Shape Blending Using Discrete Curvature-Variation Functional

By

Guo, Li

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirement for
the Degree of Master of Philosophy
in the Department of Industrial Engineering and Engineering Management

July 2005, Hong Kong

Copyright by Hong Kong University of
Science & Technology and the author.
DISTRIBUTION PROHIBITED.
Authorization

I hereby declare that I am the sole author of this thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Guo Li
To my parents, my friends
Acknowledgements

I would like to give my sincere thanks to my supervisor, Dr. Ajay Joneja, for his patience and guidance in my postgraduate study. His sound knowledge on this field not only broadens my view in academic area but also strengthen my way of thinking. I appreciate his wide and profound in his knowledge.

Special thanks are given to my thesis examination committee, Dr. Kai Tang and Dr. Jeff Hong. They give many valuable comments on my work and made my thesis more complete.

Sincere thanks are given to my parents for their support on my study.

Thanks all people who kindly help me in my study and my life.
Shape Blending Using Discrete Curvature-Variation Functional

By

Guo, Li

This is to certify that I have examined the above Mphil thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

DR. AJAY JONEJA, SUPERVISOR

PROF. CHUNG-YEE LEE, HEAD

Department of Industrial Engineering and Engineering Management

August 9, 2005
# TABLE OF CONTENTS

Title Page i
Authorization Page ii
Dedication iii
Acknowledgements iv
Signature Page v
Table of Contents vi
List of Figures ix
Abstract xi

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Problem statement</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Problem statement</td>
<td>3</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Data format</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Organization of the thesis</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2</th>
<th>Background</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Morphing based interpolation</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Linear morphing</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Nonlinear morphing</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Surface blending</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Surface extension</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Background of fairness</td>
<td>13</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Energy based fairness functional</td>
<td>14</td>
</tr>
<tr>
<td>2.4.1.1</td>
<td>Energy based fairness functional for curve</td>
<td>14</td>
</tr>
<tr>
<td>2.4.1.2</td>
<td>Energy based fairness functional for surface</td>
<td>14</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Variation based fairness functional</td>
<td>14</td>
</tr>
<tr>
<td>2.4.2.1</td>
<td>Variation based fairness functional for curve</td>
<td>15</td>
</tr>
<tr>
<td>2.4.2.2</td>
<td>Variation based fairness functional for surface</td>
<td>15</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Energy based fairness functional for discrete case</td>
<td>15</td>
</tr>
<tr>
<td>2.4.3.1</td>
<td>Energy based fairness functional for discrete curve</td>
<td>15</td>
</tr>
<tr>
<td>2.4.3.2</td>
<td>Energy based fairness functional for discrete surface</td>
<td>16</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Variation based fairness functional for discrete case</td>
<td>18</td>
</tr>
<tr>
<td>2.4.4.1</td>
<td>Variation based fairness functional for discrete curve</td>
<td>18</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>Interpolation using blending of morphing and surface extension</td>
<td>20</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>3.1</td>
<td>Main idea</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Methodology</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>Linear morphing between $P[3]$ and $Q[3]$</td>
<td>23</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Constructing correspondence</td>
<td>23</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Computing $P_{morphing}$ and $Q_{morphing}$</td>
<td>24</td>
</tr>
<tr>
<td>3.4</td>
<td>Surface Extension</td>
<td>24</td>
</tr>
<tr>
<td>3.4.1</td>
<td>On continuity</td>
<td>24</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Continuity condition</td>
<td>25</td>
</tr>
<tr>
<td>3.4.2.1</td>
<td>Curve continuity condition</td>
<td>25</td>
</tr>
<tr>
<td>3.4.2.2</td>
<td>Surface continuity condition</td>
<td>26</td>
</tr>
<tr>
<td>3.4.2.3</td>
<td>Continuity condition of parametric surface patch with single parametric boundary curve</td>
<td>28</td>
</tr>
<tr>
<td>3.4.3</td>
<td>$G^1$ extension by linear extension method</td>
<td>29</td>
</tr>
<tr>
<td>3.4.4</td>
<td>$G^2$ extension based on continuity condition</td>
<td>31</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Examples of surface extension</td>
<td>35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>Blend optimization and MVS-based interpolation</th>
<th>37</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Energy based functional (MES)</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>Curvature variation based functional (CVS)</td>
<td>38</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Local parameterization</td>
<td>38</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Principal curvature variation based functional</td>
<td>40</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Maximum curvature change rate based functional</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Convex blending of morphing and extension</td>
<td>44</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Approximation using poly line</td>
<td>45</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Approximation using quadric Bezier curve</td>
<td>46</td>
</tr>
<tr>
<td>4.4</td>
<td>MES and MCVF optimization</td>
<td>47</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Problem formulation</td>
<td>47</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Determining start point by sampling</td>
<td>48</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Multidirectional search method</td>
<td>49</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>Examples</th>
<th>52</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Examples</td>
<td>52</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Approximation using poly line</td>
<td>52</td>
</tr>
<tr>
<td>5.1.1.1</td>
<td>Blending between 2 planar surfaces</td>
<td>52</td>
</tr>
<tr>
<td>5.1.1.2</td>
<td>Blending between planar surface and curved surface</td>
<td>54</td>
</tr>
<tr>
<td>5.1.1.3</td>
<td>Blending between 2 curved surfaces</td>
<td>56</td>
</tr>
<tr>
<td>5.1.1.4</td>
<td>Shoe last example</td>
<td>58</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Approximation using quadric Bezier curve</td>
<td>60</td>
</tr>
<tr>
<td>5.1.2.1</td>
<td>Blending between 2 planar surfaces</td>
<td>60</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1.1  A typical shoe last with the toe and rear parts marked  P.2
Figure 1.2  A typical shoe last with gap between toe and rear part  P.2
Figure 1.3  Data format of shoe last scanning  P.4
Figure 2.1  Linear morphing between 2 discrete 2D curves  P.6
Figure 2.2  Using linear morphing to fill gap  P.6
Figure 2.3  Shortcoming of linear morphing  P.7
Figure 2.4  Blending between an elliptic paraboloid and a sphere  P.9
Figure 2.5  Shortcoming of surface blending  P.10
Figure 2.6  Construction of linear extension  P.12
Figure 2.7  Reflection of control points  P.12
Figure 2.8  Two smooth interpolating curves of different fairness  P.13
Figure 2.9  A vertex v and local configurations  P.17
Figure 3.1  Schematic of the blending based gap filling  P.22
Figure 3.2  Using linear morphing to fill a gap  P.24
Figure 3.3  (a) $G^0$ continuity for curve  P.26
(b) $G^1$ continuity for curve  P.26
Figure 3.4  Two patches with single parametric boundary curve  P.28
Figure 3.5  Linear extension  P.30
Figure 3.6  Alignment of original slices and extension slices  P.31
Figure 3.7  Computing $\overrightarrow{R_i}$  P.32
Figure 3.8  One-slice extension for toe part  P.35
Figure 3.9  One-slice extension for rear part  P.35
Figure 3.10  Multiple slices extension of a curved surface  P.36
Figure 4.1  Local triangulation at point $S[i][j]$  P.38
Figure 4.2  Parameterization for local surface fitting  P.39
Figure 4.3  Line of principal direction in parametric space  P.40
Figure 4.4  Mapping line of principal direction in parametric space into real space  P.40
Figure 4.5  $k$ vs. curvature change rate $\frac{dk}{ds}$  P.42
Figure 4.6  Poly line by (0,1), $(u,v)$ and $(n-1,1)$  P.46
Figure 4.7  Example of poly line that $n = 9$, $u = 5$ and $v = 0$  P.46
Figure 4.8  Example of Bezier curve that $n = 19$, $u = 5$ and $v = -1$  P.47
Figure 4.9  Multidirectional search method  P.50
Figure 5.1  Original surfaces: two planar surfaces  P.52
Figure 5.2  Blending surface between 2 planar surfaces  P.53
Figure 5.3  Comparison of PCVF/MCVF/MES surfaces  P.54
| Figure 5.4 | W [i] for optimized MES, MCVF, and PCVF surface | P.54 |
| Figure 5.5 | Original surfaces: planar surface and curved surface | P.55 |
| Figure 5.6 | Blending surface between: planar surface and curved surface | P.55 |
| Figure 5.7 | Comparison of PCVF/MCVF/MES surfaces | P.56 |
| Figure 5.8 | W [i] for optimized MES, MCVF, and PCVF surface | P.56 |
| Figure 5.9 | Original surfaces: two curved surfaces | P.57 |
| Figure 5.10 | Blending surface between two curved surfaces | P.57 |
| Figure 5.11 | Comparison of PCVF/MCVF/MES surfaces | P.57 |
| Figure 5.12 | W [i] for optimized MES, MCVF, and PCVF surface | P.58 |
| Figure 5.13 | Original surfaces: shoe last | P.58 |
| Figure 5.14 | Blending surface between two shoe last surfaces | P.59 |
| Figure 5.15 | Comparison between MES (a) and PCVF (b) surfaces | P.60 |
| Figure 5.16 | W [i] for optimized MES, MCVF, and PCVF surface | P.60 |
| Figure 5.17 | Original surfaces: two planar surfaces | P.61 |
| Figure 5.18 | Blending surface between 2 planar surfaces | P.61 |
| Figure 5.19 | Comparison of PCVF/MCVF/MES surfaces | P.62 |
| Figure 5.20 | W [i] for optimized MES, MCVF, and PCVF surface | P.62 |
| Figure 5.21 | Original surfaces: planar surface and curved surface | P.63 |
| Figure 5.22 | Blending surface between: planar surface and curved surface | P.63 |
| Figure 5.23 | Comparison of PCVF/MCVF/MES surfaces | P.64 |
| Figure 5.24 | W [i] for optimized MES, MCVF, and PCVF surface | P.64 |
| Figure 5.25 | Original surfaces: two curved surfaces | P.65 |
| Figure 5.26 | Blending surface between two curved surfaces | P.65 |
| Figure 5.27 | Comparison of PCVF/MCVF/MES surfaces | P.66 |
| Figure 5.28 | W [i] for optimized MES, MCVF, and PCVF surface | P.66 |
| Figure 5.29 | Original surfaces: shoe last | P.67 |
| Figure 5.30 | Blending surface between two shoe last surfaces | P.67 |
| Figure 5.31 | Comparison between MES (a) and PCVF (b) surfaces | P.68 |
| Figure 5.32 | W [i] for optimized MES, MCVF, and PCVF surface | P.68 |
| Figure 5.33 | Comparison of optimization result: case of two planar surfaces | P.69 |
| Figure 5.34 | Comparison of optimization result: case of planar and curved surfaces | P.70 |
| Figure 5.35 | Comparison of optimization result: case of two curved surfaces | P.71 |
| Figure 5.36 | Difference between 19-D and 2-D optimization | P.72 |
| Figure 5.37 | Comparison of optimization result: shoe last surfaces | P.73 |
| Figure 5.38 | Difference between 19-D and 2-D optimization | P.74 |
Shape Blending Using Discrete Curvature-Variation Functional

By Guo, Li

Department of Industrial Engineering and Engineering Management

The Hong Kong University of Science and Technology

Abstract

This research studies a new method for discrete surface blending. This method is based on combination of morphing and extension. Deriving from continuity conditions, extension of discrete surface is discussed. Fairness functional is also studied in thesis. Both MES (minimum energy surface) and MVS (minimum variation surface) are extended into discrete case. For MES, discrete curvature is introduced; For MVS, two approximations are presented. One is to minimize curvature change rate along principal direction; the other is to minimize maximum curvature change rate. Multidirectional searching method is applied in optimization. Plenty of examples are discussed in the end. The method presented in thesis can be applied in surface modeling.
Chapter 1

INTRODUCTION

1.1 Motivation

The work described in this thesis is motivated by a problem in Computer-Aided Design (CAD) of footwear. Design of a shoe begins with design of the shoe last, which is the mould around which a shoe is constructed. Last designers often decompose lasts into two parts: toe part (front part) and rear part (back part).

