國立交通大學

多媒體工程研究所

碩 士 論 文

以 圓 錐 可 展 曲 面 擬 合 網 格 之 演 算 法

Approximating Meshes using Cone Spline Developable Surfaces

研 究 生：劉育碩

指導教授：莊榮宏　教授

中 華 民 國　九 十 六　年 八 月

利用圓錐可展曲面擬合三角網格之演算法

研究生 ： 劉育碩　　指導教授 ： 莊榮宏 博士

國立交通大學多媒體工程研究所

摘　　要

可展曲面可以應用於衣服或紙模型的設計。可展曲面具有易於生產，無扭曲等優點，但通常需要有經驗的專家才能設計可展曲面。舊有的可展曲面擬合演算法通常只能擬合近似可展之網格[3, 6, 8]，或限制擬合曲面為受限的可展曲面[5, 7, 9]。我們在論文中，提出了更通用的演算法，能利用圓錐可展曲面來擬合網格。我們的演算法為一反覆擬合之架構，在每一次迴圈中，我們執行區域擴張並最佳化擬合之圓錐可展曲面。不同於舊有的方法利用單一的圓錐曲面，我們利用多段的圓錐曲面擬合網格以降低誤差，並保證同一區塊內，圓錐之間的C0連續性，因此我們的演算法可以利用較少的區塊，來更精確的擬合網格。

Approximating Meshes using Cone Spline Developable Surfaces

Student: Yu-Shuo Liu Advisor: Dr. Jung-Hong Chuang

Department of Computer Science
National Chiao Tung University

ABSTRACT

Developable surfaces have various applications like clothes or paper craft models design. They have many advantages like simple to construct and no distortion. Designing developable surfaces is not trivial though, so various developable surfaces approximation methods have been proposed. Previous developable surfaces approximation methods either only approximate near developable surfaces [3, 6, 8] or only use restrictive surfaces [5, 7, 9] as approximation surfaces. We propose a more general method that use cone spline surfaces as approximation surfaces. Our method is a iterative approach, in each iteration, our algorithm executes region growing and approximation surface optimization. Unlike previous approaches that use a single conic as proxy surface in the optimization phase, our method use multiple segments of conic to further reduce the error and guarantee C0 continuity between the conics in the same cone spline surface. As a result, our algorithm is more accurate and usually generates fewer patches than previous methods.


E
S
A
1896

Acknowledgments

I would like to thank my advisors, Dr. Jung-Hong Chuang, and Wen-Chieh Lin for their inspirations and guidance.

I would especially thank Tan-Chi Ho, who spared his time to discuss with me and gave me a lot helpful comments and advices.

Thanks to my colleagues in CGGM lab: Ji-Wen Chon, Chia-Lin Ko, Ya-Ching Chiu, Yung-Cheng Chen, Yueh-Tse Chen, Chih-Hsiang Chang, Hsin-Hsiao Lin, Kuang-Wei Fu, and Ying-Tsung Li for their assistances and discussions.

Special thanks to my senior colleagues Young-Cheng Cheng, we spent endless nights discussing software engineering, computer graphics and all other interesting stuff in game industry. Thanks to Yi-Chun Lin, who taught me his philosophy of life, and various special attacks in Ninja Gaiden.

Finally I would like to thank my family. Without their support I could not come through all the pain in my life.


E
S
A
1896

Contents

List of Figures


E
S
A
1896

CHAPTER 1

Introduction

1.1 Brief of Developable Surfaces

Developable surfaces are surfaces that can be unfolded into planes without any distortion. They are appealing because of their simplicity. During construction, they can be obtained by only bending a plane without any stretching or contraction, which also means the surface has fewer weak spots. Every objects made from metal, leather, wood, paper or cloth sheets without stretch are developable surfaces. Although simple in concept and presented in everyday's life, designing and modeling of developable surfaces are difficult and require experienced experts. Such difficulty often limits the applications of developable surfaces.

1.2 Motivation

Developable surfaces approximation is studied extensively in CAD realm, but in CAD realm the original surfaces are often assumed to be very close to developable surfaces, and the approximation surfaces are often assumed to have C1 or G1 continuity [3, 6, 8], which restrict these algorithms applying to general meshes.

Making paper craft models is another application of developable surfaces. Recently there are few papers propose algorithms to generate paper craft parts from meshes automatically. These algorithms use restrictive surfaces like conic surfaces [5, 9], or long triangle strips [7] as approximation surfaces. The paper craft models generated from these algorithms either have too many parts or too complex boundaries so they are still far from ideal.

Because previous algorithms either restrict the original surfaces or the approximation surfaces, we try to close the gap by relaxing the restrictions of approximation surfaces so our algorithm can be applied on near-developable surfaces as well as general meshes. And when applied on general meshes, our algorithm can fit the surface more accurately than previous paper crafting algorithms.

1.3 Contributions

Our algorithm use cone spline surfaces as approximation surfaces. Because cone spline surfaces are more general than conic surfaces and have more degree of freedom, we can approximate more complex surfaces. Compare our method to previous methods that use conics as approximation surfaces [5, 9], our method is often more efficient, which means that we can approximate a complex patch using only a single cone spline surface, where previous methods require multiple conics. And we also guarantee the C0 continuity of the conics in the same cone spline surface which can't be guaranteed in previous method [9]. Because our method suffer from the unstable boundaries between multiple cone spline surfaces as in previous method [9], we also proposed a novel boundaries optimization method which is based on Hermite interpolation [6]. Compared to the methods presented in CAD realm [3, 6, 8], we relax the constraint of the C1 and G1 continuity and only preserve C0 continuity, as a result our algorithm can approximate near developable surfaces as well as general surfaces.

1.4 Thesis Organization

In chapter 2 we first introduce the mathematical background of developable surfaces, then discuss related works of our method. Chapter 3 describes our algorithm framework. Chapter 4 shows the experiment results and discuss some problems in our method, and chapter 5 is the conclusions and discusses of future work.


E
S
A
1896

CHAPTER 2

Related Work

In this chapter, a review on related works of our method will be given. Section 2.1 introduces the mathematical background of developable surfaces. Section 2.2 discusses representation and approximation algorithms of developable surfaces.

2.1 Mathematical Background

In this section we discuss the mathematical background of developable surfaces. Section 2.1.1 discuss the properties of developable surfaces from the perspective of differential geometry. In Section 2.1.2 we discuss how duality principle in projective geometry simplifies the representation of developable surfaces.

2.1.1 Developable Surfaces in Differential Geometry

Ruled surfaces are surfaces such that each points on the surface has one tangent line (ruling) pass through it such that the tangent line is also lies on the surface. Given a one-parameter family of straight lines $\{\alpha(t), \omega(t)\}$, for each t, $\alpha(t)$ is a point and $\omega(t)$ is a vector, a ruled

surface can be represented as

$$x(t, v) = \alpha(t) + v\omega(t), t \in I, v \in R. \tag{2.1}$$

In equation 2.1, the curve generate by $\alpha(t)$ is called a $directrix$, and the straight line generated by vector $\omega(t)$ is called a $ruling$. Equation 2.1 has the geometric interpretation that a rule surface is generated by a straight line parallel to $\omega(t)$, and pass through $\alpha(t)$, sweep through the directrix.

