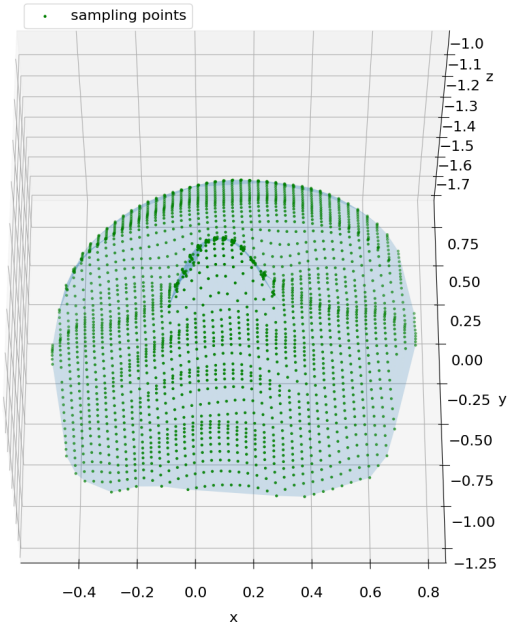


Figure 3.1: 3D human face surface model with sampling points.

Refer to subsection 2.2.2, the number of sampling points in each slice should be the same in both u and v directions, but it is not always the case. That is to say, a data selection process should be conducted on the input surface model. Wenchao Wu et al. [16] proposed a method for selecting sampling points for reconstructing a NURBS surface. To begin with, the surface model's position

and posture should be rearranged by moving the center of mass of the surface model to the origin $(0, 0)$ and aligning the three principal components of the surface model to the x, y, and z axis. By following Algorithm A.1, the rearranged surface can be produced and shown in Figure 3.2.

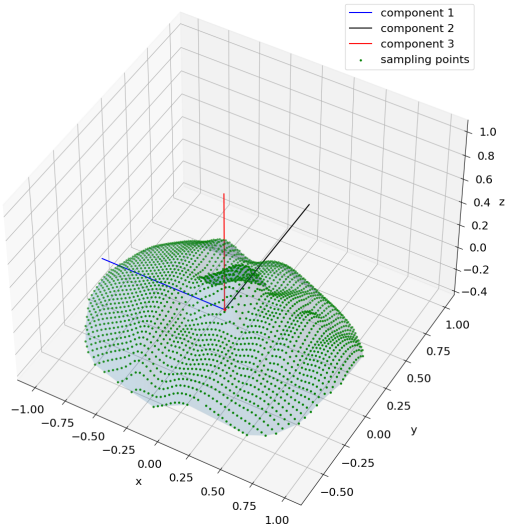
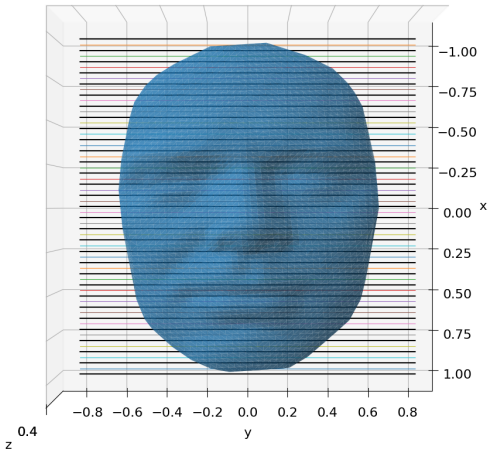
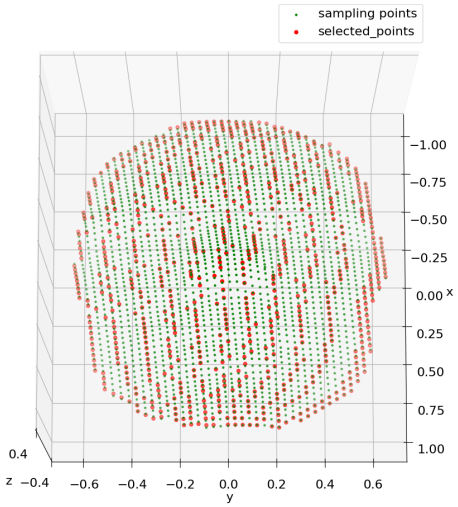


Figure 3.2: Rearranged 3D human face surface model.

Then, the surface model will be sliced into several slices, and points within the certain slice will be projected onto the central plane of this slice (Figure 3.3a). To select the same number of data points in each slice, the maximum number of data points that can be chosen is the number of points of the slice with the least points. The criterion to choose the points in each slice is always choosing the start point and the end point of this slice to make sure the edge will not be lost and the rest of the points should be chosen evenly. This process can be done by following Algorithm A.2 and the result is shown in Figure 3.3b.



(a) Slices(colored lines) and projection plane(black lines) of 3D human face surface model.



(b) Points selection of 3D human face surface model for NURBS surface reconstruction.

Figure 3.3: (a)Slicing 3D human surface model, (b)data selection of 3D human surface model for NURBS surface reconstruction.

After having the selected points of the surface model, the NURBS surface can be constructed by finding the chord length parameterization $\bar{u}$ and $\bar{v}$ in each slice of both u, v directions through Equation 2.4 and knot vectors U and V through Equation 2.3 first. Then, using these selected points as the sampling points of the NURBS surface $S(u,v)$ and $\bar{u}$, $\bar{v}$ as the input of basis functions $N_{i,p}$, $N_{j,q}$, the control points $\mathbf{P}_{i,j}$ in Equation 2.5 can be obtained by: $\mathbf{P} = \mathbf{S}\mathbf{R}^{-1}$, where $\mathbf{R}$ consists of $R_{i,j}$, which is called rational function and given in: $\frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}}$ for $i = 0,...,m-1$ and $j = 0,...n-1$, where m is the number of selected points in slices of u direction and n is the number of selected points in slices of v direction. With the control points $\mathbf{P}$ and knot vectors U, V, a NURBS surface is constructed, and the corresponding value $S(u,v)$ of each points (u,v) in the knots vectors span can be found by Algorithm 3.1 and shown in Figure 3.4.

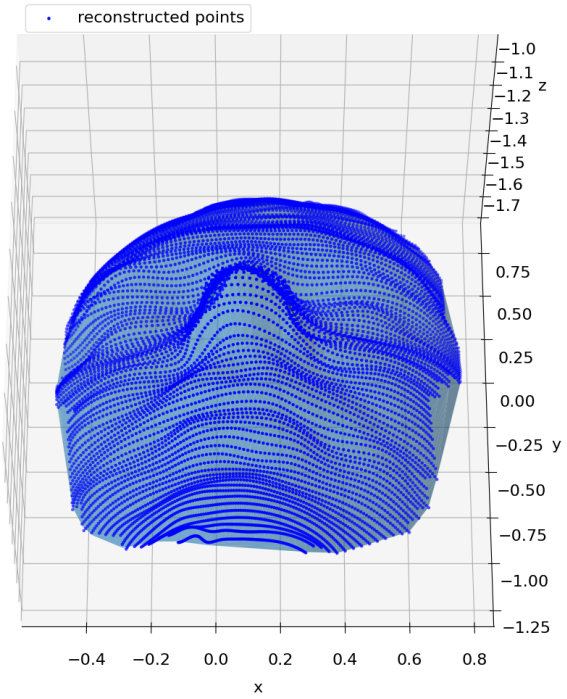


Figure 3.4: Reconstructed points $S(u,v)$ for $u = 0, 0.01, ..., 0.99, 1$ and $v = 0, 0.01, ..., 0.99, 1$ of 3D human face surface model through NURBS.

Algorithm 3.1 NURBS Surface Reconstruction

Input: u, v, $\mathbf{Q}$, n, p, q
Output: $S(u,v)$

1: $\mathbf{Q}_{rearranged} = \mathbf{DataRearrangement}(\mathbf{Q})$ ▷ Algorithm A.1
2: $\mathbf{Q}_{selected}$, $m = \mathbf{DataSelection}(\mathbf{Q}_{rearranged}, n)$ ▷ Algorithm A.2
3: $\bar{u} = \mathbf{ChordLengthParameterization}(\mathbf{Q}_{selected}.\mathbf{reshape}((n+1),(m+1),3), m)$ ▷ Algorithm A.3
4: $\mathbf{U} = \mathbf{GetKnotVector}(\bar{u}, m, p)$ ▷ Algorithm A.4
5: $ctrlpts\_temp = \mathbf{GetControlPoints}(\mathbf{Q}_{selected}.\mathbf{reshape}((n+1),(m+1),3), \bar{u}, \mathbf{U}, m, p)$ ▷ Algorithm A.5
6: $\bar{v} = \mathbf{ChordLengthParameterization}(\mathbf{Q}_{selected}.\mathbf{reshape}((n+1),(m+1),3).\mathbf{transpose}(1,0,2), n)$
7: $\mathbf{V} = \mathbf{GetKnotVector}(\bar{v}, n, q)$
8: $ctrlpts = \mathbf{GetControlPoints}(ctrlpts\_temp.\mathbf{reshape}((n+1),(m+1),3).\mathbf{transpose}(1,0,2), \bar{v}, \mathbf{V}, n, q)$
9: $U_k = \mathbf{mean}(\mathbf{U}, axis = 0)$
10: $V_k = \mathbf{mean}(\mathbf{V}, axis = 0)$

11: $N_{i,p}$ = **BasisFunction**(u, U_k, $len(U_k)$, p) ▷ Algorithm A.6
12: $N_{j,q}$ = **BasisFunction**(v, V_k, $len(V_k)$, q)
13: R = **RationalFunction**($N_{i,p}$, $N_{j,q}$, m, n, $w = 1$) ▷ Algorithm A.7
14: $S(u,v)$ = R.**dot**($ctrlpts$)

There is one thing that should be noted, in order to get the selected points for applying NURBS surface, the target surface is divided into several slices, and each slice might have different length based on the geometry of the target surface. That is to say, there are also several different knot vectors will be generated during the process of creating the NURBS surface. In the research of Wenchao Wu et al. [16], they take the mean value of corresponding knots in each knot vector as the final knot vector for the input of the NURBS surface reconstruction algorithm. However, this knot vector will have a bad effect on pattern mapping since it can't represent the distance between each knot at all. In this research, a new way to determine the knot vector for the input of the NURBS surface reconstruction algorithm is developed. By finding the point (u,v) that will be reconstructed is located in which two knot vectors, and using the linear interpolation to calculate the final knot vector, this knot vector can still represent the distance between each knot. The Algorithm 3.1 can be modified to Algorithm 3.2.

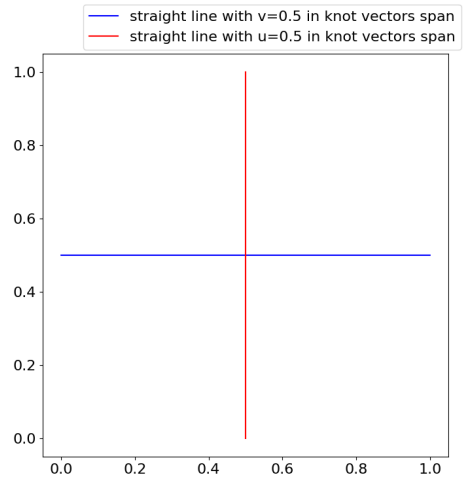
Algorithm 3.2 New NURBS Surface Reconstruction

Input: u, v, $\mathbf{Q}$, n, p, q
Output: $S(u,v)$
1: $\mathbf{Q}_{rearranged}$ = **DataRearrangement**($\mathbf{Q}$) ▷ Algorithm A.1
2: $\mathbf{Q}_{selected}$, m = **DataSelection**($\mathbf{Q}_{rearranged}$, n) ▷ Algorithm A.2
3: $\bar{u}$ = **ChordLengthParameterization**($\mathbf{Q}_{selected}$.**reshape**($(n+1),(m+1),3$), m) ▷ Algorithm A.3
4: $\mathbf{U}$ = **GetKnotVector**($\bar{u}$, m, p) ▷ Algorithm A.4
5: $ctrlpts\_temp$ = **GetControlPoints**($\mathbf{Q}_{selected}$.**reshape**($(n+1),(m+1),3$), $\bar{u}$, $\mathbf{U}$, m, p) ▷ Algorithm A.5
6: $\bar{v}$ = **ChordLengthParameterization**($\mathbf{Q}_{selected}$.**reshape**($(n+1),(m+1),3$).**transpose**($1,0,2$), n)
7: $\mathbf{V}$ = **GetKnotVector**($\bar{v}$, n, q)
8: $ctrlpts$ = **GetControlPoints**($ctrlpts\_temp$.**reshape**($(n+1),(m+1),3$).**transpose**($1,0,2$), $\bar{v}$, $\mathbf{V}$, n, q)
9: U_k, V_k = **NewKnotVector**(u, v, $\mathbf{U}$, $\mathbf{V}$, m, n) ▷ Algorithm A.8
10: $N_{i,p}$ = **BasisFunction**(u, U_k, $len(U_k)$, p) ▷ Algorithm A.6
11: $N_{j,q}$ = **BasisFunction**(v, V_k, $len(V_k)$, q)
12: R = **RationalFunction**($N_{i,p}$, $N_{j,q}$, m, n, $w = 1$) ▷ Algorithm A.7
13: $S(u,v)$ = R.**dot**($ctrlpts$)

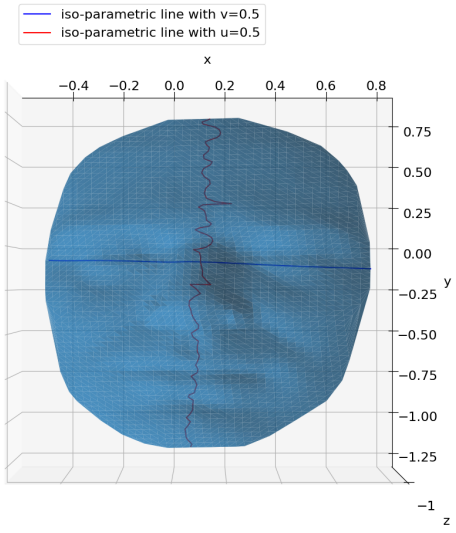
3.2 Pattern mapping

By putting the 2-dimensional pattern into the knot vectors span of the reconstructed NURBS surface, this 2-dimensional pattern could be mapped onto the 3-dimensional surface. However, as can be seen in Figure 3.3b, the selected points are not aligned straightly, and since the control points, which are the main factors to control the positions of reconstructed points, are derived from these selected points, the reconstructed points also will not be aligned straightly. In short, when a straight line in the knot vectors span is mapped onto the NURBS surface, it will no longer be a straight line. Figure 3.5 shows two iso-parametric lines (parametric line with one parameter is fixed) $u = 0.5$ and $v = 0.5$ that are mapped onto the 3D human face surface model. It is clear that the original surface is sliced along the x axis, so the selected points are aligned in the y direction but not aligned in the x axis. Due to this, the iso-parametric line in the y direction is still aligned straightly but the iso-parametric line in the x direction is distorted. If this NURBS surface is only used in representing the original surface, then this distorted iso-parametric line is not a problem since all the reconstructed points are still on the surface. Nevertheless, when it comes to mapping

patterns onto the target surface, the distorted straight line after mapping is not acceptable, and zero padding is applied to solve this problem. This will be elaborated in subsection 3.2.1.



(a) Straight line with parameter $u = 0.5$ and $v = 0.5$ in the knot vectors span.



(b) Iso-parametric line $u = 0.5$ and $v = 0.5$ mapped onto the 3D human face surface model.

Figure 3.5: Mapped Iso-parametric line $u = 0.5$ and $v = 0.5$ in (a)the knot vectors span and (b)onto the 3D human face surface model.

3.2.1 Zero padding

The reason for the distorted iso-parametric line is the different number of sampling points in each slice. To solve this problem, points with zero value that would not affect the geometry can be added to make the number of sampling points in each slice consistent. For convenience, an area with the same size as the scanning area of the laser system will be defined. In this area, the z-value of all points that don't belong to the target 3-dimensional surface model will be set to zero. For the data selection process, the sampling step along u and v directions should be defined. The smaller sampling step means more points will be selected and leading to higher precision. The origin point $(0, 0)$ will be the starting point of this process. By moving one sampling step at a time, choosing the nearest point of the surface model with zero padding as the sampling point at this step, the data selection process can be done by stepping through the whole defined scanning area. Following the Algorithm 3.3, the result of selected data points with zero padding is shown in Figure 3.6.

Algorithm 3.3 Data Selec Zero Padding

Input: $\mathbf{Q}_{rearranged}$, $scanning\_area$, $step\_size\_m$, $step\_size\_n$
Output: $\mathbf{Q}_{selected}$

$\mathbf{Q}_{selected} = []$
$m = scanning\_area[0]/step\_size\_m$; $n = scanning\_area[1]/step\_size\_n$
for $j = 0 \rightarrow n + 1$ **do**
    for $i = 0 \rightarrow m + 1$ **do**
        $sampling\_position = ([i * step\_size\_m, j * step\_size\_n])$
        Try
            $var = \mathbf{Q}_{rearranged}[\mathbf{abs}(\mathbf{Q}_{rearranged}[:, 0] - sampling\_position[0] < step\_size\_m)]$
            $var = var[\mathbf{abs}(var[:, 1] - sampling\_position[1] < step\_size\_n)]$
            $dist = \mathbf{sqrt}(\mathbf{sum}(var[:, : 2] - sampling\_position, axis = 1))$

10: $\mathbf{Q}_{selected}$.**append**($[i * step\_size\_m, j * step\_size\_n, var[$**argmin**$(dist)][2]]$)
11: **Except**
12: $\mathbf{Q}_{selected}$.**append**($[i * step\_size\_m, j * step\_size\_n, 0$)
13: **end for**
14: **end for**

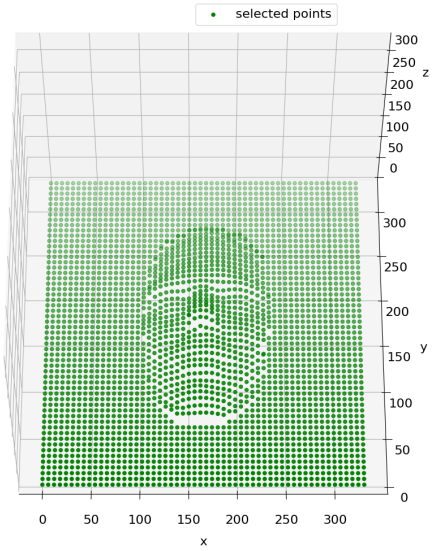
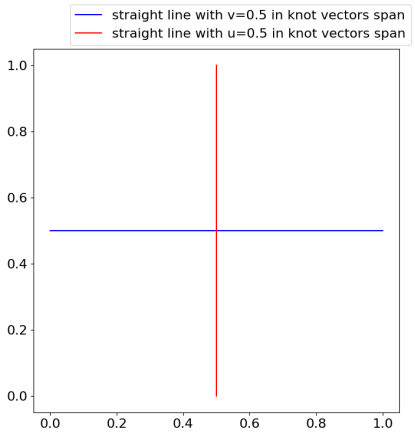
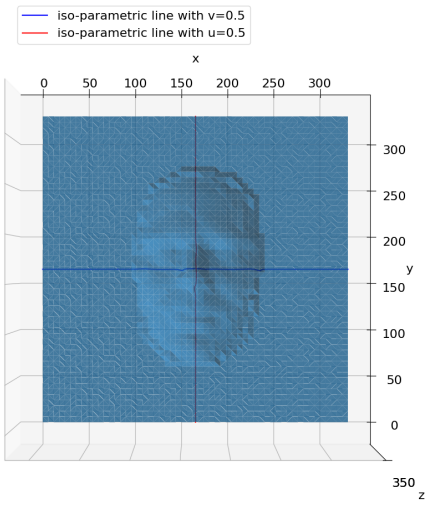


Figure 3.6: Zero padding to 3D human face surface model with scanning area 334×334.

By applying Algorithm 3.2, this human face NURBS surface can be reconstructed, and since the selected points in each slice of both u and v directions are aligned straightly, the iso-parametric line will not be distorted in this case Figure 3.7.



(a) Straight line with parameter u = 0.5 and v = 0.5 in the knot vectors span after zero-padding.



(b) Iso-parametric line u = 0.5 and v = 0.5 mapped onto the 3D human face surface modelafter zero-padding.

Figure 3.7: Mapped Iso-parametric line u = 0.5 and v = 0.5 in (a)the knot vectors span and (b)onto the 3D human face surface model.

3.2.2 Mapping domain

After applying zero padding, the 2-dimensional pattern can be mapped onto the 3-dimensional without distortion by directly putting it into the 2-dimensional knots span. Nonetheless, one should keep in mind that the knot vectors originate in chord length parameterization, so if the 2-dimensional pattern is required to be mapped onto the 3-dimensional surface with a certain size, the size of the 2-dimensional pattern should be scaled to fit the area of the target 3-dimensional surface. There is no intuitive way to scale the 2-dimensional pattern because most 3-dimensional surfaces are with nonzero Gaussian curvatures somewhere, which makes them "doubly curved surfaces" [17].

To make sure the distance between two points on the 2-dimensional pattern keeps the same after mapping, some points in the knot vectors span will be chosen evenly, and the distance between these points will be calculated through line integral on NURBS to form a mapping domain. This mapping domain is the area extended $(m \times n)$ chosen points and can be created by following continuing steps. First, the knot vector span $\in [0,1]$ in U and V direction will be evenly divided into n and m parts, and the length of iso-parametric line with $u = 0.5$ will be calculated as l_v. Second, the position where the center of 2-dimensional pattern wants to be mapped should be defined, and the distance from the center of the scanning area to the center of 2-dimensional pattern along the v direction will be calculated. By examining the ratio of this distance and l_v, the v parameter of the pattern's center can be determined. Then, using the iso-parametric line $u = 0.5$ as the central axis of the mapping domain, values of the mapping domain along $n/2$ will be given. Still then, finding the nearest point v_{center} to the v parameter of the pattern's center in divided knot vector span V, the iso-parametric line with $v = v_{center}$ will be the other central axis of the mapping domain. Thus, values of the mapping domain along the number index of $v_{center} : m_{center} \in [0, m]$ will also be given. Finally, the rest values of the mapping domain $domain(m, n)$ will be given by calculating the distance between $domain(m, n)$ and $domain(m - j, n - i)$ when $m \neq m_{center}$ and $n \neq n/2$, where $i = 1$ if $n > n/2$ else $i = -1$ and $j = 1$ if $m > m_{center}$ else $j = -1$. By following Algorithm 3.4, the mapping area of the 3D human face surface model can be created and shown in Figure 3.8. With this mapping domain, the 2-dimensional pattern can be mapped onto the 3-dimensional surface in the desired size without scaling beforehand. The detail of the pattern mapping process will be introduced in the next subsection.

Algorithm 3.4 Mapping Domain

Input: $S(u,v)$, $patten\_center$, m, n, $\mathbf{U}$, $\mathbf{V}$, num, s
Output: $mapping\_domain$

$l_v = \mathbf{LineLength}(S(u,v), 0.5, 0.5, 0, 1, m, n, \mathbf{U}, \mathbf{V}, s)$ ▷ Algorithm A.10
$center = patten\_center[1]/l_v + 0.5$
$mapping\_domain = \mathbf{zeros}([num + 1, num + 1, 2])$
$intv = \mathbf{linspace}(0, 1, num + 1)$
$c = \mathbf{argmin}(\mathbf{abs}(intv - center))$
$h = \mathbf{int}(num/2)$
for $i = 1 \rightarrow h + 1$ **do**
    $u_{start} = h$; $u_{end} = h$; $v_{start} = h + i - 1$; $v_{end} = h + i$
    $mapping\_domain[:, h][h + i, 1] = mapping\_domain[:, h][h + i - 1, 1]$
      $+\mathbf{LineLength}(S(u,v), intv[h], intv[h], intv[h + i - 1], intv[h + i], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[:, h][h - i, 1] = mapping\_domain[:, h][h - i + 1, 1]$
      $-\mathbf{LineLength}(S(u,v), intv[h], intv[h], intv[h - i + 1], intv[h - i], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[c][h + i, 0] = mapping\_domain[c][h + i - 1, 0]$
      $+\mathbf{LineLength}(S(u,v), intv[h + i - 1], intv[h + i], intv[c], intv[c], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[c][h - i, 0] = mapping\_domain[c][h - i + 1, 0]$
      $-\mathbf{LineLength}(S(u,v), intv[h - i + 1], intv[h - i], intv[c], intv[c], m, n, \mathbf{U}, \mathbf{V}, s)$
end for

$mapping\_domain[c][h+1:,1] = mapping\_domain[c][h,1]$
$mapping\_domain[c][:h,1] = mapping\_domain[c][h,1]$
for $j = 1 \to c+1$ **do**
  for $k = 1 \to h+1$ **do**
    $mapping\_domain[c-j][h+k,0] = mapping\_domain[c-j+1][h+k-1,0]$
    $+\mathbf{LineLength}(S(u,v), intv[h+k-1], intv[h+k], intv[c-j], intv[c-j], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[c-j][h-k,0] = mapping\_domain[c-j+1][h-k+1,0]$
    $-\mathbf{LineLength}(S(u,v), intv[h-k+1], intv[h-k], intv[c-j], intv[c-j], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[c-j][h+k,1] = mapping\_domain[c-j+1][h+k-1,1]$
    $-\mathbf{LineLength}(S(u,v), intv[h+k], intv[h+k], intv[c-j+1], intv[c-j], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[c-j][h-k,1] = mapping\_domain[c-j+1][h-k+1,1]$
    $-\mathbf{LineLength}(S(u,v), intv[h-k], intv[h-k], intv[c-j+1], intv[c-j], m, n, \mathbf{U}, \mathbf{V}, s)$
  end for
end for
for $j = 1 \to c+1$ **do**
  for $k = 1 \to h+1$ **do**
    $mapping\_domain[c+j][h+k,0] = mapping\_domain[c+j-1][h+k-1,0]$
    $+\mathbf{LineLength}(S(u,v), intv[h+k-1], intv[h+k], intv[c+j], intv[c+j], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[c+j][h-k,0] = mapping\_domain[c+j-1][h-k+1,0]$
    $-\mathbf{LineLength}(S(u,v), intv[h-k+1], intv[h-k], intv[c+j], intv[c+j], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[c+j][h+k,1] = mapping\_domain[c+j-1][h+k-1,1]$
    $+\mathbf{LineLength}(S(u,v), intv[h+k], intv[h+k], intv[c+j-1], intv[c+j], m, n, \mathbf{U}, \mathbf{V}, s)$
    $mapping\_domain[c+j][h-k,1] = mapping\_domain[c+j-1][h-k+1,1]$
    $+\mathbf{LineLength}(S(u,v), intv[h-k], intv[h-k], intv[c+j-1], intv[c+j], m, n, \mathbf{U}, \mathbf{V}, s)$
  end for
end for

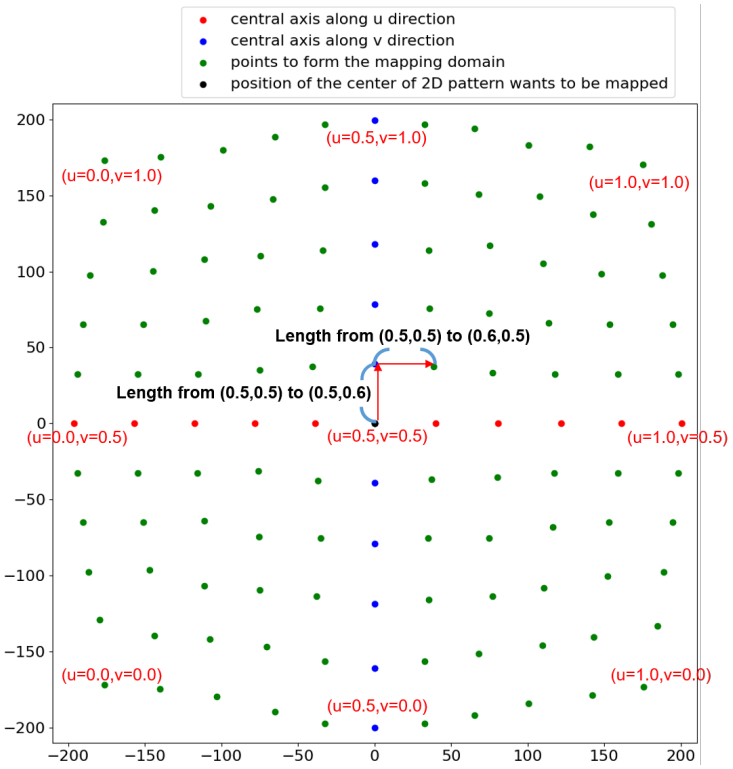


Figure 3.8: The mapping area of 3D human face surface model created by (11×11) points.