Typically the geometry of rear part is controlled by comfort requirements so it is determined by geometry of foot; this part has less freedom for design. It is common for companies to maintain a library of standard back-part designs (a library is required since the geometry depends on the heel height, footwear style, e.g. sandal, pump, boot etc., and the anthropomorphic statistics of the target population).

However, much freedom is allowed in the shape of the toe, which is largely dictated by aesthetic concerns. A convenient design strategy is to first design the toe part, and then selects a suitable rear part from library and ‘combine’ these two components to create the last. Another common technique used by practitioners is to use the toe shape of one style of shoe and adapt it to fit with the rear part of a different design to create new designs. Figure 1.1 shows a CAD model of a simple last, with the rear and the toe parts marked. In practice to give freedom for designer to smooth out whole shape, there is a gap between toe part and rear part, otherwise combining a toe and rear that come from different designs will result in a discontinuity between the two shapes at the interface. This is shown in figure 1.2: there is a gap of
approximately 15-20mm between any pair of arbitrarily selected toe and back-parts. In this work, it will be assumed that the new toe has been scaled appropriately and located in the correct position with respect to the back part. This scaling and positioning may be done with some user interaction, since the new toe shape must satisfy comfort and aesthetic guidelines that are sometimes subjective. Then a surface must be created in the gap between the toe and the back part, to get a complete surface describing the last.

![Figure 1.1](image1.jpg)  
**Figure 1.1** A typical shoe last with the toe and rear parts marked

![Figure 1.2](image2.jpg)  
**Figure 1.2** A typical shoe last with gap between toe and rear part

1.2 Problem statement
1.2.1 Problem statement

We are given two disjoint surfaces, $S_1$ and $S_2$, with curves $C_1$ and $C_2$ lying respectively on $S_1$, $S_2$. The curves $C_1$ and $C_2$ must not have self-intersections; both must either be simultaneously open- or closed-loop. The objective is to develop interpolating function(s) that will generate an aesthetic (this will be discussed later) surface blending $S_1$ and $S_2$ across $C_1$ and $C_2$. A further goal of this research is to apply the results obtained to a practical problem of gap filling in footwear CAD as described in the motivation section above.

1.2.2 Data format

Although the techniques developed in the thesis are based on differential properties, in practical implementation, it will be convenient to use discretized data. The data used in thesis conforms to the simplest possible form, point clouds. This format is universally obtainable from any digitizing hardware, e.g. CMM, laser scanner, etc. and so it quite versatile. It is further assumed, though not necessary, that the point clouds are ordered, being composed of several planar slices. Each slice contains an equal number of points. It is relatively easy to convert any other digitized data into this format by triangulation, slicing and sampling the slices. Figure 1.3 shows point cloud of a shoe last, and in this example each slice contains 90 points.
1.3 Organization of the thesis

In the next chapter, relevant background on surface interpolation is discussed, along with some examples showing why existing methods do not always yield practically convenient solution to our problem. Chapter 3 discusses some of the theoretical background for the interpolation technique we shall adopt. Chapter 4 presents a simple surface interpolation method using a weighted interpolation of linear morphing and surface blending functions, and interpolation scheme based on minimum rate of change of curvature is also presented in chapter 4. The methods of Chapters 4 are implemented, and some examples of the use of these methods are shown in Chapter 5, along with some details of the implementation. Finally, some concluding remarks and possible future work are presented in chapter 6.
Chapter 2

BACKGROUND

The research described in thesis is related to 3D shape operations, particularly morphing, surface extension, blending and surface fairing. The background knowledge and literature review of relevant works is presented in this chapter.

2.1 Morphing based interpolation

Morphing is a continuous deformation from one key frame or 3D model to another. In 3D this is often achieved by approximating a surface with a triangular mesh that can then be continuously deformed. In 2D, it is generally performed by either distortion or deformation. Its primary use is in animation or graphic effects.

If $v_1 \in O_1$ and $v_2 \in O_2$ are corresponding points, an animation path between them by morphing is defined as:

$$\varphi(t) = (1 - f(t))v_1 + f(t)v_2 \quad f(t) \in [0,1], t \in [0,1], \quad f(0) = 0 \text{ and } f(1) = 1$$

2.1.1 Linear morphing

When $f(t)$ is linear function, morphing is also called linear morphing or linear interpolation.

$$\varphi(t) = (1-t)v_1 + tv_2 \quad t \in [0,1]$$

It is easy to see that, $\varphi(0) = v_1$ and $\varphi(1) = v_2$

Figure 2.1 show an example of linear morphing between 2 discrete 2D curves.
There are two controls in morphing – (1) generating a one-to-one correspondence between points on the initial and final curves and (2) the morphing function. Thus, in linear morphing, each intermediate figure is only determined by the correspondence map. There are several ways to construct such a correspondence. Typical way is to construct correspondence by optimization, for example, minimizing energy consumed [1]; and a relatively simple way is to match closest points [2] [3]: for point
$v_i \in O_1$, point $v_2 \in O_2$ is its correspondence point if and only if $v_2$ is the closest point to $v_i$ among all points in $O_2$.

2.1.2 Nonlinear morphing

Here, the function $f(t)$ is nonlinear. Nonlinear morphing is widely used especially under constraints; this operation has many elements in common with the more general shape operation called blending.

However, most morphing-based functions do not explicitly account for (different levels of) continuity at the boundary. This is partly because the operation itself is more often used for animation than for blending. Therefore classical morphing often results in discontinuity at the interface. An example of such a problem is shown using linear morphing between a toe part and rear part of a shoe last in figure 2.3.

![Figure 2.3 Shortcoming of linear morphing](image)

2.2 Surface blending

A large amount of blending research in the past has concentrated on the creation of variable- or constant-radius fillet features in CAD models. These essentially result in exact or approximated (parametric) forms of canal surfaces or their restricted form, pipe surfaces [4-7]. A large class of blending models is based on some form or
generalization of the notion of filleting, including the use of cyclides. Some researchers have also considered issues such as smoothness of the blending surface. We are aware of at least two different methods to do so. In [8], the smoothness of the blending surface was derived by filtering out all high frequency variations of the surface in the region of the blend. The tool used to do so was Fourier transforms. However, this work obviously requires the geometry of the underlying surface(s) to be known before the blend is applied. Another typical problem in generation of blends is that the complexity of the geometry and topology in regions where several components of the blending surface intersect. This occurs, for example, in the neighborhood of a vertex where the blending surfaces for all the incident edges of a part meet. To tackle this problem, a method to generate a minimum energy surface to form the blend was developed in [9].

Another approach is that of generation of “fill” surfaces – namely a surface that smoothly interpolates a set of given curves forming a network. A simple case is when the interpolated curves form a closed loop that is marked as a boundary of the filling surface. At other times, the network is in the form of a series of curves, and a skinning operation can be applied [10]. However, this approach fails to give sufficient control on the resulting surface, partly due to the lack of appropriate guide curves bridging the gap. Smoothly interpolating fill surfaces have also been generated using techniques that are based on subdivision surfaces [11]. However, this approach yielded surfaces for which subsequent shape modification of the shoe last was stunted due to lack of tools.
Another research direction of surface blending is based on partial differential equations (PDE) [12-13]. In the PDE method, a blending surface is achieved by solving a partial differential equation, with given boundary curve(s) and first derivative at boundary curve as boundary conditions. PDE’s are usually based on a physical model. One such form is the elliptic equation:

\[
\left( \frac{\partial^2}{\partial u^2} + a^2 \frac{\partial^2}{\partial v^2} \right)^2 f(u,v) = 0
\]

With boundary conditions:

\[
\begin{align*}
    f(0,v) &= G_0(v), f(1,v) = G_1(v) \\
    f_u(0,v) &= S_0(v), f_u(1,v) = S_1(v)
\end{align*}
\]

Solving this equation can yield a smooth surface because the solution is transcendental function. See figure 2.4 [13].

![Figure 2.4 Blending between an elliptic paraboloid and a sphere](image_url)
PDE-based blending has some drawbacks. The geometry of the boundary must be known; complex boundary geometry makes the PDE computationally intractable; also, blending surfaces have little freedom for control.

For our specific problem of gap filling for footwear CAD, surface blending is a potential solution. However, due to the complexity of constraints on the blended shape, this approach cannot generate satisfying surface. See figure 2.5 for an example, where the ergonomic constraints on the last shape are violated, although the blending operator yields a smooth and perhaps even a fair gap-filling surface.

![Diagram](image)

**Figure 2.5 Shortcoming of surface blending**

### 2.3 Surface Extension

Curve/surface extension refers to extrapolation across boundary point/curve. In solid modeling, curve/surface extension is often used, for example, in operations of shelling, blending, and drafting. Common used extension methods include [14]:

1. **Natural extension**

Natural extension is the straightforward extrapolation for parametric surface. Assume a parametric surface \( S(u, v) \) is given, whose parametric domain
is \( u \in [u_0, u_1], v \in [v_0, v_1] \). Natural extension is performed by applying surface function with a value outside parametric domain. For example, if this surface needs to be extended across boundary curve \( S(u, v_i) \), we just apply surface function on \( u \in [u_0, u_1], v > v_i \). Natural extension keeps continuity at boundary curve (continuity order is determined by order of surface function); in some cases such extension will lead to instability such as surface singularity or self-intersection. Also, shape control is limited. Usually, this approach works well only for very small gaps.

2. Linear extension

Linear extension is using the derivative across the extension boundary to infer new geometry. New control points along extension direction are computed as:

\[
d_{n+1,i} = d_{n,i} + \alpha v_i
\]

\( d_{n,i} \) is the last row of control points;

\( d_{n+1,i} \) are the new control points;

\( v_i = d_{n,i} - d_{n-1,i} \) is the tangent direction;

\( \alpha \) is extension magnitude, set by user.

Figure 2.6 shows how linear extensions can be constructed.
3. Reflection extension

This method is introduced in [15]. The main idea is to reflect control points with respect to normal plane (for curve extension) / normal plane of iso-parametric curve (for surface). For rational Bezier or spline curve/surface, the weight can be found by solving the equation derived from the continuity condition(s). Figure 2.7 shows the reflection of control points for curve extension [15].
Besides these methods there are some papers discuss surface extension based on continuity condition [16]. These will be discussed later.

