

國立交通大學

資訊工程系

碩士論文

基於疊代最佳化之網格參數化及其於三維著色系統之應用

An Iterative-Optimization based Parameterization
for Surface Painting Systems



研究生：鄭仁豪

指導教授：莊榮宏 博士

中華民國九十四年八月

基於疊代最佳化之網格參數化及其於三維著色系統之應用

An Iterative-Optimization based Parameterization

for Surface Painting Systems

研究生：鄭仁豪

Student : Ren-Hao Cheng

指導教授：莊榮宏 博士

Advisor : Dr. Jung-Hong Chuang

國立交通大學

資訊工程學系



Institute of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Computer Science and Information Engineering

August 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年八月

基於疊代最佳化之網格參數化 及其於三維著色系統之應用

研究生：鄭仁豪

指導教授：莊榮宏 博士

國立交通大學資訊工程學系



三維著色為一讓使用者直接於三維表面著色之程序。著色筆觸儲存於一張由網格參數化技術產生之二維貼圖。傳統三維著色的系統，網格參數化為事先產生，此參數化平面即為三維模型之材質貼圖。在著色過程中，網格參數化是固定的，並不會根據繪圖筆觸，於顏色訊號變化較為劇烈之處給予較多的貼圖取樣。為了提高三維著色系統的效能，必須針對使用者的繪圖筆觸，於著色過程中調整網格參數化使得顏色變化劇烈的區域得到較多的貼圖取樣，且須在互動時間內完成。在這篇論文中，我們提出一個基於疊代最佳化之網格參數化架構。於互動時間內，針對顏色訊號變化較為劇烈的區域在貼圖區域上給予較多的取樣空間。

關鍵字：網格參數化、三維著色系統、材質貼圖產生

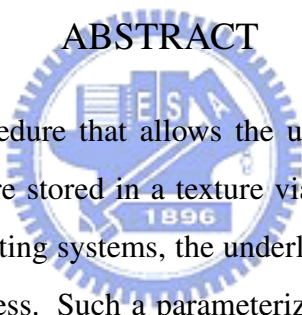
An Iterative-Optimization based Parameterization for Surface Painting Systems

Student: Ren-Hao Cheng

Advisor: Dr. Jung-Hong Chuang

Department of Computer Science and Information Engineering
National Chiao Tung University

ABSTRACT

The logo of National Chiao Tung University is a circular emblem with a gear-like border. Inside the circle, there is a stylized representation of a book and a torch, with the letters 'E', 'I', 'S', and 'A' arranged around them. The year '1896' is inscribed at the bottom of the inner circle.

Surface painting is a procedure that allows the users to paint onto a surface directly. The painting strokes are stored in a texture via surface parameterization techniques. In current surface painting systems, the underlying surface parameterization is fixed during the painting process. Such a parameterization is not sensitive to the frequency spectrum of the color signal introduced by painting strokes. To associate the regions of higher color signal variation with more texture samples, we need to do the re-parameterization according to users strokes at interactive rates. In this thesis, we propose a re-parameterization scheme that is based on an iterative-optimization aiming to allocate more texture samples for regions of high signal variation and to perform at an interactive rate as well.

Keywords: surface painting, surface parameterization, texture generation.

Acknowledgments

First of all, I would like to thank my advisors, Dr. Jung-Hong Chuang, for his inspirations, encouragement and guidance in the past two year.

I would like to thank Dan-Chi, Wen-Zheng and Yu-Hsian for their help and advice on my work. Besides, I would like to thank Wang-Yeh and Yi-Jun's help in my daily life for the last two years.

I also appreciate all my colleagues in CGGM lab, Chih-Chun, Jong-Hon, Chi-Han, Chao-Wei, Yong-Cheng, Zheng-Li, Min-Sheng and Roger from the bottom of my heart.

At last, I would like to thank my parents, my sister and my girlfriend for their support, encouragement, and love.



Contents

Abstract	ii
Acknowledge	iii
List of Figures	vi
List of Tables	ix
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	3
1.3 Thesis Organization	3
2 Related Work	4
2.1 Theoretic background of parameterization	4
2.1.1 General setup and notation	5
2.1.2 First fundamental form	6
2.1.3 Some stretch metrics	9
2.2 Mesh Parameterization	10
2.3 Surface Painting	16

3	A Re-parameterization Framework for Surface Painting	23
3.1	Approach overview	23
3.2	Topological surgery	24
3.3	Initial parameterization	27
3.4	Stroke sampling	27
3.5	Importance map and geometry stretch map	28
3.6	The L_s^2 stretch	31
3.7	Rapid re-parameterization	33
3.7.1	Iterative optimization based on uniform grid	33
3.7.2	Re-parameterization	36
3.7.3	Stroke resampling over optimized parameterization	36
3.8	Two-stage re-parameterization framework	37
4	Results and Performance Analysis	45
4.1	Results of surface painting	46
4.2	Analysis of interactive application	47
4.3	Signal-specialized parameterization versus our method	49
4.4	Comparison to painting detail	53
5	Conclusion and Future Work	57
5.1	Conclusion	57
5.2	Future work	58
	Bibliography	60

List of Figures

2.1	Parameterization ϕ and embedding ψ [3].	5
2.2	Γ and γ represent the largest and smallest local stretch [3].	8
2.3	Determine λ_{vw} for v that is inside of a triangle.	12
2.4	Determine λ_{vw} for v inside a n -sided polygon [6].	12
2.5	A cat head model and its harmonic map [5].	13
2.6	Gray-coded deformation energy of different parameterizations [11]. . .	14
2.7	Examples of geometric-stretch and signal-stretch parameterizations [18].	14
2.8	Comparison of two mesh parameterization schemes [26]. Top : Stretch minimization of Sander et al [17], Down : Yoshizawa et al. [26]	15
2.9	Texture mapping for a cat head model [22]. (a) cat head model. (b) ABF parameterization. (c) ABF texturing result. (d) Uniform Grid G_1 . (e) Smoothed grid G_2 . (f) Texturing result after applying G_2	17
2.10	Texturing result of geometry stretch and signal stretch [18].	18
2.11	Each stroke is stored in an individual atlas [13].	19
2.12	Uniform mesh atlas for a cloud textured moon [1].	20
2.13	Rhino sculpted from wood and its area-weighted mesh atlas [1].	20
2.14	Recursively partition the mesh using Metis algorithm [1].	21