3.2.3 Pattern mapping

The distance between two points of the mapping domain is the distance between these two points on the 3-dimensional surface, so the mapping domain is just like the flattened surface. If a 2-dimensional pattern is going to be mapped onto a 3-dimensional surface, it can be easily finished by putting this 2-dimensional pattern into the mapping domain of this 3-dimensional surface and finding the corresponding (u, v) value for applying NURBS surface reconstruction algorithm. The way to find the (u, v) parameters can be done by locating the point that should be mapped into a quadrilateral formed by four points of the chosen points to create a mapping area first. Then, the (u, v) parameters can be found by applying the inverse bilinear interpolation to the four vertices of this quadrilateral. See Figure 3.9, the inverse bilinear interpolation is given in:

$$\begin{aligned} u &= u_{00} + (u_{01} - u_{00}) \cdot x + (u_{10} - u_{00}) \cdot y + (u_{00} - u_{01} + u_{11} - u_{10}) \cdot x \cdot y \\ v &= v_{00} + (v_{01} - v_{00}) \cdot x + (v_{10} - v_{00}) \cdot y + (v_{00} - v_{01} + v_{11} - v_{10}) \cdot x \cdot y \end{aligned} \tag{3.1}$$

where

$$\begin{aligned} x &= \frac{H_i - F_i \cdot y}{E_i - G_i \cdot y} \\ y &= \frac{-k_1 \pm \sqrt{k_1^2 - 4 \cdot k_0 \cdot k_2}}{2 \cdot k_2} \end{aligned}$$

There will be two roots for y, the root $\in [0, 1]$ will be chosen, and

$$\begin{aligned} H &= p(x, y) - p_{00}(x, y) \\ E &= p_{01}(x, y) - p_{00}(x, y) \\ F &= p_{10}(x, y) - p_{00}(x, y) \\ G &= p_{00}(x, y) - p_{01}(x, y) + p_{11}(x, y) - p_{10}(x, y) \\ k_2 &= G_i \cdot F_j - Gj \cdot F_i \\ k_1 &= E_i \cdot F_j - E_j \cdot F_i + H_i \cdot G_j - H_j \cdot G_i \\ k_0 &= H_i \cdot E_j - H_j \cdot E_i \end{aligned}$$

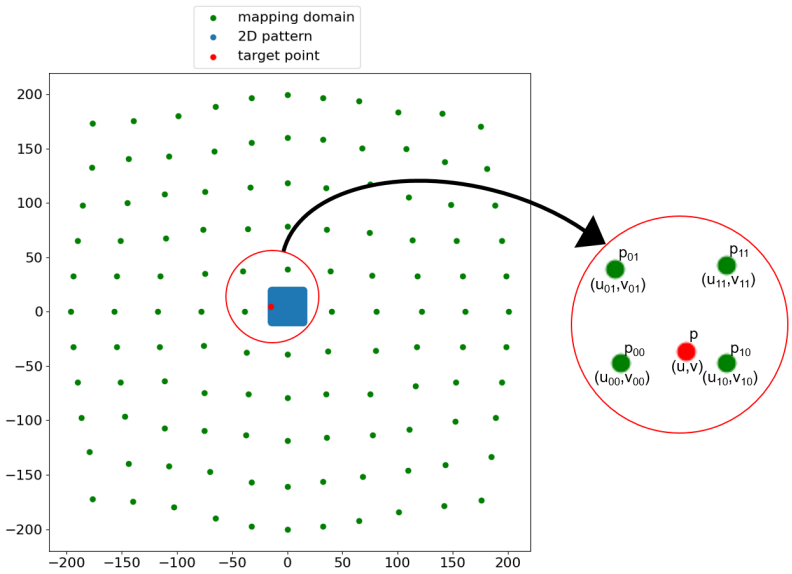
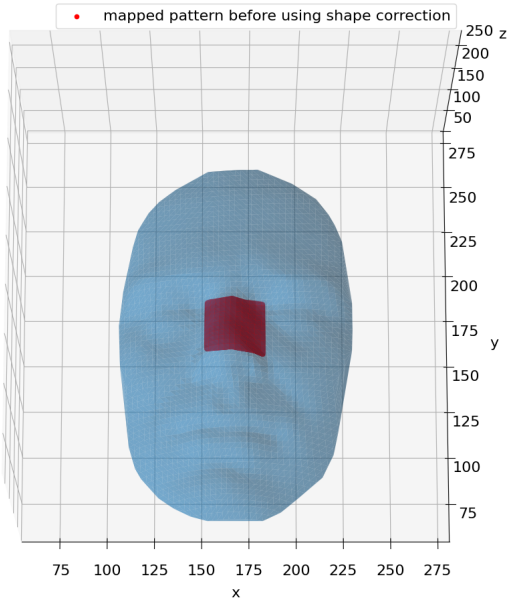
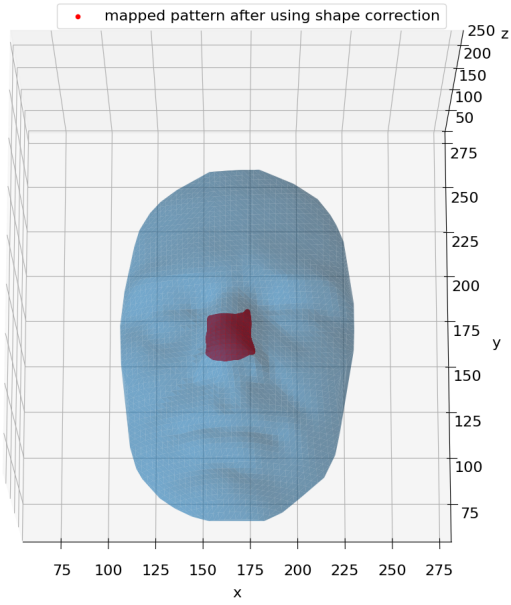


Figure 3.9: How to get the (u, v) parameters of a certain point for applying NURBS surface.

By calculating (u, v) parameters of all points in the 2-dimensional pattern (Algorithm 3.5), it can be mapped onto the 3-dimensional surface through putting all (u, v) parameters in Algorithm 3.2, and this process is the shape correction for mapping 2-dimensional pattern onto the 3-dimensional surface without distortion. Figure 3.10 shows a square with side length 30 that is being mapped onto the 3D human face surface model before and after using shape correction.



(a) Mapped square before applying shape correction.



(b) Mapped square after applying shape correction.

Figure 3.10: A square with side length 30 is mapped onto the 3D human face surface model (a) before applying shape correction and (b) after using shape correction.

Algorithm 3.5 Mapping Parameter

Input: $mapping\_domain$, pt, num
Output: u_k, v_k

$dist = pt - mapping\_domain$
$idx = \textbf{argmin}(\textbf{sqrt}(dist[:,0]^2 + dist[:,1]^2))$
$var = pt - mapping\_domain[idx]$
$f_u = idx\%(num+1)$; $f_v = idx//(num+1)$
if $var[0] \geq 0$ **and** $var[1] \geq 0$ **then**
    $u_{00} = f_u/num$; $u_{01} = (f_u+1)/num$; $u_{10} = f_u/num$; $u_{11} = (f_u+1)/num$
    $v_{00} = f_v/num$; $v_{01} = f_v/num$; $v_{10} = (f_v+1)/num$; $v_{11} = (f_v+1)/num$
else if $var[0] < 0$ **and** $var[1] \geq 0$ **then**
    $u_{00} = (f_u-1)/num$; $u_{01} = f_u/num$; $u_{10} = (f_u-1)/num$; $u_{11} = f_u/num$
    $v_{00} = f_v/num$; $v_{01} = f_v/num$; $v_{10} = (f_v+1)/num$; $v_{11} = (f_v+1)/num$
else if $var[0] \geq 0$ **and** $var[1] < 0$ **then**
    $u_{00} = f_u/num$; $u_{01} = (f_u+1)/num$; $u_{10} = f_u/num$; $u_{11} = (f_u+1)/num$
    $v_{00} = (f_v-1)/num$; $v_{01} = (f_v-1)/num$; $v_{10} = f_v/num$; $v_{11} = f_v/num$
else
    $u_{00} = (f_u-1)/num$; $u_{01} = f_u/num$; $u_{10} = (f_u-1)/num$; $u_{11} = f_u/num$
    $v_{00} = (f_v-1)/num$; $v_{01} = (f_v-1)/num$; $v_{10} = f_v/num$; $v_{11} = f_v/num$
end if
$idx_{00} = \textbf{round}(v_{00} * num * (num+1) + u_{00} * num)$

19: idx_{01} = **round**$(v_{01} * num * (num + 1) + u_{01} * num)$
20: idx_{10} = **round**$(v_{10} * num * (num + 1) + u_{10} * num)$
21: idx_{11} = **round**$(v_{11} * num * (num + 1) + u_{11} * num)$
22: $H = pt - mapping\_domain[idx_{00}]$
23: $E = mapping\_domain[idx_{01}] - mapping\_domain[idx_{00}]$
24: $F = mapping\_domain[idx_{10}] - mapping\_domain[idx_{00}]$
25: $G = mapping\_domain[idx_{00}] - mapping\_domain[idx_{01}] + mapping\_domain[idx_{11}] - mapping\_domain[idx_{10}]$
26: $k2 = G[0] * F[1] - G[1] * F[0]$
27: $k1 = E[0] * F[1] - E[1] * F[0] + H[0] * G[1] - H[1] * G[0]$
28: $k0 = H[0] * E[1] - H[1] * E[0]$
29: **if** $k2 == 0$ **then**
30: $v = -k0/k1$
31: **else**
32: $v1 = (-k1 + \mathbf{sqrt}(k1^2 - 4 * k0 * k2))/(2 * k2)$
33: $v2 = (-k1 - \mathbf{sqrt}(k1^2 - 4 * k0 * k2))/(2 * k2)$
34: **if** $v1 > 0$ **and** $v1 < 1$ **then**
35: $v = v1$
36: **else**
37: $v = v2$
38: **end if**
39: **end if**
40: $u = (H[0] - F[0] * v)/(E[0] + G[0] * v)$
41: $u_k = u_{00} + (u_{01} - u_{00}) * u + (u_{10} - u_{00}) * v + (u_{00} - u_{01} + u_{11} - u_{10}) * u * v$
42: $v_k = v_{00} + (v_{01} - v_{00}) * u + (v_{10} - v_{00}) * v + (v_{00} - v_{01} + v_{11} - v_{10}) * u * v$

Chapter 4

Methodology

In this research, 2-dimensional patterns should be able to be marked onto 3-dimensional surfaces with both given CAD models and unknown shapes. If the target surface is given by the CAD model, then it can be used in the shape correction directly. On the other hand, if the target surface's model is not provided, a shape-detecting sensor should be applied to obtain the surface model. In this chapter, the shape-detecting system for obtaining surfaces' model and the laser marking system for leaving markings on the surfaces will be introduced. Then, a mouse will be served as the surface model with the given CAD model case and a mandarin will be served as the surface with an unknown shape. The visualized simulation results generated through python will also be shown.

4.1 Experimental setup

In this section, the sensor used in obtaining 3-dimensional surfaces with unknown shapes and the laser marking system that can mark on a 3-dimensional surface will be discussed.

4.1.1 Shape detecting system

There are three main non-contact ways to acquire the shape of an object [18]:

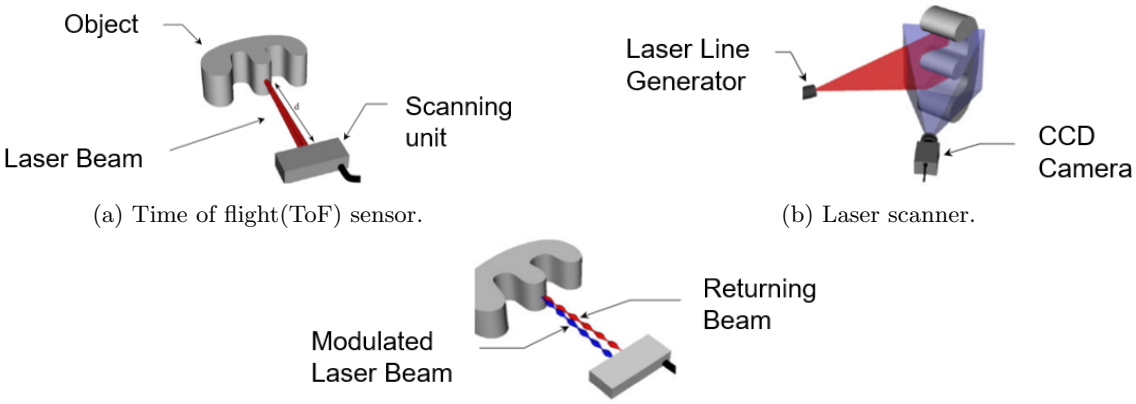
- Time of flight(TOF) sensor
- Laser scanner
- LiDAR

The time of flight sensor(Figure 4.1a) measures the time-consuming of a photon traveling from the sensor and reflected by the target object and then back to the sensor. Since the speed of light is a known value, the distance between the sensor and the target can be gotten by multiplying half of the time-consuming and the speed of light.

The laser scanner(Figure 4.1b) projects a line laser onto the target object and uses a camera to record the deformation of this line laser caused by the geometry of the target object. By observing the triangulation relationship between the laser generator, the target object, and the image taken by the camera, the shape of the target object can be derived.

By emitting a modulated laser beam to the target object and comparing it with the returning beam, LiDAR(Figure 4.1c) can acquire the shape of the target object. There is a phase shift between the emitted laser beam and the returning beam, which is caused by the distance traveled by the modulated laser beam. Because the wave number of a modulated laser beam is a constant with a known value, the distance traveled by this beam can be calculated by the phase shift.

Each of these methods has its own pros and cons. The ToF sensor is easy to implement and is not affected much by the ambient light, but its accuracy is relatively low compared with the other two methods; for the laser scanner, the advantage of it is its high resolution and accuracy, but the result will be deteriorated by the effect of the ambient light and the processing time is much longer than the other two methods due to the high resolution; the accuracy of LiDAR is also high and it is also not affected much by the ambient light much, but the phase wrapping is always the issue when using phase shift to examine the distance. Besides that, to use the triangulation relationship to acquire the shape of the target object, the laser line generator/camera will also be put at an angle. This angle prevents some regions of the target object to be illuminated/recorded. Still, the working distance of LiDAR is usually much longer than the other two methods, which increases the size of the whole shape-detecting system, making it not suitable for implementation in the production line. Consequently, the ToF sensor will be used in this research to obtain the surface information of unknown-shape objects.



(a) Time of flight(ToF) sensor.

(b) Laser scanner.

(c) LiDAR.

Figure 4.1: Sketches of different ways to acquire the shape of an object [18].

Figure 4.2 shows the shape-detecting system used in this research. This system comprises the pmd flexx2 ToF camera with specifications in Table 4.1, a USB cable for connecting to PC, a lifting platform that can adjust the distance between the target object and the sensor, and all components are attached to an optical cable.

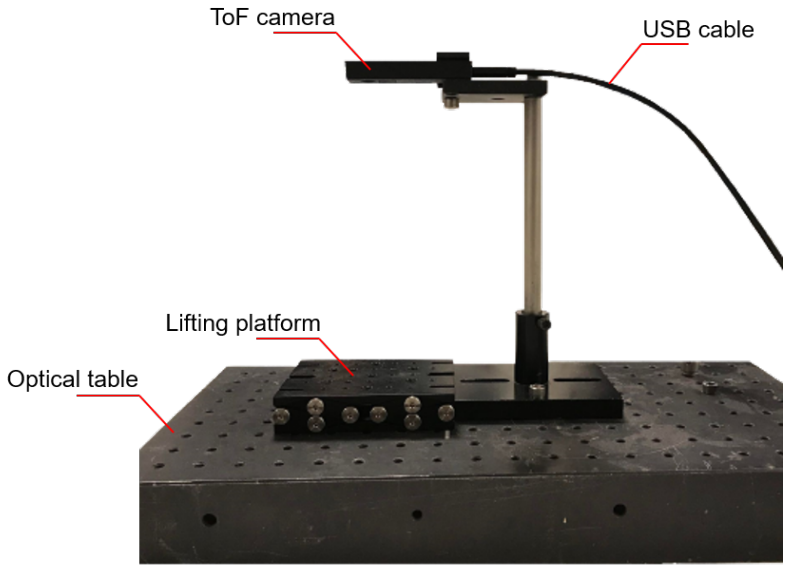
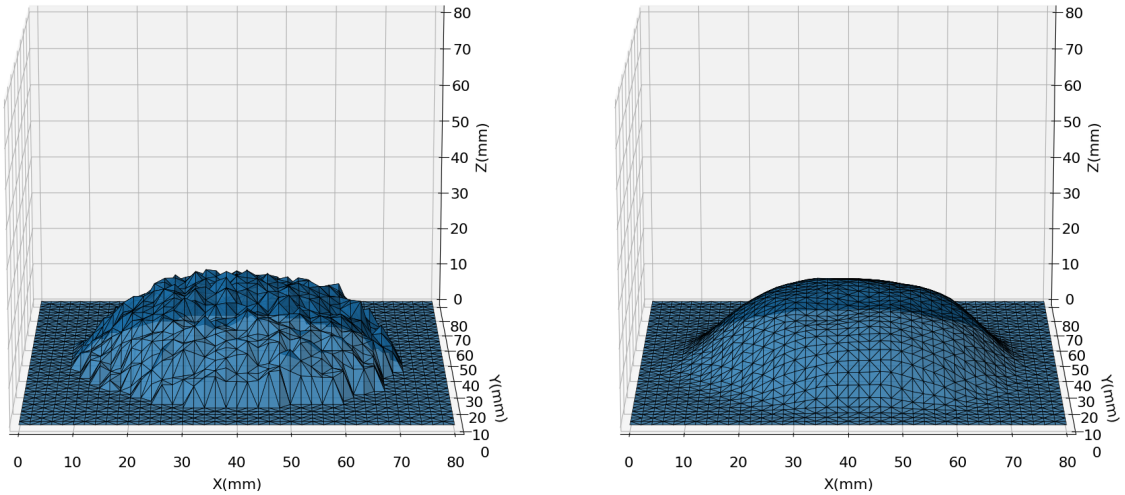


Figure 4.2: Shape-detecting system.

Table 4.1: Specifications of pmd flexx2 ToF camera

Resolution	224 × 172 Depth pixels
Measurement range	0.1-4m
Frame rate	Up to 60 fps
Illumination	940nm, VCSEL, Laser Class 1
Viewing angle($H \times V$)	56° × 44°
Depth resolution	≤ 1% of distance
Sunlight tolerance	At 100K Lumens(Full Sunlights), Loses10% max range vs. Indoors

When using this system, the target object will be put on the lifting platform to ensure the whole object is in the sensor's measurement range. The detection result will inevitably contain some noise. Thus, a low-pass filter will be implemented to remove the noise. Figure 4.3 shows the detected result of a mandarin's surface. To eliminate the noise (the zigzags in Figure 4.3a), the frequency of the zigzags needs to be observed first. The width of the zigzags is around 8 mm in this case, and each zigzag can be regarded as half a triangle wave, so the frequency of the noise is $1/16 = 0.0625/mm$. The points of this mandarin surface are sampled every 2 mm, leading the sampling rate to be $0.5/mm$ and Nyquist frequency will be $0.25/mm$, which is four times that of the noise. Hence, a low-pass filter with a cut-off frequency equal to a quarter of the Nyquist frequency is applied to the detected mandarin surface and the filtered result is shown in Figure 4.3b.



(a) The detected mandarin surface before filtering.

(b) The detected mandarin surface after filtering.

Figure 4.3: The detected mandarin surface (a) before filtering and (b) after filtering.

4.1.2 Laser marking system

The laser marking system used in this research is shown in Figure 4.4. As the typical laser marking system, this system also consists of a laser source, two mirrors attached to two galvanometers with perpendicular rotation axis to each other, and the f-theta lens that can focus the laser beam onto the target surface. However, with the assistance of inPhocal's optical module, the outgoing laser beam will have a long depth of focus. The lifting platform is used for adjusting the distance between the galvanometric scanner and the target object. Air suction is used in removing the gas generated during the marking. All components are attached to the optical table. The specifications of the laser marking system are given in Table 4.2.

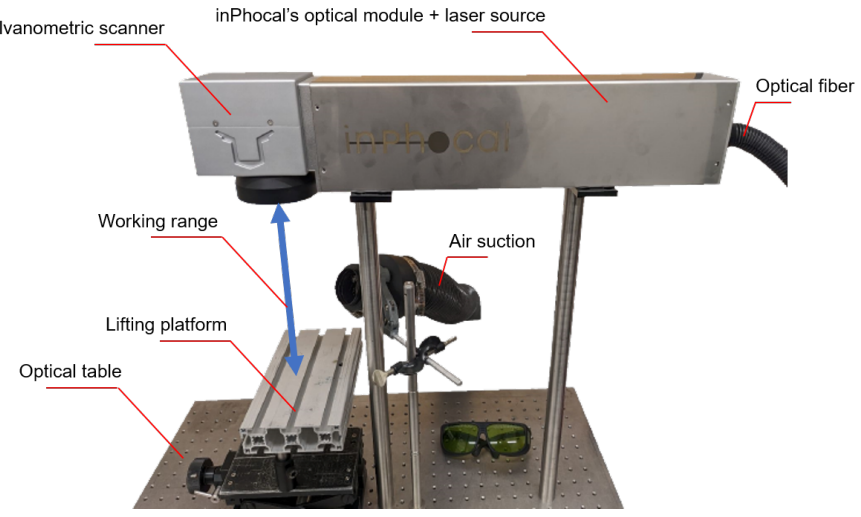


Figure 4.4: Laser marking system.

Table 4.2: Specifications of laser marking system

Power	30W
Laser type	Pulsed fiber laser, Laser Class 4
Wavelength	1064nm
Working range(distance from the galvanometric scanner to the target object)	291mm-341mm
Depth of focus	50mm
Beam spot	0.21mm

When using this laser system, the target object will be placed onto the lifting platform, so the distance between the galvanometric scanner and the target object can be adjusted to the focus range of the laser beam. In the best case, top of the target object will be at the distance equal to the lower limit of the working range from the galvanometric scanner, so the whole working range is possible to be exploited. Then, the 2-dimensional pattern will be processed by the shape correction method, stored in a vector type file such as dxf file, and imported into BeamConstruct software (Appendix B). The vector type 2-dimensional corrected pattern (Figure 4.5) will be the signal to control the galvanometers.

Figure 4.5: Vector type 2-dimensional pattern visualization and the coordinates of this pattern for controlling the galvanometers in BeamConstruct software.

In the traditional laser marking system, the laser beam usually has a few mm depth of focus, so the laser beam will quickly lose its focus when moving along a curved surface directed by the galvanometric scanner. Hence, either a device to change the focal length of the laser beam or a device to change the distance between the target object and the laser source in real time is needed, and both of these devices will increase the complexity of the whole system. However, the laser beam originating from the inPhocal's optical module has a 50 mm depth of focus. It means when thickness of the target surface is less than this value, the laser marking process can be easily finished by directly moving the laser spot to the position information provided by the 2-dimensional pattern.