2.4 Background of fairness

In solid modeling, smooth curve/surface are desired and well defined. Smoothness refers to mathematical continuity. A curve/surface is said to be smooth if it satisfies $G^1$ or higher continuity conditions at all points (more discussion is in chapter 3).

But in some cases, smooth curve/surface doesn’t “look smooth” even though it satisfies mathematical definition, especially when bumps and wiggles are included.

Here is an example of two smooth curves; both are interpolation of 3 given points, but curve (b) looks “smoother” than curve (b).

![Figure 2.8 Two smooth interpolating curves of different fairness](image)

This motivated the search for “visually smooth” curves/surfaces, called fair curves/surfaces. Compared to smoothness, there is no commonly accepted mathematical definition of fairness. Several approaches have been used, as discussed in the following sub-sections.
2.4.1 Energy based fairness functional

2.4.1.1 Energy based fairness functional for curve

The functional for minimum energy curve (MEC) is derived from a physical model. The functional is simple and widely used in either of the following two forms [17]:

\[ E_{\text{curve}} = \int \kappa(s)^2 ds \]

Or: \[ E_{\text{curve}} = \int |\kappa(s)| ds \]

In which, \( \kappa \) refers to curvature and \( s \) is the arc length. MEC is intuitive, and relatively easy to compute using numerical techniques.

2.4.1.2 Energy based fairness functional for surface

Similar to MEC, minimum energy surface (MES) is based on a physical model. The energy-based functionals can be classified into two types. The first category is derived from physical deformation, typically elastic sheet deformation [18] [19]:

\[ E_{\text{surface}} = \iint \left( \alpha_1 W_u^2 + 2\alpha_{12} W_u W_v + \alpha_{22} W_v^2 + \beta_1 W_{uu}^2 + \beta_{12} W_{uv}^2 + \beta_{22} W_{vv}^2 \right) - 2Wf(u,v) \] \] 

where \( W_u, W_v \) are partial derivatives;
\( W_{uv} \) is the mixed derivative;
\( \alpha, \beta \) are given coefficients;
\( f(u,v) \) is a given vector function.

The second category is derived from geometry properties. A popular form is:

\[ E_{\text{surface}} = \int (\kappa_1^2 + \kappa_2^2) dA \]

In which, \( \kappa_1, \kappa_2 \) are principal curvatures. This functional is independent of the parameterization.

2.4.2 Variation based fairness functional

In some cases, MEV/MES will generate curves/surfaces with sharp curvature change,
which makes curve/surface looks unsmooth. Recently some papers present another functional based on curvature variation [20].

2.4.2.1 Variation based fairness functional for curve

Minimum curvature variation curve (MCV) is easy to define:

\[ V_{\text{curve}} = \int \left( \frac{d\kappa}{ds} \right)^2 ds \]

Or \[ V_{\text{curve}} = \int \left| \frac{d\kappa}{ds} \right| ds \]

Similar research also includes optimizing curve with as few as possible monotone curvature pieces [21]; this effectively combines the MEC functional and an MVC functional [22].

2.4.2.2 Variation based fairness functional for surface

Minimum variation surface is a little complex because direction is also involved when computing curvature derivative; [20] proposed MVS a functional as

\[ V_{\text{surface}} = \iint_{\text{surface}} [\left( \frac{d\kappa_1}{de_1} \right)^2 + \left( \frac{d\kappa_2}{de_2} \right)^2 ] dA \]  \hspace{1cm} (4.1)

This evaluates how the principal curvature changes along principal direction. This functional is parameter-independent but it is not easy to compute. Another way to achieve MVS is convert MVS problem into an MVC problem [23]. Instead of computing curvature derivative on surface, groups of curves forming a mesh on the surface are optimized to be MVC. This approach is easier to implement; the difficulty is how to choose a reasonable curve mesh.

2.4.3 Energy based fairness functional for discrete case

2.4.3.1 Energy based fairness functional for discrete curve
In subsequent chapters, it will be necessary to use discrete representations of surfaces and estimations of differential properties based on discrete models. In this section, some background is presented. To apply MEC functional in discrete case, estimating curvature is necessary. There are two ways to estimate curvature:

- using finite difference

From [24], Given points series $f_0, \ldots, f_i, \ldots, f_m$,

$$\Delta s_i = |f_i - f_{i-1}| \quad i = 1, \ldots, m$$

$$\Delta f_i = \frac{f_i - f_{i-1}}{\Delta s_i} \quad i = 1, \ldots, m$$

$$\Delta^2 f_i = \frac{\Delta f_{i+1} - \Delta f_{i-1}}{(\Delta s_i + \Delta s_{i+1})/2} \quad i = 1, \ldots, m - 1$$

Accordingly, the required functional is:

$$E(f) = \sum_{i=1}^{m} \left| \Delta^2 f_i \right|^2 \Delta s_i$$

- using touch circle

For detailed description of this approach, see [25]. Discrete curvature of $f_i$ is defined as circle radius determined by $f_{i-1}, f_i, f_{i+1}$.

$$\kappa(f_i) = \frac{|f_i - f_{i-1}| \cdot |f_{i+1} - f_{i-1}| \cdot |f_{i+1} - f_i|}{A}$$

$A$ is the area of the triangle defined by $f_{i-1}, f_i, f_{i+1}$.

Accordingly, the functional is:

$$E(f) = \sum_{i=1}^{m-1} \kappa(f_i)$$

**2.4.3.2 Energy based fairness functional for discrete surface**

Energy based fairness functional has been studied in the context of
mesh/triangulation fairing [26] [27]. In [26], fairness of the triangulation is estimated by first constructing a locally continuous mapping by local parameterization; then the MES functional can be applied by the use of numerical techniques as discussed before. In [27] an energy functional using discrete curvature is presented. The definition for discrete curvature is given as:

- **Gaussian Curvature**

\[ K = 2\pi - \sum_{i=1}^{n} \alpha_i \]

\(\alpha_i\) is the angle between two successive edges \(\vec{e}_i\) and \(\vec{e}_{i+1}\): \(\alpha_i = \angle \vec{e}_i, \vec{e}_{i+1}\)

- **Mean Curvature**

\[ H = \frac{1}{4} \sum_{i=1}^{n} |e_i||\beta_i| \]

\(\beta_i\) is the angle between normal vectors of two successive triangles: \(\beta_i = \angle \vec{n}_i, \vec{n}_{i-1}\)

![Figure 2.9 A vertex \(v\) and local configurations](image)

From basic differential geometry,

\[ K = \kappa_1 \kappa_2 \]

\[ H = \frac{1}{2} (\kappa_1 + \kappa_2) \]

where \(\kappa_1, \kappa_2\) are principal curvatures.
\[ |\kappa_1| + |\kappa_2| = \begin{cases} 
\frac{2|H|}{2\sqrt{|H|^2 - K}} 
\end{cases} \]

Accordingly, the energy functional is defined as:

\[ E_{\text{triangulation}} = \sum K^2 \]

or

\[ E_{\text{triangulation}} = \sum H^2 \]

or

\[ E_{\text{triangulation}} = \sum |\kappa_1| + |\kappa_2| \]

2.4.4 Variation based fairness functional for discrete case

2.4.4.1 Variation based fairness functional for discrete curve

Finally, a brief discussion of the curvature variation functionals and their computation is discussed for the discrete case. Here, a finite sampling of points on the underlying surface is the only available data. Using finite difference [24],

\[ \Delta s_i = |\overrightarrow{f_i} - \overrightarrow{f_{i-1}}| \quad i = 1, \ldots, m \]

\[ \Delta f_i = \frac{\overrightarrow{f_i} - \overrightarrow{f_{i-1}}}{\Delta s_i} \quad i = 1, \ldots, m \]

\[ \Delta^2 f_i = \frac{\Delta f_{i+1} - \Delta f_i}{(\Delta s_i + \Delta s_{i+1})/2} \quad i = 1, \ldots, m - 1 \]

\[ \Delta^3 f_i = \frac{\Delta^2 f_i - \Delta^2 f_{i-1}}{\Delta s_i} \]

As before, it is conventional to minimize the integral (approximate) of the variation of the curvature over the curve/surface; so a typical objective function is:

\[ E(f) = \sum_{i=1}^{m} \left| \Delta^3 f_i \right|^2 \Delta s_i \]

Further, [25] introduces another fairing algorithm. A "bad point" is defined as a point with successive sign change of discrete curvature, \( K_{i-1} \cdot K_i < 0, K_i \cdot K_{i+1} < 0 \); the algorithm modifies each successive "bad point" until a given stopping criterion based
on fairness is met.

To our knowledge, there is no work reported on surface fairing using a curvature-variation (CVS) based fairness functional. In this research, a discrete algorithm based on CVS is developed for gap filling. This operator will be discussed in detail in Chapter 4.
Chapter 3

INTERPOLATION USING BLENDING OF MORPHING AND SURFACE EXTENSION

3.1 Main idea

In this chapter, a new operator is introduced to generate interpolated gap-filling. Throughout this chapter and the next, it will be assumed that only a discrete form of the interpolation surfaces is known. Further, it will be assumed that the discretization is in the form of a point cloud organized in axis-parallel slices (see figure 3.1). It is easy to transform inputs in other formats to this form, within any specified accuracy. Thus, the task of generation of the gap-filling surface is reduced to the task of generating a sufficient number of intermediate slices in the gap. As presented in chapter 1, intermediate slices must achieve:

- (gradually) transition from one shape to the other;
- smoothness at interface;
- fairness of whole surface;

The operator presented in this chapter is based on a (non-linear) combination of linear morphing and surface extension. Note that linear morphing provides a gradual transition (but not continuity), while surface extension guarantees continuity (but not fairness). A suitable blending-function will be created to interpolate between morph and extension so as to provide all three desirable characteristics.

3.2 Methodology

Given two original surfaces, \( P \) and \( Q \), each contains at least 3 slices and each
slice has same number of points (say, \( m \) points in each slice). To simplify notation, it is assumed that the distance between neighboring slices \( P \) and \( Q \) is the same, and the slices are evenly distributed along the \( z \) axis. The gap will be filled by generating \( n \) intermediate slices (\( n \) is variable and depends on the size of the gap). Each intermediate slice is a convex combination (blend) of two slices generated on that plane – the first resulting from linear morphing, and the second resulting from an extension of either \( P \) or \( Q \), whichever is closer. The blend weight is denoted \( W[k] \), \( k \in \{0, \ldots, n-1\} \), at slice \( k \). Thus, for the \( n \) slices, there are a total of \( n \) blend weights, stored in an array denoted \( W \).

\( P[i], Q[i] \) denote the \( i \)-th slice on surface \( P \), \( Q \) respectively (figure 3.1); \( P[i][j] \) and \( Q[i][j] \) denote the \( j \)-th point of \( i \)-th slice on surface \( P \), \( Q \) respectively. Similarly, points on the intermediate slices are denoted \( S[i][j] \), \( i = 0,1,\ldots,n-1; \ j = 0,1,\ldots,m-1 \).