Developable surfaces are ruled surfaces with the additional constraint

$$(\omega, \omega', \alpha') = 0. \tag{2.2}$$

The constraint can be interpreted as that, for the directrix $\alpha(t)$ of a developable surface **x**, such that $\alpha(t)$ is on a surface **S**, each points on the same ruling have the same tangent plane $T_{\alpha(t)}(S)$ of the surface **S**. The interpretation implies that each rulings of the developable surfaces **x** are the limiting positions of the intersection of neighboring tangent planes of the family $\{T_{\alpha(t)}(S)\}$, and the developable surface **x** is called the envelope of the family of tangent planes of S along $\alpha(t)$.

2.1.2 Developable Surfaces in Projective Geometry

Using duality principle in projective three space, developable surfaces representation can be greatly simplified. In this section we introduce the basic concept of projective space and the duality principle. Then we show the the dual form of developable surfaces in projective three space.

2.1.2.1 Projective space

Let the coordinates of a point **P** in the projective three-space be (x_1, x_2, x_3, x_4), which are also known as the homogeneous coordinates of a point. The corresponding point **P** in the Euclidean space is obtained by

$$(x, y, z) = (\frac{x_1}{x_4}, \frac{x_2}{x_4}, \frac{x_3}{x_4}) \tag{2.3}$$

The points at infinity can be expressed by setting $x_4 = 0$, in which case the first three coordinates define the direction of a point at infinity. Finite points can be represented using their Cartesian coordinates and unity as their fourth coordinate. The coordinates (x_1, x_2, x_3, x_4) and $(kx_1, kx2, kx_3, kx_4)$, where k is nonzero real number, represent the same point **P** in Euclidean space.

2.1.2.2 Duality

In projective space, there is an important principle called duality, which means that every theorems in projective space actually have two versions. For example, in projective three-space, points and planes are dual to each other, which means the coordinates (x_1, x_2, x_3, x_4) could be a point or a plane, and every theorem for the point have a dual theorem for the plane. For example, for three points a, b and c being co-planar and the plane is D, the condition must be satisfied is that (a, b, c) = 0. The dual statement of above theorem is that if three planes A, B and C (A, B and C have the same coordinates with a, b and c respectively) intersect at one point, they must satisfy the condition (A, B, C) = 0, and the intersection point is d.

In section 2.2, we see that a developable surface can be represented as a one parameter family of tangent planes. In projective three-space, each tangent planes are dual to a points, so the dual representation of a developable surface in projective three-space is a curve. This property inspires many recent developable surfaces representation methods.

2.2 Developable Surface Representations

The representation of curves and surfaces in NURBS form is standard in CAD realm. Earlier NURBS developable surfaces representations[1] methods treat the developable surfaces as ruled surfaces then derive conditions that have to be imposed additionally in order to achieve developability. These additional conditions however, are complex nonlinear equations, which restrict the practical usage of these methods.

Bodduluri and Ravani [2] first proposed the dual algorithm. They consider a developable

rational surfaces as a rational curve in dual projective space. This dual consideration transformed the original problem to curves representation problem. They choose the directrix of the developable surface as the dual curve and compute the tangent plane, then the tangent planes are represented in Bézier or B-spline form. The problem of their algorithm is that the developable surfaces are represented in 1D Bézier or B-spline form, not in standard tensor product form. Because they choose the directrix as the dual curve, their algorithm can not be applied to cylinders or cones, in such cases, the differential of the directrix degenerate and their algorithm fail.

Pottmann and Farin [8] extended Bodduluri and Ravani's work by generalizing the dual approach so that developable surfaces can be represented in standard tensor product form. Their method first construct the developable surfaces in 1D B-spline form, then intersect the surface with two other planes in projective space. The intersection of the developable surface and the two planes are two curves, connect the two curves at corresponding parameters forms the tensor product form of the developable surface. Because they don't use the directrix to compute the tangent planes, their algorithm can represent cylinders or cones which can't be represented in Bodduluri and Ravani's algorithm.

The problem of the dual approaches is that during the design process, users are controlling control planes instead of control points of the NURBS surface which is not very intuitive. All previous developable surfaces representation methods can only represent $(1, n)$ developable surfaces, general (m, n) developable surfaces representation is still an open problem. Such constraint limit the applications of previous methods, for example, paper craft models often have many cuts that singular points in a patch, which can't be accomplished by previous methods.

Because it's hard to directly approximate a surface by general NURBS developable surfaces, cone spline surfaces are often used instead. Cone spline surfaces are piece wise conic surfaces, which are more suitable for local approximation.

Leopoldseder and Pottmann [6] proposed Hermite interpolation method for the conic surfaces. A Hermite element of a conic surface is a ruling plus the tangent plane pass though the ruling. Leopoldseder and Pottmann first derive the interpolation solutions of the one parame-

ter family of the two G1 connected (have the same tangent plane between them) conics with the boundaries being two given Hermite elements. Because the solutions of two consecutive Hermite elements are known, the same technique can be extended to a sequence of Hermite elements, the interpolated surface will be at least G1 everywhere. In their paper, the original surfaces are developable surfaces, so the Hermite elements sequence is well behaved, but it's difficult to find such sequence on general meshes

2.3 Paper Crafting

Automatically making paper craft parts form meshes is first introduced by Mitani and Suzuki [7]. The algorithm first segment the meshes into parts based on features. Then these parts are approximated with triangle strips. The same process repeat until all triangles are covered by some triangle strips. The triangle strips generate from their algorithm tend to have long boundaries which are not convenient for gluing. Another problem is that the method dose not consider any error metric, the only way to control the error is the predefined width of the triangle strips, which is not flexible.

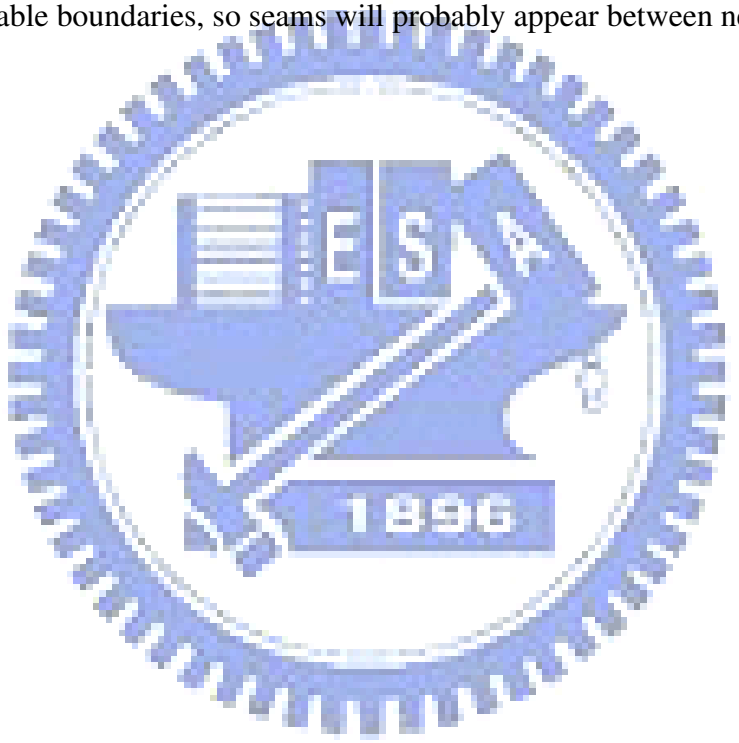
Julius et al. [5] propose using developability as error metric to segment the meshes. Their algorithm is based on region growing framework and use a Lloyd scheme. For each patch, a conic is used as a proxy surface. Because the angle between the conic's axis and normal on the surface is constant, the developable error metric is defined as