4.2 Simulation

In this section, a picture file consisting of three characters A, B, and C will be the 2-dimensional pattern for laser marking(Figure 4.6). A mouse model will be the target surface with a given CAD model(Figure 4.7a) and a mandarin will be the target surface with an unknown shape(Figure 4.7b). Then how the 2-dimensional pattern will be mapped onto these two 3-dimensional surfaces will be visualized through python.

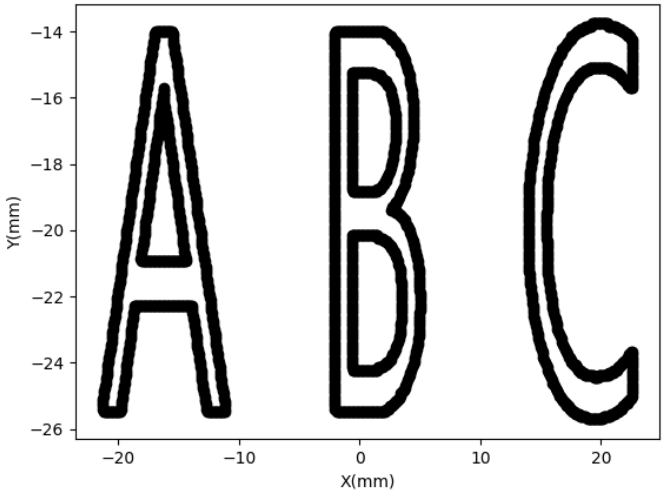
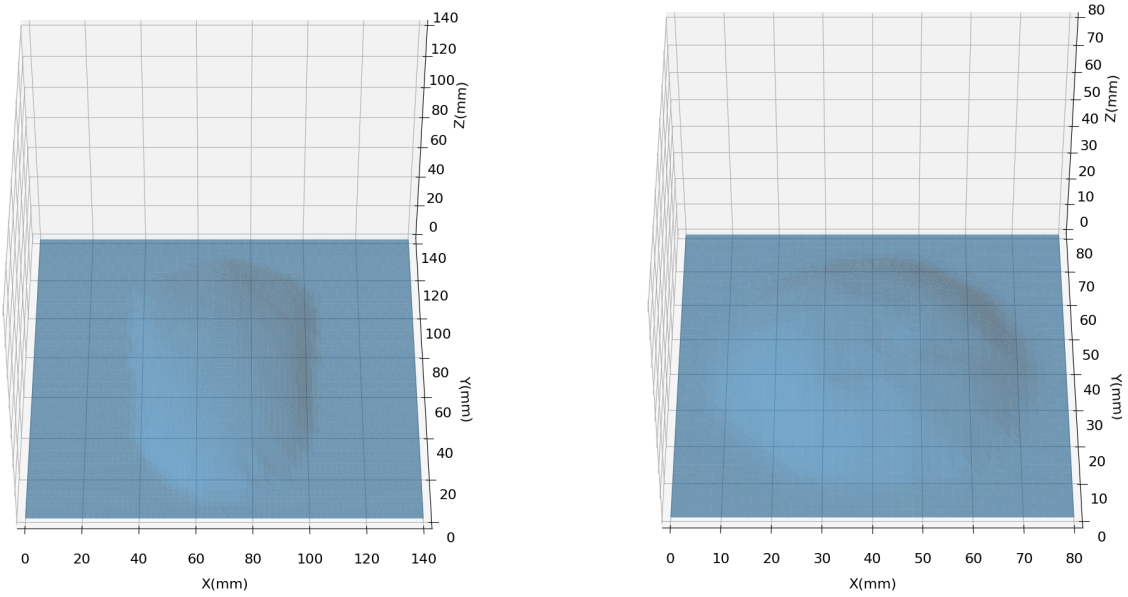


Figure 4.6: 2-dimensioanl pattern consisting of three characters A, B, and C.

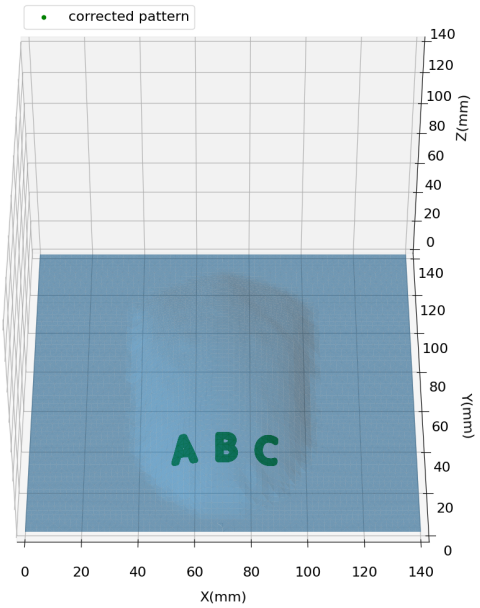


(a) The mouse surface model.

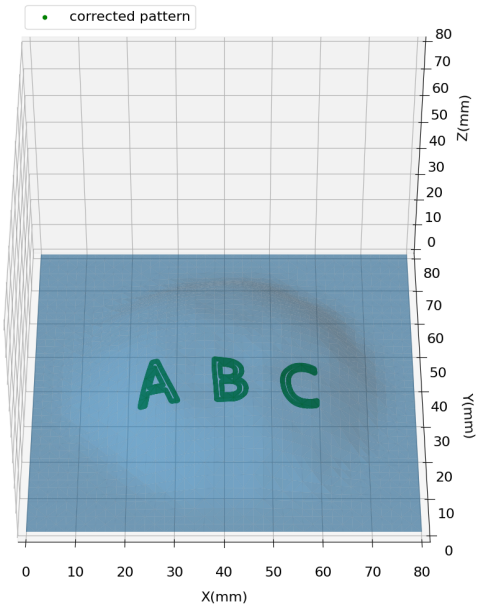
(b) The mandarin surface model.

Figure 4.7: The target surfaces that will be marked (a) the mouse surface model and (b) the mandarin surface model.

First, the picture file will be loaded by specifying the size and the center position that needs to be marked. Then, this pattern will be processed by shape correction for both the mouse model and the mandarin. Figure 4.8 shows what a 2-dimensional pattern looks like after being mapped onto 3-dimensional surfaces. In Figure 4.9, the pattern with red color is mapped through orthogonal projection. Compared to the corrected pattern(green color), it is clear that the pattern mapped through orthogonal projection will be elongated in the position with large curvature. On the contrary, the corrected pattern still keeps its shape even after being mapped onto the large curvature position. The laser marking onto the real surfaces model will be shown in the next chapter.

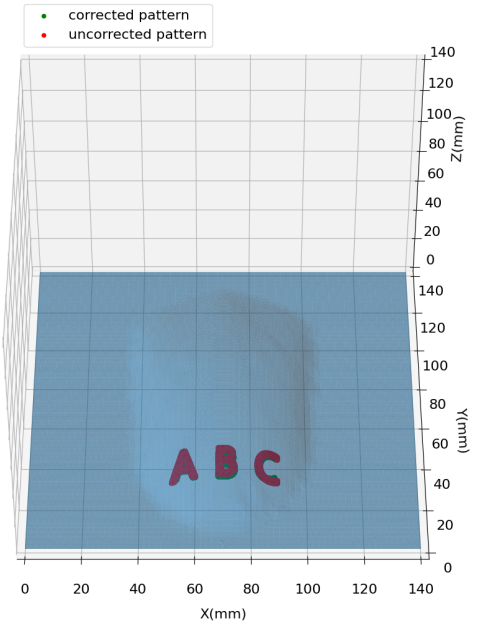


(a) The mouse surface model mapped with corrected ABC pattern.

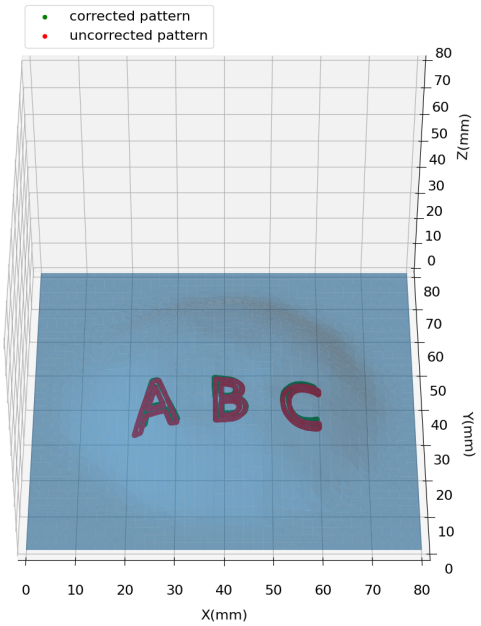


(b) The mandarin surface model mapped with corrected ABC pattern.

Figure 4.8: The target surfaces that are mapped with corrected ABC pattern (a) the mouse surface model and (b) the mandarin surface model.



(a) The mouse surface model mapped with corrected ABC pattern(green) and uncorrected pattern(red).



(b) The mandarin surface model mapped with corrected ABC pattern(green) and uncorrected pattern(red).

Figure 4.9: The target surfaces that are mapped with both corrected and uncorrected ABC pattern (a) the mouse surface model and (b) the mandarin surface model.

Chapter 5

Experimental results

In this section, the 2-dimensional pattern will be marked onto a mouse and a mandarin by inPhocal's laser marking system. Before doing the actual marking, the simulation results will be shown first, then, two performance evaluation methods of the shape correction method will be applied to the mapped pattern to see whether the distortion is eliminated or not. Finally, the corrected pattern will be imported into the BeamConstruct software and marked onto the 3-dimensional surfaces.

To evaluate the performance of the shape correction method, the similarity of a 2-dimensional pattern before and after being mapped should be considered. In this research, the perimeter and the area enclosed by this perimeter of the 2-dimensional pattern will be the criterion. To be specific, the contour of the 2-dimensional pattern is composed of many little line segments and the pattern itself consists of little pixels. By summing up the square root of the square of the length difference between each line segment before and after being mapped, the total length error e_l in perimeter can be obtained:

$$\text{length difference} = l_{3d\_s} - l_{2d\_s}$$

$$e_l = \sum_{s=0}^{n} |l_{3d\_s} - l_{2d\_s}| \tag{5.1}$$

where n is the total number of the line segments, l_{2d_s} is the s^{th} line segment before being mapped, and l_{3d_s} is the s^{th} line segment after being mapped. Also, the total area error e_a can be obtained by summing up the square root of the square of the area difference between each pixel before and after being mapped:

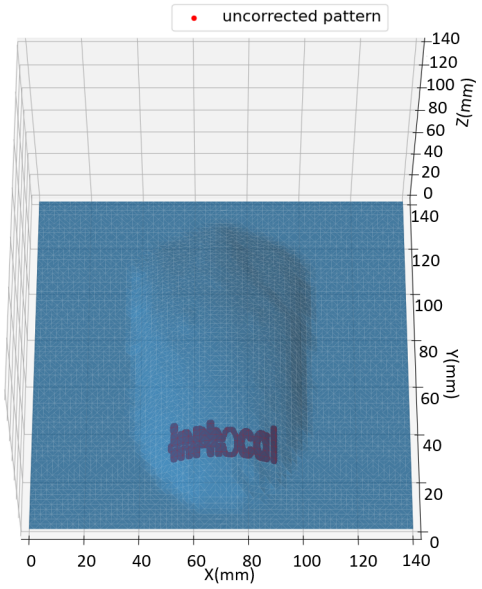
$$\text{area difference} = a_{3d\_s} - a_{2d\_s}$$

$$e_a = \sum_{s=0}^{n} |a_{3d\_s} - a_{2d\_s}| \tag{5.2}$$

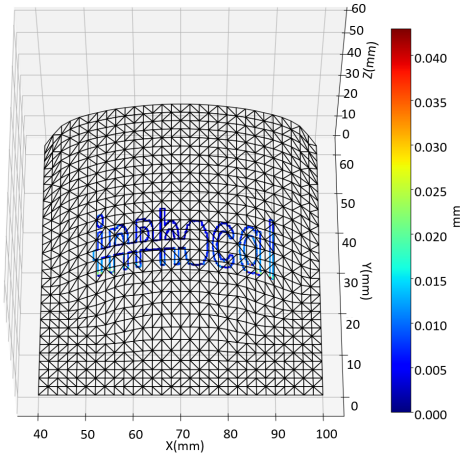
where n is the total number of the pixels enclosed by the perimeter, a_{2d_s} is the s^{th} pixel before being mapped, and l_{3d_s} is the s^{th} pixel after being mapped. It is obvious that when e_l and e_a are low, the similarity between the pattern before and after being mapped is high, which also means the good performance of the shape correction method.

5.1 Marking on the object with given well-designed surface model: computer mouse

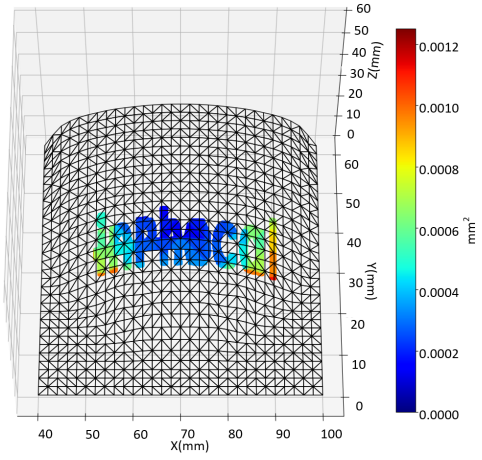
Figure 5.1 shows the simulation and evaluation result of mapping an uncorrected pattern onto the mouse's surface. The original 2-dimensional pattern is a picture file of inPhocal's logo with 1200pixel × 200pixel. This pattern is scaled in $39mm \times 13mm$ and orthogonally projected onto the target surface and it will be elongated at the parts of the target surface with large curvature naturally. The original perimeter and area enclosed by this perimeter of this pattern are 318.14mm and 165.69mm^2. After being projected, the perimeter and the area will be increased to 353.86mm and 197.27mm^2, and the total length error e_l is $35.72mm$ and the total area error e_a is $31.58mm^2$.



(a) Simulation result.



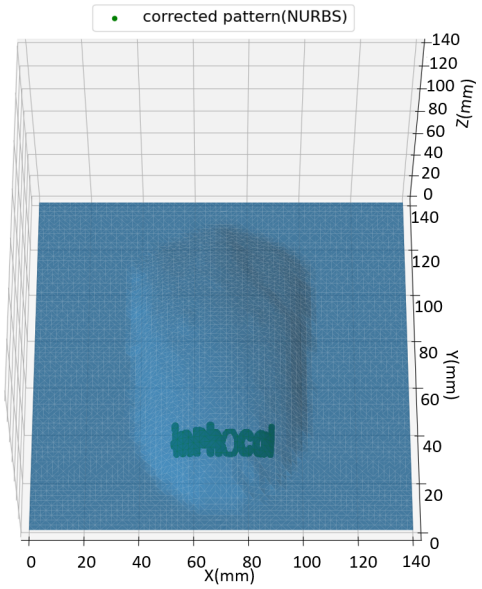
(b) Visualization of the length difference, $e_l = 35.72mm$.



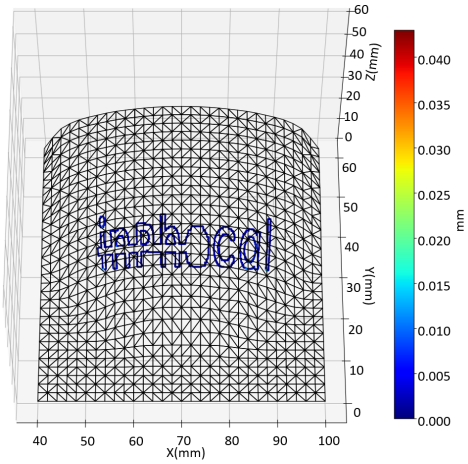
(c) Visualization of the area difference, $e_a = 31.58mm^2$.

Figure 5.1: Simulation of mapping uncorrected pattern onto the mandarin's surface. The pattern is the logo of inPhocal and it is in 1200pixel × 200pixel and mapped into $39mm \times 13mm$. (a) is the simulation result, (b) is the visualization of the length difference, and (c) is the visualization of the area difference.

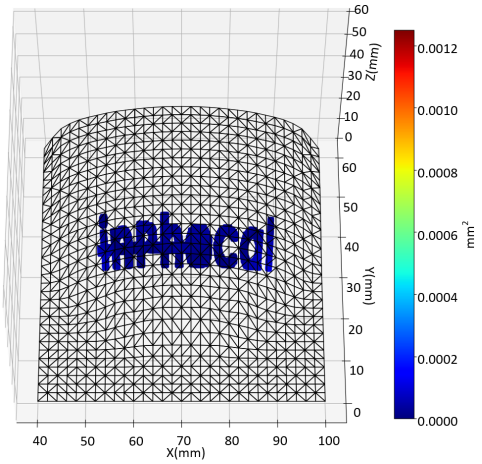
In Figure 5.2, the pattern is corrected by the shape correction method through NURBS, which uses 30points × 30points to create the mapping domain and 50 subintervals in Simpson's rule to calculate distance between those points, and mapped into $39mm \times 13mm$. This process costs 74.55 seconds. It can be found that the distortion has been largely eliminated, especially in the left bottom corner and the right bottom corner of the mapped pattern. Through the evaluation method, the decreasing of e_l (from 35.72mm to 5.42mm) and e_a (from 31.58mm^2 to 2.17mm^2) can also be observed. The perimeter and the area of the mapped pattern are 321.42mm and 166.82mm^2



(a) Simulation result.



(b) Visualization of the length difference, e_l = 5.42mm.

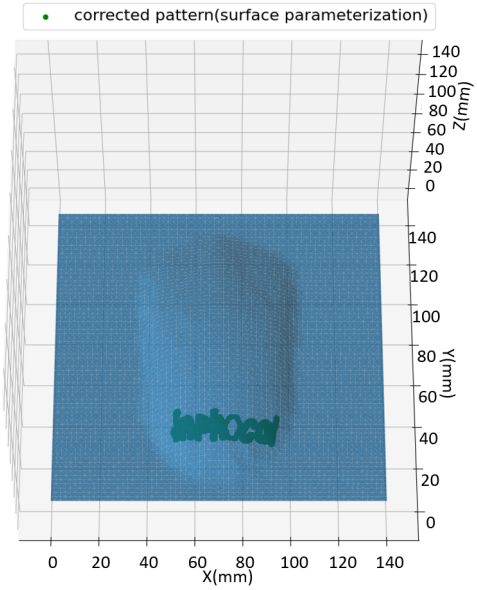


(c) Visualization of the area difference, e_a = 2.17mm^2.

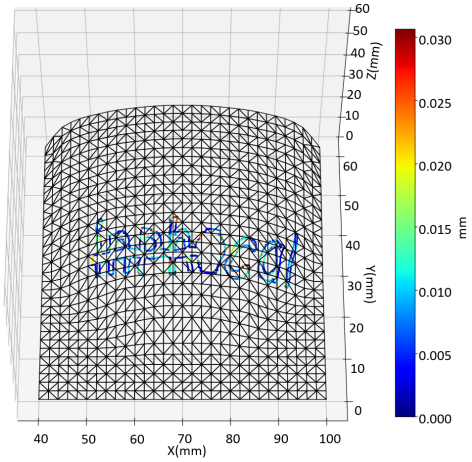
Figure 5.2: Simulation of mapping corrected pattern onto the mandarin's surface. The shape correction method uses 30points × 30points to create the mapping domain and 50 subintervals in Simpson's rule to calculate distance between those points, and the pattern is mapped into $39mm \times 13mm$. The time cost is 74.55 seconds. (a) is the simulation result, (b) is the visualization of the length difference, and (c) is the visualization of the area difference.

Figure 5.3 shows the simulation result of mapping the corrected pattern created by surface parame-

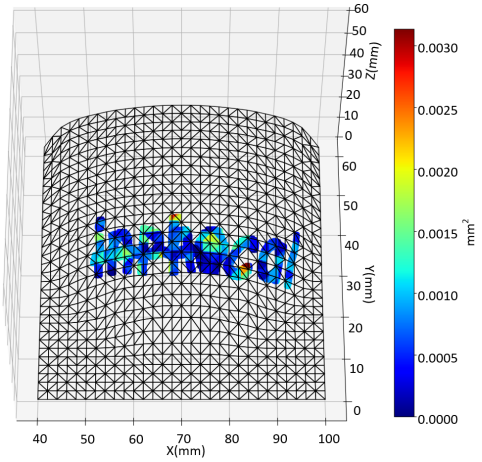
terization onto the mouse's surface. As mentioned in subsection 1.2.1, the surface is parameterized into a unit square, so it is hard to scale the pattern into desired size. In this case, the pattern is simply scaled in $1/39mm \times 1/13mm$ and put into the parameterized plane for mapping on the target surface. One can easily imagine that the dimensions in the x and y directions of the mouse are different, so the ratio of the mapped pattern will not keep the same. The size and the posture of the mapped pattern might be slightly different as well. Although both the e_l (from 35.72mm to 48.29mm) and the e_a (from 31.58mm^2 to 54.21mm^2) are increased (due to the size-changing) comparing to that of uncorrected pattern, the elongation at the left bottom corner and the right bottom corner are still eliminated. The whole process takes 1273.23 seconds, and the perimeter and the area of the mapped pattern are 335.74mm and 218.11mm^2



(a) Simulation result.



(b) Visualization of the length difference, e_l = 48.29mm.



(c) Visualization of the area difference, e_a = 54.21^2.

Figure 5.3: Simulation of mapping corrected pattern created by surface parameterization onto the mouse's surface. The pattern is scaled in $1/39mm \times 1/13mm$. The total time cost is 1273.23 seconds. (a) is the simulation result, (b) is the visualization of the length difference, and (c) is the visualization of the area difference.

Figure 5.4 is the actual marking result of the simulation shown in Figure 5.2. The corrected pattern for being imported into the BeamConstruct software is obtained by extracting the x and y coordinates of the mapped pattern (Figure 5.5). Since the marking process can be done without changing the distance between the laser source and the target surface through inPhocal's laser marking system, the z coordinate of the mapped pattern doesn't need to be recorded and exploited.

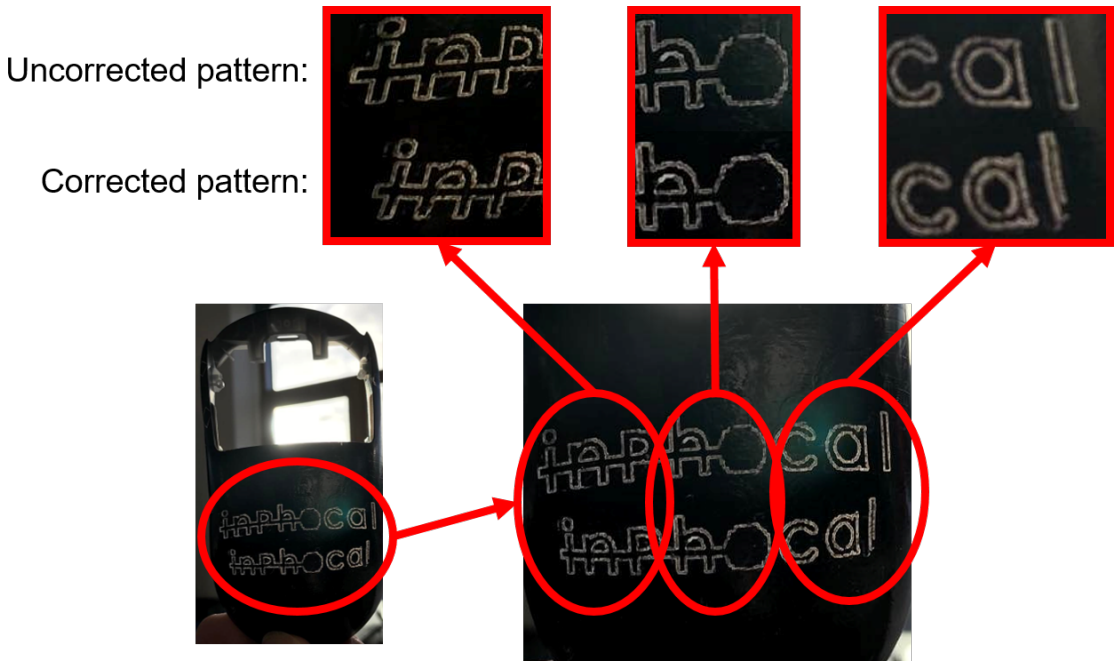
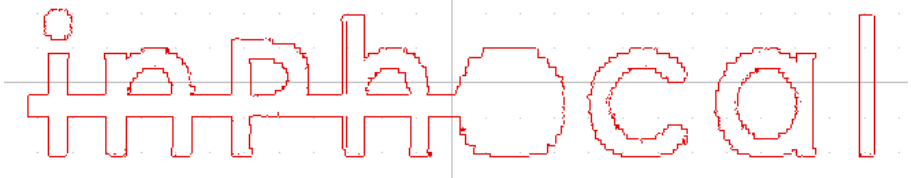


Figure 5.4: Actual marking result on the mouse's surface by using inPhocal's laser marking system.

(a) Corrected pattern for being imported into the BeamConstruct software and controlling the galvanometric scanner to mark on the mouse.

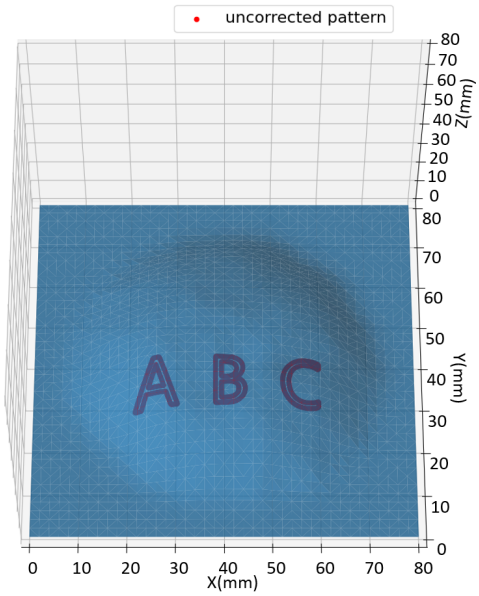


(b) Uncorrected pattern for being imported into the BeamConstruct software and controlling the galvanometric scanner to mark on the mouse.