2.15	A multiresolution meshed atlas of a cow. Each node in the tree corresponds to (a) a cluster of triangles (b) and a square region in the texture domain. (c) Each node's cluster is the union of its children's clusters [2].	22
3.1	The overall process of our method.	25
3.2	Procedure of parameterizing closed-surface [3].	25
3.3	Slice the closed-mesh along the cut [3].	26
3.4	Texture domain.	29
3.5	Problems of four-tap filter.	30
3.6	Our filter.	30
3.7	Four-tap filter and our filter.	31
3.8	Face model with its parameterization, geometry stretch map and importance map.	32
3.9	The optimization result base on one iteration.	35
3.10	Chart diagram of the optimization process.	35
3.11	Venus model and optimized parameterization.	39
3.12	Parasaur model : single stage optimization using 256x256 uniform sample points	40
3.13	High resolution uniform grid points.	40
3.14	Low resolution uniform grid points.	41
3.15	Venus model and two-stage optimized parameterization.	42
3.16	Parasaur model : Two-stage optimization using 16×16 and 256×256 grids.	43
3.17	Comparison of single and two stage optimization	44
4.1	Painting results of the venus model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).	47

4.2	Back-view of the painting results of the venus model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).	48
4.3	Painting results of the triceratops model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).	49
4.4	Painting results of the face model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).	50
4.5	Painting results of the face model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).	51
4.6	An image is texture mapped to simulate painting strokes. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).	52
4.7	Four images is texture mapped to simulate painting strokes. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).	54
4.8	The optimization process of triceratops model.	55
4.9	Comparison of our result with signal specialized parameterization under different texture map resolutions.	56

List of Tables

4.1 Statistics of initial parameterization, two-stage optimization and re-parameterization time (sec.) for four different models. 53



Introduction

1.1 Motivation

Surface painting(also called 3D painting) is a technique that allows the users to paint directly onto a 3D surface. If the discretization of the surface is fine enough, user can directly paints on the vertices of the surface. However, in general, the desired precision for the color is greater than the geometric detail of the model. Assuming that a surface is provided with a parameterization, it is convenient to store colors in the parameterized texture space. In current surface painting systems, the underlying mesh parameterization is predefined and fixed during the painting process. Such a parameterization is not sensitive to the frequency spectrum of the color signal as the result of painting strokes, and in consequence, may introduce distortion at arbitrary locations and waste texture space in areas of no stroke. Moreover, current surface painting systems parameterize the surface based only on the geometric aspects. Even though these systems provide tools allowing users to adjust the underlying parameterization, but it is not intuitive for normal users.

Most surface parameterization schemes assume no prior knowledge of the signal,

and take only surface's geometry information into account. For surface painting, we want to allocate more texture samples in regions of greater signal detail by doing the re-parameterization on the fly according to the painting strokes. Moreover, the re-parameterization should be fast enough to achieve an interactive rate.

Sander's signal-specialized parameterization [18] minimizes the signal stretch for a given mesh with signal information. It is, however, not suitable for surface painting since it utilizes an expensive global optimization, and cannot support the interactive re-parameterization required after each stroke is painted. To support painting systems, very few methods have been proposed. In Igarashi and Cosgrove [13], painting strokes are stored as separate charts which are packed into a texture atlas. The method generates each chart such that only the region with painting strokes are included, and from which fine details can be reproduced. However, overlapping strokes made at different poses are stored in separate charts, and hence additional texture space are required. Carr et al. proposed a dynamic re-parameterization scheme based on their prior work on multi-resolution meshed atlas (MMA) [1, 2]. The dynamic re-parameterization is performed by changing the so called MMA tree hierarchy. This framework is complicated and, in general, achieves better results only for the model with large polygon count.

Our proposed method first derives an initial parameterization, and then, during the painting process, analyze the color signal frequency introduced by painting strokes, and utilizes an iterative optimization to do the re-parameterization to interactively allocate more texture samples for regions with high color signal variation. The proposed method is simple to implement and works well for models with either low or large polygon count.

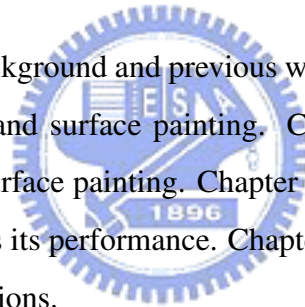
1.2 Contributions

This thesis describes a novel and simple framework of the re-parameterization necessary for future surface painting systems. Along the way to achieve this goal, we present the following contributions:

- Propose a modified signal metric L_s^2 that measures the geometric and signal stretch of a parameterization.
- Propose an interactive approach for the re-parameterization aiming to increase the sampling ratio in regions with high signal variation.

1.3 Thesis Organization

Chapter 2 introduces some background and previous works related to the thesis, including surface parameterization and surface painting. Chapter 3 presents our two-stage optimization framework for surface painting. Chapter 4 demonstrates the results of the proposed method and analyzes its performance. Chapter 5 summarizes our method and mentions some research directions.



CHAPTER 2

Related Work

In this chapter, reviews on related work of parameterization and surface painting will be given.