$$(N_C \cdot n_t - cos\theta_C)^2. \tag{2.4}$$

where N_C is the axis of the conic, n_t is the normal of the triangle, θ is the constant angle. They additionally consider the compactness and boundary smoothness of the patch as error metrics and use the product of the three weighted error metrics as the region growing error metric. At each iteration, faces with the smallest error are inserted into the patches until all faces are covered, then the optimized proxy conics are computed to fit the patches, the process repeat until converge. Because the algorithm only segment the mesh, a parametrization method must

be used to unfold patches into a plane. Because there are no isometric parametrization for general patches, distortion will still be introduced in the process.

Shatz et al. [9] also use conic surfaces as proxy surfaces, but additionally consider the error between the triangles and the conic surface. The proxy conic surfaces are treated as the final approximation surfaces and can be directly unfolded. Because of the error between the proxy conic and the origin meshes, sometimes the boundaries between the neighboring conic surfaces will be unstable and must be specially dealt with. They only apply additional optimization process on such unstable boundaries, so seams will probably appear between neighboring conics.



CHAPTER 3

Mesh Approximation using Cone Spline Surfaces

3.1 Approach overview

The aim of our algorithm is to approximate a mesh by several cone spline surfaces. The overview flow of our method is shown in Fig. 3.1. First the mesh is pre-segmented into several patches. For each patch, we perform region growing from the two boundaries of a cone spline surface to approximate the patch until the boundaries contain no faces. If the patch contains faces that are not covered by the cone spline surface, these faces will form new patches and the whole process repeat until every face is covered by one cone spline surface.

In section 3.2 we discuss the options of pre-segmentation method. In section 3.3, we discuss the detailed single patch approximation method. Because the boundaries between multiple cone spline surfaces may be unstable, in section 3.4, we discuss the possible boundaries optimization method.

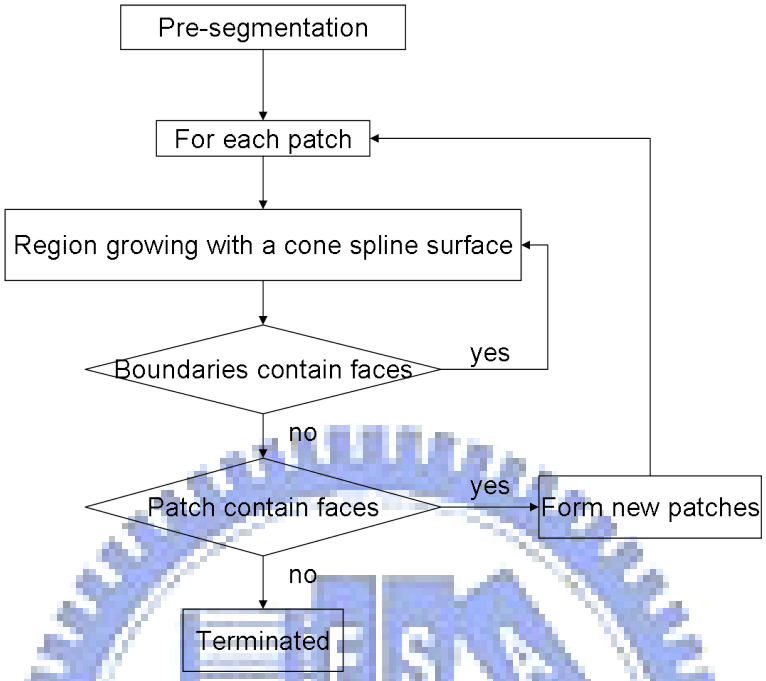


Figure 3.1: Overview of our algorithm

3.2 Pre-segmentation

Because our method can only locally approximate a patch, so pre-segmentation is required to segment the mesh into several patches before the patches could be approximated by cone spline surfaces. There are several types of mesh segmentation algorithm. Ideal segmentation for our method should be developability and feature based, which segment the mesh into near developable and meaningful parts. With such segmentation, our method could be applied to generate paper craft models from meshs which contain several meaningful parts, and each part can be approximated by a cone spline surface. To our best knowledge though, D-charts [5] is the only developability based segmentation, but it actually only uses conics to measure developability, which will limit our method. Because currently there is no ideal segmentation method fulfill our need, we segment the mesh manually into visually near developable and meaningful parts. Manually segment the mesh is very time-consuming, so we hope that in the future, more advanced mesh segmentation method that fulfill our need will be proposed, so the pre-segmentation could

be fully automated.

3.3 Single patch approximation

After the pre segmentation, we use a region growing approach to generate cone spline surfaces that approximate the patch. In this section, we discuss in detail the region growing process. Because we use cone spline surfaces as approximation surfaces, the consecutive conics in the same cone spline surface must at least C0 continuous. In section 3.3.1, we discuss the conic parametrization that preserve C0 continuity between consecutive conics. In section 3.3.2, we discuss the region growing cost function and conic optimization function. Finally in section 3.3.3 we discuss the region growing framework using the conic parametrization and error functions defined in 3.3.1 and 3.3.2.

3.3.1 Cone Spline Surfaces Parametrization

We followed the work of Shatz et al. [9] and define a conic as follow: Let c be the center of the cone base, n be the cone axis, d be the distance from the cone, θ be the constant angle between and the normals and the cone axis. Then a conic (n, c, d, θ) is defined by:

$$n_x \cdot (x - c) = d, \tag{3.1}$$

where r_x and n_x are defined as follows:

$$r_x = \frac{(x - c) - ((x - c) \cdot n)n}{\|(x - c) - ((x - c) \cdot n)n\|}, \tag{3.2}$$

$$n_x = r_x \cdot sin\theta + n_x \cdot cos\theta. \tag{3.3}$$

A plane is a conic where $c = (0, 0, 0)$ and $\theta = 0$.

A cone spline surface is a piece wise conic surfaces. In the work of Leopoldseder and Pottmann [6], the consecutive conic surface share a G1 Hermite element which means the tangent planes of the connected boundaries of the two conic surface is the same. Such constraint

is actually too strong and restrict the shape of the cone spline. In a lot of cases such as paper crafting, only the C0 continuity is required for the cone spline. As shown in Fig. 3.2, the condition for two conics being G1 continuous is that they share the same generator and the tangent planes are the same, it's obvious that both conics' axes lie on the same normal plane.