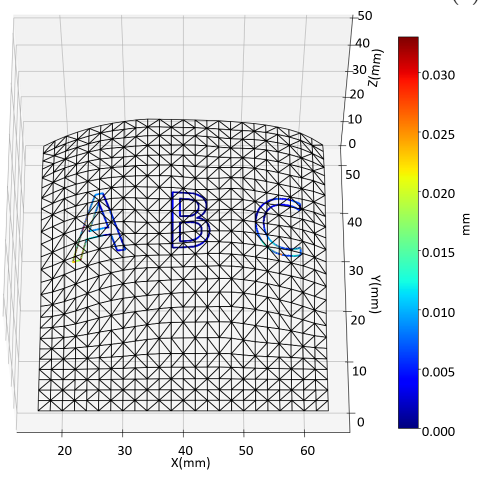
Figure 5.5: The visualized dxf files in the BeamConstruct software for marking on the mandarin.

5.2 Marking on the object with unknown shape surface model: mandarin

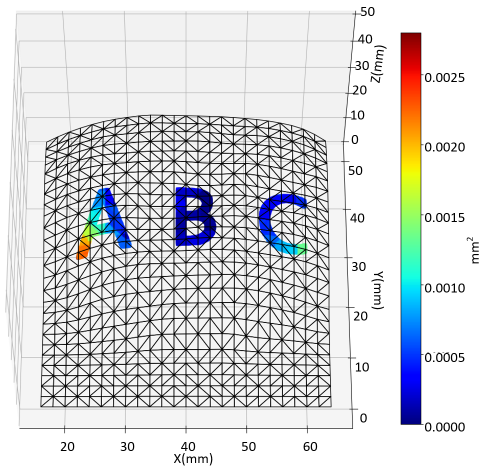
Figure 5.6 shows the simulation and evaluation result of mapping an uncorrected pattern onto the mandarin's surface. The original 2-dimensional pattern is a picture file of a string with 572pixel × 192pixel. This pattern is scaled in $39mm \times 13mm$ and orthogonally projected onto the target surface and again, it will be elongated at the parts of the target surface with large curvature. The original perimeter and area enclosed by this perimeter of this pattern are 150.58mm and 81.33mm^2. After being projected, the perimeter and the area will be increased to 161.38mm and 81.33mm^2, and the total length error e_l is $10.80mm$ and the total area error e_a is $9.58mm^2$.



(a) Simulation result.



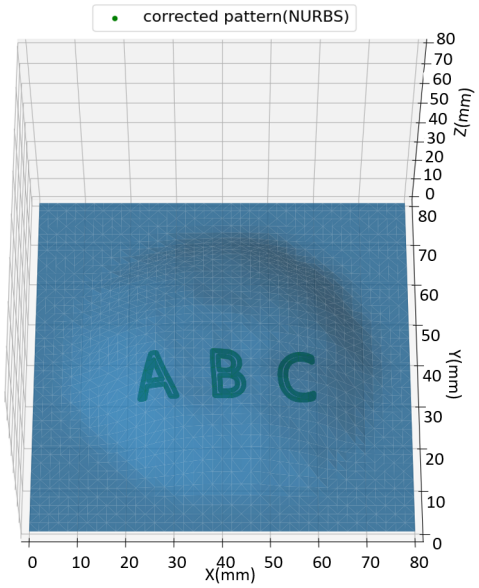
(b) Visualization of the length difference, e_l = $10.80mm$.



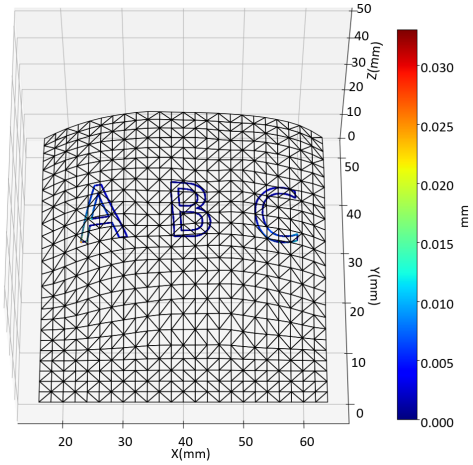
(c) Visualization of the area difference, e_a = $9.58mm^2$.

Figure 5.6: Simulation of mapping uncorrected pattern onto the mandarin's surface. The pattern contains 3 characters A, B, and C and it is in 572pixel × 192pixel and mapped into $39mm \times 13mm$. (a) is the simulation result, (b) is the visualization of the length difference, and (c) is the visualization of the area difference.

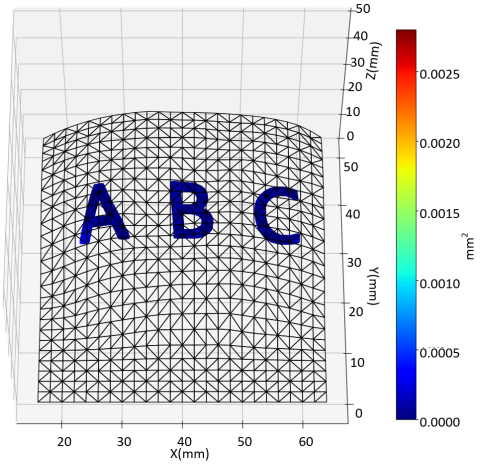
In Figure 5.7, the pattern is corrected by the shape correction method through NURBS, which uses 30points × 30points to create the mapping domain and 50 subintervals in Simpson's rule to calculate distance between those points, and mapped into $39mm \times 13mm$. This process costs 48.13 seconds. Comparing to the mapped uncorrected pattern, the distortion, especially in the left and the right bottom part, is eliminated. Through the evaluation method, the decreasing of e_l (from 10.80mm to 3.29mm) and e_a (from 9.58mm^2 to 0.99mm^2) can also be observed. The perimeter and the area of the mapped pattern are 152.53mm and 81.66mm^2



(a) Simulation result.



(b) Visualization of the length difference, e_l = 3.29mm.

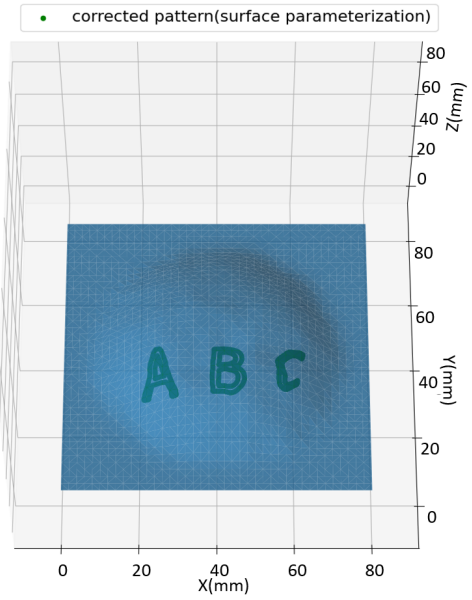


(c) Visualization of the area difference, e_a = 0.99mm^2.

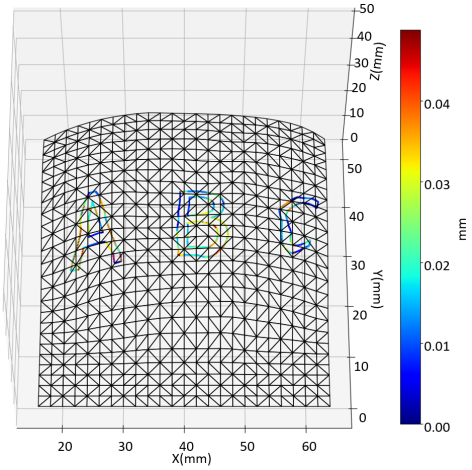
Figure 5.7: Simulation of mapping corrected pattern onto the mandarin's surface. The shape correction method uses 30points × 30points to create the mapping domain and 50 subintervals in Simpson's rule to calculate distance between those points, and the pattern is mapped into $39mm \times 13mm$. The time cost is 48.13 seconds. (a) is the simulation result, (b) is the visualization of the length difference, and (c) is the visualization of the area difference.

Figure 5.8 shows the simulation result of mapping the corrected pattern created by surface parameterization onto the mandarin's surface. The pattern is scaled into $1/39mm \times 1/13mm$ and

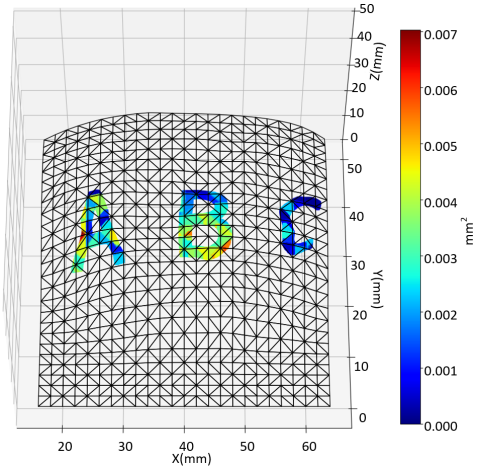
put into the parameterized plane for mapping. Due to the size-changing mentioned before, the e_l is increased from 10.80mm to 38.68mm and the e_a is in creased from 9.58mm^2 to 41.02mm^2 when comparing to the that of uncorrected pattern. However, the elongation at the left bottom and the right bottom parts is still diminished. The whole process takes 135.23 seconds, and the perimeter and the area of the mapped pattern are 178.75mm and 118.48mm^2



(a) Simulation result.



(b) Visualization of the length difference, e_l = 38.68mm.



(c) Visualization of the area difference, e_a = 41.02^2.

Figure 5.8: Simulation of mapping corrected pattern created by surface parameterization onto the mandarin's surface. The pattern is scaled into $1/39mm \times 1/13mm$. The total time cost is 135.23 seconds. (a) is the simulation result, (b) is the visualization of the length difference, and (c) is the visualization of the area difference.

Figure 5.9 is the actual marking result of the simulation shown in Figure 5.7. The corrected pattern for being imported into the BeamConstruct software is obtained by extracting the x and y coordinates of the mapped pattern (Figure 5.10).

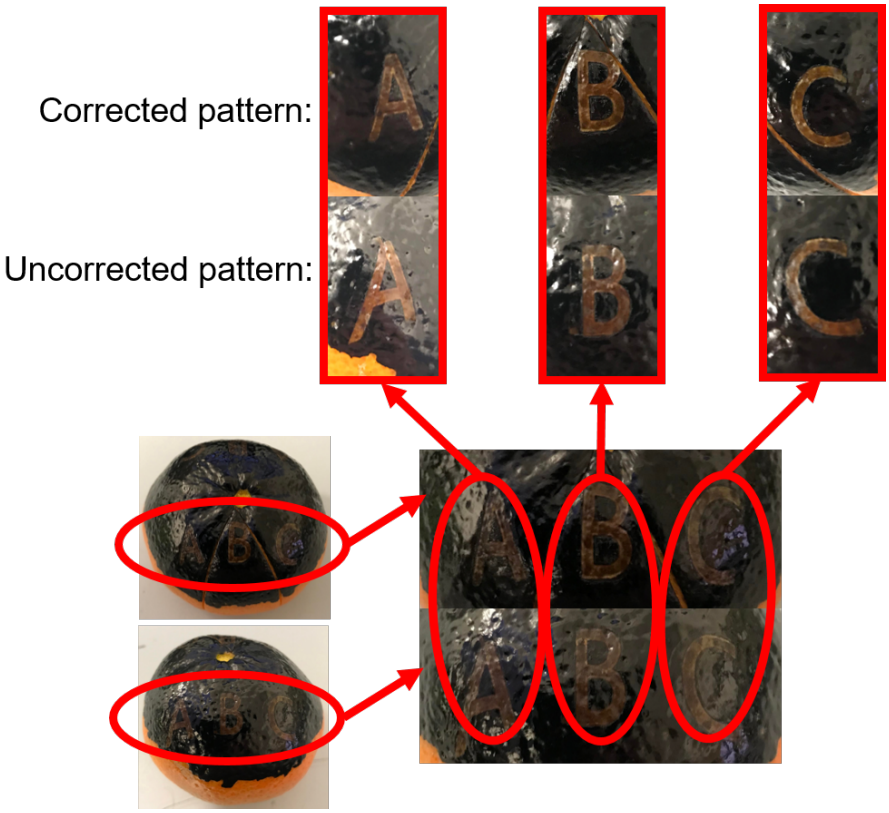
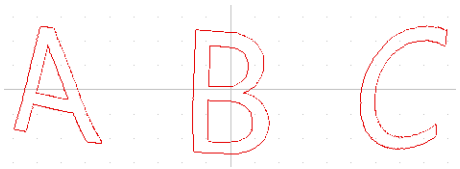
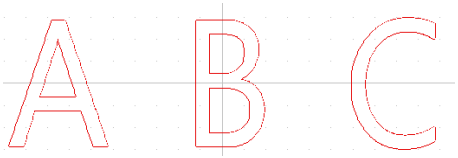


Figure 5.9: Actual marking result on the mandarin's surface by using inPhocal's laser marking system.



(a) Corrected pattern for being imported into the BeamConstruct software and controlling the galvanometric scanner to mark on the mandarin.



(b) Uncorrected pattern for being imported into the BeamConstruct software and controlling the galvanometric scanner to mark on the mandarin.

Figure 5.10: The visualized dxf files in the BeamConstruct software for marking on the mandarin.

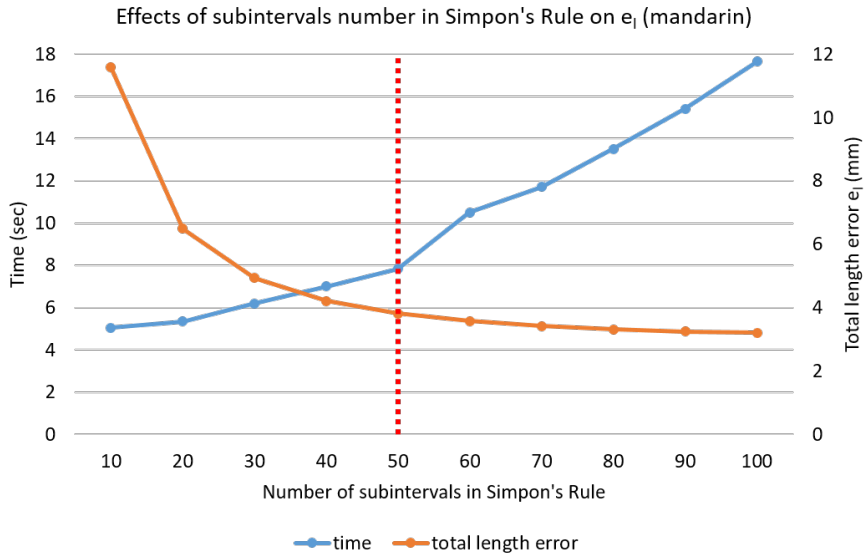
Chapter 6

Discussion

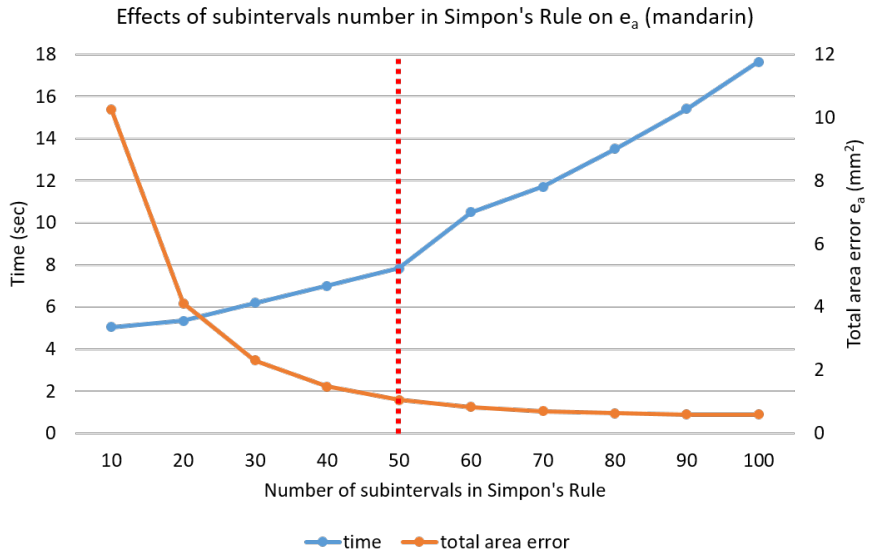
In chapter 5, the corrected patterns are created through the shape correction method based on NURBS by using 30points × 30points to create the mapping domain and 50 subintervals in Simpson's rule to calculate distance between those points. There are always errors when constructing such mapping domains and using the numerical integration method to do the integral. It is straightforward that both the mapping domains and the integral value can be created as precisely as possible when using a large number of points and subintervals, but it will also increase the computational load. Therefore, a trade-off has to be made between the time cost and the accuracy. In order to choose the proper number of points for creating the mapping domain and subintervals in Simpson's rule, both how the number of points to create the mapping domain and how the number of subintervals used in Simpson's rule will affect the precision of the shape correction method will be discussed. What's more, the current state-of-the-art surface parameterization-based shape correction method will be compared with the new NURBS-based shape correction method proposed in this chapter.

6.1 Effects of number of subintervals in Simpson's Rule on correction error

To find the effects of the number of subintervals in Simspon's Rule on correction error, different numbers of subintervals will be chosen to apply Simspon's Rule. Then, in each case, the total length error e_l and the total area error e_a will be calculated (simulation results can be found in Appendix C). Figure 6.1 shows the run chart of the number of subintervals verse the cost time (blue line) and verse the shape correction error (orange line). It can be found that if the number of subintervals is more than 50, both the total length error e_l and the total area error e_a are not decreasing much when the cost time is significantly increased. Therefore, unless the precision of shape correction is highly demanded, using 50 subintervals in Simspon's rule can good a good corrected pattern with acceptable time consumption.



(a) Effects of the number of subintervals in Simpson's rule on total length error e_l.

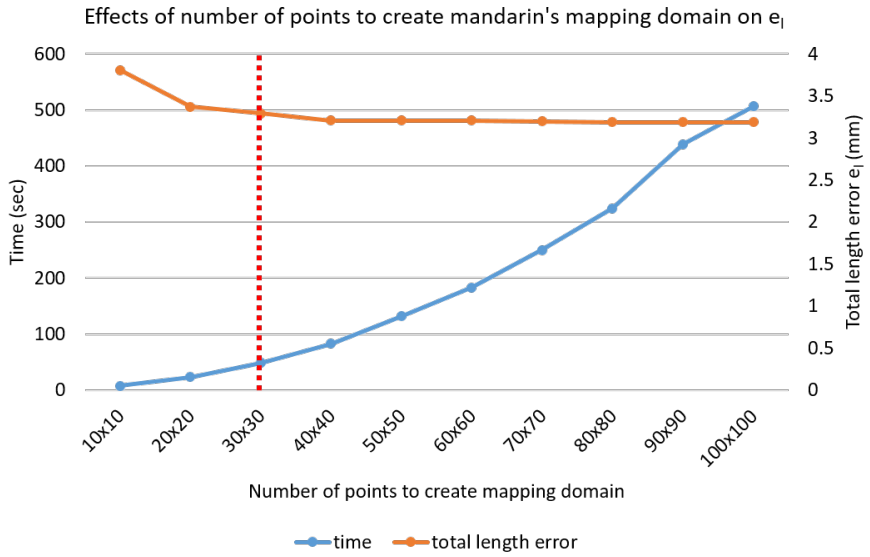


(b) Effects of the number of subintervals in Simpson's rule on total area error e_a.

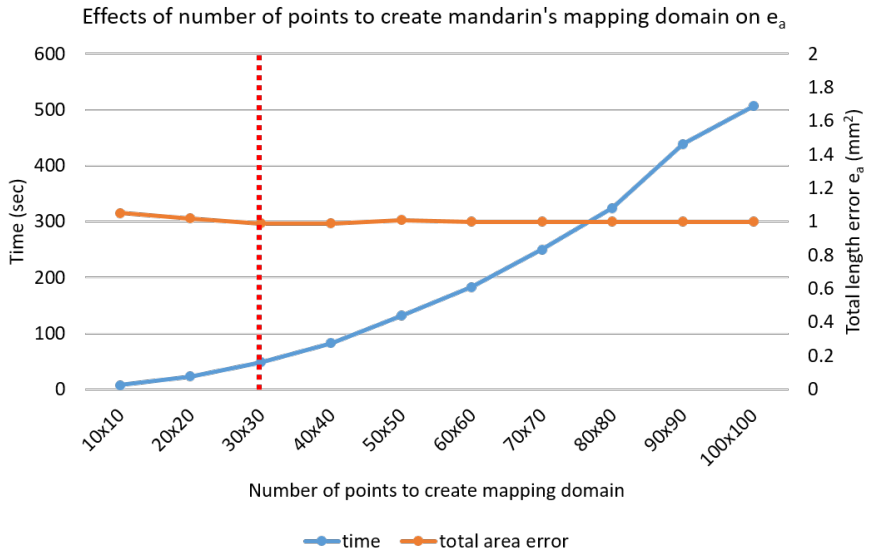
Figure 6.1: Effects of the number of subintervals in Simpson's rule on the shape correction error.

6.2 Effects of number of points to create mapping domain on correction error

In Figure 6.2b, different numbers of points are chosen to create the mandarin's mapping domain (see the simulation results in Appendix D). It can be seen that the total length error e_l is decreasing when the number of points is increasing between 10×10 and 30×30. After the number of points is more than 30×30, the total length error e_l doesn't change a lot. In the area length error e_a. there is a slightly decreasing when the number of points increases from 10×10 to 30×30, but after increasing to more than 30×30 points, the variation can not be observed anymore.



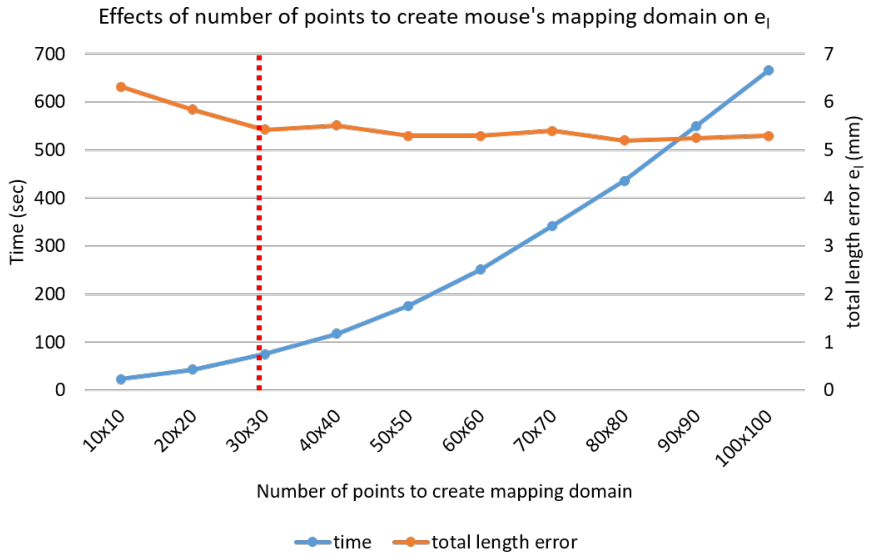
(a) Effects of the number of points to create the mandarin's mapping domain on total length error e_l.



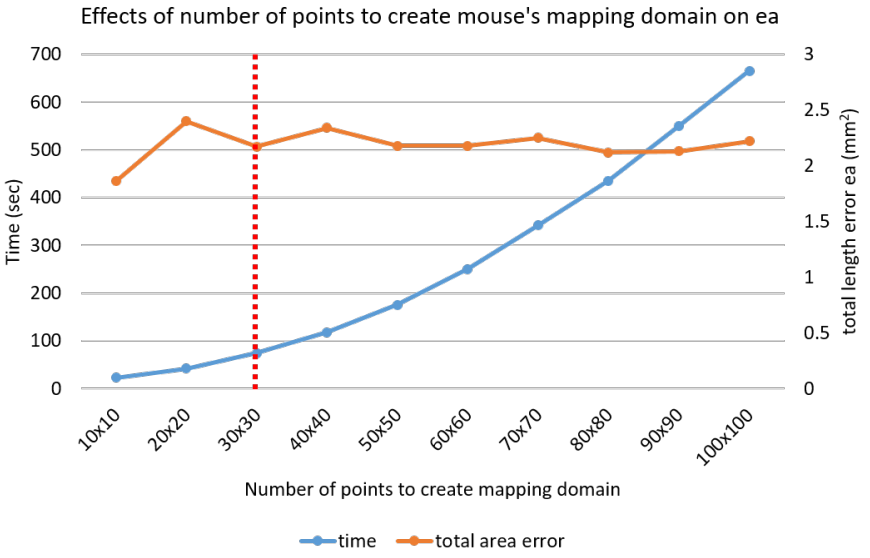
(b) Effects of the number of points to create the mandarin's mapping domain on total area error e_a.

Figure 6.2: Effects of the number of points to create the mandarin's mapping domain on the shape correction error.

In Figure 6.3, different numbers of points are chosen to create the mouse's mapping domain (see the simulation results in Appendix D). The effect of the number of points on the total length error e_l, in this case, is very similar to that of the mandarin case. The decrease of e_l is obvious when the number of points is less than 30×30, but it becomes unapparent when the number of points is more than 30×30. For the area length error e_a, the values fluctuate when the number of points is increasing, but the variation is less than 1mm^2, which is a low value compared to the total mapped area (165.69mm^2).



(a) Effects of the number of points to create the mouse's mapping domain on total length error e_l.



(b) Effects of the number of points to create the mouse's mapping domain on total area error e_a.

Figure 6.3: Effects of the number of points to create the mouse's mapping domain on the shape correction error.

Based on these discussions, it can be said that using 30×30 points to create the mapping domain is the optimal scenario to do the shape correction proposed in this research.