2.1 Theoretic background of parameterization

Several schemes have been proposed that flatten a surface region and establish a parameterization over the last ten years in computer graphics. Parameterization is a mapping from a two dimension domain to a high dimension space. All previous proposed techniques explicitly aim at producing least-distorted parameterizations, and vary only in the objective function (distortion metric) and the minimization processes used. The main distinction between the functions is how they measure the difference of the resulting parameterization from an isometric one (a mapping preserving lengths and angles). In this section, we introduce several metrics that are directly related to our research.

2.1.1 General setup and notation

Before reviewing the fundamental theory, we first introduce some basic concepts and notations that are useful for introducing the fundamental theory.

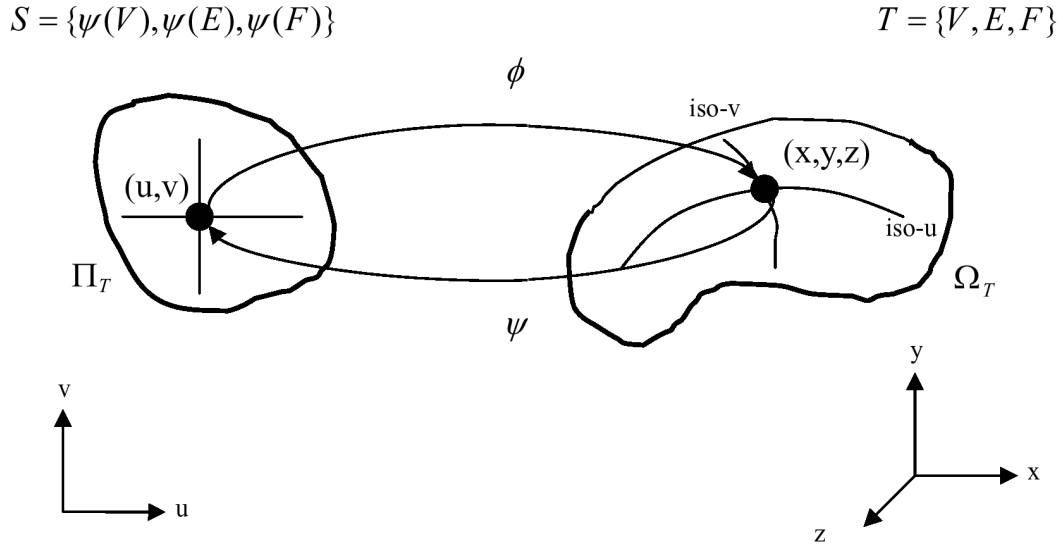


Figure 2.1: Parameterization ϕ and embedding ψ [3].

As shown in Figure 2.1, we call $\Omega_T \in R^3$ the surface of triangulation T and further let $V = V(T)$ denote the set of vertices, $E = E(T)$ the set of edges, and $F = F(T)$ the set of faces of T . If Ω_T has a boundary we distinguish between the disjoint sets of interior and boundary vertices as V_I and V_B . Two distinct vertices $v, w \in V$ are neighbors if they are the end points of an edge $e = [v, w] \in E$. For each $v \in V$, $N_v = \{w \in V : [v, w] \in E\}$ denotes the set of neighbors of v . In general, parameterization ϕ of a triangulation T over a parameter domain $\Pi_T \in R^2$ is a homeomorphism between this domain and the surface of T ; that is,

$$\phi : \Pi_T \rightarrow \Omega_T$$

From differential geometry, we know that such a homeomorphism and its inverse

$\psi = \phi^{-1}$ exist if and only if Π_T and Ω_T are topologically equivalent, i.e. Π_T is a 2-manifold with the same number of boundaries and handles as Ω_T . The number of handles is also called the genus of the manifold. For example, a sphere has genus zero and the genus of a torus is one, etc. If Ω_T and Π_T are not topologically equivalent, some topological operations are required to change the topology of Ω_T such that the result is topologically equivalent to Π_T .

Most parameterizations are piecewise linear functions. Due to the piecewise linearity, ψ induces a triangulation S which is equivalent to T in the sense that vertices, edges, and triangles of S and T naturally correspond to each other; that is,

$$\psi(V(T)) = V(S), \psi(E(T)) = E(S), \psi(T) = S,$$

and

$$\phi(V(S)) = V(T), \phi(E(S)) = E(T), \phi(S) = T.$$

We call triangles of S parameter triangles and S itself the parameter triangulation. Note that ϕ and ψ are uniquely determined by the images $\psi(v)$, which we call the parameter points or parameter values of the vertices $v \in V$. Hence the task of parameterizing T amounts to finding parameter values $\psi(v) \in \Pi_T$, one for each vertex $v \in V$. Since we also expect ψ to be bijective, we have to assure that the parameter points are arranged such that the parameter triangles do not overlap or flip (adjacent triangles in Ω_T with opposite orientation).

2.1.2 First fundamental form

In this section, we introduce more complicated metrics of parameterizing Ω_T based on the first fundamental form from differential geometry. Given a differentiable surface Ω and its parameterization ϕ , we can regard the parameterization ϕ as a mapping from R^2 to R^3 by

$$\phi(u, v) = [x(u, v), y(u, v), z(u, v)]$$

with the following differential forms

$$\begin{aligned} dx &= \frac{\partial x}{\partial u} du + \frac{\partial x}{\partial v} dv, \\ dy &= \frac{\partial y}{\partial u} du + \frac{\partial y}{\partial v} dv, \\ dz &= \frac{\partial z}{\partial u} du + \frac{\partial z}{\partial v} dv. \end{aligned}$$