The following steps summarize the procedure to generate the intermediate slices.

a. Read slices \( P[1], P[2], P[3] \) (the last 3 slices of \( P \)), slices \( Q[1], Q[2], Q[3] \) (the first 3 slices of \( Q \));

$P\_extension$;

c. Compute extension of $Q$, beyond $Q[3]$, denoted as $Q\_extension$;

d. Compute linear morphing slices between $P[3]$ and $Q[3]$, next to $P[3]$ and

next to $Q[3]$, denoted as $P\_morphing$ and $Q\_morphing$;

e. Generate two new slices, one on each end of the gap, using convex combination:

$R_i = W[i] \cdot P\_extension + (1 - W[i]) \cdot P\_morphing$ (on left side),

and

$R_{n-i} = W[n-i] \cdot Q\_extension + (1 - W[n-i]) \cdot Q\_morphing$ (on right side)


g. If gap can accommodate two or more slices, got to step (a); else,

If there is only one slice left, fill the gap by linear morphing of $P[3]$, $Q[3]$
Note that the scalar weights $W[n]$, can be changed to get different intermediate slices; this gives some freedom to control the interpolated shape, e.g. to increase fairness.

The following sections give details of the steps in the algorithm.

3.3 Linear morphing between $P[3]$ and $Q[3]$

3.3.1 Constructing correspondence

Since $P[3]$ and $Q[3]$ are closed loops, before computing $P\_morphing$ and $Q\_morphing$, correspondence between $P[3]$ and $Q[3]$ should be constructed (in some literature this is also called matching start points). There are several ways to construct such correspondence. One way is to construct correspondence by optimization, for example, minimizing energy consumed [1]; another way is to match closest points [2] [3]. A simple closest point algorithm is as follows (assuming each slice is a closed loop with $m$ points):

(a) Closest = $i = 1$; Current closest distance = Infinity (suitably large real number);

(b) For $i = 1$ to $m$

Find closest point $Q[3]$, e.g. $Q[3][r]$ to $P[3][i]$

Closest distance = Euclidean distance from $P[3][i]$ to $Q[3][r]$

If Closest distance < Current closest distance, then

Current closest distance = Closest distance

Closest = $i$

(c) Report $P[3][Closest]$, and its nearest neighbor on $Q$;

Due to the nature of $P[i]$ and $Q[i]$, the algorithm can be significantly improved in efficiency by remembering the previous closest point, and beginning the search for
the next pair from there. For clarity, these details are omitted here.

3.3.2 Computing \( P \_morphism \) and \( Q \_morphism \)

When there are \( x \) \( (x \leq n) \) slices between \( P[3] \) and \( Q[3] \), morphing slices next to \( P[3] \) and \( Q[3] \) are:

\[
P \_morphism[j] = \frac{x}{x+1} \cdot P[3][j] + \frac{1}{x+1} \cdot Q[3][j];
\]

\[
Q \_morphism[j] = \frac{x}{x+1} \cdot Q[3][j] + \frac{1}{x+1} \cdot P[3][j];
\]

\( 0 \leq j \leq m - 1; \)

Here is example of linear morphing.

![Figure 3.2 Using linear morphing to fill a gap](image)

3.4 Surface Extension

3.4.1 On continuity

Before developing a method for surface extension, some basics on continuity of surfaces is presented. There are several notions of continuity on surfaces meeting at a shared boundary.
- **Parametric continuity** (also called C-continuity): Two pieces (curves or surfaces, solids) have rth order parametric continuity, $C^r$, if and only if all their 0th to rth derivatives agree at the common points [28] [29]. Parametric continuity is parameterization-dependent.

- **Visual continuity** (also called G-continuity): Two pieces have rth order geometric continuity, $G^r$, if and only if there exists parameterization, so that the parameterized pieces have $C^r$ continuity. $G^1$ continuity is also called tangent continuity; $G^2$ continuity also called curvature continuity; Visual continuity is parameterization-independent.

In general, parametric continuity is more stringent, and therefore more difficult to satisfy. In many most applications requiring continuity above $C^1$, it is conventional to use the parameterization-independent visual continuity.

**3.4.2 Continuity condition**

**3.4.2.1 Curve continuity condition**

Two curves $C_-(t)$ and $C_+(t)$ join in common point $t=0$ with $G^n$ continuity if and only if:

$$\begin{bmatrix} C_- & C_-' & \ldots & C_-'^{(n)} \end{bmatrix}_{0} = \begin{bmatrix} C_+ & C_+ ' & \ldots & C_+^{(n)} \end{bmatrix}_{0}$$

$$= \begin{bmatrix} 1 & 0 & \ldots & 0 \\ \alpha & \ldots & * \\ \ldots \\ \alpha^n \end{bmatrix}$$

In short, $C_- = C_+ A$, where the upper triangular matrix $A$ is called connection matrix. A is related to re-parameterization. For example, the connection matrix for $G^3$ continuity is:
\[ A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \alpha & \beta & \gamma \\ 0 & 0 & \alpha^2 & 3\beta\gamma \\ 0 & 0 & 0 & \alpha^3 \end{bmatrix} \]

If we can find non-zero number \(\alpha\), \(\beta\) and \(\gamma\) that make \(C_\gamma = C_{\alpha}A\) satisfied, then \(C_{\gamma}(t)\) and \(C_{\alpha}(t)\) are said to join in common point \(t=0\) with \(G^3\) continuity.

This condition can be derived from definition of \(G\)-continuity, see [28] [29]. The geometric meaning for \(G^n\) continuity of curves is:

- \(G^0\) continuity: share point;
- \(G^1\) continuity: share tangent direction;
- \(G^2\) continuity: share Frenet frame and curvature;
- \(G^3\) continuity: share torsion;
- When \(n > 3\): the curvature is \(C^{r-2}\) continuous, and the torsion is \(C^{r-3}\) continuous.

Figure 3.3 shows example of \(G^0\) continuity and figure 3.4 shows example of \(G^1\) continuity.

![Figure 3.3 (a) \(G^0\) continuity for curve (b) \(G^1\) continuity for curve](image)

3.4.2.2 Surface continuity condition
Surface continuity condition is a little more complex.

- Two surface $P$ and $\tilde{P}$ are $G^0$ continuous if and only if they share a boundary curve;

- Two surface $P$ and $\tilde{P}$ are $G^1$ continuous if and only if for each point on common boundary we can find a matrix $R$ that:

$$\overline{A} = RA$$

- Two surface $P$ and $\tilde{P}$ are $G^2$ continuity if and only if for each point on common boundary we can find such a matrix $R$ that:

$$\overline{A} = RA$$

$$\overline{G} = RGR^T$$

$$\overline{D} = RDR^T$$

$$G = AA^T = \begin{pmatrix} P_u \cdot P_u & P_u \cdot P_v \\ P_v \cdot P_u & P_v \cdot P_v \end{pmatrix}$$

Where

$$D = \begin{pmatrix} N \cdot P_{uu} & N \cdot P_{uv} \\ N \cdot P_{uv} & N \cdot P_{vv} \end{pmatrix}$$

$$N = \frac{P_x \times P_v}{\|P_x \times P_v\|}$$

$$A = \begin{pmatrix} P_u \\ P_v \end{pmatrix}$$

The geometric meaning for $G^0$ continuity is:

- Two surface $P$ and $\tilde{P}$ are $G^1$ continuous if and only if they share same boundary curve;
- Two surface $P$ and $\overline{P}$ are $G^1$ continuous if and only if for each point on
  the boundary curve the two patches share the surface normal direction;
- Two surface $P$ and $\overline{P}$ are $G^2$ continuous if and only if for each point on
  boundary curve, the two patches share the surface normal direction, the
  principal curvatures and the principle directions.

3.4.2.3. Continuity condition of parametric surface patch with single parametric
boundary curve

As a special case, continuity conditions of parametric surface patch with single
parametric boundary curve get more attention because it is often applied in real
problem.

![Two patches with single parametric boundary curve](image)

Figure 3.4 Two patches with single parametric boundary curve

From [15], [16], patches $B$ and $C$ are $G^1$ continuous if for each point on
boundary curve, tangent planes for two patch are identical; or

\[
\alpha(v) \frac{\partial}{\partial u} B(u,v) \bigg|_{u=0} + \beta(v) \frac{\partial}{\partial s} C(s,v) \bigg|_{s=0} + \gamma(v) \frac{\partial}{\partial v} C(s,v) \bigg|_{s=0} = 0 \quad (3.1)
\]

$B$ and $C$ are $G^2$ continuous if and only if:

\[
\alpha(v) \frac{\partial}{\partial u} B(u,v) \bigg|_{u=0} + \beta(v) \frac{\partial}{\partial s} C(s,v) \bigg|_{s=0} + \gamma(v) \frac{\partial}{\partial v} C(s,v) \bigg|_{s=0} = 0
\]
\[(\alpha(v))^2 \frac{\partial^2}{\partial u^2} B(u, v) \bigg|_{u=0} - (\beta(v))^2 \frac{\partial^2}{\partial s^2} C(s, v) \bigg|_{s=0} - 2\beta(v)\gamma(v) \frac{\partial}{\partial s} C(s, v) \bigg|_{s=0} - (\gamma(v))^2 \frac{\partial^2}{\partial v^2} C(u, v) \bigg|_{v=0} = \delta(v) \frac{\partial}{\partial s} C(s, v) \bigg|_{s=0} + \eta(v) \frac{\partial}{\partial v} C(u, v) \bigg|_{v=0}\]

(3.2)

Without losing generality, \(\alpha(v)\) is set as 1.