6.3 Comparison between shape correction method based on NURBS and surface parameterization

Table 6.1 and Table 6.2 show the time consumption, total length difference e_l and total area error e_a by using the NURBS-based shape correction method with 50 subintervals and different numbers of points in the mapping domain, and by using the surface parameterization-based shape correction method.

In Table 6.1, the pattern is a picture file of a string with 572pixel × 192pixel, and in Table 6.2, the pattern is a picture file of inPhocal's logo with 1200pixel × 200pixel. Both patterns are scaled in 39mm × 13mm, and mapped onto the mandarin's surface and the mouse's surface respectively. In both cases, the e_l and ea by using the NURBS-based shape correction method are much lower than that of the surface parameterization-based shape correction method. In the mandarin case, when using less than 50×50 points in the mapping domain by using the NURBS method, the time consumption can also be lower than that of the surface parameterization method. However, in the mouse case, even using 100×100 points in the mapping domain, the time consumption is still lower than that of using the surface parameterization method. It is because there are more points (1200pixel × 200pixel) that need to be mapped in inPhocal's logo than the string pattern (572pixel×192pixel) used in the mandarin case and the pattern mapping process takes a lot of time in the surface parameterization method. Hence, the advantage in efficiency of the NURBS-based shape correction method can be magnified when mapping patterns with high resolution.

Table 6.1: Time consumption, total length difference e_l and total area error e_a comparison by using different methods of shape correction in mapping on the mandarin's surface.

Method	time consumption (sec)	e_l(mm)	e_a(mm^2)
10×10 mapping domain points NURBS	7.83	3.81	1.05
20×20 mapping domain points NURBS	22.97	3.37	1.02
30×30 mapping domain points NURBS	48.13	3.29	0.99
40×40 mapping domain points NURBS	82.75	3.21	0.99
50×50 mapping domain points NURBS	131.35	3.21	1.01
60×60 mapping domain points NURBS	183.06	3.21	1.00
70×70 mapping domain points NURBS	249.78	3.2	1.00
80×80 mapping domain points NURBS	324.53	3.19	1.00
90×90 mapping domain points NURBS	438.86	3.19	1.00
100×100 mapping domain points NURBS	506.75	3.19	1.00
Surface parameterization	135.23	38.68	41.02

Table 6.2: Time consumption, total length difference e_l and total area error e_a comparison by using different methods of shape correction in mapping on the mouse's surface.

Method	time consumption (sec)	e_l(mm)	e_a(mm^2)
10×10 mapping domain points NURBS	23.02	6.32	1.86
20×20 mapping domain points NURBS	42.26	5.84	2.4
30×30 mapping domain points NURBS	74.55	5.42	2.17
40×40 mapping domain points NURBS	117.48	5.51	2.34
50×50 mapping domain points NURBS	175.25	5.29	2.18
60×60 mapping domain points NURBS	250.654	5.29	2.18
70×70 mapping domain points NURBS	342.22	5.4	2.25
80×80 mapping domain points NURBS	435.24	5.2	2.12
90×90 mapping domain points NURBS	549.85	5.25	2.13
100×100 mapping domain points NURBS	665.32	5.29	2.22
Surface parameterization	1273.23	48.29	54.21

Chapter 7

Conclusion

By using the long depth of focus laser marking system developed by inPhocal, fast laser marking on 3-dimensional surfaces is not impossible anymore. Through the help of the shape correction method, the patterns will not be distorted after being marked onto the 3-dimensional surfaces. Therefore, the dominant way to print information onto products in the production line, ink and stickers, which pollute our planet severely, can now be replaced by laser marking.

When marking the products with simple shapes such as cans (cylindrical surfaces, see Appendix E), shape correction can be done by examining and eliminating the distortion of projected patterns, but the products do not always have such a simple shape. Hence, a more powerful shape correction method is needed to correct patterns for free-form surfaces. The new shape correction method proposed in this research gives a fast and precise way to map 2-dimensional patterns onto 3-dimensional surfaces. Compared to the existing shape correction method based on surface parameterization, this new shape correction method is based on NURBS exempts the pre-processing of the 2-dimensional patterns and the time-consuming pattern mapping process. Since products sometimes are not all in the same shape (e.g. fruits), pre-processing the patterns case-by-case is required when applying the surface parameterization-based method, and it also costs a lot of time. Such a time-consuming method will slow down the speed of production, so it is more efficient to use this NURBS-based shape correction method, especially when marking high-resolution patterns onto surfaces with complicated shapes.

In the production line, if the products all have the same shapes, the corrected pattern can be generated by the shape correction method beforehand. However, when the products are not in the same shape, the surface information needs to be captured by a shape-detecting system. Then, the unique corrected pattern will be generated for each product. In this case, the shape-detecting system can be placed before the laser marking system at a certain distance, so when it is marking one product, there is another product can be detected at the same time. This distance can be calculated by multiplying speed of the production line by the time to complete shape correction. For example, when a string pattern wants to be mapped onto mandarins as the case in section 5.2, it will take around 50sec to finish the shape correction process. If speed of the production line is 10cm/second, the shape-detecting system should be placed 5m before the laser marking system, so when the mandarin arrives at the place of laser marking system, the corrected pattern has just been generated.

To apply NURBS, the target surface should be sliced into several slices and the number of sample points in each slice should be the same. In this research, zero padding is used to make sure the same number of sample points are in these slices, but it will add additional points in the surface model, increasing the calculation time. It is better to look for another way to do this in the future.

Besides, the performance evaluation of the corrected pattern is only applied to the simulation results since there is still a lack of an effective way to measure the pattern on the real object. Hence, further research on this topic is still required.

In this research, all patterns are marked on the surfaces by pointwise laser marking, but there is another way to leave patterns by projecting a laser beam onto the target surfaces, which is known as laser lithography. In this method, a plate with holes that form the pattern that wants to be marked, which is also known as a mask, is placed between the laser source and the target surface. When the mask is illuminated by a laser beam, the beam can only pass through the holes, so a visible mark can be left on the target surface. The corrected pattern created by the shape correction method can also be used in producing the mask for distortion-free laser lithography. However, the laser beam might go through some optical components to control the spot size, making it not a parallel light beam. In this situation, the marked pattern will change size when putting the mask in different locations between the laser source and the target surface, so more researches on how the position of the mask will affect the size of marked pattern still need to be conducted.

Appendix A

Algorithms for NURBS surface

A.1 Data rearrangement algorithm

Algorithm A.1 Data Rearrangement

Input: $\mathbf{Q}$
Output: $\mathbf{Q}_{rearranged}$
$x_{center} = \mathbf{sum}(\mathbf{Q}[:,0])/len(\mathbf{Q}[:,0])$; $y_{center} = \mathbf{sum}(\mathbf{Q}[:,1])/len(\mathbf{Q}[:,1])$
$\mathbf{Q}[:,0] = \mathbf{Q}[:,0] - x_{center}$; $\mathbf{Q}[:,1] = \mathbf{Q}[:,1] - y_{center}$ ▷ reposition to the center of mass
$\mathbf{Q}_{rearranged} = \mathbf{PCA\ fitting}(\mathbf{Q})$ ▷ fit the first/second principal component to x/y axis.

A.2 Data selection algorithm

Algorithm A.2 Data Selection

Input: $\mathbf{Q}_{rearranged}$, n
Output: $\mathbf{Q}_{selected}$, m
$w_s = (\mathbf{max}(\mathbf{Q}_{rearranged}[:,0]) - \mathbf{min}(\mathbf{Q}_{rearranged}[:,0]))/n$ ▷ width of slices
$proj\_planes = [\mathbf{min}(\mathbf{Q}_{rearranged}[:,0])]$
for $i = 1 \rightarrow n+1$ **do**
    $proj\_planes.\mathbf{append}(\mathbf{min}(\mathbf{Q}_{rearranged}[:,0]) + (i-0.5) * w_s)$
end for
$proj\_planes.\mathbf{append}(\mathbf{max}(\mathbf{Q}_{rearranged}[:,0]))$ ▷ creating projection planes
$slice = [\,]$; $min\_num = \mathbf{inf}$
for $ppl = 1 \rightarrow \mathbf{len}(proj\_planes) - 1$ **do**
    $pts\_in\_slice = [p$ **for** p **in** $\mathbf{Q}_{rearranged}$ **if** $p[0] \geq proj\_planes[ppl] - w_s/2$ **and**
        $p[0] \leq proj\_planes[ppl] + w_s/2]$
    $pts\_in\_slice = [:,0] = proj\_planes[ppl]$
    if $\mathbf{len}(pts\_in\_slice) < min\_num$ **then**
        $min\_num = \mathbf{len}(pts\_in\_slice)$
    end if
    $pts\_in\_slice = pts\_in\_slice[\mathbf{argsort}(pts\_in\_slice.T,\ axis = 1).T[:,1]]$
    for pt **in** pts_in_slice **do**
        $slice.\mathbf{append}([pt[0],\ pt[1],\ pt[2]])$
    end for
end for

19: $\mathbf{Q}_{selected} = [\,],\ m = min\_num$
20: **for** $ppl = 1 \rightarrow$ **len**$(projection\ planes) - 1$ **do**
21: $\quad pts\_selected = [p$ **for** p **in** $slice$ **if** $p[0]\ ==\ proj\_planes[ppl]]$
22: $\quad \mathbf{Q}_{selected}$.**append**$(pts\_selected[0])$
23: $\quad$ **for** $i = 1 \rightarrow m - 1$ **do**
24: $\quad\quad \mathbf{Q}_{selected}$.**append**$(pts\_selected[$**int**$((\textbf{len}(pts\_selected) - 2)/(m - 2) * i)])$
25: $\quad$ **end for**
26: $\quad \mathbf{Q}_{selected}$.**append**$(pts\_selected[-1])$
27: **end for**

A.3 Chord length parameterization algorithm

Algorithm A.3 Chord Length Parameterization

Input: $\mathbf{Q}_{selected},\ m$
Output: $\bar{u}$
1: $\bar{u} = [\,]$
2: **for** $l = 0 \rightarrow$ **len**$(\mathbf{Q}_{selected})$ **do**
3: $\quad u_l = 0$
4: $\quad$ **for** $i = 1 \rightarrow m + 1$ **do**
5: $\quad\quad u_l = u_l + \|\mathbf{Q}_{selected}[l][i] - \mathbf{Q}_{selected}[l][i-1]\|$
6: $\quad$ **end for**
7: $\quad \bar{u}_l = [\,]\ ;\ u_o = 0$
8: $\quad \bar{u}_l$.**append**(u_o)
9: $\quad$ **for** $p = 1 \rightarrow m + 1$ **do**
10: $\quad\quad u_o = u_o + \|\mathbf{Q}_{selected}[l][i] - \mathbf{Q}_{selected}[l][i-1]\|/u_l$
11: $\quad\quad \bar{u}_l$.**append**(u_o)
12: $\quad$ **end for**
13: $\quad \bar{u}$.**append**$(\bar{u}_l)$
14: **end for**

A.4 Finding knot vector algorithm

Algorithm A.4 Get Knot Vector

Input: $\bar{u},\ m,\ p$
Output: $\mathbf{U}$
1: $\mathbf{U} = [\,]$
2: **for** $i = 0 \rightarrow$ **len**$(\bar{u})$ **do**
3: $\quad U_l = [\,]$
4: $\quad$ **for** $u = 0 \rightarrow p + 1$ **do**
5: $\quad\quad U_l$.**append**(0)
6: $\quad$ **end for**
7: $\quad$ **for** $i = 0 \rightarrow m - p + 1$ **do**
8: $\quad\quad u_{j,p} =$ **sum**$(\bar{u}[i : i + p])/p$
9: $\quad\quad U_l$.**append**$(u_{j,p})$
10: $\quad$ **end for**
11: $\quad$ **for** $u = 0 \rightarrow p + 1$ **do**
12: $\quad\quad U_l$.**append**(1)
13: $\quad$ **end for**
14: $\quad U$.**append**(U_l)
15: **end for**

16: *projection planes*.**append**(max(**Q**[:, 0]))

A.5 Control points algorithm

Algorithm A.5 Get Control Points

Input: $\mathbf{Q}_{selected}$, $\bar{u}$, **U**, m, p
Output: *ctrlpts*

```
ctrlpts = []
for l = 0 → len(ū) do
    R = zeros([m + 1, m + 1])
    for i = 0 → len(ū[l]) do
        N_{i,p} = BasisFunction(ū[l][i], U[l], len(U[l]), p)        ▷ Algorithm A.6
        rational_function = N_{i,p}/sum(N_{i,p})
        R[i,:] = rational_function[0, m + 1]
    end for
    ctrl = R^{-1}.dot(Q[l])
    for i = 0 in ctrl do
        ctrlpts.append(i)
    end for
end for
```

A.6 Basis function algorithm

Algorithm A.6 Basis Function

Input: u, **U**, num, p
Output: $N_{i,p}$

```
N = zeros([num, p + 1])
for pp = 0 → p + 1 do
    for i = 0 → num − pp − 1 do
        if pp == 0 then
            if u ≥ U[i] and u < U[i + 1] then
                N[i][pp] = 1
            else
                N[i][pp] = 0
            end if
        else
            if N[i][pp − 1] == 0 then
                a = 0
            else
                a = ((u − U[i])/(U[i + pp] − U[i])) ∗ N[i][pp − 1]
            end if
            if N[i + 1][pp − 1] == 0 then
                b = 0
            else
                b = ((U[i + pp + 1] − u)/(U[i + pp + 1] − U[i + 1])) ∗ N[i + 1][pp − 1]
            end if
            N[i][pp] = a + b
        end if
    end for
end for
```

25: $N_{i,p} = N[:, -1]$

A.7 Rational function algorithm

Algorithm A.7 Rational Function

Input: $N_{i,p},\ N_{j,q},\ num_u,\ num_v,\ w = 1$
Output: R
1: $tensor = \textbf{tensordot}(N_{i,p}[0 : num_u + 1],\ N_{j,q}[0 : num_v + 1],\ axes = 0)$
2: $tensor = tensor.\textbf{reshape}((num_u + 1) * (num_v + 1))$
3: $R = tensor/\textbf{sum}(tensor)$

A.8 Get knot vector algorithm

Algorithm A.8 New Knot Vector

Input: $u,\ v,\ \mathbf{U},\ \mathbf{V},\ m,\ n$
Output: $\mathbf{U}_k,\ \mathbf{V}_k$
1: **if** $u == 1$ **and** $v == 1$ **then**
2: $\quad \mathbf{U}_k = \mathbf{U}[-1]$
3: $\quad \mathbf{V}_k = \mathbf{V}[-1]$
4: **else if** $v == 1$ **then**
5: $\quad \mathbf{U}_k = \mathbf{U}[-1]$
6: $\quad v_{idx} = \textbf{int}(u * m)$
7: $\quad \mathbf{V}_k = \mathbf{V}[v_{idx}] * ((v_{idx} + 1)/m - u) * m + \mathbf{V}[v_{idx} + 1] * (u - v_{idx}/m) * m$
8: **else if** $u == 1$ **then**
9: $\quad \mathbf{V}_k = \mathbf{V}[-1]$
10: $\quad u_{idx} = \textbf{int}(v * n)$
11: $\quad \mathbf{U}_k = \mathbf{U}[u_{idx}] * ((u_{idx} + 1)/n - v) * n + \mathbf{U}[u_{idx} + 1] * (v - u_{idx}/n) * n$
12: **else**
13: $\quad v_{idx} = \textbf{int}(u * m)$
14: $\quad u_{idx} = \textbf{int}(v * n)$
15: $\quad \mathbf{V}_k = \mathbf{V}[v_{idx}] * ((v_{idx} + 1)/m - u) * m + \mathbf{V}[v_{idx} + 1] * (u - v_{idx}/m) * m$
16: $\quad \mathbf{U}_k = \mathbf{U}[u_{idx}] * ((u_{idx} + 1)/n - v) * n + \mathbf{U}[u_{idx} + 1] * (v - u_{idx}/n) * n$
17: **end if**

A.9 First fundamental form

Algorithm A.9 First Fundamental Form

Input: $S(u, v),\ u,\ v,\ dudt,\ dvdt$
Output: γ
1: $d\sigma du = S(u, v)_u\ ;\ d\sigma dv = S(u, v)_v$
2: $E = d\sigma du.\textbf{dot}(d\sigma du)$
3: $F = d\sigma du.\textbf{dot}(d\sigma dv)$
4: $G = d\sigma dv.\textbf{dot}(d\sigma dv)$
5: $\gamma = \textbf{sqrt}(E * dudt^2 + 2 * F * dudt * dvdt + G * dvdt^2)$

A.10 Line length on NURBS algorithm

Algorithm A.10 Line Length

Input: $S(u,v)$, u_{start}, u_{end}, v_{start}, v_{end}, m, n, $\mathbf{U}$, $\mathbf{V}$, s
Output: l

1: $c = [4, 2]$
2: $dudt = u_{end} - u_{start}$; $dvdt = v_{end} - v_{start}$
3: $l =$ **FirstFundForm**$(S(u_{start}, v_{start}), u_{start}, v_{start}, dudt, dvdt)$ ▷ Algorithm A.9
4: **for** $i = 1 \rightarrow s$ **do**
5: $u_k = u_{start} + i * dudt/s$; $v_k = v_{start} + i * dvdt/s$
6: $l = l + c[i\%2] *$ **FirstFundForm**$(S(u_k, v_k), u_k, v_k, dudt, dvdt)$
7: **end for**
8: $l =$ **FirstFundForm**$(S(u_{end}, v_{end}), u_{end}, v_{end}, dudt, dvdt)$
9: $l = 1/(3 * s) * l$

Appendix B

BeamConstruct software

The software used to control the laser marking system (rotation of galvanometric scanners, laser beam's power, laser beam's pulse frequency, marking speed, etc) is called BeamConstruct.

Before doing laser marking, the scanning area should be defined first. Since the laser beam is directed by rotating the two galvanometric scanners, whose rotational axes are perpendicular to each other, the sides of scanning area can be obtained by multiplying the distance between the scanners and the target object by the maximum angle the scanners can rotate. In the BeamConstruct software, this value needs to be entered in the settings of "working area". For example, if the scanning area is a 100mm × 100mm square, the "Field left Position" should be filled in by −50mm, the "Field upper position" should be filled in by 50mm, and both the "Field width" and the "Field height" should be filled in by 100mm (see Figure B.1).

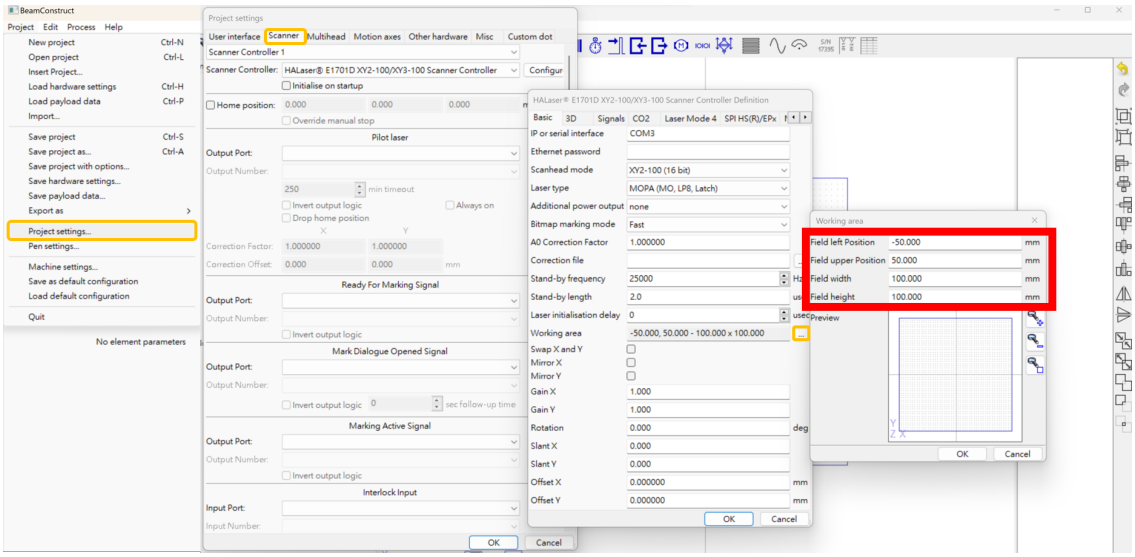


Figure B.1: "Working area" settings in the BeamConstruct software.

Then, the "Editing area" in the software should be fitted in the size of the working area to fully exploit the scanning area. See Figure B.2, the first two values of "Editing area upper left" would be −50mm and 50mm and the third value should be zero. For the "Editing area size", both the

first two values would be 100mm and the default value given by the software of the third one can remain the same.

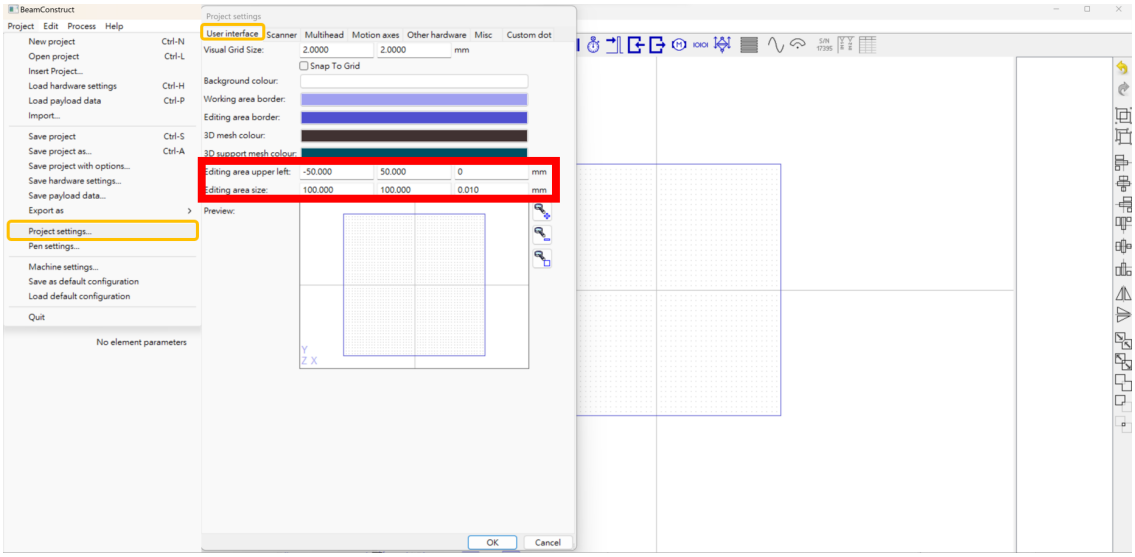


Figure B.2: "Editing area" settings in the BeamConstruct software.

After that, the pattern that wants to be marked can be imported into the software (see Figure B.3, noted that the pattern should be a vector-type file such as a dxf file, and the information in this vector-type file would be the signal to control the galvanometric scanner).

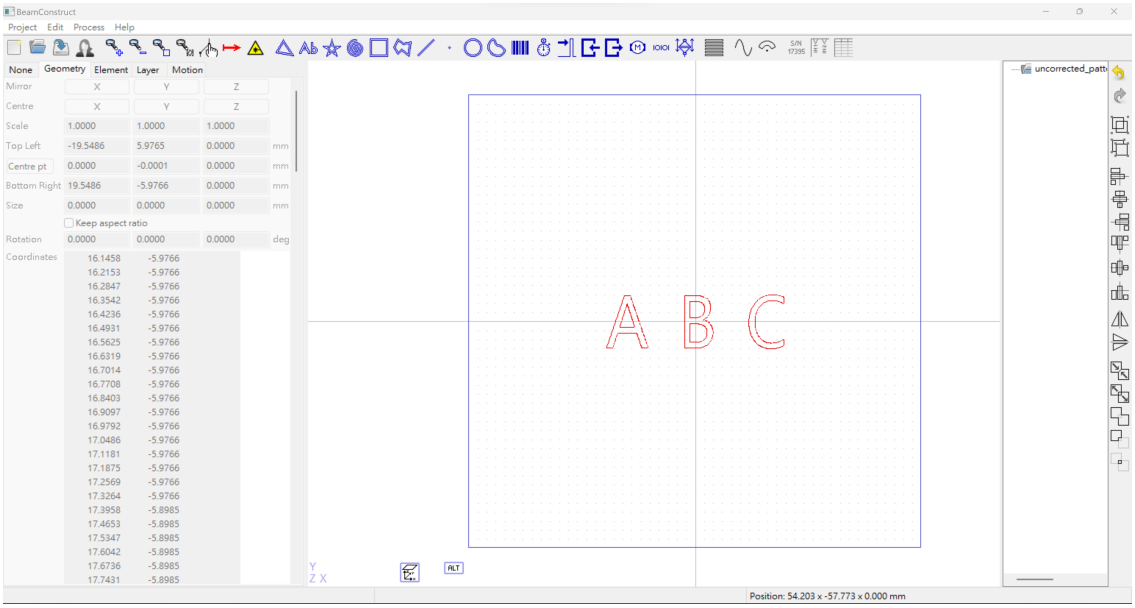


Figure B.3: The pattern that wants to be marked is imported to the BeamConstruct software.

By specifying the laser's configuration (e.g. power, frequency, marking speed, etc., see Figure B.4, the optimal configuration is the other project of inPhocal [19].), the pattern can be marked onto the target surface.