We define the line element as $dl^2 = dx^2 + dy^2 + dz^2$ or alternatively,

$$dl^2 = E(u, v)du^2 + 2F(u, v)dudv + G(u, v)dv^2, \quad (2.1)$$

where

$$\begin{aligned} E(u, v) &= \frac{\partial \phi}{\partial u} \cdot \frac{\partial \phi}{\partial u} = \left(\frac{\partial x}{\partial u} \right)^2 + \left(\frac{\partial y}{\partial u} \right)^2 + \left(\frac{\partial z}{\partial u} \right)^2, \\ F(u, v) &= \frac{\partial \phi}{\partial u} \cdot \frac{\partial \phi}{\partial v} = \left(\frac{\partial x}{\partial u} \right) \cdot \left(\frac{\partial x}{\partial v} \right) + \left(\frac{\partial y}{\partial u} \right) \cdot \left(\frac{\partial y}{\partial v} \right) + \left(\frac{\partial z}{\partial u} \right) \cdot \left(\frac{\partial z}{\partial v} \right), \\ G(u, v) &= \frac{\partial \phi}{\partial v} \cdot \frac{\partial \phi}{\partial v} = \left(\frac{\partial x}{\partial v} \right)^2 + \left(\frac{\partial y}{\partial v} \right)^2 + \left(\frac{\partial z}{\partial v} \right)^2. \end{aligned}$$

$E(u, v)du^2 + 2F(u, v)dudv + G(u, v)dv^2$ in Equation 2.1 is known as the the first fundamental form which determines the arc length of a curve on the surface. Arranging the coefficient in a symmetric matrix form

$$I = \begin{bmatrix} E & F \\ F & G \end{bmatrix},$$

we have

$$dl^2 = \begin{pmatrix} du & dv \end{pmatrix} I \begin{pmatrix} du \\ dv \end{pmatrix}.$$

The matrix I is called the metric tensor, which can be decomposed as

$$I = J^T J$$

where

$$J = \begin{bmatrix} \frac{\partial \phi}{\partial u} & \frac{\partial \phi}{\partial v} \end{bmatrix}$$

is the Jacobian matrix of ϕ . We can show that if Γ and γ are singular values of J , Γ^2 and γ^2 will be eigenvalues of I . Since J can be seen as a local affine mapping from R^2 to R^3 , denoted by

$$q = \begin{pmatrix} J & o \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix}, \quad q, o \in R^3, \quad p \in R^2,$$

where o is the original point for completing the affine mapping, it is intuitive to use Γ and γ for describing the lengths and angles of vectors in R^2 after being mapped by ϕ . In other words, the singular values Γ and γ represent the largest and smallest length obtained after mapping unit-length vectors from R^2 to R^3 , i.e. Γ and γ can represent the largest and smallest local “stretch” of ϕ .

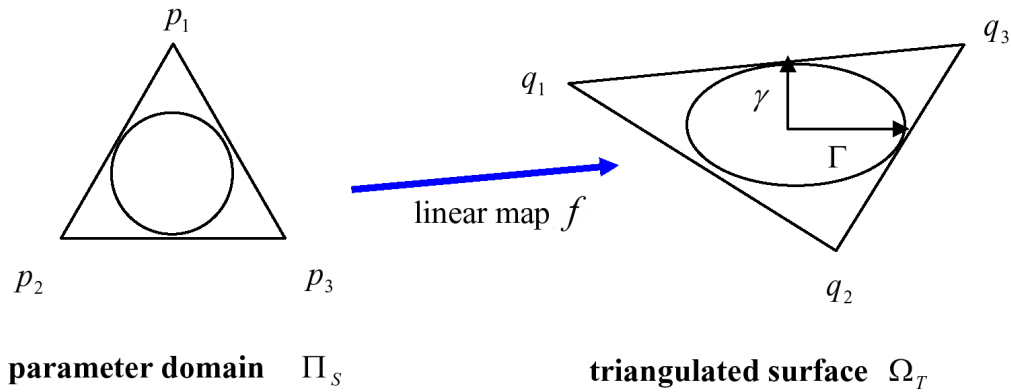


Figure 2.2: Γ and γ represent the largest and smallest local stretch [3].

Because triangulated surface Ω_T is piecewise linear, ϕ can be seen as the piecewise linear map f as shown in Figure 2.2. The common way to represent f is the barycentric mapping $B(p)$ defined as

$$B(p) = (\langle p, p_2, p_3 \rangle q_1 + \langle p, p_3, p_1 \rangle q_2 + \langle p, p_1, p_2 \rangle q_3) / \langle p_1, p_2, p_3 \rangle$$

where $\langle a, b, c \rangle$ denotes the area of $\triangle abc$ and p is a point on the $\triangle abc$. We can

discretize J and denote it as J_T

$$J_T = \begin{bmatrix} \partial x/\partial u & \partial y/\partial u & \partial z/\partial u \\ \partial x/\partial v & \partial y/\partial v & \partial z/\partial v \end{bmatrix}^t = \begin{bmatrix} \partial B/\partial u \\ \partial B/\partial v \end{bmatrix}^t$$

where

$$\partial B/\partial u = B_u = (q_1(v_2 - v_3) + q_2(v_3 - v_1) + q_3(v_1 - v_2))/(2A)$$

$$\partial B/\partial v = B_v = (q_1(u_3 - u_2) + q_2(u_1 - u_3) + q_3(u_2 - u_1))/(2A)$$

and

$$A = \langle p_1, p_2, p_3 \rangle = ((u_2 - u_1)(v_3 - v_1) - (u_3 - u_1)(v_2 - v_1))/2$$

Thus the Jacobian matrix of Ω_T is

$$J_T = \frac{1}{2A} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix} \begin{bmatrix} (v_2 - v_3) & (u_3 - u_2) \\ (v_3 - v_1) & (u_1 - u_3) \\ (v_1 - v_2) & (u_2 - u_1) \end{bmatrix}$$

and the largest and smallest singular values of J_T are given respectively by

$$\Gamma = \sqrt{1/2 \left((a + c) + \sqrt{(a - c)^2 + 4b^2} \right)},$$

$$\gamma = \sqrt{1/2 \left((a + c) - \sqrt{(a - c)^2 + 4b^2} \right)}.$$

where $a = B_u \cdot B_u$, $b = B_u \cdot B_v$, and $c = B_v \cdot B_v$.