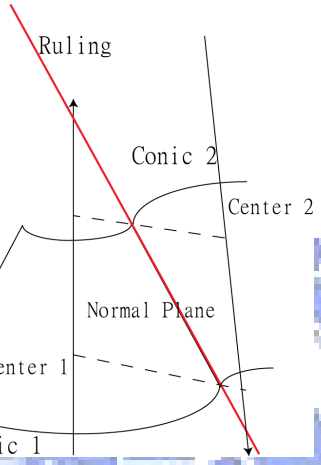


Figure 3.2: Two conics are G1 continuous

In our method, the constraint is relaxed so the consecutive conics are only C0 continuous. As shown in Fig. 3.3, the two conics only share the same ruling, but the tangent planes at the ruling are different. As a result, the axes of the two conics now could lie on two different normal plane, which increase the the degree of freedom during the approximation process. Such relaxation will result in a "folding line" between the two conics, but the developability is still preserved.

Following we discuss the parametrization of the three kinds of conic: cone, cylinder and plane respectively. The parametrization assume the neighboring conic and its boundary Hermite element is known so the conic can be expressed using the boundary information.

Cone

In the case of a cone, the vertex of the cone must reside on the ruling so we can use a parameter t_v to express the position of the vertex using the generator's line coordinate. Once

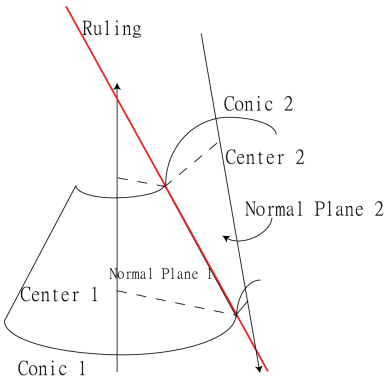


Figure 3.3: Two conics only preserve developability

the vertex position is determined, a local coordinates is defined using the vertex as origin. As shown in Fig. 3.4, the local coordinates are (u, v, w), where u is the normal of the normal plane, v is the direction of the generator and $w = u \times v$. Using the local coordinates, the direction of the new axis can be expressed using three parameters (u_0, v_0, w_0). Because the vertex and axis actually defined two cones, one in the upper and another in the lower of the vertex, so another parameter t_c is added to distinguish the two cases, $t_c > 0$ represent the upper cone, and $t_c < 0$ represent the lower cone. So a cone can be parameterized by $(t_v, u_0, v_0, w_0, t_c)$.

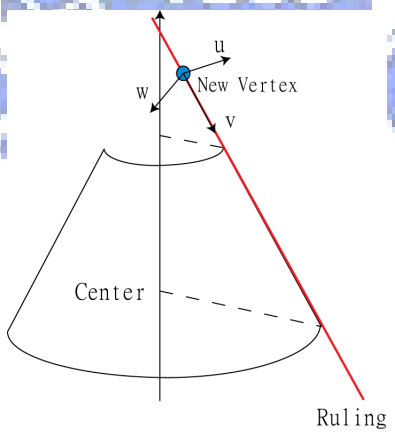


Figure 3.4: The local coordinate of the cone

Cylinder

In the case of a cylinder, the axis direction must be the same as the ruling, so we only need to specify the radius and the direction of the axis. As shown in Fig. 3.5, two parameters (θ, r) are used to express the cylinder, where r is the radius of the cylinder and θ is the angle between the local coordinate direction u and the vector x which is the direction from generator to the cylinder center.

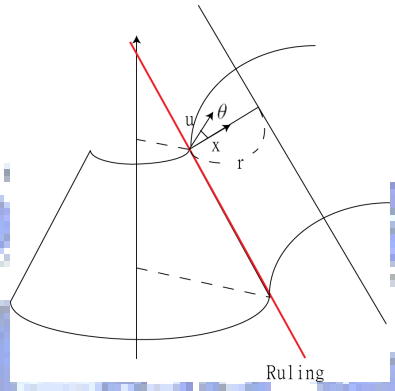


Figure 3.5: The local coordinate of the cylinder

Plane

In the case of a plane, the plane must pass through the generator, so the normal direction must be perpendicular to the generator. As shown in Fig. 3.6, the normal of the plane is parameterized by (β), which is the angle between the local coordinate direction u and the plane normal.

3.3.2 Cost Function

In this section, we define various cost functions used in our method. Because we hope the cone spline is grown in "good shape", beside distance cost function, we define additional cost functions that change the weight of the distance cost of each face. The total region growing cost function is defined as the product of these cost functions. Following we discuss the three cost functions used in our algorithm.

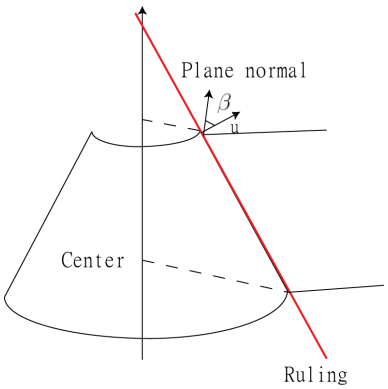


Figure 3.6: The local coordinate of the plane

Distance cost function

Distance cost function measure the distance between the faces and the conic surface. The cost is defined by:

$$Dist(face, conic) = \Sigma_{v \in face} \| v - Proj_{conic}(v) \|. \tag{3.4}$$

where v is the vertex in the face and $Proj_{conic}(v)$ is the closest point of v on the conic.

Normal difference cost function

Normal difference measure the normal difference between the face and the approximation conic. The cost is defined by:

$$NormDiff(face, conic) = 1 + \lambda \cdot \Sigma_{v \in face}(1 - \| N_{conic}(v) \cdot N(face) \|). \tag{3.5}$$

Where $N_{conic}(v)$ is the normal of the conic at the projection of v, $N(face)$ is the normal of the face, and λ is a user- defined parameter.

Compactness cost function

To prevent the generated patches to be strip-like, compactness cost is added to generate relatively ”round” patches [5], which is defined by:

$$CompactCost(face, patch) = \pi \frac{D(S_{patch}, face)^2}{A_{patch}}. \tag{3.6}$$

Where S_{patch} is the seed face of the patch, $D(S_{patch}, face)$ is the geodesic distance between the face and the seed, A_{patch} is the area of the patch. For triangles on the boundary of a circle (which is ideally compact) this metric evaluates to one.

Total cost

The cost of a face added to a patch is the combination of the three cost functions, which is defined by:

$$Cost(face, Conic, Patch) = Dist(face, conic)^{\alpha} NormDiff(face, conic)^{\beta} CompactCost(face, patch)^{\gamma}. \tag{3.7}$$

Where α, β and γ are user-defined parameters, in our algorithm, we use $\alpha = \beta = \gamma = 1$.

3.3.3 Region Growing Framework

The region growing framework is shown in Fig. 3.7. First, we find a conic surface that locally approximate the patch, the conic is then treated as the initial cone spline surface, then region growing is performed from the two boundaries of the cone spline surface. At each iteration, the face at the boundaries and with smallest total cost is selected and assigned to the cone spline surface. If the total cost of the assigned face is greater than a threshold $T_{triangle}$, a new conic surface that preserve C0 continuity and better approximate the face is created from the two boundaries of the initial conic. The growing stopped when the boundaries contain no face. Following we discuss every process in detail.

Get initial conic by local approximation the patch

If the pre segmentation is developability based like d-charts [5], then an initial guess could possibility be derived from the segmentation process. Because currently we manually segment the mesh, such information is not available, so we first find a conic by optimizing the distance cost of the conic and the whole patch, then the part of the patch that has smallest cost is selected, finally the initial conic is found by optimizing the distance cost to fit the part.