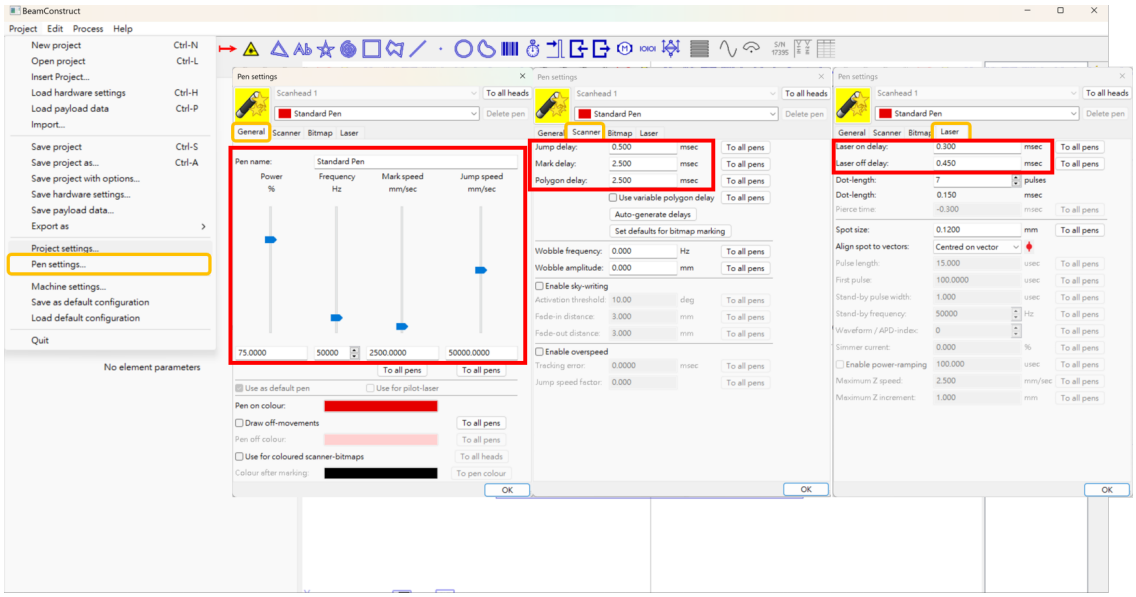


Figure B.4: Laser configuration settings in the BeamConstruct software.

Appendix C

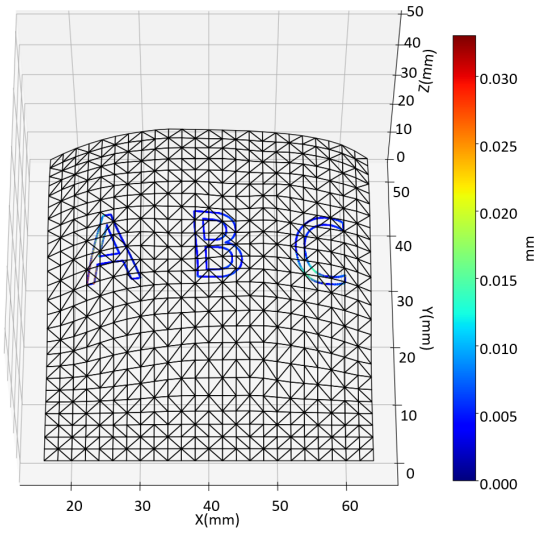
Simulation results of effects of subintervals number in Simpson's Rule on correction error

Table C.1 shows the effects of subintervals number in Simpson's Rule on the time consumption and correction error.

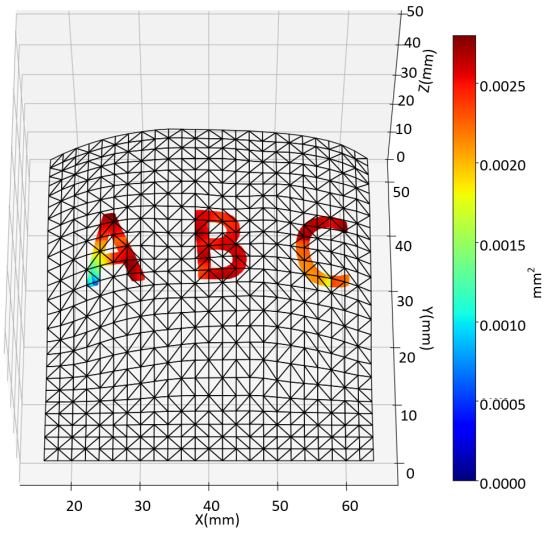
Table C.1: Time consumption, total length difference e_l and total area error e_a by using different subintervals in Simpson's rule when applying the NURBS- based shape correction method on the mandarin's surface.

Number of subintervals	time consumption (sec)	e_l(mm)	e_a(mm^2)
10	5.05	11.58	10.25
20	6.49	6.49	4.12
30	4.93	4.93	2.31
40	4.21	4.21	1.48
50	3.81	3.81	1.05
60	3.57	3.57	0.83
70	3.41	3.41	0.7
80	3.31	3.31	0.483
90	3.24	3.24	0.48
100	3.21	3.21	0.48

From Figure C.1 to Figure C.10, the simulation of the mapping pattern onto the mandarin with 10×10 points to create the mapping domain and different numbers of subintervals in Simpson's rule are shown.

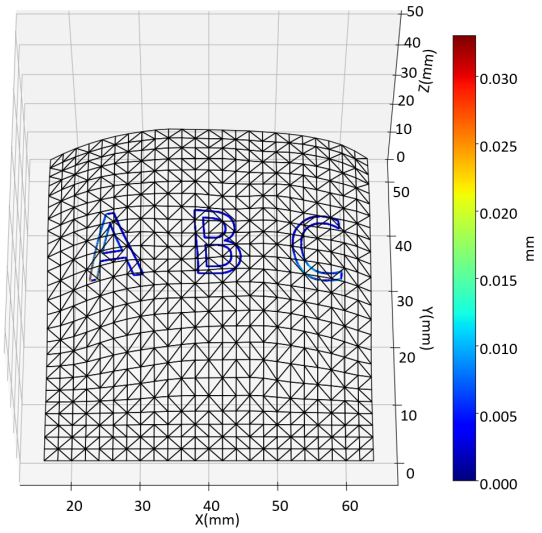


(a) Visualization of the length difference with 10×10 points to create mapping domain and 10 subintervals in Simpson's rule, $e_l = 11.58$mm.

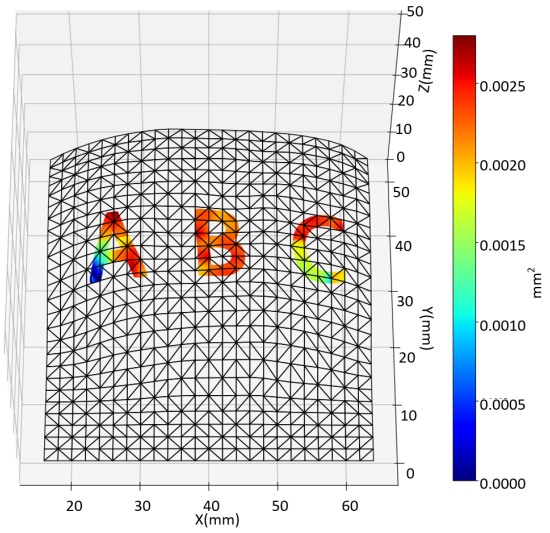


(b) Visualization of the area difference with 10×10 points to create mapping domain and 10 subintervals in Simpson's rule, $e_a = 10.25\text{mm}^2$.

Figure C.1: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 10 subintervals in Simpson's rule.

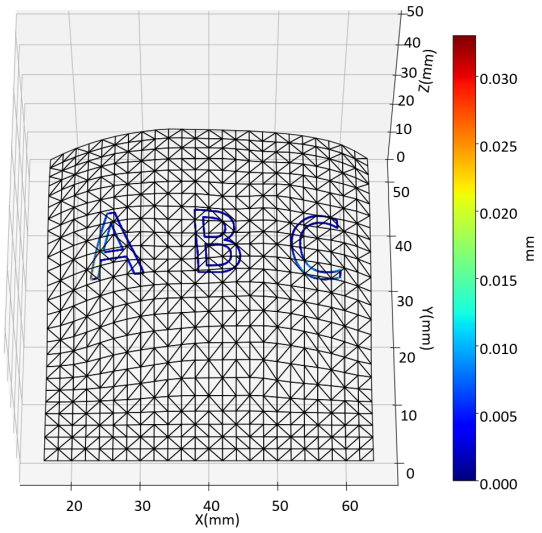


(a) Visualization of the length difference with 10×10 points to create mapping domain and 20 subintervals in Simpson's rule, $e_l = 6.49$mm.

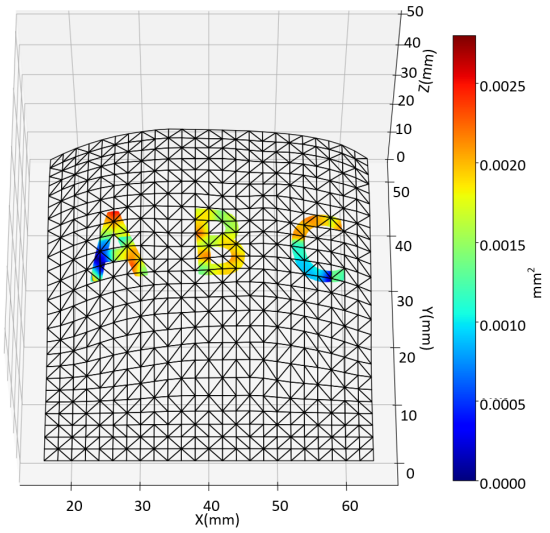


(b) Visualization of the area difference with 10×10 points to create mapping domain and 20 subintervals in Simpson's rule, $e_a = 4.12\text{mm}^2$.

Figure C.2: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 20 subintervals in Simpson's rule.

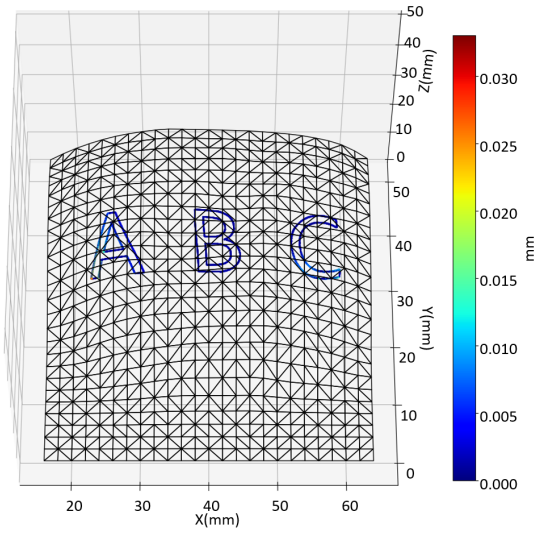


(a) Visualization of the length difference with 10×10 points to create mapping domain and 30 subintervals in Simpson's rule, $e_l = 4.93$mm.

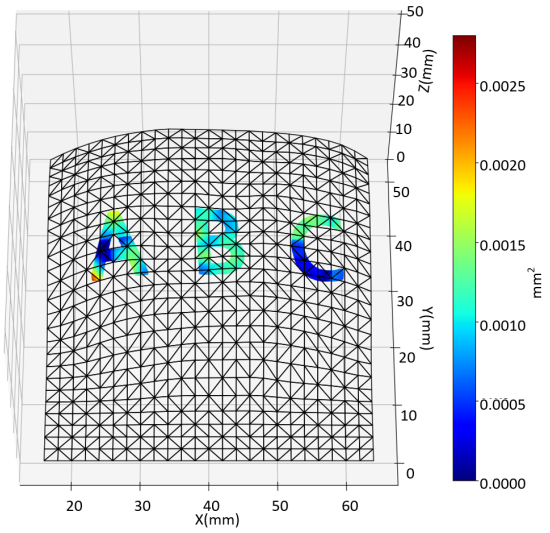


(b) Visualization of the area difference with 10×10 points to create mapping domain and 30 subintervals in Simpson's rule, $e_a = 2.31\text{mm}^2$.

Figure C.3: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 30 subintervals in Simpson's rule.

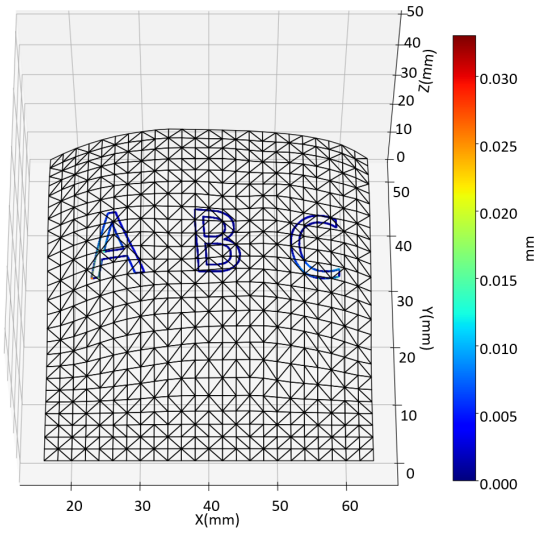


(a) Visualization of the length difference with 10×10 points to create mapping domain and 40 subintervals in Simpson's rule, $e_l = 4.21$mm.

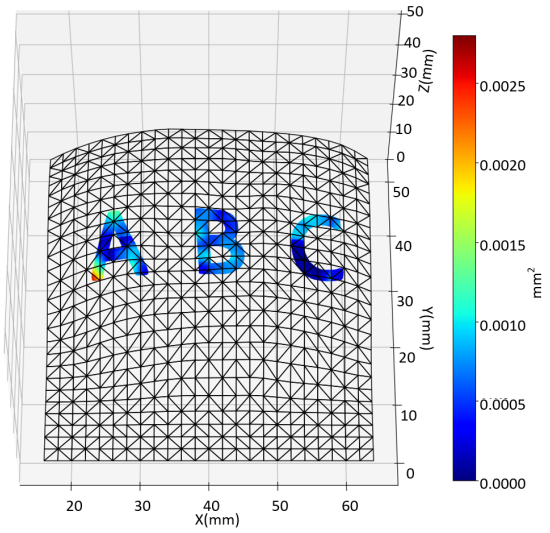


(b) Visualization of the area difference with 10×10 points to create mapping domain and 40 subintervals in Simpson's rule, $e_a = 1.48\text{mm}^2$.

Figure C.4: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 40 subintervals in Simpson's rule.

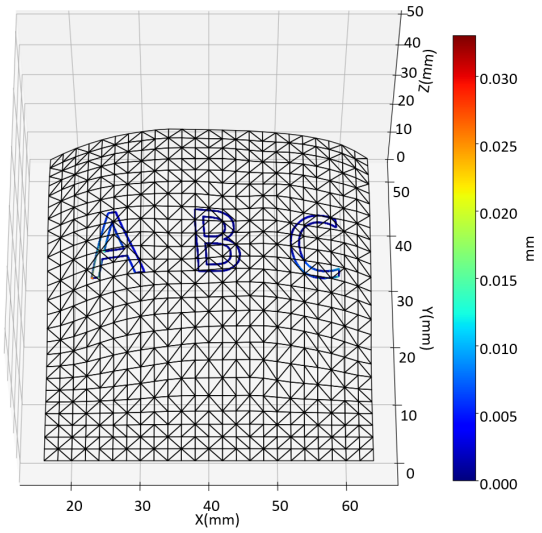


(a) Visualization of the length difference with 10×10 points to create mapping domain and 50 subintervals in Simpson's rule, $e_l = 3.81$mm.

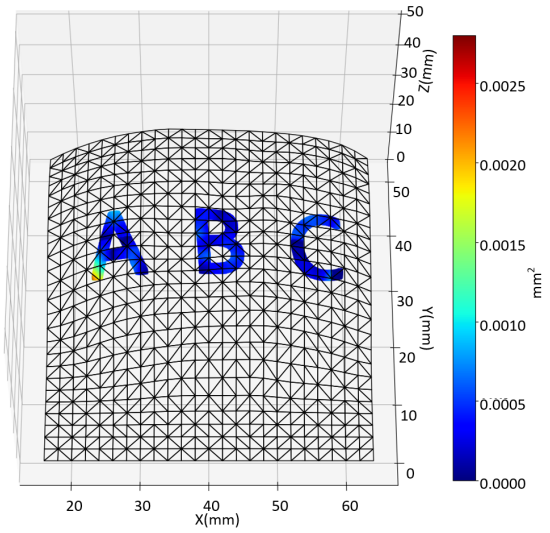


(b) Visualization of the area difference with 10×10 points to create mapping domain and 50 subintervals in Simpson's rule, $e_a = 1.05\text{mm}^2$.

Figure C.5: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 50 subintervals in Simpson's rule.

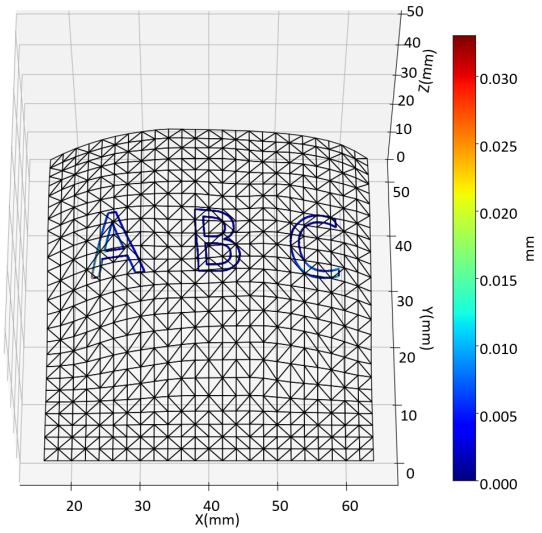


(a) Visualization of the length difference with 10×10 points to create mapping domain and 60 subintervals in Simpson's rule, $e_l = 3.57$mm.

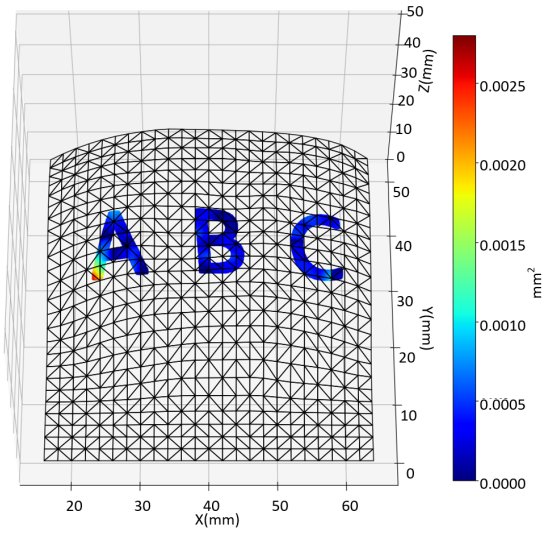


(b) Visualization of the area difference with 10×10 points to create mapping domain and 60 subintervals in Simpson's rule, $e_a = 0.83\text{mm}^2$.

Figure C.6: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 60 subintervals in Simpson's rule.

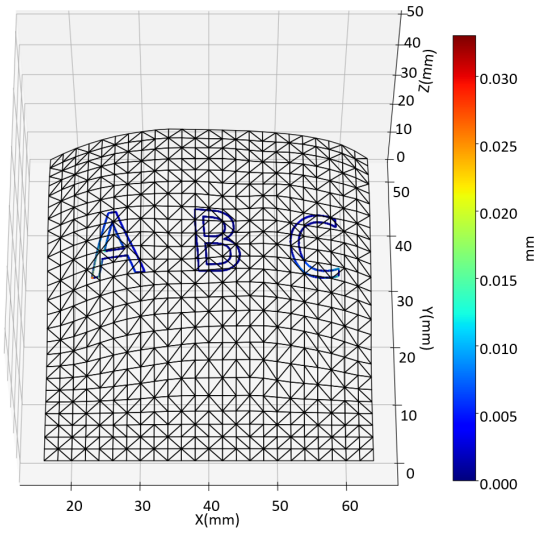


(a) Visualization of the length difference with 10×10 points to create mapping domain and 70 subintervals in Simpson's rule, $e_l = 3.41$mm.

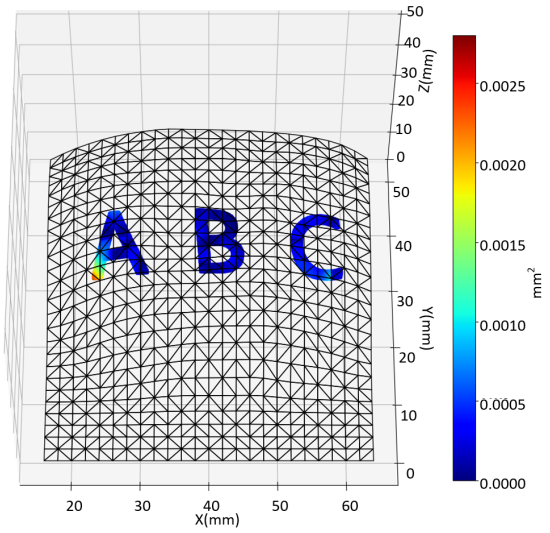


(b) Visualization of the area difference with 10×10 points to create mapping domain and 70 subintervals in Simpson's rule, $e_a = 0.7\text{mm}^2$.

Figure C.7: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 70 subintervals in Simpson's rule.

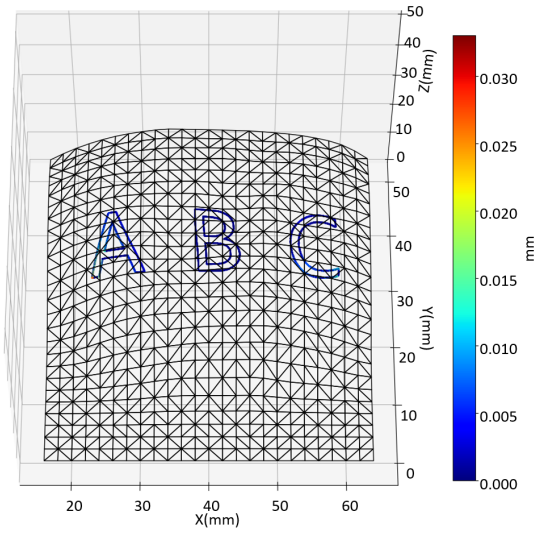


(a) Visualization of the length difference with 10×10 points to create mapping domain and 80 subintervals in Simpson's rule, $e_l = 3.31$mm.

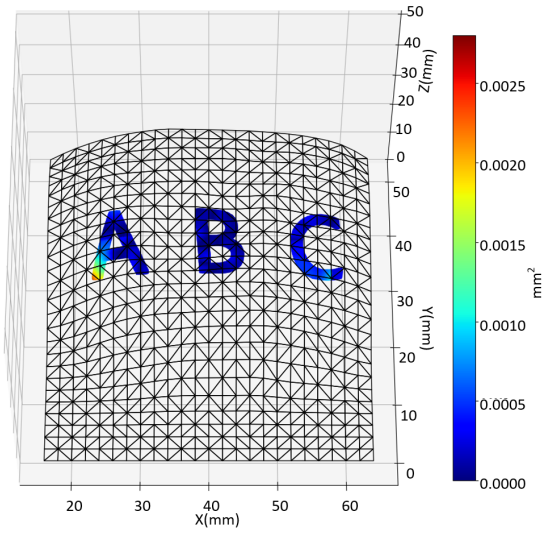


(b) Visualization of the area difference with 10×10 points to create mapping domain and 80 subintervals in Simpson's rule, $e_a = 0.483\text{mm}^2$.

Figure C.8: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 80 subintervals in Simpson's rule.

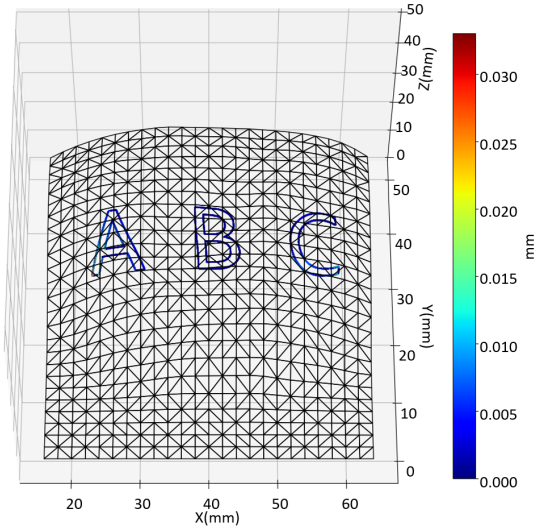


(a) Visualization of the length difference with 10×10 points to create mapping domain and 90 subintervals in Simpson's rule, $e_l = 3.24$mm.

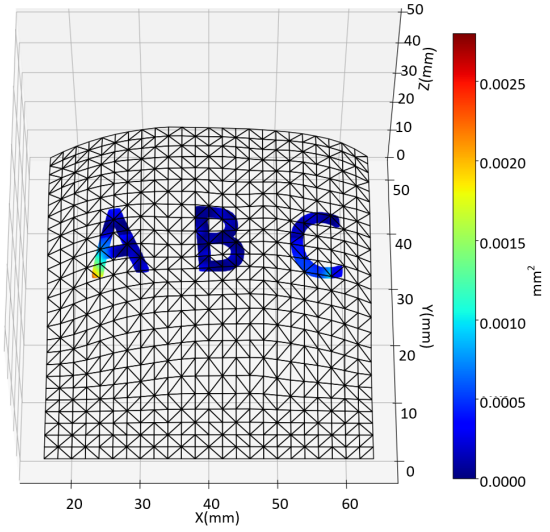


(b) Visualization of the area difference with 10×10 points to create mapping domain and 90 subintervals in Simpson's rule, $e_a = 0.48\text{mm}^2$.

Figure C.9: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 90 subintervals in Simpson's rule.



(a) Visualization of the length difference with 10×10 points to create mapping domain and 100 subintervals in Simpson's rule, $e_l = 3.21$mm.



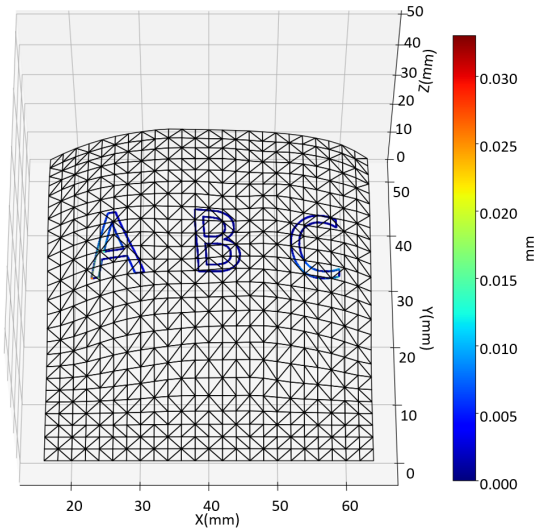
(b) Visualization of the area difference with 10×10 points to create mapping domain and 100 subintervals in Simpson's rule, $e_a = 0.48\text{mm}^2$.