2.1.3 Some stretch metrics

Based on above deduction, various stretch metrics base on the versatile Γ and γ have been proposed. For example, Sander et al. [17] define an L^2 distortion measure by taking the root-mean-square of Γ and γ , and define L^∞ as the largest singular value Γ :

$$L^2 = \sqrt{\frac{\Gamma^2 + \gamma^2}{2}}$$

$$L^\infty = \Gamma$$

Hormann et al. [11] defines a deformation metric as

$$L_d = \frac{\Gamma}{\gamma} + \frac{\gamma}{\Gamma}$$

Sorkine et al. [24] defines a geometric distortion as

$$L_g = \max \left\{ \Gamma, \frac{1}{\gamma} \right\}$$

Khodakovsky et al. [15] defines an area distortion as

$$L_{area} = \Gamma \cdot \gamma$$

and an anisotropic distortion as

$$L_{angle} = \frac{\Gamma}{\gamma}$$

2.2 Mesh Parameterization

Over the last years, a lot of research has been done in the area of surface parameterization. In the context of parameterization, harmonic maps were first used by Eck et al. [5]; see Figure 2.5. However, the texture coordinates for boundary vertices must be fixed a priori and harmonic maps may contain face flips, which violate the bijectivity of the parameterization.

Based on earlier work by Tutte [25], Floater proposed a specific weight based on the barycentric maps to obtain a mapping that is shape-preserving [6]. It guarantees the embedding for convex boundaries and find a parameterization by solving a linear system.

The first step of the method is to specify the parameter points $\psi(v)$ of the boundary vertices $v \in V_B$. Then, set each interior vertex $v \in V_I$ to be a convex combination of its neighbors. For each interior vertex v , a set of strictly positive convex weights λ_{vw} , $w \in N_v$, is chosen such that

$$\sum_{w \in N_v} \lambda_{vw} = 1.$$

For all interior vertices, the mapping $\psi(v)$, $v \in V_I$, is determined by solving the following linear system of equations:

$$\psi(v) = \sum_{w \in N_v} \lambda_{vw} \psi(w), \quad v \in V_I.$$

This equation can be rewritten as

$$\psi(v) - \sum_{w \in N_v \text{ and } w \in V_I} \lambda_{vw} \psi(w) = \sum_{w \in N_v \text{ and } w \in V_B} \lambda_{vw} \psi(w), \quad v \in V_I,$$

and further as

$$Bx = c.$$

The problem remained is how to determine the value of λ_{vw} . There are two cases needed to be discussed. The first case is that v is inside of a triangle as shown in Figure 2.3. We can represent v as

$$v = \frac{\text{area}(v, w_1, w_2)}{\text{area}(w_1, w_2, w_3)} \cdot w_3 + \frac{\text{area}(v, w_2, w_3)}{\text{area}(w_1, w_2, w_3)} \cdot w_1 + \frac{\text{area}(v, w_3, w_1)}{\text{area}(w_1, w_2, w_3)} \cdot w_2,$$

and then set

$$\begin{aligned} \lambda_{vw_1} &= \frac{\text{area}(v, w_2, w_3)}{\text{area}(w_1, w_2, w_3)} \\ \lambda_{vw_2} &= \frac{\text{area}(v, w_3, w_1)}{\text{area}(w_1, w_2, w_3)} \\ \lambda_{vw_3} &= \frac{\text{area}(v, w_1, w_2)}{\text{area}(w_1, w_2, w_3)} \end{aligned}$$

The second case is that v has more than three neighbor vertices. As shown in Figure 2.4, since $\triangle w_1 w_4 w_5$ encloses v , we can solve λ_{vw_1} , λ_{vw_4} and λ_{vw_5} using the previous method. Similarly, for each triangle k that has an end point w_i and encloses v , we compute $\lambda_{vw_i}^k$ and set λ_{vw_i} as

$$\lambda_{vw_i} = \frac{1}{d} \sum_k \lambda_{vw_i}^k$$

where d is the degree of v .

Floater later proposed an improved method, called *mean value coordinates*, which derives the weights by using the mean value theorem [7]. The method also guarantees

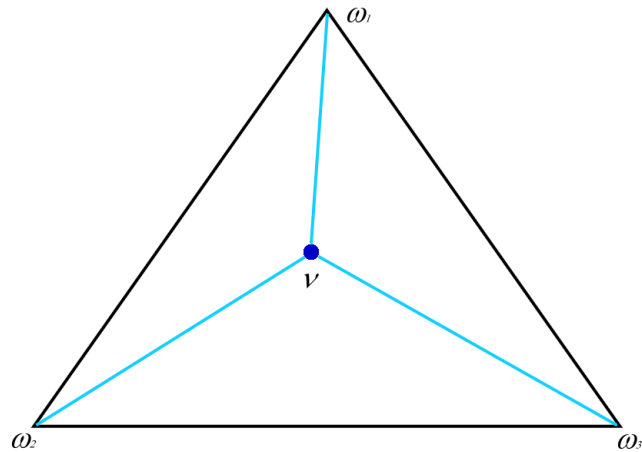


Figure 2.3: Determine λ_{vw} for v that is inside of a triangle.