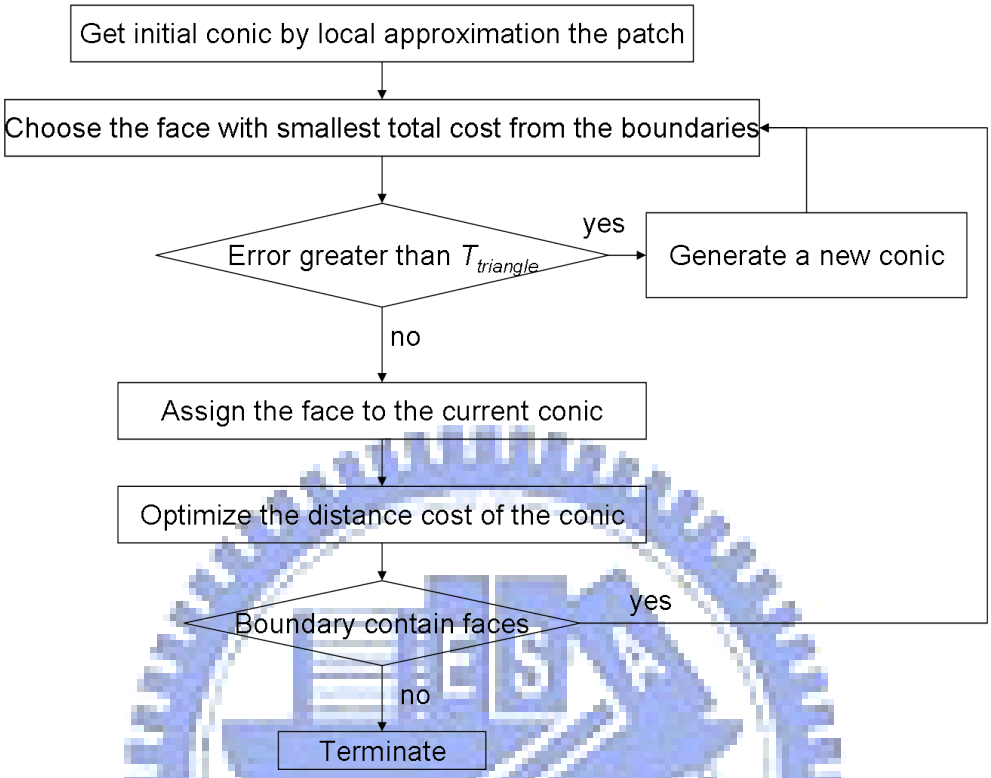


Figure 3.7: The region growing framework

Optimize the distance cost of the conic

After the initial conic is computed, at each of its two boundaries, a conic is used to approximate the patch. We keep a priority queue that store faces that reside at the boundary, at each iteration of region growing, the face in the queue that has minimal total cost defined in equation 3.7 is extracted from the queue and assigned to the new conic and its neighbor faces are inserted into the queue, then the approximation conic are optimized by the distance cost function defined in equation 3.4. We first treat the conic as a cone, and the cone is optimized by

$$argmin_{t_v,u_0,v_0,w_0,t_c}(\Sigma_{face \in cone} Dist(face, cone(t_v, u_0, v_0, w_0, t_c))). \tag{3.8}$$

If the cone degenerates to a cylinder, then the cylinder is optimized by:

$$argmin_{\theta,r}(\Sigma_{face \in cylinder} Dist(face, cylinder(\theta, r))). \tag{3.9}$$

And if the cone degenerates to a plane, then the plane is optimized by:

$$argmin_{\beta}(\Sigma_{face \in plane} Dist(face, plane(\beta))). \tag{3.10}$$

Generate a new conic

The region growing will continue until the new conic contain at least $MinFace$ faces, which is about one tenth of the total faces in the patch. After the number of faces in the conic exceed $MinFace$, the cost of the newly inserted face is checked, if it's greater than a predefined threshold $T_{triangle}$, a new conic that preserve C0 continuity and better approximate the face is created from the boundary. Because there are various possible choices of the new conic, a set of initial guesses of the new conic are used to perform region growing and the one with smallest total cost will be chosen. The cone spline surface will grow until the boundaries of the two sides contain no face. If there are remaining faces in the patch, these faces will form new patches and the process repeats until all faces are assigned to some cone spline surfaces.

3.4 Boundaries Optimization

In applications such as paper craft models, the boundaries between neighboring cone splines must be accurate enough to allow the parts to be glued back together to construct the models. As pointed out by Shatz and Leifman [9], when the normal difference between the two conic is small, the boundaries of the two conics will be unstable. In such case, they perform additional optimization passes to optimize the conics' boundaries. In our case, however, we can not optimize a single conic's boundary in a cone spline surface, because such optimization will break the continuity of the cone spline surface.

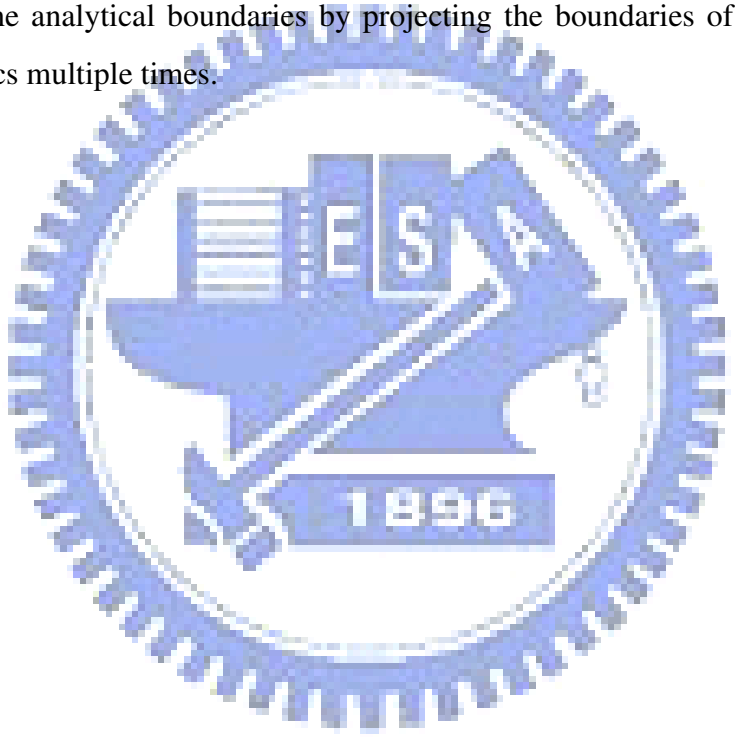
Instead of directly optimizing a conic, we follow the idea of Hermite interpolation [6], which allow one parameter family of two G1 continuous conics to be generated if the two boundaries' Hermite element are given. For each cone spline surface, we first specify the segment that contain unstable boundaries, then the Hermite elements at the two boundaries of the segments

are extracted. Using Hermite interpolation, we optimize the segment by:

$$argmin_t \Sigma_{face \in segment} Dist(face, conic(t)) + \beta \cdot \Sigma_{vertex \in bounary} Dist(vertex, conic(t)). \quad (3.11)$$

Where t is the parameter used in Hermite interpolation. If the optimization result is not good enough, the segment can be further divided and extract more Hermite elements and increase the degree of freedom in the optimization process.