3.4.3 \(G^1\) extension by linear extension method

Now we discuss how to generate \textit{discrete extension} (extrapolation) of a given surface beyond its boundary curve. In this case, for each point \(\overline{Q}_i\) on boundary, its extending direction should satisfy:

- \textit{Perpendicular to normal direction of the surface on} \(\overline{Q}_i\) \textit{(for tangent continuity)};

- \textit{Perpendicular to tangent direction of the boundary curve on} \(\overline{Q}_i\) \textit{(to avoid surface shear/twisting)};

Let \(\overline{R}_i\) be the extension point corresponding to a boundary point \(\overline{Q}_i\), along direction \(\vec{v}\); let \(d\) be the extension distance. Tangent continuity is guaranteed when \(d \to 0\).

\[\overline{R}_i = \overline{Q}_i + \vec{v} \cdot d\]

(3.3)

Consider point \(\overline{Q}_i\) and its neighboring 2 points on last slice \(\overline{Q}_{i-1}, \overline{Q}_{i+1}\), and its corresponding point on previous slice, say \(\overline{P}_i\).

\[\text{29}\]
Denote:
\[ \vec{a} = \overrightarrow{Q_{i+1}} - \overrightarrow{Q_i} \]
\[ \vec{b} = \overrightarrow{P_i} - \overrightarrow{Q_i} \]
\[ \vec{c} = \overrightarrow{Q_{i-1}} - \overrightarrow{Q_i} \]

Since \( \vec{v} \) is perpendicular to tangent direction of the boundary curve on \( \overrightarrow{Q_i} \), we have:
\[ (\overrightarrow{Q_{i+1}} - \overrightarrow{Q_{i-1}}) \cdot \vec{v} = 0 \]  \hspace{1cm} (3.4)

Estimate normal vector of \( \overrightarrow{Q_i} \) as sum of surrounding triangles’ normal direction, that is: \( a \times b + b \times c + c \times v + v \times a \); since \( \vec{v} \) should be perpendicular to the surface normal at \( \overrightarrow{Q_i} \), we have:
\[ (a \times b + b \times c + c \times v + v \times a) \cdot \vec{v} = 0 \]
\[ \Rightarrow (a \times b + b \times c) \cdot \vec{v} + (c \times v) \cdot \vec{v} + (v \times a) \cdot \vec{v} = 0 \]
\[ \Rightarrow (a \times b + b \times c) \cdot \vec{v} + c \cdot v \times v + a \cdot v \times v = 0 \]
\[ \Rightarrow (a \times b + b \times c) \cdot \vec{v} = 0 \]  \hspace{1cm} (3.5)

From (3.3) and (3.4), we know \( \vec{v} \) is perpendicular to \( \overrightarrow{Q_{i+1}} - \overrightarrow{Q_{i-1}} \) and \( a \times b + b \times c \), so:
\[ \vec{v} = (a \times b + b \times c) \times (\overrightarrow{Q_{i+1}} - \overrightarrow{Q_{i-1}}) \]  \hspace{1cm} (3.6)
Substitute (3.6) into (3.3) to compute \( \vec{R}_i \) (for a selected slice separation, \( d \)):

\[
\vec{R}_i = \vec{Q}_i + [(\vec{a} \times \vec{b} + \vec{b} \times \vec{c}) \times (\vec{Q}_{i+1} - \vec{Q}_{i-1})] \cdot d
\]

### 3.4.4 \( G^2 \) extension based on continuity condition

To apply continuity conditions (3.1) and (3.2), original surface (slices \( O, P \) and \( Q \)) should be parameterized. Parameterization is shown in figure 3.7; slices \( O, P \) and \( Q \) are original slices (denoted as patch \( B \)), and \( Q, R, S \) are extending slices (patch \( C \)). Patch \( B \) and \( C \) have common boundary, slice \( Q \).

Define \( u, s \) directions as extending directions, and \( v \) as the direction along the slice. That is to say, points \( \vec{P}_i, \vec{Q}_i \) (and so on) have same \( u \) parameter value and point \( \vec{P}_i, \vec{P}_j \) have same \( v \) parameter.

To achieve \( G^2 \) continuity, two extension slices are needed; here slice \( R \) (Figure 3.6) is temporary slice. Because these slices are evenly distributed along \( z \) axis, without losing generality we can set \( z \) coordinate of slice \( O, P, Q, R, S \) as \(-2d, -d, 0, d/2, \) and \( d \).

![Figure 3.6 Alignment of original slices and extension slices](image)

Using finite difference \( G^1 \) condition (3.1) is discretized at point \( \vec{Q}_i \) as:

\[
\frac{\vec{R}_i - \vec{Q}_i}{\Delta u} + \beta \frac{(\vec{P}_i - \vec{Q}_i)}{2\Delta s} + \gamma \frac{(\vec{Q}_{i+1} - \vec{Q}_{i-1})}{2\Delta v} = 0
\]

(3.7)
For symmetry consideration we let $\Delta u = \Delta s = k \Delta v$. From (3.7) we have:

$$\bar{R}_i = \left(\frac{3\bar{Q}_i}{2} - \frac{\bar{P}_i}{2}\right) - \gamma' (\bar{Q}_{i+1} - \bar{Q}_{i-1})$$  \hspace{1cm} (3.8)

In which $\gamma' = \frac{\gamma}{k}$ or $\gamma' = \frac{2\gamma}{k}$.

Since $\gamma'$ can be chosen arbitrarily, there are infinite solutions for $\bar{R}_i$, and these points lie on a line. To fix $\bar{R}_i$ an additional constraint is required; in the proposed method, the point closest to $\bar{Q}_i$ is selected. Thus:

$$\bar{R}_i - \bar{Q}_i = \left(\frac{\bar{Q}_i - \bar{P}_i}{2}\right) - \gamma' (\bar{Q}_{i+1} - \bar{Q}_{i-1})$$

When $\gamma' = \frac{(\bar{Q}_i - \bar{P}_i) \cdot (\bar{Q}_{i+1} - \bar{Q}_{i-1})}{2[\bar{Q}_{i+1} - \bar{Q}_{i-1}]^2}$, $|\bar{R}_i - \bar{Q}_i|$ is minimized. The geometric meaning is seen from the following figure:

![Figure 3.7 Computing $\bar{R}_i$](image)

When $\bar{R}_i - \bar{Q}_i$ is perpendicular to $\bar{Q}_{i+1} - \bar{Q}_{i-1}$, $|\bar{R}_i - \bar{Q}_i|$ is minimized;

So $\gamma' |\bar{Q}_{i+1} - \bar{Q}_{i-1}| = |\bar{Q}_i - \bar{P}_i| \cos \theta = \frac{(\bar{Q}_i - \bar{P}_i) \cdot (\bar{Q}_{i+1} - \bar{Q}_{i-1})}{2|\bar{Q}_{i+1} - \bar{Q}_{i-1}|}$

$$\Rightarrow \gamma' = \frac{(\bar{Q}_i - \bar{P}_i) \cdot (\bar{Q}_{i+1} - \bar{Q}_{i-1})}{2|\bar{Q}_{i+1} - \bar{Q}_{i-1}|^2}$$  \hspace{1cm} (3.9)

Substitute (3.9) into (3.8) we can get $\bar{R}_i$. 

32
Now $\overline{R}_i$ is fixed, we can move on $G^2$ condition to compute slice $S$.

The discrete form of (3.2) can be written as:

$$\frac{(\overline{S}_i + \overline{Q}_i - 2\overline{R}_i)}{(\Delta u)^2} - \frac{(\overline{O}_i + \overline{Q}_i - 2\overline{P}_i)}{4(\Delta s)^2} - \gamma \left( \frac{\overline{P}_{i+1} - \overline{P}_{i-1} - \overline{Q}_{i+1} + \overline{Q}_{i-1}}{4} \right) - \gamma^2 \left( \frac{\overline{Q}_{i+1} + \overline{Q}_{i-1} - 2\overline{Q}_i}{(\Delta s\Delta v)^2} \right)$$

$$= \eta \frac{\overline{Q}_{i+1} - \overline{Q}_{i-1}}{2\Delta v} + \delta \frac{\overline{P}_i - \overline{Q}_i}{2\Delta s}$$

(3.10)

By z coordinate, we have $\delta = 0$ ; substituting $\gamma = \frac{2\gamma'}{k}$ into (3.10), we get:

$$\frac{(\overline{S}_i + \overline{Q}_i - 2\overline{R}_i)}{(\Delta u)^2} - \frac{(\overline{O}_i + \overline{Q}_i - 2\overline{P}_i)}{4} - \gamma \left( \frac{\overline{P}_{i+1} - \overline{P}_{i-1} - \overline{Q}_{i+1} + \overline{Q}_{i-1}}{2} \right) - 4\gamma^2 \left( \frac{\overline{Q}_{i+1} + \overline{Q}_{i-1} - 2\overline{Q}_i}{(\Delta s\Delta v)^2} \right)$$

$$= \eta \cdot k \cdot \Delta u \frac{\overline{Q}_{i+1} - \overline{Q}_{i-1}}{2}$$

Let $\eta' = \frac{\eta \cdot k \cdot \Delta u}{2}$ we simplify (3.10) into

$$\frac{(\overline{S}_i + \overline{Q}_i - 2\overline{R}_i)}{(\Delta u)^2} - \frac{(\overline{O}_i + \overline{Q}_i - 2\overline{P}_i)}{4} - \gamma \left( \frac{\overline{P}_{i+1} - \overline{P}_{i-1} - \overline{Q}_{i+1} + \overline{Q}_{i-1}}{2} \right) - 4\gamma^2 \left( \frac{\overline{Q}_{i+1} + \overline{Q}_{i-1} - 2\overline{Q}_i}{(\Delta s\Delta v)^2} \right)$$

$$= \eta' (\overline{Q}_{i+1} - \overline{Q}_{i-1})$$

Note that $\eta'$ can be chosen so $\overline{S}_i$ is on a line (one freedom left). Similarly we minimize $|\overline{S}_i - \overline{R}_i|$ to fix $\eta'$ ($\overline{S}_i$ is accordingly fixed);

Let:

$$\tilde{a} = (\overline{Q}_i - 2\overline{R}_i) - \frac{(\overline{O}_i + \overline{Q}_i - 2\overline{P}_i)}{4} - \gamma \left( \frac{\overline{P}_{i+1} - \overline{P}_{i-1} - \overline{Q}_{i+1} + \overline{Q}_{i-1}}{2} \right) - 4\gamma^2 \left( \frac{\overline{Q}_{i+1} + \overline{Q}_{i-1} - 2\overline{Q}_i}{(\Delta s\Delta v)^2} \right)$$

$$\tilde{b} = \overline{Q}_{i+1} - \overline{Q}_{i-1}$$

$$\overline{S}_i + a = \eta' \tilde{b}$$

(3.11)

$$\Rightarrow \overline{S}_i = \eta' \tilde{b} - a$$
To minimize $|\eta \vec{b} - a - \vec{R}_i|$ we have

$$\eta' = \frac{(a + \vec{R}_i) \cdot \vec{b}}{|\vec{b}|^2}$$

(3.12)

Substituting (3.12) into (3.11), we get $\vec{S}_i$ as:

$$\vec{S}_i = \frac{(a + \vec{R}_i) \cdot \vec{b}}{|\vec{b}|^2} \vec{b} - \vec{a}$$

An interesting property of this method is that the surface extension result is independent of the step lengths $\Delta u$, $\Delta v$ and $\Delta s$. By repeatedly applying the above method, a new set of points can be obtained giving slice $\mathcal{S}$; the surface generated using slice $\mathcal{S}$ in addition to the previously available slices is $G^2$ continuous.

The above extension method above can’t be applied at end points $\vec{Q}_0$ and $\vec{Q}_{m-1}$. In this thesis, a natural extension method (as discussed in section 3.1) is applied extension at end points. To adapt a parameter extrapolation approach, a B-spline (or Bezier) curve is first fitted in the neighborhood of the end point: e.g., interpolate points $\overrightarrow{O}_0$, $\overrightarrow{P}_0$, and $\overrightarrow{Q}_0$ into a degree-2 Bezier curve, $B(u)$, with $u \in [0, 1]$. By using values of $t > 1$, the curve can be extrapolated smoothly beyond $\overrightarrow{Q}_0$. For completeness, the Bezier curve computation is summarized below.

$$B(u) = \overrightarrow{O}_0(1-u)^2 + \left[ -\overrightarrow{O}_0 + 2\overrightarrow{P}_0 - \overrightarrow{Q}_0 \right] 2u(1-u) + \overrightarrow{Q}_0 u^2$$

In which $B(0) = \overrightarrow{O}_0$, $B(0.5) = \overrightarrow{P}_0$, and $B(1) = \overrightarrow{Q}_0$.