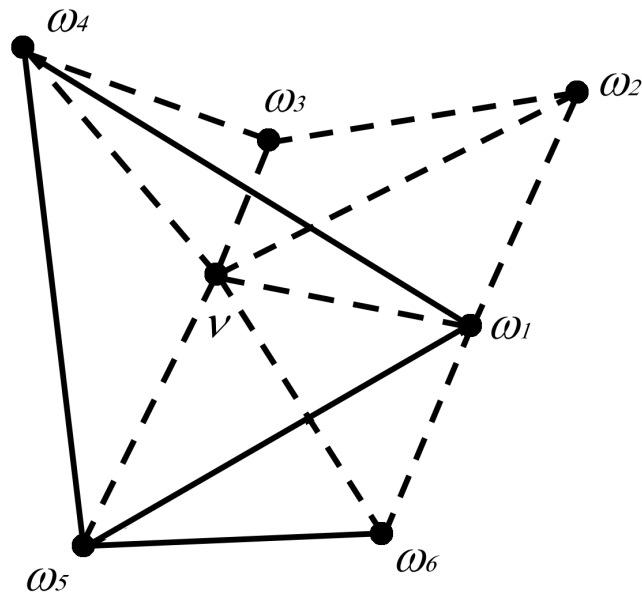


Figure 2.4: Determine λ_{vw} for v inside a n -sided polygon [6].

the existence of bijective mapping and is faster than the shape-preserving parameterization.

Desbrun et al. defined a space of measures spanned by a discrete version of the Dirichlet energy and a discrete authalic energy [4]. While the authalic energy remedies

local area deformations, it requires fixed boundaries and, moreover, produces results that are no better than the one computed by the parameterization using global length preservation.

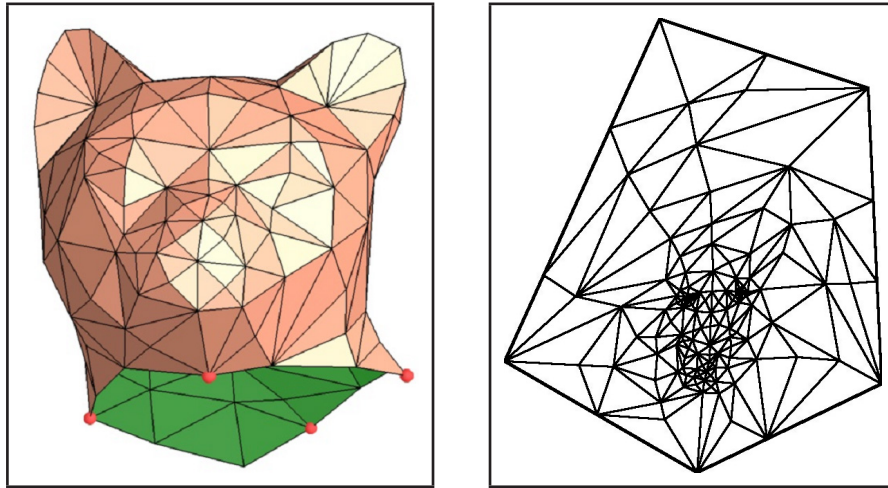


Figure 2.5: A cat head model and its harmonic map [5].

Hormann and Greiner proposed MIPS (Most Isometric Parameterizations) algorithm, which attempts to preserve the ratio of singular values over the parameterization using a hierarchical solver [11, 12]. The method finds a parameterization with “natural boundary” that minimizes the highly non-linear stretch metric. Figure 2.6 shows the MIPS parameterization with natural boundary. However, the metric disregards absolute stretch scale over the surface. As a result, a small domain area can map to a large region on the surface.

Sander et al. proposed a non-linear stretch that integrate the sum of squared singular values over the map. We refer to this metric as geometric stretch. The parameterization is derived by a coarse-to-fine optimization scheme that minimizes the geometric stretch over the map. Note that the resulting parameterization may encounter parametric crack problem.

Sander et al. developed a signal-stretch metric that combines both surface area and surface signal bandwidth [18]. It is shown that the stretch metric is related to SAE

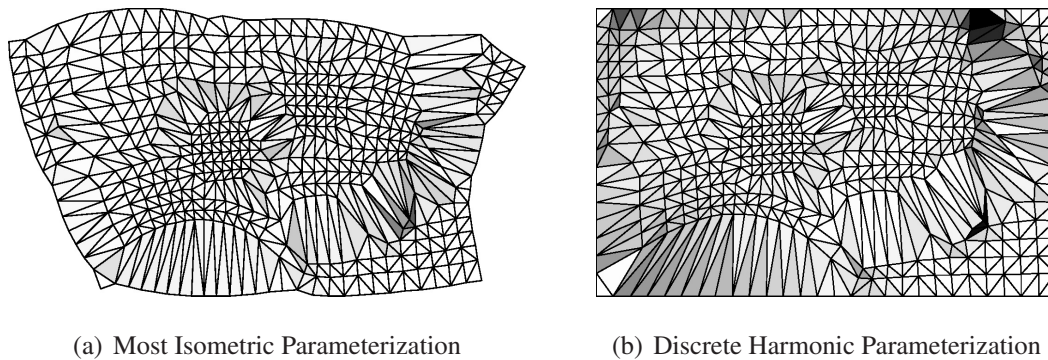


Figure 2.6: Gray-coded deformation energy of different parameterizations [11].

(Signal-Approximation Error) - the difference between a signal defined on the surface and its reconstruction. Sander's signal stretch can be seen as the extension of the geometry stretch. Figure 2.7 shows the results of the parameterization using geometric-stretch and signal-stretch parameterizations.