After the boundaries have been optimized, we can follow the work of Shatz and Leifman [9] and extract the analytical boundaries by projecting the boundaries of the patch onto the neighboring conics multiple times.




E
S
A
1896

CHAPTER 4

Experimental Results and Discussion

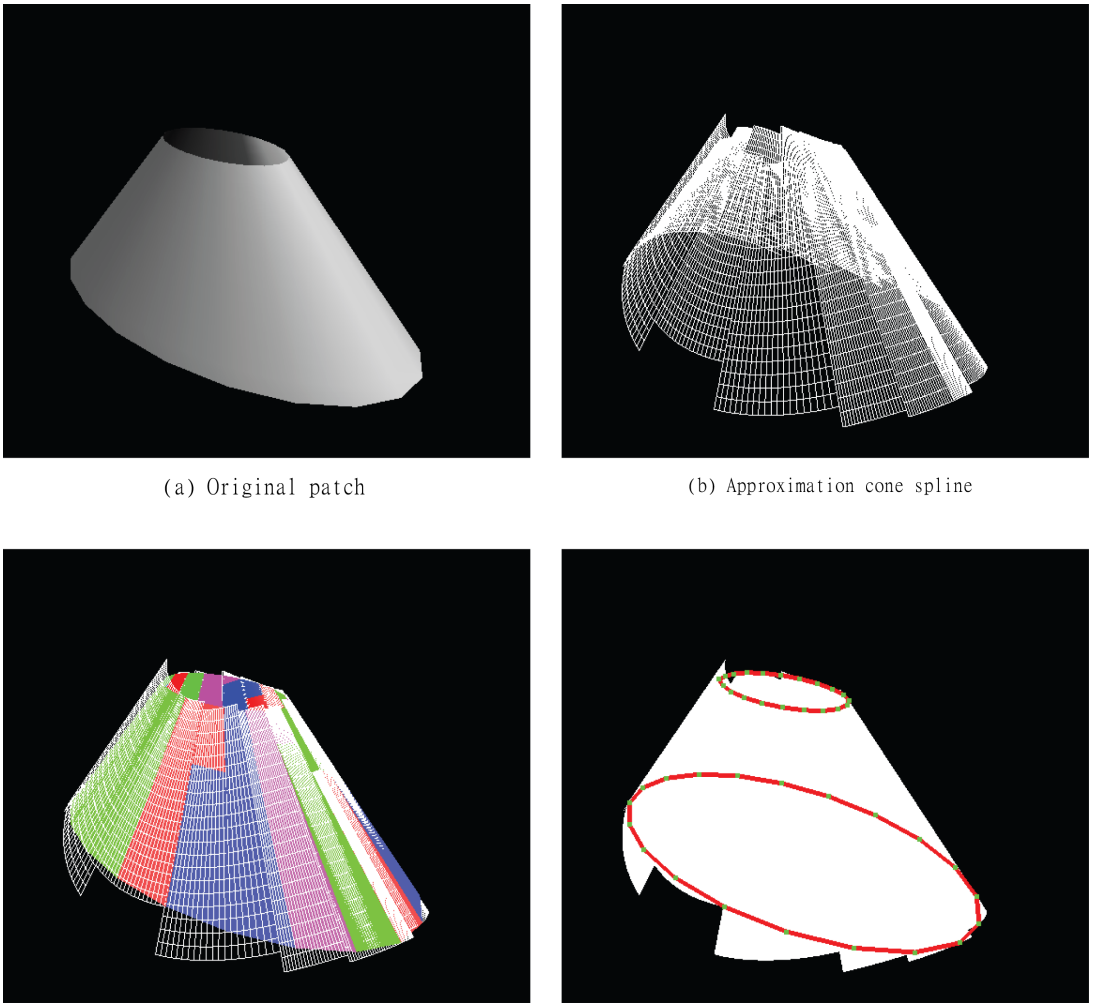
In this section, we present the experimental results and discussion of our method. In section 4.1, we first show the results when applying our algorithm on patches that are close to developable surfaces. In section 4.2, we show the results when applying our algorithm on more general meshes. In section 4.3, we discuss the unsolved problems and limitations of our method.

4.1 Near developable meshes

Fig. 4.1 shows the approximation result of a general cone. As shown in Fig. 4.1 (c) and (d), our method can approximate the shape quite well with multiple segments of cone. The cones are all connected with C0 continuity so we can unfold the cone spline in only one patch. In contrast, previous method [9] that use conics as approximation surfaces will approximate the patch with multiple separate cones and the boundaries between the cones are not guaranteed to be C0 continuous.

Fig. 4.2 shows the approximation result of a part of the fandisk model. The patch can be seen as a general cylinder. As shown in Fig. 4.2 (d), the part can be approximated efficiently

with a cone spline with only four segments of cylinder.



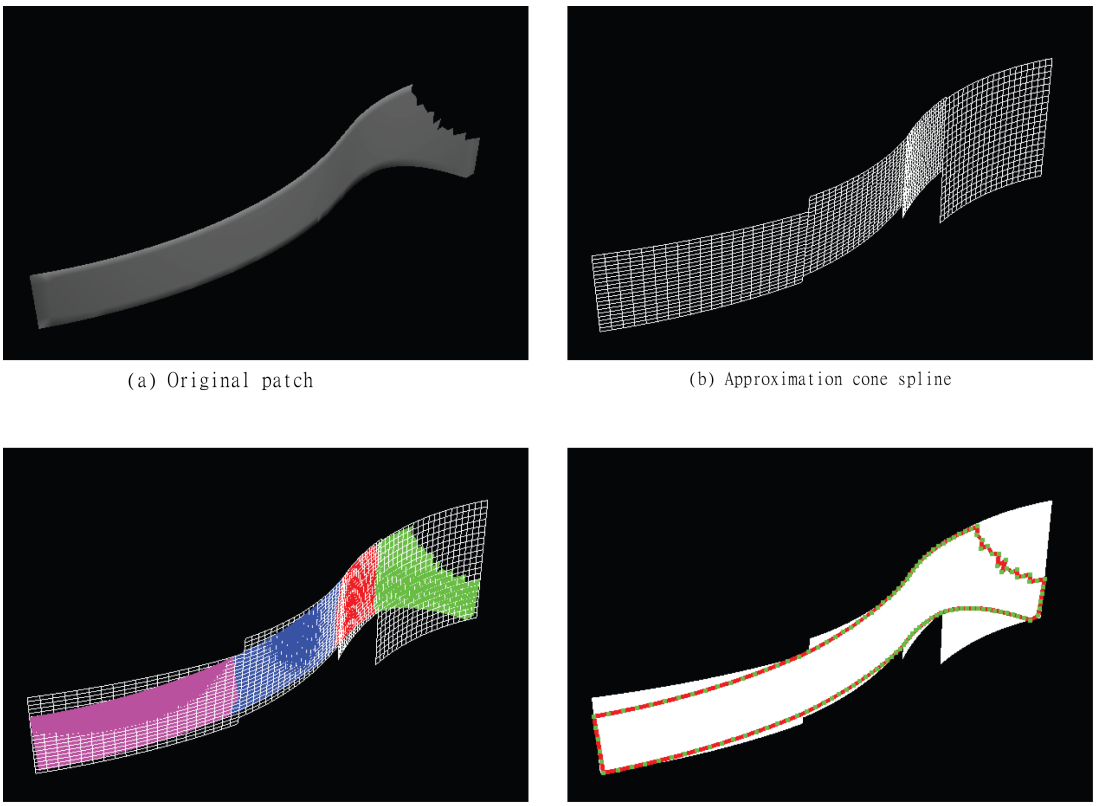
(a) Original patch (b) Approximation cone spline

(c) Original patch face assigment (d) Approximation Cone spline with projected boundaries

Figure 4.1: The approximation result of a general cone

4.2 General meshes

Fig. 4.3 and Fig. 4.4 show the approximation results of a segment of bunny model. As can be seen in Fig. 4.4 the cone spline can fit the surface accurately which is not possible using only a cylinder.