To guarantee points $\overrightarrow{O}_0$, $\overrightarrow{P}_0$, $\overrightarrow{Q}_0$ and $\overrightarrow{R}_0$ are evenly distributed along $z$ axis, we have:

$$\overrightarrow{R}_0 = B(1.5)$$
\[
\overrightarrow{Q} = \frac{1}{4} \overrightarrow{Q}_0 + \frac{3}{2} (\overrightarrow{Q}_0 - 2\overrightarrow{P}_0 + \overrightarrow{Q}_0) + \frac{9}{4} \overrightarrow{Q}_0
\]

The extension at $\overrightarrow{Q}_{m-1}$ is similar.

**3.4.5 Examples of surface extension**

Figure 3.7 shows the outcome of extending the discrete data making a toe shape by one slice. Figure 3.8 shows the extension of the back part data for a shoe last. Figure 3.9 shows a series of slices, generated iteratively, extending a curved surface. In each of the examples, the original point cloud is in lighter colored dots, and the extension data points are bold.

![Figure 3.8 One-slice extension for toe part](image1)

![Figure 3.9 One-slice extension for rear part](image2)
Figure 3.10 Multiple slices extension of a curved surface
Chapter 4

BLEND OPTIMIZATION AND MVS-BASED INTERPOLATION

In the previous chapter, it was proposed that common operators such as morphing and surface extension can be blended to generate gap-interpolating surfaces with desirable properties. In this chapter, three different approaches based on this idea will be presented. First, user-controlled blend functions will be used to control interpolating surface shape (Section 4.3). Next, a surface energy based optimization scheme will be used to set the optimal blending weight matrix $W[k]$. Finally, a minimum curvature variation based approach will be introduced to interpolate surfaces for gap-filling. The first part of this chapter discusses the background on the energy- and curvature based functional that will be used for the latter two approaches.

4.1 Energy based functional (MES)

As described in chapter 3, energy based functional for triangulation is:

$$E_{\text{triangulation}} = \sum K^2$$

Or $$E_{\text{triangulation}} = \sum H^2$$

Or $$E_{\text{triangulation}} = \sum |\kappa_1| + |\kappa_2|$$

To apply MES to our problem, local triangulation is constructed at each point (except at points on edge). The curvature at point $S[i][j]$ ($j$ th point in $i$ th slice, $i = 0,1,\ldots,n-1$, $j = 1,2,\ldots,m-2$) is estimated by the four surrounding points $S[i][j-1]$, $S[i][j+1]$, $S[i-1][j]$, $S[i+1][j]$. A local triangulation is shown in
The mean curvature $H$ is calculated as described in Chapter 2, section 2.4.3.2. The functional $E_{\text{triangulation}} = \sum H^2$ will be used in this thesis to generate a minimum energy surface (MES).

4.2 Curvature variation based functional (CVS)

4.2.1 Local parameterization

Local surface fitting will be used to approximate the estimation of curvature at a point. Curvature variation of a target point $S[i][j]$ is estimated from the differential properties of this interpolating surface. This approach is easily generalized to improve accuracy by increasing the size of the neighborhood where the surface is fitted. In the following, the base case is developed, using the target point at which we need to compute curvature and its derivative, plus its eight neighboring points: the target point $S[i][j]$, two adjacent points in its slice $S[i][j-1]$, $S[i][j+1]$ and six corresponding points in adjacent slices $S[i\pm1][j-1]$, $S[i\pm1][j]$, $S[i\pm1][j+1]$. 

Figure 4.1 Local triangulation at point $S[i][j]$
A quadratic, parametric surface $P(u,v)$ is generated interpolating the nine points, with parametrization as shown in Figure 4.3.

The interpolated surface is

$$P(u,v) = [u^2 \quad u \quad 1] \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} v^2 \\ v \\ 1 \end{bmatrix}, \quad u \in [-1,1], v \in [-1,1]$$

where

$$a_{00} = \frac{(S[i-1][j-1] - 2S[i-1][j] + S[i-1][j+1]) + (S[i+1][j-1] - 2S[i+1][j])}{4}$$

$$a_{10} = \frac{(S[i-1][j-1] - 2S[i][j-1] + S[i][j+1] - (S[i-1][j+1] - 2S[i][j+1] + S[i+1][j+1])}{4}$$

$$a_{01} = \frac{S[i-1][j] - 2S[i][j] + S[i+1][j]}{4}$$

$$a_{11} = \frac{S[i+1][j-1] - S[i-1][j-1] - S[i+1][j+1] + S[i-1][j+1]}{2}$$

$$a_{12} = \frac{S[i+1][j] - S[i-1][j]}{2}$$

$$a_{20} = \frac{S[i][j-1] - 2S[i][j-1] + S[i][j+1]}{2}$$

$$a_{21} = \frac{S[i][j-1] - S[i][j+1]}{2}$$

$$a_{22} = S[i][j]$$

Figure 4.2 Parameterization for local surface fitting
4.2.2 Principal curvature variation based functional

A reasonable functional to approximate the MVS is [20]:

\[ V_{\text{surface}} = \iint_{\text{surface}} \left[ \left( \frac{dK_1}{de_1} \right)^2 + \left( \frac{dK_2}{de_2} \right)^2 \right] dA \] (4.1)

The principal curvature variation based functional is derived from (4.1). However, computing \( \frac{dK_i}{de_i} \) is expensive in the context of iterative algorithms. The following simplification can reduce this effort. We use curvature derivative of curve \( C_i(u) = P(u, k_i u) \ (i = 1, 2) \) to estimate \( \frac{dK_i}{de_i} \), in which direction \( du : dv = k_i \) is principal direction. See figures 4.3 and 4.4.

Figure 4.3 Line of principal direction in parametric space

Figure 4.4 Mapping line of principal direction in parametric space into real space
The principal directions \( k_i \) at the target point \( S[i][j] \) are the roots of equation

\[
(FN - MG)k^2 + (EN - LG)k + (EM - LF) = 0
\] (4.2)

In which \( E, F, G, L, M, N \) are coefficients of first and second fundamental forms [30].

\[
E = P_u \cdot P_u
\]

\[
F = P_u \cdot P_v
\]

\[
G = P_v \cdot P_v
\]

\[
L = P_{uu} \cdot (P_u \times P_u)
\]

\[
M = P_{uv} \cdot (P_u \times P_v)
\]

\[
N = P_{vv} \cdot (P_u \times P_v)
\]

Mapping the two lines \( v = k_iu \) onto surface \( P \) we can get curves \( P(u, k_i u) \), denoted by \( C_i(u) \). We can use the chain rule to compute the derivative of the curvature with respect to the arc length \( s \):

\[
\frac{d\kappa_i}{ds} = \frac{du}{ds} \frac{du}{ds}
\]

\[
\frac{d}{du} \left( \frac{C_i'(u) \times C_i''(u)}{|C_i'(u)|^3} \right) \]

\[
= \frac{[C_i'(u) \times C_i'''(u)][C_i'(u)]^2 - 3[C_i'(u) \times C_i''(u)]^2[C_i'(u) \cdot C_i''(u)]}{|C_i'(u)|^6[C_i'(u) \times C_i''(u)]}
\] (4.3)

From \( C_i(u) = P(u, k_i u) \) we have

\[
C_i'(u) \big|_{u=0} = k_i a_{i1} + a_{i2}
\]

\[
C_i''(u) \big|_{u=0} = 2k_i^2 a_{20} + 2k_i a_{11} + 2a_{02}
\]

\[
C_i'''(u) \big|_{u=0} = 6k_i^2 a_{0} + 6k_i a_{01}
\] (4.4)
Substituting (4.4) into (4.3), we get $\frac{d\kappa_i}{ds}$ for the target point $S[i][j]$. Substituting (4.3) into $\sum (\frac{d\kappa_1}{ds})^2 + (\frac{d\kappa_2}{ds})^2$ we can get principal curvature variation based functional, denoted as PCVF. It should be pointed out that at some points, principal directions $k_i$ cannot be found. Such points are called umbilical points. In optimization $\frac{d\kappa_i}{ds}$ are set as 0 at umbilical points.

4.2.3 Maximum curvature change rate based functional

As the value of $k$ is changed, the rate of change of curvature $\frac{d\kappa}{ds}$ of curve $P(u,k,\mu)$ varies smoothly (for a well behaved surface). The following graph shows a typical relationship between $\left| \frac{d\kappa}{ds} \right|$ and $k$.

![Graph showing curvature change rate vs. k](image)

Figure 4.5 $k$ vs. curvature change rate $\left| \frac{d\kappa}{ds} \right|$.

From figure 4.5 it can be seen $\left| \frac{d\kappa}{ds} \right|$ has a maximum value. Denote maximum $\left| \frac{d\kappa}{ds} \right|$ as $\left| \frac{d\kappa}{ds} \right|_{\text{max}}$. The maximum curvature change rate based functional (MCFV) can
be defined as:

\[ V_{\text{surface}} = \sum \left| \frac{d\kappa}{ds} \right|_{\text{max}} \]

Mapping \( v = ku \) onto surface \( P \) we can get a curve, denoted by \( C(u) \):

\[ C(u) = k^2 u^4 a_{00} + k^2 u^3 a_{10} + k^2 u^2 a_{20} + ku^2 a_{01} + ku^2 a_{11} + kua_{21} + u^2 a_{02} + ua_{12} + a_{22} \quad (4.5) \]

Since \( C(u) \) is \( C^\infty \), and curvature change rate \( \frac{d\kappa}{ds} \) is only related to 3\textsuperscript{rd} and lower derivatives, \( \frac{d\kappa}{ds} \) is \( C^1 \) continuous at point \( C(u)|_{u=0} \), so we only take \( C(u), u \geq 0 \) part into consideration. For \( C(u), u < 0 \) part, \( \frac{d\kappa}{ds} \) is identical. Using line segment to approximate arc length \( s \):

\[ s(u) = |C(u) - C(0)| \]

\[ \Rightarrow s(u) = |k^2 u^4 a_{00} + k^2 u^3 a_{10} + k^2 u^2 a_{20} + ku^2 a_{01} + ku^2 a_{11} + kua_{21} + u^2 a_{02} + ua_{12}| \]

Since only a small neighborhood of \( C(u)|_{u=0} \) is considered, we can ignore high order terms, and so \( s(u) \approx |kua_{21} + ua_{12}| \). So we get:

\[ u = \frac{s}{|ka_{21} + a_{12}|} \]

Substituting into (4.5) and differentiating thrice:

\[ \left. \frac{d\kappa}{ds} \right|_{s=0} = \left. \frac{d^3 C(s)}{ds^3} \right|_{s=0} \]

\[ = \frac{k^2 a_{10} + ka_{01}}{|ka_{21} + a_{12}|^3} \]

Accordingly,

\[ \left| \frac{d\kappa}{ds} \right| = \frac{|k^2 a_{10} + ka_{01}|}{|ka_{21} + a_{12}|^3} \quad (4.6) \]

We need to compute the value of \( k \) that maximizes \( \frac{d\kappa}{ds} \). For degree three Bezier
interpolated surfaces, this is conveniently computed as follows. Writing:

\[ f(k) = \left| \frac{d\kappa}{ds} \right|^2 = \frac{|k^2a_{10} + ka_{01}|^2}{|ka_{21} + a_{12}|^6} \]

\[ A = a_{10} \cdot a_{10}, \quad B = a_{10} \cdot a_{01}, \quad C = a_{01} \cdot a_{01}, \quad D = a_{12} \cdot a_{21}, \quad E = a_{21} \cdot a_{12}, \text{ and } \quad F = a_{12} \cdot a_{12} \]

We have:

\[ f(k) = \left| \frac{d\kappa}{ds} \right|^2 = \frac{k^4A + 2k^3B + k^2C}{(k^2D + 2kE + F)^3} \]

To maximize,

\[ \frac{d}{dk} f(k) = 0 \]

\[ \Rightarrow ADk^4 + (3BD - AE)k^3 + (2CD - 2AF)k^2 + (CE - 3BF)k - CF = 0 \quad (4.7) \]

This is a degree four polynomial with a closed form solution, yielding four roots: \( k_1, k_2, k_3 \) and \( k_4 \). Select the value that gives the maximum point as \( k^* \), substitute into (4.6) to get the required MCVF:

\[ \left| \frac{d\kappa}{ds} \right|_{\text{max}} = \frac{k^* a_{10} + k^* a_{01}}{|k^* a_{21} + a_{12}|} \]

With the above background, we now develop the different gap-filling interpolation functions in the following sub-sections.