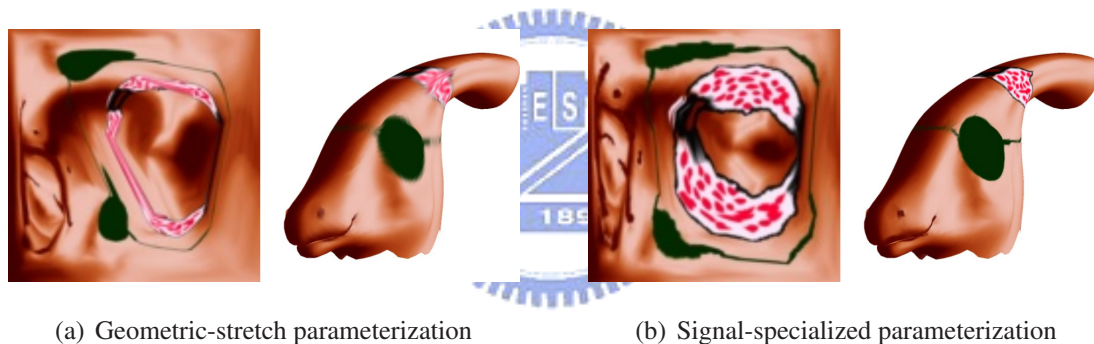


Figure 2.7: Examples of geometric-stretch and signal-stretch parameterizations [18].

Parameterization using either geometric stretch or signal stretch involves a process of expensive non-linear optimization. Yoshizawa et al. developed a simple and fast method that computes the parameterization of low geometry stretch [26]. Floater's shape preserving parameterization [6] is used as an initial parameterization, which is then optimized gradually. At each step, the parameterization generated at previous step is optimized by updating the set of positive convex weights λ_{vw} for each interior vertex

$v \in V_I$ using geometry stretch. The formula is as follows:

$$\lambda_{vw}^{new} = \frac{\lambda_{vw}^{old}}{\sigma_w}, \quad w \in N_v$$

where

$$\sigma_w = \sqrt{\sum A(T_u) \sigma(U_u)^2 / \sum A(T_u)},$$

$$\sigma(U) = \sqrt{(\Gamma^2 + \gamma^2)/2}, \quad U = \langle u_1, u_2, u_3 \rangle \text{ in the parametric plane,}$$

and $A(T_u)$ is the area of triangle T_u and the sums are taken over all triangles T_u surrounding the vertex. After the weights are updated, the new parameterization is computed by solving a linear system, which is fast. Moreover, the parametric cracks that may happen in Sander's global optimization will not be encountered. This method is, however, heuristic and lack of rigorous mathematical support. Nevertheless, it is fast and powerful for generating parameterization with low geometry stretch. See Figure 2.8 for the comparison.

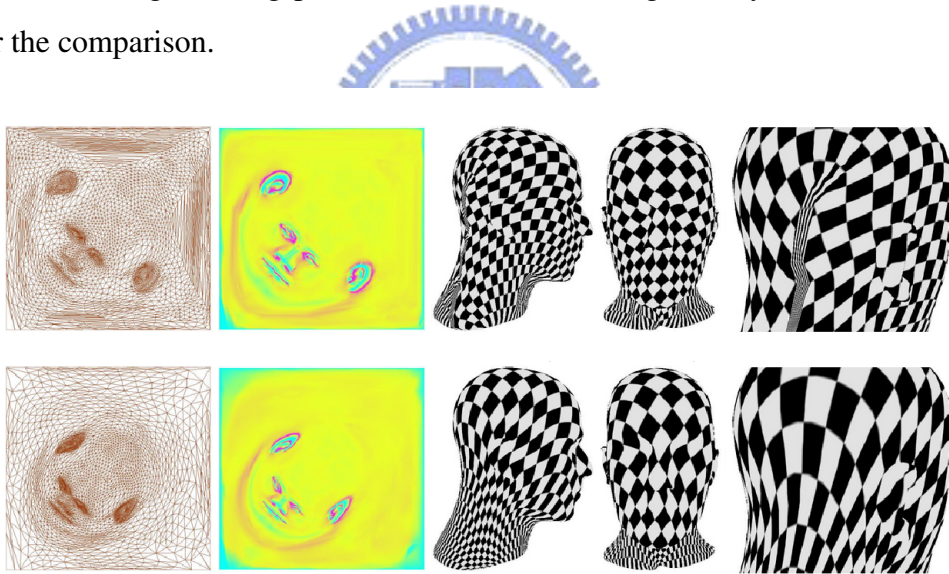


Figure 2.8: Comparison of two mesh parameterization schemes [26]. Top : Stretch minimization of Sander et al [17], Down : Yoshizawa et al. [26]

Another approach that minimizes angular distortion is proposed by Sheffer and Sturler [21]. The parameterization is derived by minimizing the relative distortion of the

planar angles with respect to their counterparts in the three-dimensional space. Though the minimization problem is linear, it becomes non-linear as some other non-linear constraints have to be taken into account in order to generate a valid solution. As a type of discrete conformal parameterization, the method suffers more area distortion, especially in region closed to the center of the surface. The area distortion leads to linear distortion in texture mapping. Sheffer et al. tried to solve the problem by applying a mesh smoothing procedure to the uniform grid overlaying on the parametric domain [22]. The smoothing uses a sizing function which is based on the ratios between the lengths of the edges in the three-dimensional surface and their counterparts in parametric domain. Figure 2.9 demonstrates the result of texture mapping by overlaying a regular grid.

Levy et al. computed quasi-conformal parameterizations by measuring the violation of the Cauchy-Riemann equation in the least square sense [16]. They also show that the quasi-conformal parameterization exists uniquely, independent of resolution and preserves orientations. Using a standard numerical conjugate gradient solver they are able to compute least squares approximations to continuous conformal maps very efficiently without requiring fixed boundary texture coordinates.