(a) Original patch (b) Approximation cone spline

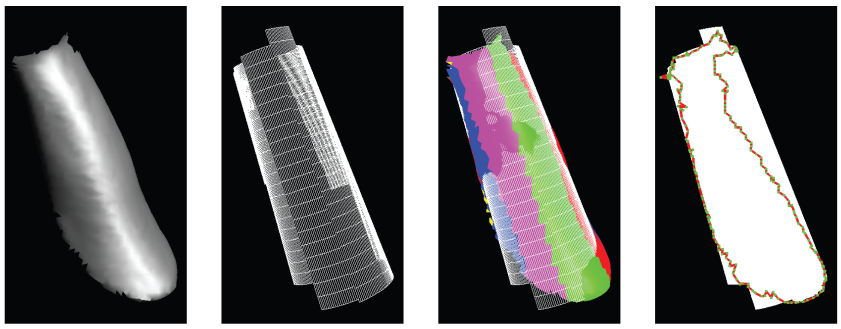
(c) Original patch face assigment (d) Approximation Cone spline with projected boundaries

Figure 4.2: The approximation result of a segment of fandisk model

Fig. 4.5 and Fig. 4.6 show the approximation results of a segment of the horse model. Although the detail on the patch is disappeared after the approximation, the overall shape of the patch is still preserved. As can be seen in Fig. 4.6, the upper and lower boundaries of the patch are preserve quite well in the cone spline approximation.

4.3 Discussion

Currently our algorithm still has some unsolved problems, we discuss these problems in this section.



(a) Original patch (b) Approximation cone spline (c) Original patch face assigment (d) Approximation Cone spline with projected boundaries

Figure 4.3: The approximation result of a segment of bunny model

(a) Original patch (b) Original patch and approximation cone

Figure 4.4: The approximation result of a segment of bunny model

4.3.1 Initial guess and greedy approach

The initial conic can greatly affect the approximation results, take the Fig. 4.7 for example, there are two possible initial guesses of the axis. In (a), the hyperboloid will be approximated by two cones, but in (b) the result will be many strip-like general cylinders. In this case, both approximations can be considered good, but the number of patches generated are greatly varied. In some cases however, bad initial guess will generate many unnecessary patches. In our implementation, if the result is not ideal, we allow the users to adjust the initial guess manually.

During region growing, the initial guess of the growing conics also greatly affect the result.

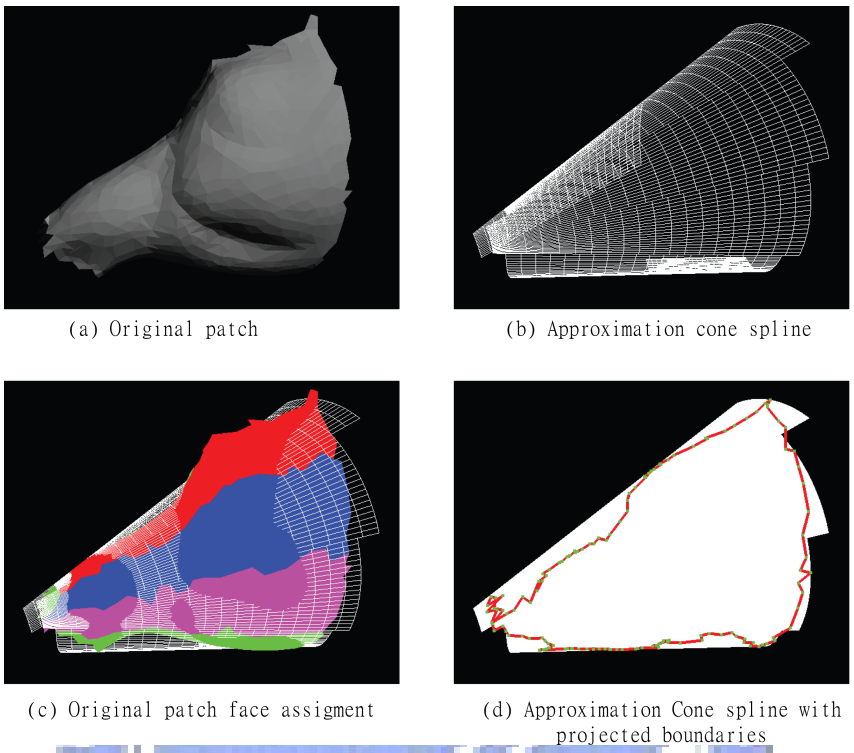


Figure 4.5: The approximation result of a segment of horse model

Because we take a greedy approach, the conic that has smallest cost will be chosen, such approach can't guarantee an optimal result. In Julius et al. [5] and Shatz et al.'s [9] works, the region growing are accompanied with a Lloyd scheme, so their approximation will "move" at each iteration and converge to a good result. But in our method, the number of parameters of each cone spline is not known a priori, so its hard to integrate our method with such scheme.

4.3.2 Faces assignment problem

Sometimes if the meshes have thin structures, the approximation algorithm will assign the faces to the wrong conic, which results in incorrect results. For example, in Fig. 4.8, (a) incorrectly assign the two sides of patches to the same conic surface where the correct result should be two different conic surfaces like (b). If such problem occurs, the incorrect faces will be inserted into the priority queue, and the algorithm fails.