**4.3 Convex blending of morphing and extension**

As described in chapter 3, a simple mechanism to generate the interpolating surface is by a user-defined set of blending weights \( W[n] \). This completely defines the
interpolating surface as a convex blend of morphing and extension. Without losing
generality, assume first intermediate slice \( S[0] \) is next to \( P[3] \), the last slice of \( P \);
last intermediate slice \( S[n-1] \) is next to \( Q[3] \), the last slice of \( Q \);
Intermediate slice \( S[k] \) \((k = 0, 1, ..., n-1)\) is combination of extension slice and
morphing slice:

In case that \( n \) is odd number:

\[
S[k][i] = \begin{cases} 
W[k] \cdot P\_extension[i] + (1-W[k]) \cdot P\_morphing[i] & k < \frac{n}{2} \\
\frac{1}{2}(P[3][i]+Q[3][i]) & k = \frac{n}{2} \\
W[k] \cdot Q\_extension[i] + (1-W[k]) \cdot Q\_morphing[i] & k > \frac{n}{2}
\end{cases}
\]

In case that \( n \) is even number:

\[
S[k][i] = \begin{cases} 
W[k] \cdot P\_extension[i] + (1-W[k]) \cdot P\_morphing[i] & k < \frac{n}{2} \\
W[k] \cdot Q\_extension[i] + (1-W[k]) \cdot Q\_morphing[i] & k \geq \frac{n}{2}
\end{cases}
\]

Note that at the two ends of the gap, \( W[k] \) should be \( \approx 1 \) to provide continuity at
the interface; in the middle part, \( W[k] \) should be \( \approx 0 \), (i.e., morphed surface
assumes more weight) to avoid discontinuity in middle slice. These observations
allow a simple set of guidelines for creating blending functions as follows.

4.3.1 Approximation using poly line

The simplest method to assign \( W[n] \) is to use a piecewise linear function, or a
poly-line (see figure 4.6). Denote the lowest point as \((u,v), \ 0 \leq u \leq n-1
and 0 \leq v \leq 1\). Then three points \((0,1), \ (u,v) \) and \((n-1,1)\) define two line segments
intersect at point \((u,v)\). Figure 4.7 shows an example with \( n = 9, \ u = 5 \) and \( v = 0 \). By
varying the lowest point, the relative influence of the left and right surfaces that are being interpolated can be controlled.

Figure 4.6 Poly line by (0,1), (u, v) and (n-1,1)

![Graph depicting a poly line with points (0,1), (u, v), and (n-1,1).]

Figure 4.7 Example of poly line that \( n = 9 \), \( u = 5 \) and \( v = 0 \)

4.3.2 Approximation using quadric Bezier curve

Similar to 4.3.1, \( W[k] \) can be approximated by a degree two Bezier curve, with three control points at (0,1), \((u, v)\) and \((n-1,1)\). To guarantee \( W[k] \in [0,1] \), set \( 0 \leq u \leq n-1 \) and \(-1 \leq v \leq 1\). Figure 4.8 shows an example where \( n = 19, u = 5 \), and \( v = -1 \).
4.4 MES and MCVF optimization

4.4.1 Problem formulation

Finally, we discuss how to generate the interpolating surface automatically, by setting $W[k]$ either a Minimum energy formulation (MES), or a Minimum curvature change formulation (MCVF). The evaluation of the selected optimization functional was derived in section 4.2. In this section, a simple method to search for the optimum configuration, $W'[k]$ is discussed. For simplicity, $F(u, v)$ will be used to denote the selected fairness functional (MES, PCVF or MCVF). The problem formulation for optimizing the poly-lines is:

Min $F(u, v)$

S.t. $0 \leq u \leq n - 1$

$0 \leq v \leq 1$  

For approximation using Bezier curve:

Min $F(u, v)$

S.t. $0 \leq u \leq n - 1$

$-1 \leq v \leq 1$  

(a)
Both (a) and (b) are 2D, nonlinear, constrained optimization problems due to the nature of $F(u, v)$.

4.4.2 Determining start point by sampling

Since $F(u, v)$ is a complex function, it is difficult to solve (a) and (b) analytically. A searching method is applied to get an approximate solution. It is possible that $F(u, v)$ has more than one stationary point, so the search may miss the global optimal. To decrease this probability, a sampling method is applied before executing searching to determine the start point (initial triangle). This is summarized as follows.

1. Take $N_u \cdot N_v$ sample points; these sample points are uniformly distributed in $u$ axis and $v$ axis.

2. Compute $F(u_i, v_j), i = 0, 1, ..., N_u - 1, j = 0, 1, ..., N_v - 1$ and find the minimum point $F(u_{i*}, v_{j*})$ among them.

3. Determine start point (initial triangle):

If the minimum (among the sampled $N_u \cdot N_v$ points) is in the interior of the grid, i.e. if $0 < i^* < N_u - 1$ and $0 < j^* < N_v - 1$:

the initial triangle is $(u_{i^{*}+1}, v_{j^{*}-1}), (u_{i^{*}}, v_{j^{*}+1})$;

If the minimum point is on the boundary of the grid, i.e. if $0 < i^* < N_u - 1$ and $j^* = \begin{cases} 0 \\ N_v - 1 \end{cases}$;

the initial triangle is $(u_{i^{*}+1}, v_{j^{*}}), (u_{i^{*}}, v_{j^{*}+1})$ when $j^* = 0$;

$(u_{i^{*}+1}, v_{j^{*}}), (u_{i^{*}}, v_{j^{*}+1})$ when $j^* = N_v - 1$;
If \( i^* = \begin{cases} 0 & \text{and } 0 < j^* < N_v - 1 \quad \text{(minimum point is on edge):} \\ N_u - 1 & \end{cases} \)

the initial triangle is \((u_{i^*}, v_{j^*+1}), (u_{i^*+1}, v_{j^*}) \) when \( i^* = 0 \);
\((u_{i^*}, v_{j^*+1}), (u_{i^*}, v_{j^*}) \) when \( j^* = N_v - 1 \);

_in case that minimum point is on a corner:_

If \( i^* = 0 \) and \( j^* = 0 \), the initial triangle is \((u_0, v_0), (u_0, v_1) \) and \((u_1, v_0) \);

If \( i^* = 0 \) and \( j^* = N_v - 1 \), the initial triangle is \((u_0, v_{N_v-1}), (u_0, v_{N_v-2}) \) and \((u_1, v_{N_v-1}) \);

If \( i^* = N_u - 1 \) and \( j^* = 0 \), the initial triangle is \((u_{N_u-1}, v_0), (u_{N_u-2}, v_0) \) and \((u_{N_u-1}, v_1) \);

If \( i^* = N_u - 1 \) and \( j^* = N_v - 1 \), the initial triangle is \((u_{N_u-1}, v_{N_v-1}), (u_{N_u-2}, v_{N_v-1}) \) and \((u_{N_u-1}, v_{N_v-2}) \).

### 4.4.3 Multidirectional search method

Multidirectional search method [31] is an iterative search method without requiring derivatives. The search is initialized with three points at the vertices of a triangle. Denote these initial points as \( v_0, v_1, \) and \( v_2 \), where \( v_0 \) gives smallest value of objective function among these three points. The algorithm is summarized below.
Algorithm:

Initialize: Set triangle $v_0 - v_1 - v_2$ as current triangle.

a. Reflect current triangle through vertex $v_0$, new triangle is $v_0 - r_1 - r_2$;

b. Examine triangle $v_0 - r_1 - r_2$: evaluate $F(u, v)$ at $r_1, r_2$

   If $r_1$ or $r_2$ improves on $v_0$, expand $v_0 - r_1 - r_2$ to $v_0 - e_1 - e_2$ by a factor $f > 1$.

   If $e_1$ or $e_2$ is the best point, update $v_0 - e_1 - e_2$ as current triangle, keeping $v_0$ as best point;

   If neither $e_1$ nor $e_2$ is the best point, update $v_0 - r_1 - r_2$ as current triangle, keeping $v_0$ as best point;

   If neither $r_1$ nor $r_2$ is the best point, contract $v_0 - r_1 - r_2$ to $v_0 - c_1 - c_2$ by factor $s < 1$, update $v_0 - c_1 - c_2$ as current triangle, keeping $v_0$ as best point;

c. If area of current triangle is smaller than a preset threshold, report the centroid of current triangle as solution;
Otherwise, return to step a.

In the above algorithm, the factors $f$ and $s$ can be statically set (e.g. $f = 2$, $s = 0.5$), or they can be adaptively set depending on the ratio of the evaluation of $F(u, v)$ at $v_0$ and the better of the objective values at the remaining two vertices of the current solution.

The operators described in this chapter have been implemented, and some examples of the use of these operators will be presented in the next chapter.
Chapter 5

EXAMPLES

5.1 Examples

Three different examples are presented, including blending between two planar surfaces, one planar surface and one curved surface, and two curved surfaces. Application of the method to shoe last design is also discussed. The examples are generated using a single processor PC, and programmed using Visual Basic programming language. The images are generated by using the display utilities of CATIA™, a commercial CAD system. The functions can easily be incorporated as shape operators within CATIA via the Visual Basic API interface.

5.1.1 Approximation using poly line

5.1.1.1 Blending between 2 planar surfaces

In this example, each slice contains 180 points, and a total of 19 intermediate slices are generated in the gap. The blending methods generate point clouds, but for visual clarity, interpolated tessellated (using triangulation) surfaces are also shown for the data.