Figure C.10: Simulation of mapping pattern onto the mandarin's with 10×10 points to create mapping domain and 100 subintervals in Simpson's rule.

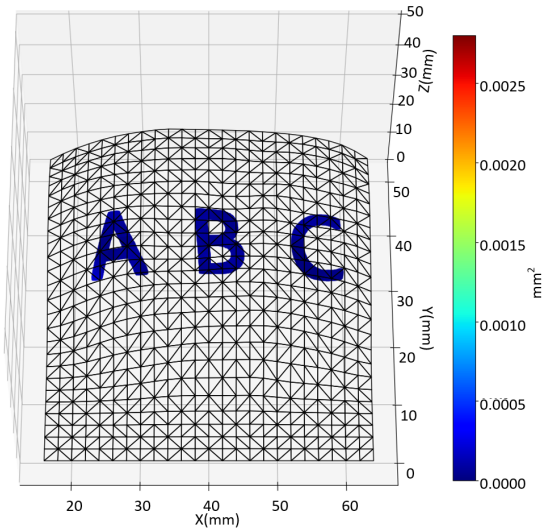
Appendix D

Simulation results of effects of mapping domain points number on correction error

Figure D.1 to Figure D.10 show the effects of number of points to create the mapping domain on the time consumption and correction error. The pattern is a picture file of a string with 572pixel × 192pixe. This pattern is scaled into 39mm × 13mm and mapped onto the mandarin's surface.

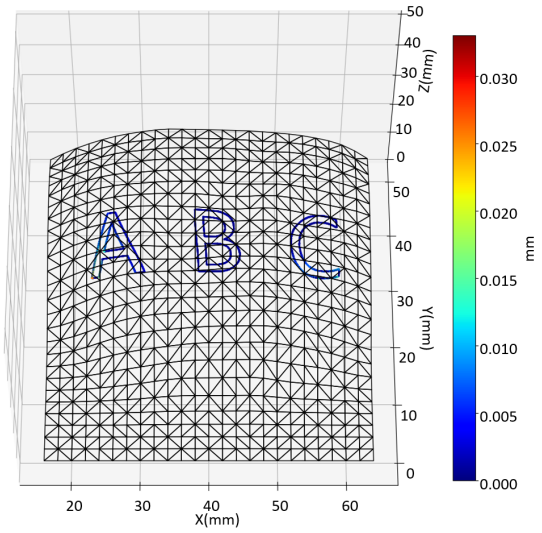


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 10 × 10 points to create mapping domain and, e_l = 3.81mm.

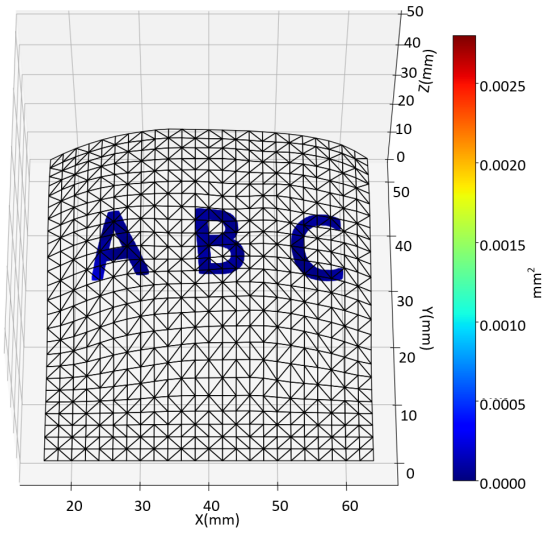


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 10 × 10 points to create mapping domain and, e_a = 1.05mm^2.

Figure D.1: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 10 × 10 points to create mapping domain.

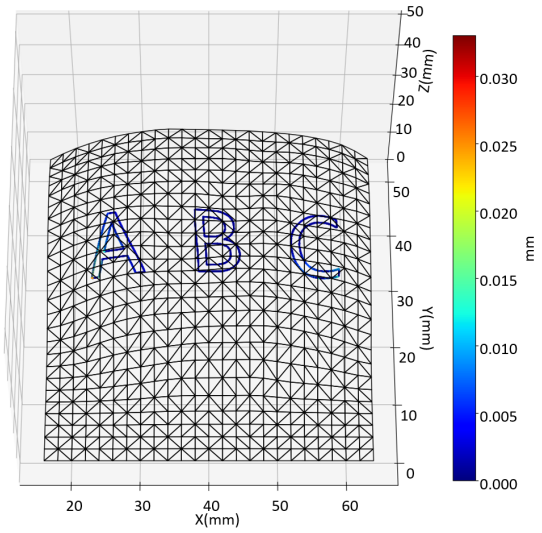


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 20×20 points to create mapping domain and, $e_l = 3.37$mm.

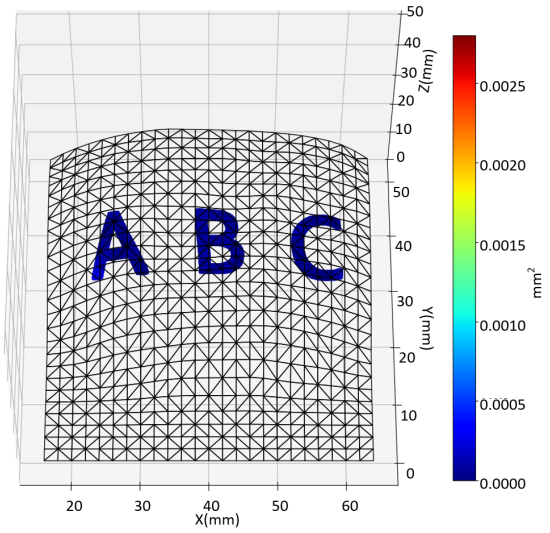


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 20×20 points to create mapping domain and, $e_a = 1.02\text{mm}^2$.

Figure D.2: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 20×20 points to create mapping domain.

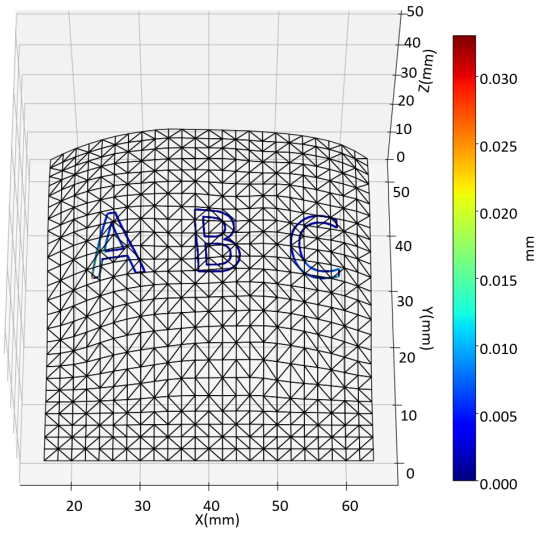


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 30×30 points to create mapping domain and, $e_l = 3.29$mm.

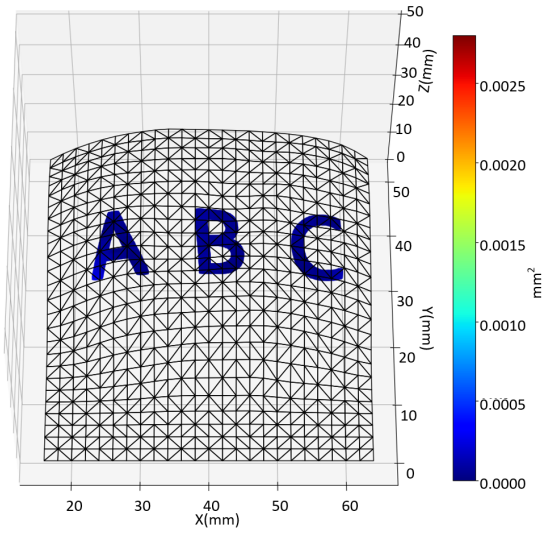


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 30×30 points to create mapping domain and, $e_a = 0.99\text{mm}^2$.

Figure D.3: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 30×30 points to create mapping domain.

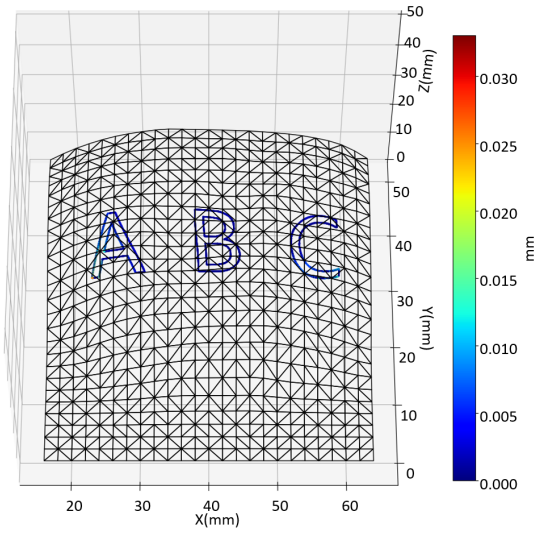


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 40×40 points to create mapping domain and, $e_l = 3.21$mm.

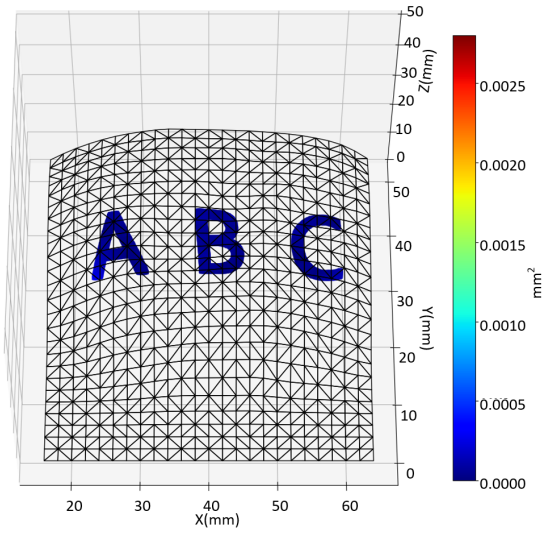


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 40×40 points to create mapping domain and, $e_a = 0.99\text{mm}^2$.

Figure D.4: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 40×40 points to create mapping domain.

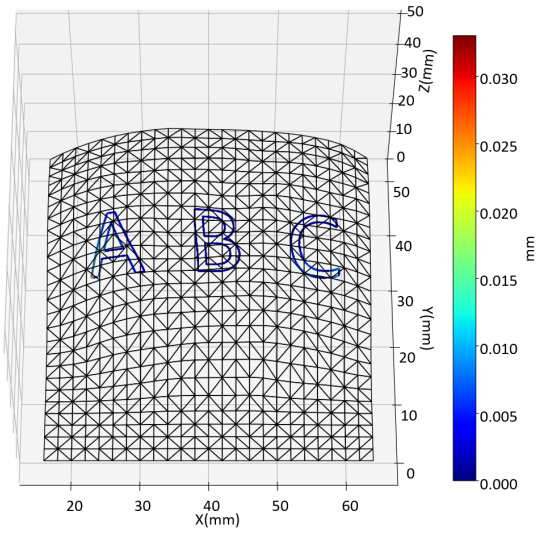


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 50×50 points to create mapping domain and, $e_l = 3.21$mm.

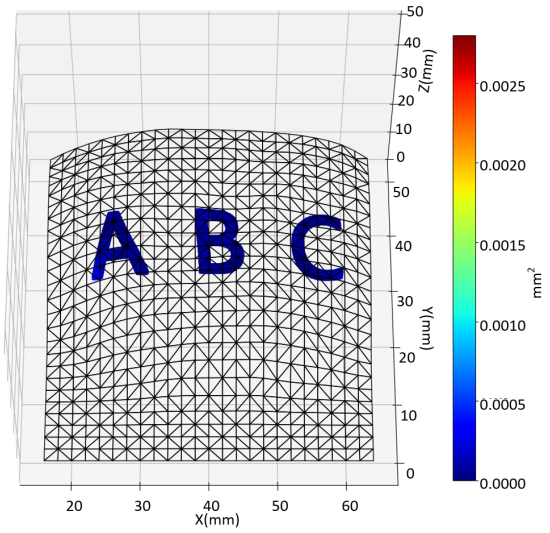


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 50×50 points to create mapping domain and, $e_a = 1.01\text{mm}^2$.

Figure D.5: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 50×50 points to create mapping domain.

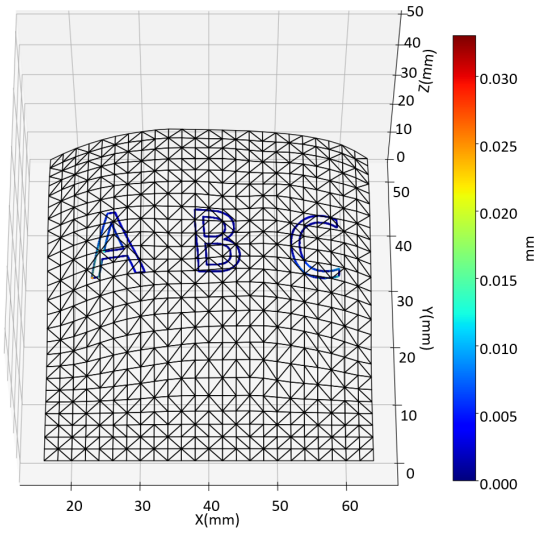


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 60×60 points to create mapping domain and, $e_l = 3.21$mm.

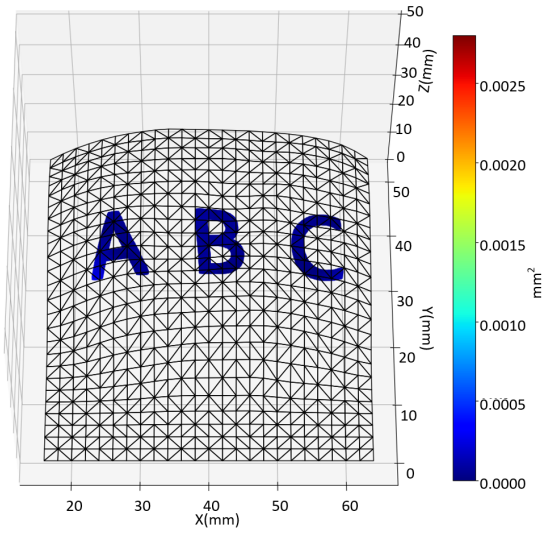


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 60×60 points to create mapping domain and, $e_a = 1.00\text{mm}^2$.

Figure D.6: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 60×60 points to create mapping domain.

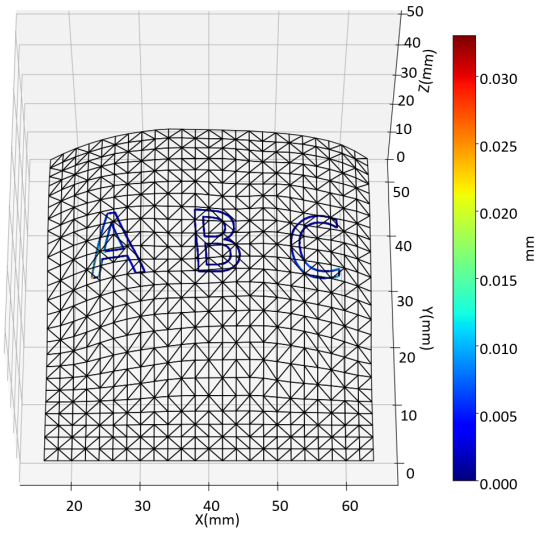


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 70×70 points to create mapping domain and, $e_l = 3.2$mm.

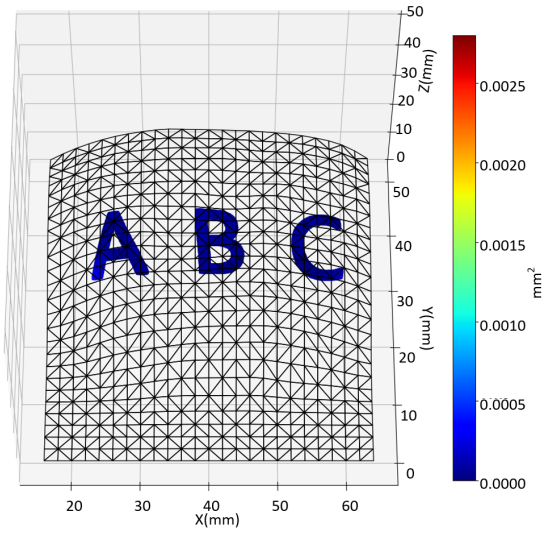


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 70×70 points to create mapping domain and, $e_a = 1.00\text{mm}^2$.

Figure D.7: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 70×70 points to create mapping domain.

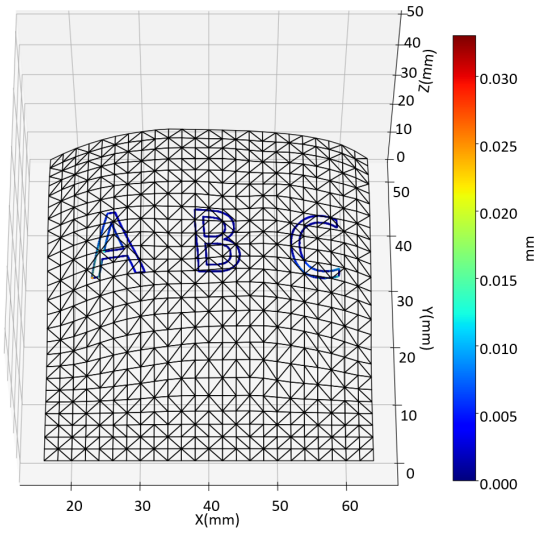


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 80×80 points to create mapping domain and, $e_l = 3.19$mm.

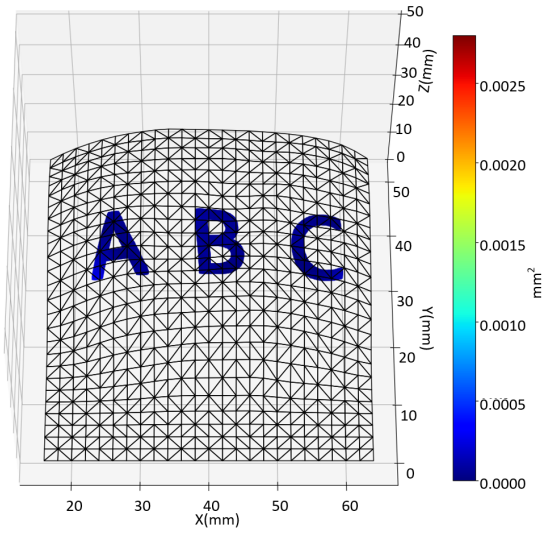


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 80×80 points to create mapping domain and, $e_a = 1.00\text{mm}^2$.

Figure D.8: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 80×80 points to create mapping domain.

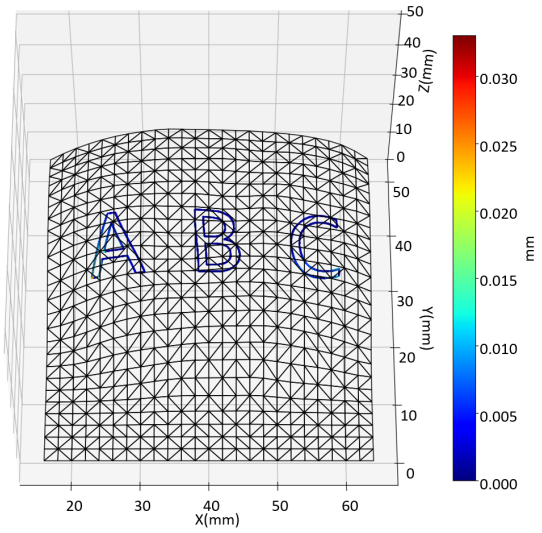


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 90×90 points to create mapping domain and, $e_l = 3.19$mm.

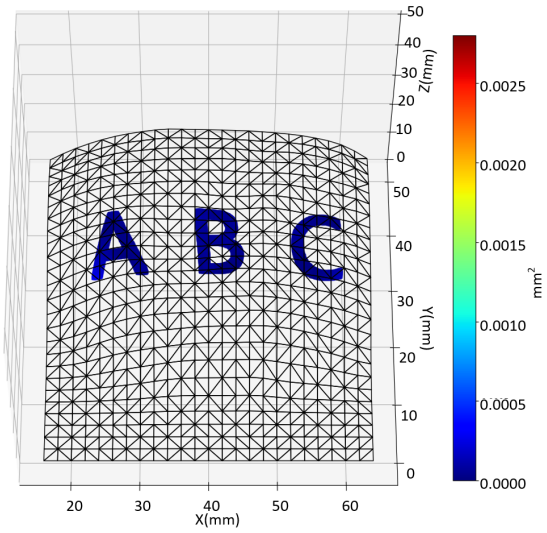


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 90×90 points to create mapping domain and, $e_a = 1.00\text{mm}^2$.

Figure D.9: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 90×90 points to create mapping domain.



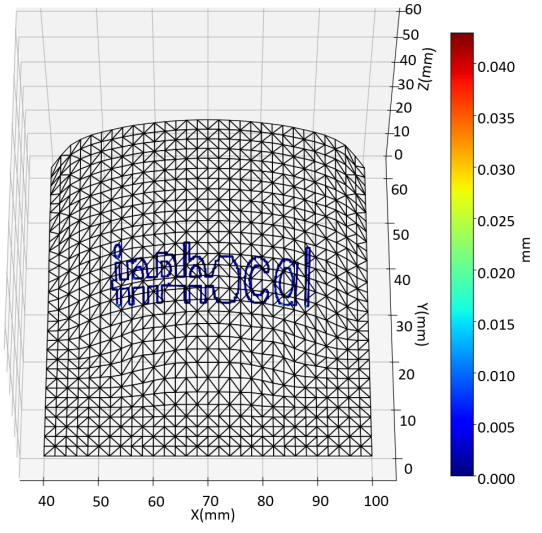
(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 100×100 points to create mapping domain and, e_l = 3.19mm.



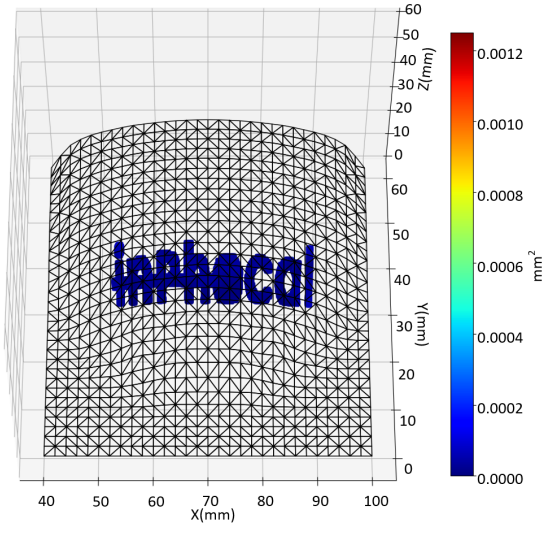
(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 100×100 points to create mapping domain and, e_a = 1.00mm^2.

Figure D.10: Simulation of mapping pattern onto the mandarin's with 50 subintervals in Simpson's rule and 100×100 points to create mapping domain.

Figure D.1 to Figure D.10 show the effects of number of points to create the mapping domain on the time consumption and correction error. The pattern is a picture file of nPhocal's logo with 1200pixel × 200pixe. This pattern is scaled into 39mm × 13mm and mapped onto the mouse's surface.

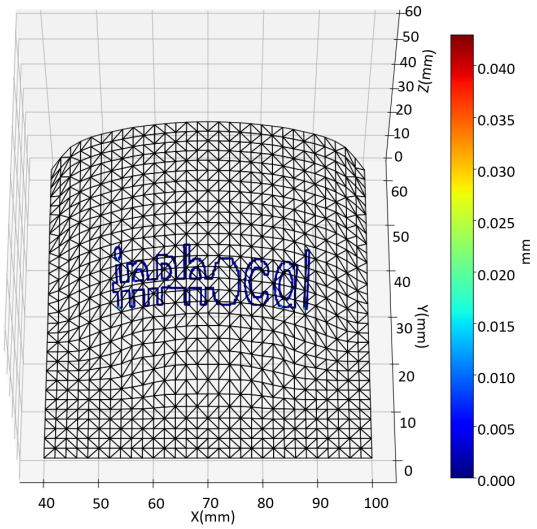


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 10×10 points to create mapping domain and, e_l = 6.32mm.

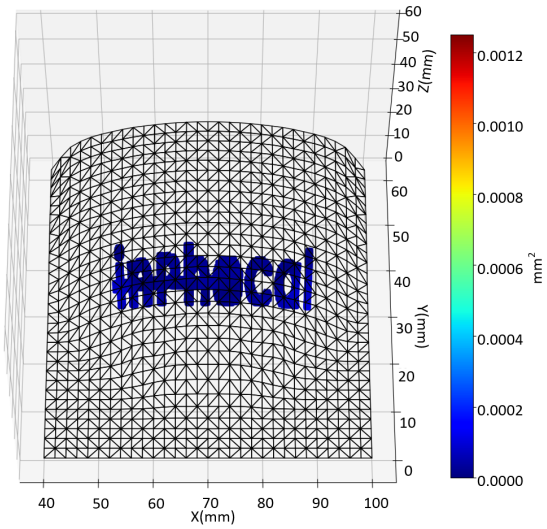


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 10×10 points to create mapping domain and, e_a = 1.86mm^2.

Figure D.11: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 10×10 points to create mapping domain.