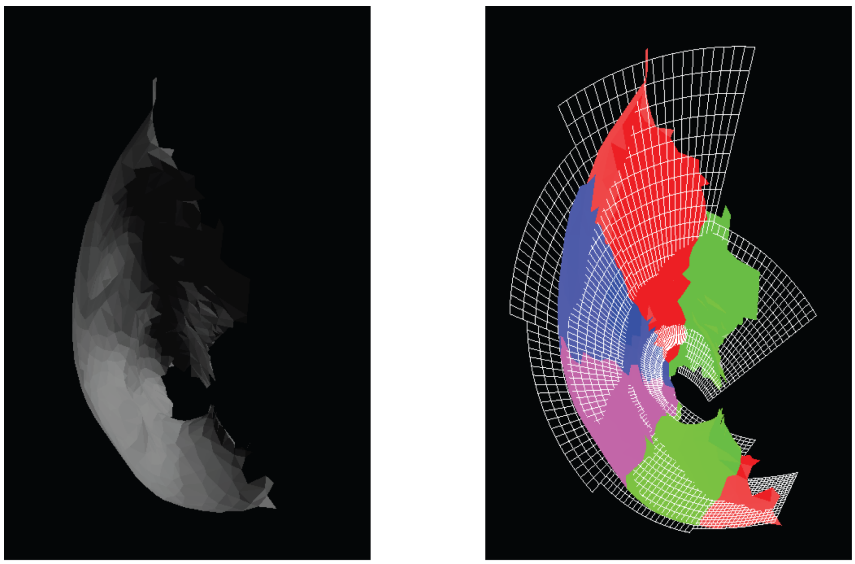


Figure 4.6: The approximation result of a segment of horse model

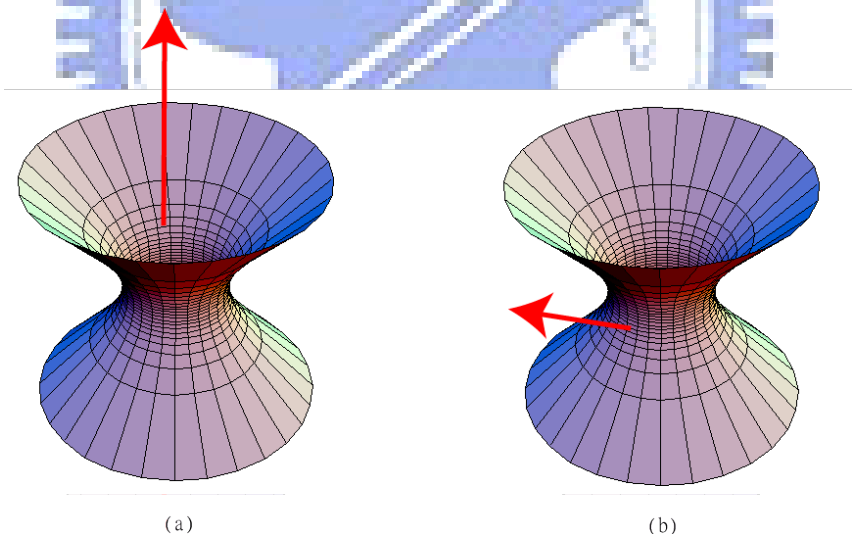


Figure 4.7: Two possible initial guesses' axis direction

This problem may be automatically detected in the pre-segmentation process. But as discuss in section 3.2, currently there is no appropriate segmentation method for us, so currently we don't deal with the problem. If the problem occurs, we manually segment the thin structure into

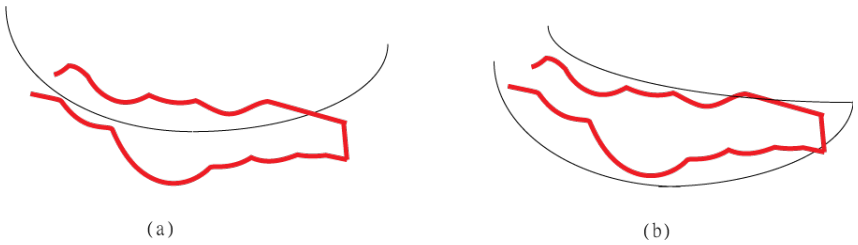


Figure 4.8: Two possible approximations

two or more patches.


E
S
A
1896

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

We have proposed a novel cone spline developable surface approximation method that can be applied on near developable surfaces as well as more general meshes. Our algorithm can be seen as a generalize form of the paper craft algorithms that use single conic as approximation surface. Currently our algorithm has some problems discussed in section 4.3, but we believe these problems can be solved by further analysis of the meshes and a better approximation framework.

5.2 Future work

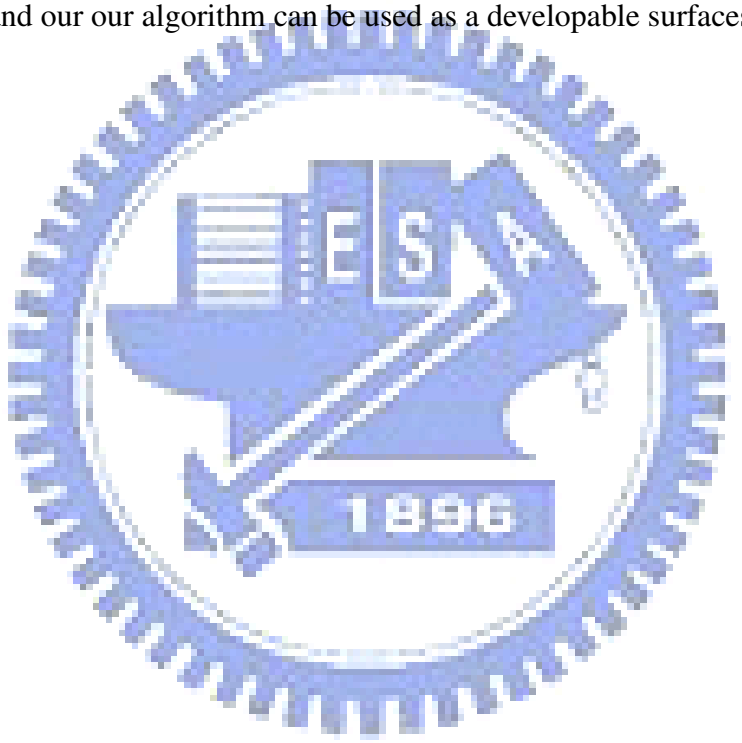
Currently our algorithms can generate dramatically different results when applying different initial guesses, and it's difficult to get an ideal initial guess by local approximation of the patches, so we want to apply a global analysis on the patch to generate better initial guesses.

Our algorithm can't automatically identify the sharp edge on the patches and can be affected by the noisy surfaces, so we want to consider and integrate local approximation methods such

as moving least square [4] that can identify outliers and features.

To our best knowledge, there is no developability based mesh segmentation algorithm that use general developable surfaces as proxies. So our ultimate goal is to use cone spline surfaces as proxies and combine a cluster scheme to automatically segment the mesh. Such algorithm would generalize our work so the segmentation and approximation is performed simultaneously and the result will automatically converge to a good result.

Finally, we also want to incorporate user editing system so the approximated results can be easily modified, and our our algorithm can be used as a developable surfaces design system.



Bibliography

[1] G. Aumann. Interpolation with developable bézier patches. In *Computer Aided Geometric Design*, 1991.

[2] R. Bodduluri and B. Ravani. Design of developable surfaces using duality between plane and point geometries. In *Computer Aided Design*, 1992.

[3] H.-Y. Chen, I.-K. Lee., S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. On surface approximation using developable surfaces. In *Graphical models and image processing: GMIP*, 1999.

[4] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. In *Proc. ACM Transactions on Graphics*, 2005.

[5] D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. In *Proc. Eurographics*, 2005.

[6] S. Leopoldseder and H. Pottmann. Approximation of developable surfaces with cone spline surfaces. In *Computer Aided Design*, 1998.

[7] J. Mitani and H. Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. In *Proc. ACM Transactions on Graphics*, 2004.

[8] H. Pottmann and G. Farin. Developable rational bezier and b-spline surfaces. In *Computer Aided Geometric Design*, 1995.

[9] I. Shatz, A. Tal, and G. Leifman. Paper craft models from meshes. In *The Visual Computer*, 2006.