Figure 5.1 original surfaces: two planar surfaces
Figure 5.2 Blending surface between 2 planar surfaces:

(a) MES surface  (b) MCVF surface  (c) PCVF surface
Figure 5.3 Comparison of PCVF/MCVF/MES surfaces (the right image is an expanded view of the circled region)

Figure 5.4 W/[i] for optimized MES, MCVF, and PCVF surface

In figure 5.4, the three series refer to optimized W/[i] values for the following:

Series 1: MES; Series 2: MCVF; Series 3: PCVF

5.1.1.2 Blending between planar surface and curved surface

Same as in example 1, each slice on the disjoint surfaces contains 180 points, and a total of 19 slices are generated in the gap.
Figure 5.5 Original surfaces: planar surface and curved surface

Figure 5.6 Blending surface between: planar surface and curved surface:

(a) MES surface  (b) MCVF surface  (c) PCVF surface
Figure 5.7 Comparison of PCVF/MCVF/MES surfaces

Figure 5.8 W [i] for optimized MES, MCVF, and PCVF surface

In figure 5.8, the three series refer to optimized W[i] values for the following:

Series 1: MES; Series 2: MCVF; Series 3: PCVF

5.1.1.3 Blending between 2 curved surfaces

In the following example, both interpolated surfaces are curved. The data sets are similar to the previous examples.
Figure 5.9 Original surfaces: two curved surfaces

(a)

(b)

(c)

Figure 5.10 Blending surface between two curved surfaces:

(a) MES surface  (b) MCVF surface  (c) PCVF surface

Figure 5.11 Comparison of PCVF/MCVF/MES surfaces
In figure 5.12, the three series refer to optimized $W[i]$ values for the following:

Series 1: MES; Series 2: MCVF; Series 3: PCVF

5.1.1.4 Shoe-last example

In this example, the different methods are applied to real life data for shoe lasts. In the example, the back part and toe part geometry of two different lasts are used, with the gap geometry being generated by interpolation.

Figure 5.13 Original surfaces: shoe last
Figure 5.14 Blending surface between two shoe last surfaces:

(a) MES surface  (b) MCVF surface  (c) PCVF surface
Figure 5.15 Comparison between MES (a) and PCVF (b) surfaces

Figure 5.16 $W[i]$ for optimized MES, MCVF, and PCVF surface

In figure 5.16, the three series refer to optimized $W[i]$ values for the following:

Series 1: MES; Series 2: MCVF; Series 3: PCVF

5.1.2 Approximation using quadric Bezier curve

5.1.2.1 Blending between 2 planar surfaces

In this example, each slice contains 180 points, and a total of 19 intermediate slices are generated in the gap. The blending methods generate point clouds, but for visual clarity, interpolated tessellated (using triangulation) surfaces are also shown for the data.
Figure 5.17 original surfaces: two planar surfaces

(a)

Figure 5.18 Blending surface between 2 planar surfaces:

(a) MES surface  (b) MCVF surface  (c) PCVF surface
Figure 5.19 Comparison of PCVF/MCVF/MES surfaces

Figure 5.20 $W_i$ for optimized MES (series 1), MCVF (series 2) and PCVF (series 3)

5.1.2.2 Blending between planar surface and curved surface

Same as in example 1, each slice on the disjoint surfaces contains 180 points, and a total of 19 slices are generated in the gap.
Figure 5.21 Original surfaces: planar surface and curved surface

Figure 5.22 Blending surface between: planar surface and curved surface:

(a) MES surface          (b) MCVF surface          (c) PCVF surface
5.1.2.3 Blending between 2 curved surfaces

In the following example, both interpolated surfaces are curved. The data sets are similar to the previous examples.
Figure 5.25 Original surfaces: two curved surfaces

Figure 5.26 Blending surface between two curved surfaces:

(a) MES surface    (b) MCVF surface    (c) PCVF surface
5.1.2.4 Shoe-last example

In this example, the different methods are applied to real life data for shoe lasts. In the example, the back part and toe part geometry of two different lasts are used, with the gap geometry being generated by interpolation.
Figure 5.29 Original surfaces: shoe last

Figure 5.30 Blending surface between two shoe last surfaces:

(a) MES surface  
(b) MCVF surface  
(c) PCVF surface
5.3 Justification of polyline/quadric Bezier approximation

Notice that ideally, the blending weight $W[i]$ for each intermediate layer should be treated as an independent variable. However, this results in a fairly large optimization problem, which, using MATLAB, requires several hours to converge to a solution. The computation was significantly accelerated (by over two orders of magnitude) with the use of simplified blending forms such as the poly-line or the degree-2 Bezier. In each of these, the set of $n (=19$, one for each intermediate layer in our examples)
variables in the optimization is reduced to two. We now investigate how good this approximation is. Using MATLAB, the complete 19-dimensional optimization model is solved for several examples, and the resulting $W[i]$ and surface functional values are compared with the approximate models.

5.3.1 Blending between two planar surfaces

Figure 5.33 shows optimization result of 19-dimensional optimization and polyline/quadric Bezier approximation.

![Graph showing comparison of optimization results](image)

Figure 5.33 Comparison of optimization result: case of two planar surfaces

Series 1: 19-dimensional optimization, Series 2: Quadric Bezier approximation (2-dimensional optimization), and Series 3: polyline approximation (also 2-dimensional).

Clearly, the form of the blending function is similar. The table below compares the functional values for these cases.
<table>
<thead>
<tr>
<th></th>
<th>PCVF value</th>
<th>%-deviation from 19-dimensional value</th>
<th>Computation time (units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-dimensional optimization</td>
<td>20.5605</td>
<td>-</td>
<td>123</td>
</tr>
<tr>
<td>Quadric Bezier curve</td>
<td>21.3097</td>
<td>3.64%</td>
<td>1</td>
</tr>
<tr>
<td>Polyline</td>
<td>24.5524</td>
<td>19.41%</td>
<td>1</td>
</tr>
</tbody>
</table>

### 5.3.2 Blending between planar and curved surfaces

Figure 5.34 shows optimization result of 19-dimensional optimization and polyline/quadric Bezier approximation.

![Graph](image)

Figure 5.34 Comparison of optimization result: case of planar and curved surfaces

Series 1: 19-dimensionsal optimization, Series 2: Quadric Bezier approximation (2-dimensional optimization), and Series 3: polyline approximation (also 2-dimensional).

Again, the form of the blending function is similar. The table below compares the functional values for these cases.
<table>
<thead>
<tr>
<th></th>
<th>PCVF value</th>
<th>%-deviation from 19-dimensional value</th>
<th>Computation time (units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-dimensional optimization</td>
<td>56.5667</td>
<td>-</td>
<td>154</td>
</tr>
<tr>
<td>quadric Bezier curve</td>
<td>66.2744</td>
<td>9.71%</td>
<td>1</td>
</tr>
<tr>
<td>polyline</td>
<td>64.1797</td>
<td>7.91%</td>
<td>1</td>
</tr>
</tbody>
</table>

### 5.3.3 Blending between two curved surfaces

Figure 5.35 shows optimization result of 19-dimensional optimization and polyline/quadric Bezier approximation.

![Figure 5.35 Comparison of optimization result: case of two curved surfaces](image)

Series 1: 19-dimensional optimization, Series 2: Quadric Bezier approximation (2-dimensional optimization), and Series 3: polyline approximation (also 2-dimensional).

As before, the blending functions have similar shapes.

<table>
<thead>
<tr>
<th></th>
<th>PCVF value</th>
<th>%-deviation from 19-dimensional value</th>
<th>Computation time (units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-dimensional optimization</td>
<td>164.0957</td>
<td>-</td>
<td>467</td>
</tr>
<tr>
<td>quadric Bezier curve</td>
<td>191.1611</td>
<td>16.49%</td>
<td>1</td>
</tr>
<tr>
<td>polyline</td>
<td>202.6603</td>
<td>23.46%</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 5.36 shows the difference between 19-D optimization and 2-D optimization:

![Diagram showing difference between 19-D and 2-D optimization]

Figure 5.36 Difference between 19-D and 2-D optimization

5.3.4 Shoe last example

Figure 5.37 shows optimization result of 19-dimensional optimization and polyline/quadric Bezier approximation.
Figure 5.37 Comparison of optimization result: shoe last surfaces

Series 1: 19-dimensional optimization, Series 2: Quadric Bezier approximation (2-dimensional optimization), and Series 3: polyline approximation (also 2-dimensional).

As before, the blending functions have similar shapes. A comparison of functional values is shown below, and Fig 5.38 shows the difference in the corresponding surfaces.

<table>
<thead>
<tr>
<th></th>
<th>PCVF value</th>
<th>%-deviation from 19-dimensional value</th>
<th>Computation time(units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-dimensional optimization</td>
<td>388.4832</td>
<td>-</td>
<td>987</td>
</tr>
<tr>
<td>Quadric Bezier curve</td>
<td>447.9693</td>
<td>15.31%</td>
<td>1</td>
</tr>
<tr>
<td>Polyline</td>
<td>461.7789</td>
<td>18.87%</td>
<td>1</td>
</tr>
</tbody>
</table>
5.4 Discussion

In this chapter, the different methods for interpolation discussed in Chapters 3 and 4 are compared by the use of several examples. The examples give evidence that the methods developed in this work can be used to make practical interpolation functions in practice, and can certainly be adopted for the particular problem of shoe last design that was described in Chapter 1. The main point of interest from the development is that PCVF and MES yield interpolation surfaces that are significantly different – therefore even for aesthetic design, it is beneficial for the designer to have access to both these methods.
Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusions

This project was motivated by a practical problem related to CAD of footwear. The abstracted shape interpolation formulation has applications in many related areas. Several existing methods for shape interpolation were studied, and two new solution approaches were devised. The first method utilizes a convex blend of morphing and $G^2$ continuous extension. Based on the continuous form of $G^2$ extensions, a discrete format is developed that can generate interpolating data in the gap as an arbitrarily dense cloud of points. The second method is based on a new fairness functional. Past research has looked at generating fair surfaces using minimum energy (MES) as well as minimum curvature formulations. The main contributions include the development of two new faring functionals based on the minimum variation of curvature of the surface (MVS). The first minimizes curvature change rate along principal directions; the second minimizes the maximum curvature change rate. The main point of interest from the development is that PCVF and MES yield interpolation surfaces that are significantly different. Therefore, for aesthetic design, it is beneficial for the designer to have access to both these methods.

6.2 Future Work

There are several aspects of the work that remain to be explored. The methods presented in thesis work for structured data sets. In practice, it is useful to extend it to
arbitrary point clouds. There are several existing techniques that can re-organize a
mesh to create a required topology within any given tolerance level. A possible
approach is to first create a tessellation of a given unstructured point cloud, followed
by a re-sampling along parallel cross-sections. This approach is computationally
efficient, and easy to implement.

Another important extension is the construction of a blending surface between
surfaces with different topology, or blending between more than two surfaces. The
linear morphing followed by MVS formulation can be easily adopted for this
problem. The main problem will the computation of the principal curvatures at points
whose neighborhood has a points that are arranged in a form that is difficult to
interpolate using a local Bezier patch.

Another potential research direction is to construct constrained blending surfaces.
The method described in this thesis is unconstraint; it cannot work for a problem
where, for instance, a set of guide curves are provided along the gap, and the
blending surface must pass through these curves.
References