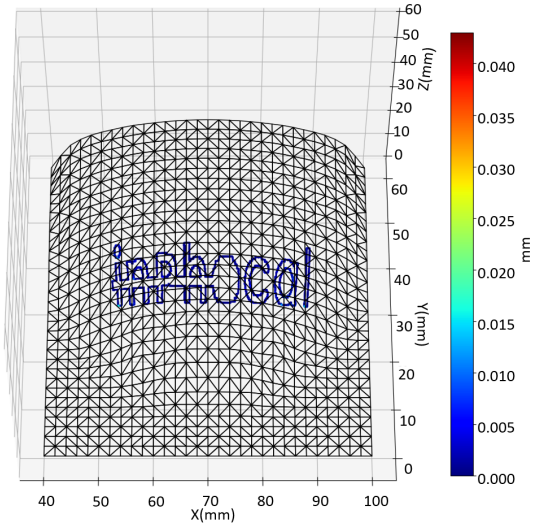


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 20×20 points to create mapping domain and, $e_l = 5.84$mm.

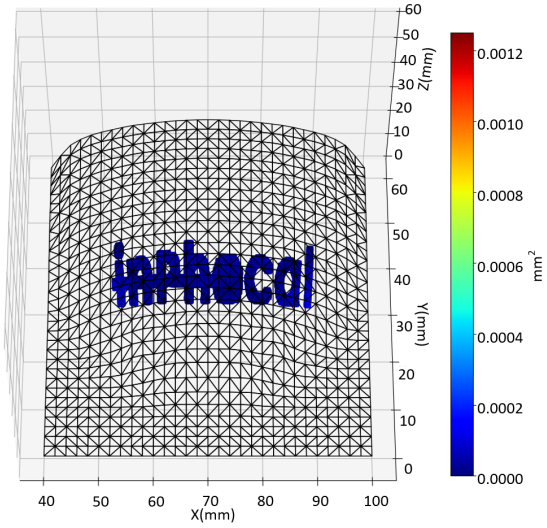


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 20×20 points to create mapping domain and, $e_a = 2.4\text{mm}^2$.

Figure D.12: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 20×20 points to create mapping domain.

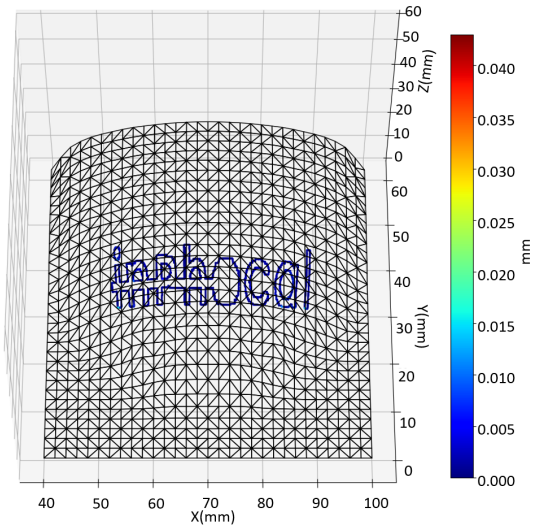


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 30×30 points to create mapping domain and, $e_l = 5.42$mm.

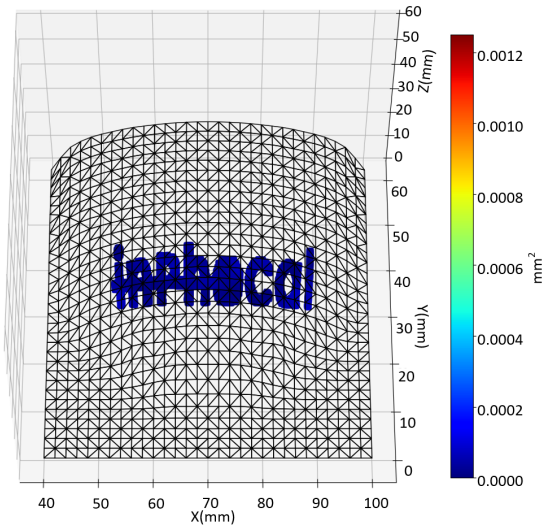


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 30×30 points to create mapping domain and, $e_a = 2.17\text{mm}^2$.

Figure D.13: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 30×30 points to create mapping domain.

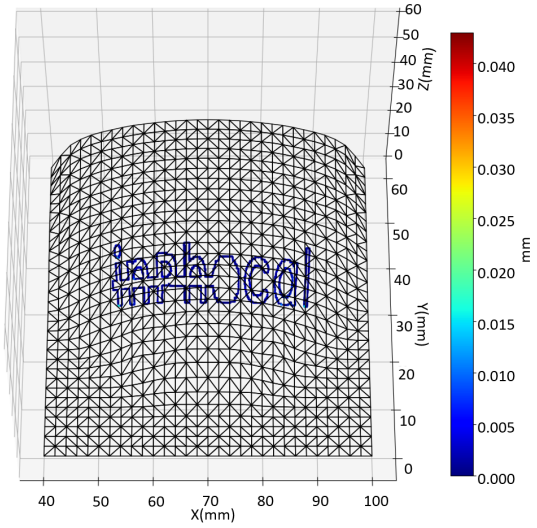


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 40×40 points to create mapping domain and, $e_l = 5.51$mm.

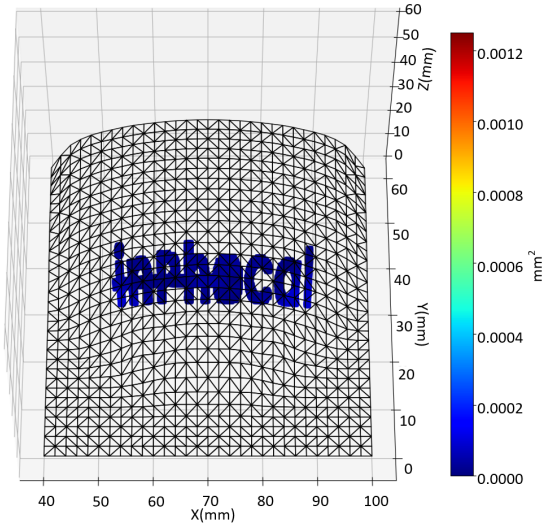


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 40×40 points to create mapping domain and, $e_a = 2.34\text{mm}^2$.

Figure D.14: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 40×40 points to create mapping domain.

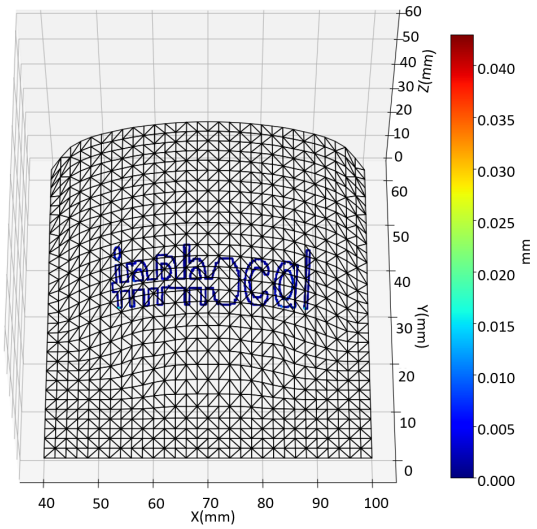


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 50×50 points to create mapping domain and, $e_l = 5.29$mm.

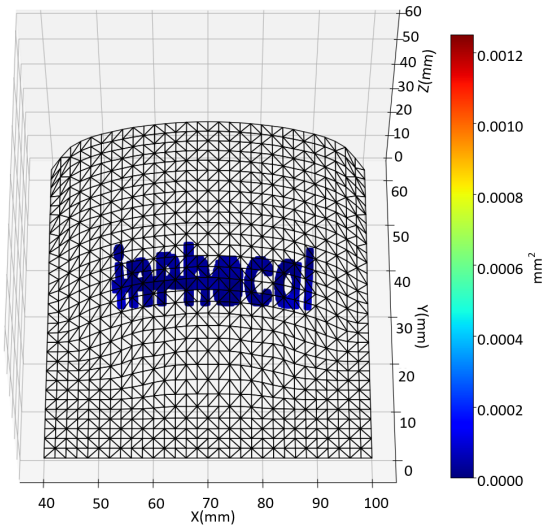


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 50×50 points to create mapping domain and, $e_a = 2.18\text{mm}^2$.

Figure D.15: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 50×50 points to create mapping domain.

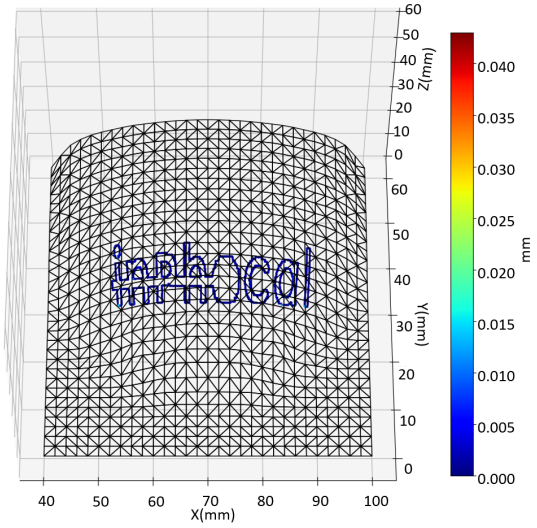


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 60×60 points to create mapping domain and, $e_l = 5.29$mm.

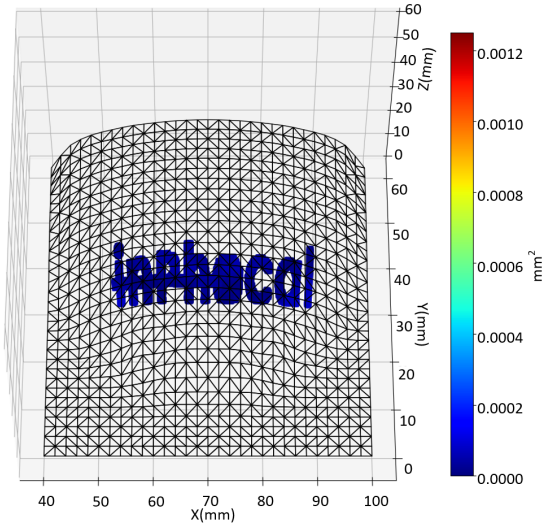


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 60×60 points to create mapping domain and, $e_a = 2.18\text{mm}^2$.

Figure D.16: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 60×60 points to create mapping domain.

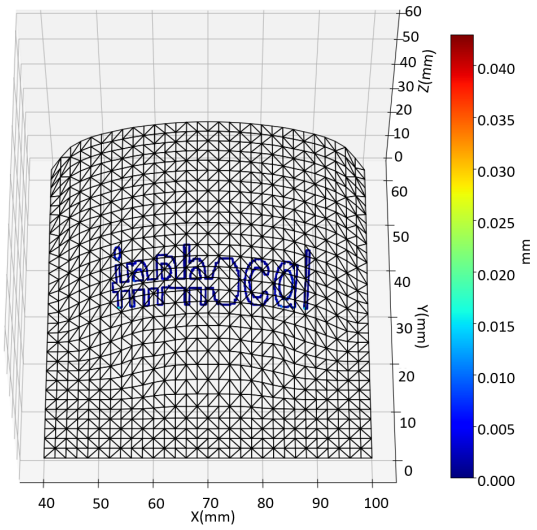


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 70×70 points to create mapping domain and, $e_l = 5.4$mm.

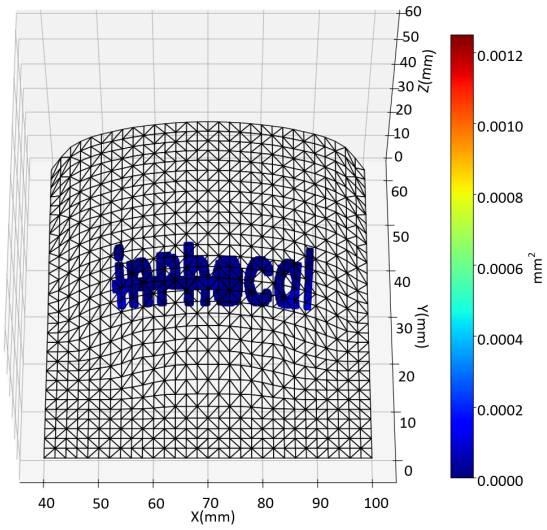


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 70×70 points to create mapping domain and, $e_a = 2.25\text{mm}^2$.

Figure D.17: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 70×70 points to create mapping domain.

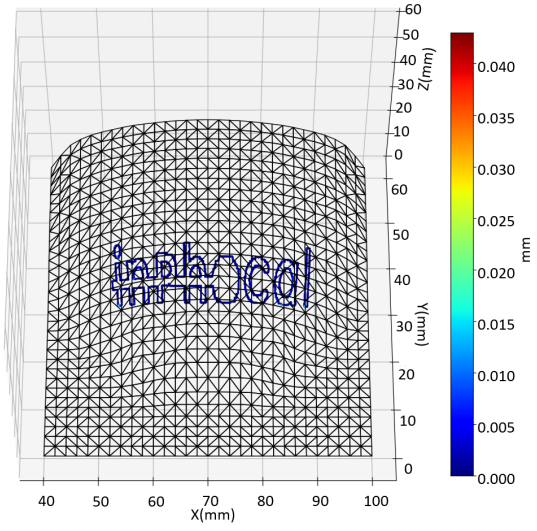


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 80×80 points to create mapping domain and, $e_l = 5.2$mm.

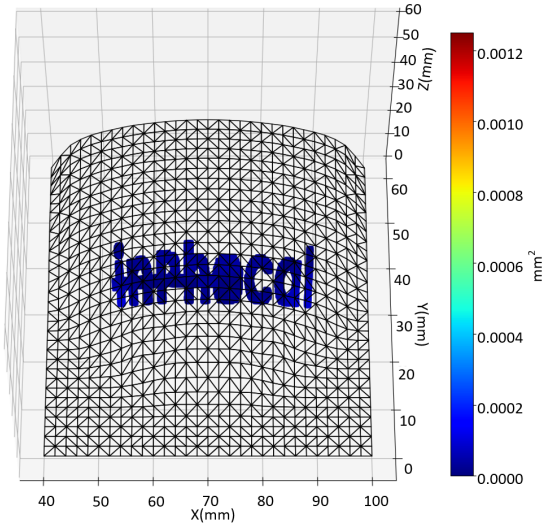


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 80×80 points to create mapping domain and, $e_a = 2.12\text{mm}^2$.

Figure D.18: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 80×80 points to create mapping domain.

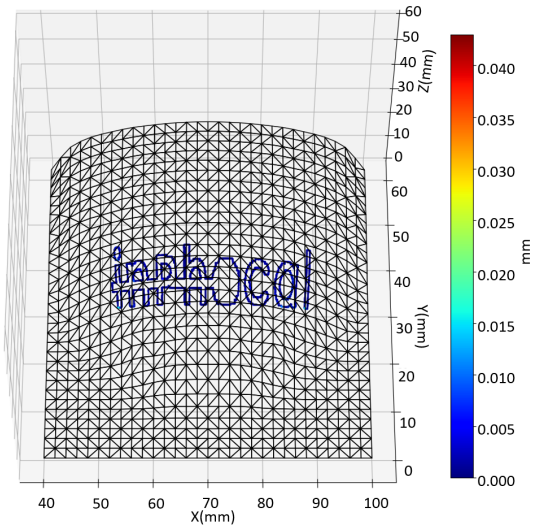


(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 90×90 points to create mapping domain and, $e_l = 5.25$mm.

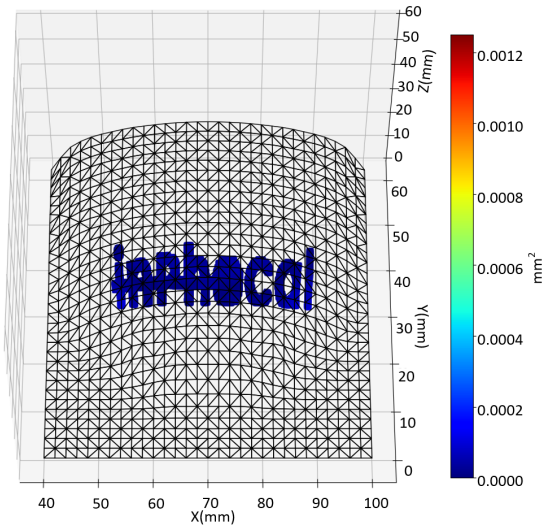


(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 90×90 points to create mapping domain and, $e_a = 2.13\text{mm}^2$.

Figure D.19: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 90×90 points to create mapping domain.



(a) Visualization of the length difference with 50 subintervals in Simpson's rule and 100×100 points to create mapping domain and, $e_l = 5.29$mm.



(b) Visualization of the length difference with 50 subintervals in Simpson's rule and 100×100 points to create mapping domain and, $e_a = 2.22\text{mm}^2$.

Figure D.20: Simulation of mapping pattern onto the mouse's with 50 subintervals in Simpson's rule and 100×100 points to create mapping domain.

Appendix E

Inspecting shift of the same point on original and orthogonally projected pattern

The process of a straightforward method to do shape correction is shown in Figure E.1. Since this method uses curvature of the curved surfaces directly to specify the workspace and correct the pattern, the correction error could be zero. Due to the same reason, however, it can only be used on surfaces with simple shapes.

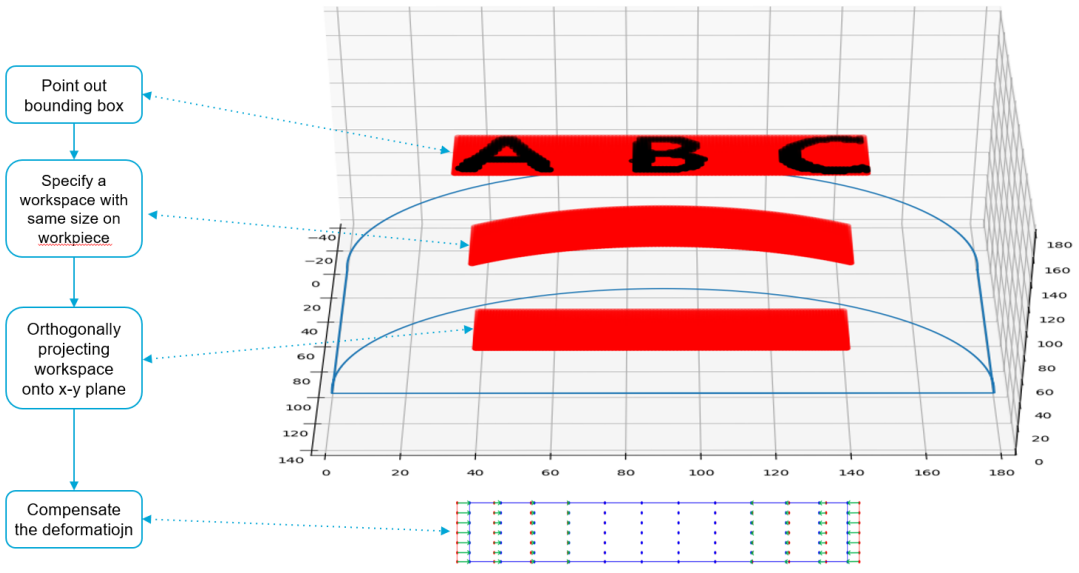
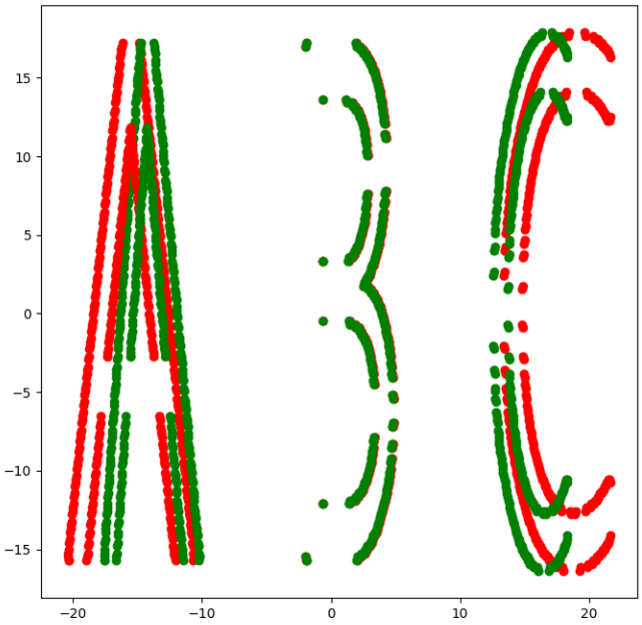


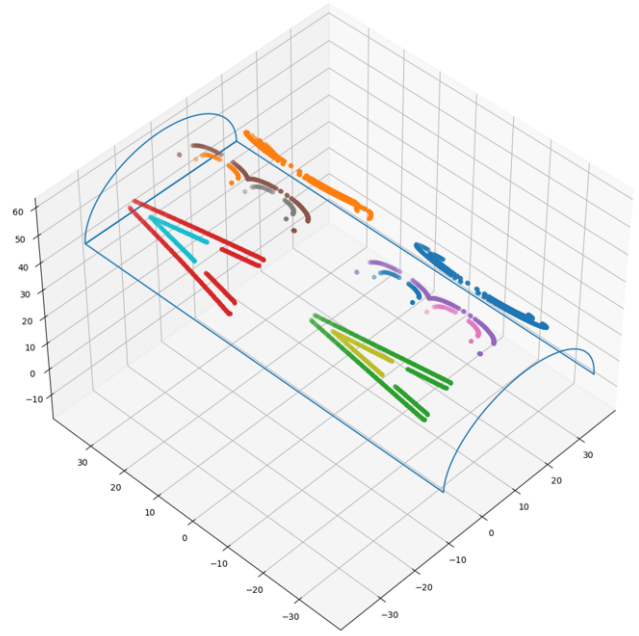
Figure E.1: Process flow of inspecting shift of the same point on original and orthogonally projected pattern

Figure E.2 shows the simulation result of shape correction by inspecting the shift of the same point on the original and orthogonally projected pattern. In Figure E.2b it can be found that the widths of the two sides of character 'A' are not the same in the uncorrected pattern (the upper

left one) because of the curvature of the cylindrical surface, but they are the same in the corrected pattern (the bottom right one).



(a) Corrected pattern obtained from inspecting shift of the same point on the original and the orthogonally projected pattern, the corrected pattern is shown in green dots and the uncorrected pattern is shown in red dots.



(b) Simulation result of mapping corrected (the bottom right one) and uncorrected (the upper left one) pattern on a cylindrical surface.

Figure E.2: Simulation of shape correction through inspecting shift of the same point on original and orthogonally projected pattern

Bibliography

[1] Wikipedia contributors. Laser — Wikipedia, the free encyclopedia, 2023. URL `https://en.wikipedia.org/w/index.php?title=Laser&oldid=1138345238`. [Online; accessed 20-February-2023].

[2] M. Šulc. An optical system for producing a structured beam, Wold Intellectual Property Organisation. Nov. 2019.

[3] G. Chen, L.S. Zhou, L.L. An, and K. Zhang. A novel surface flattening method based on mesh edges. In *2012 Second International Conference on Intelligent System Design and Engineering Application*, pages 129–133. IEEE, 2012.

[4] B. Lévy, S. Petitjean, N. Ray, and volume=21 number=3 pages=362–371 year=2002 publisher=ACM New York, NY, USA J. Maillot, journal=ACM transactions on graphics (TOG). Least squares conformal maps for automatic texture atlas generation.

[5] W.T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1): 743–767, 1963.

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

[7] J. Guo, G. Liu, G. Zhang, Y. Guan, and H. Zhang. Projection algorithm for 3d laser marking. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2199–2204. IEEE, 2015.

[8] Wikipedia contributors. Bessel beam — Wikipedia, the free encyclopedia, 2022. URL `https://en.wikipedia.org/w/index.php?title=Bessel_beam&oldid=1087417680`. [Online; accessed 21-February-2023].

[9] M. Kwaaitaal. Structured laser beams for laser marking. Delft University of Technology internship report, Nov. 2020.

[10] T. Westerveld. Concentric laser beam characterization for laser marking. Delft University of Technology internship report, Apr. 2021.

[11] F.A. van den Boom. A novel concentricanna van den boommaarten kwaaitaal. Fontys University of Applied Sciences project report, Jun. 2021.

[12] Les A. Pieg and W. Tiller. *Rational B-spline Curves and Surfaces*, pages 117–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. ISBN 978-3-642-59223-2. doi: 10.1007/978-3-642-59223-2_4. URL `https://doi.org/10.1007/978-3-642-59223-2_4`.

[13] A. Pressley. *The first fundamental form*, pages 121–157. Springer London, London, 2010. ISBN 978-1-84882-891-9. doi: 10.1007/978-1-84882-891-9_6. URL `https://doi.org/10.1007/978-1-84882-891-9_6`.

[14] J. Procházková and D. Procházka. Implementation of nurbs curve derivatives in engineering practice. 2007.

[15] Wikipedia contributors. Simpson's rule — Wikipedia, the free encyclopedia, 2023. URL https://en.wikipedia.org/w/index.php?title=Simpson%27s_rule&oldid=1137854162. [Online; accessed 25-February-2023].

[16] W. Wu, Y. Hu, and Y. Lu. Parametric surface modelling for tea leaf point cloud based on non-uniform rational basis spline technique. *Sensors*, 21(4):1304, 2021.

[17] B.K. Hinds, J. McCartney, and G. Woods. Pattern development for 3d surfaces. *Computer-Aided Design*, 23(8):583–592, 1991. ISSN 0010-4485. doi: https://doi.org/10.1016/0010-4485(91)90060-A. URL https://www.sciencedirect.com/science/article/pii/001044859190060A.

[18] G. Bradshaw. Non-contact surface geometry measurement techniques. Technical report, Trinity College Dublin, Department of Computer Science, 1999.

[19] S.B.J. Hankel. Characterization of an optical system with a longer focus depth for laser marking. Fontys University of Applied Sciences project report, Jun. 2023.