2.3 Surface Painting

Surface painting system allows users to paint directly onto a three-dimensional surface and stores the painting result in a texture. The mapping between mesh and texture space is built via a parameterization. In current surface painting systems, the parameterization is pre-computed and remains fixed during the painting process. One and the only thing done by surface painting systems is to sample painting strokes into the texture space. In this way, it is often that insufficient samples will be found in the regions with high painting detail. The left side of Figure 2.10 shows that the aliasing occurs in the region

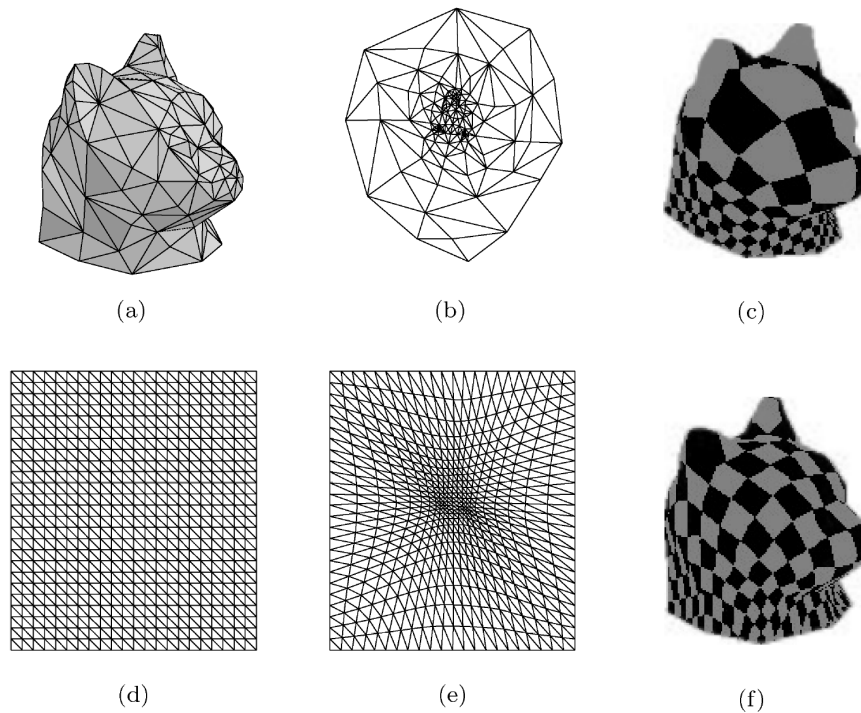


Figure 2.9: Texture mapping for a cat head model [22]. (a) cat head model. (b) ABF parameterization. (c) ABF texturing result. (d) Uniform Grid G_1 . (e) Smoothed grid G_2 . (f) Texturing result after applying G_2 .

spaced in-between black and white. The right part of Figure 2.10 depicts same model with reduced aliasing, as the result of using a parameterization that takes the signal variation into account.

Hanrahan and Haeberli firstly proposed the concept of three dimensional surface painting, in which the color signal is stored directly in mesh vertices [10]. Based on this method, the shading result is interpolated between mesh vertices, though we could not reveal rich texture detail.

Igarashi and Cosgrove stored the paint strokes image that occurred for each pose as separate charts packed into a texture atlas [13]. As shown in Figure 2.11, the eyes and mouth were first painted. Mesh triangles affected by these strokes are found and

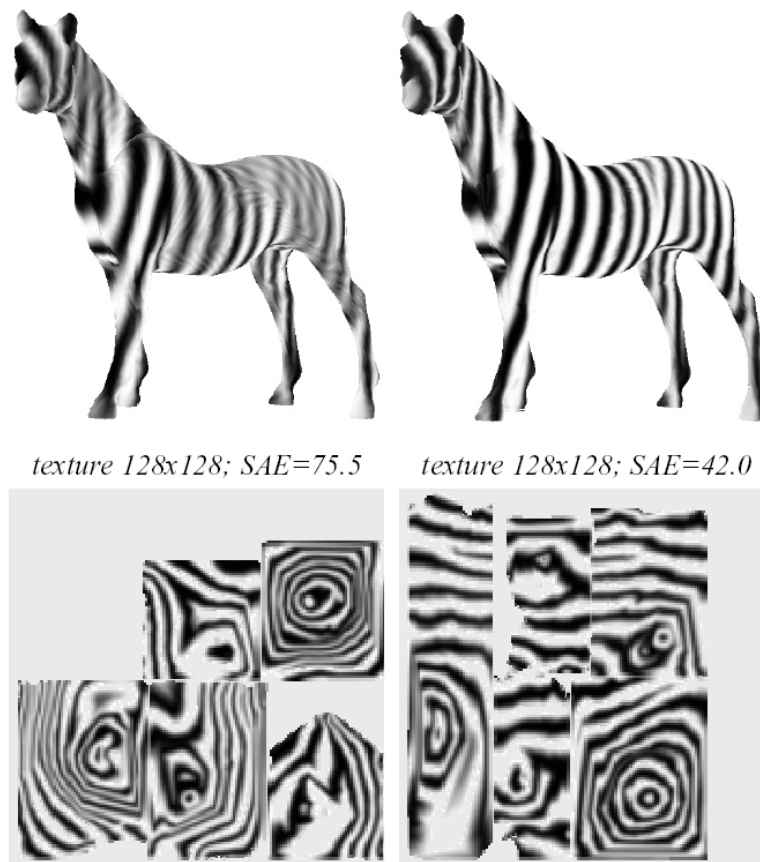


Figure 2.10: Texturing result of geometry stretch and signal stretch [18].

projected onto a two dimensional domain to form an atlas. Similarly, each subsequent stroke is stored in a new atlas. When the painting process complete, all the atlas are packed together to form the final texture atlas. The major disadvantage of the method is that a stroke that overlapping other strokes may appear in more than one atlas; as shown in Figure 2.11. In such cases, texture space may be wasted.

Before going on the approach proposed by Carr et al., we address the concept of procedural texturing. The simplest form of texturing is texture mapping. The texture space is usually created from an image file, often a photograph or artist's rendering of the material. The major disadvantage is that aliasing occurs when the texturing is under

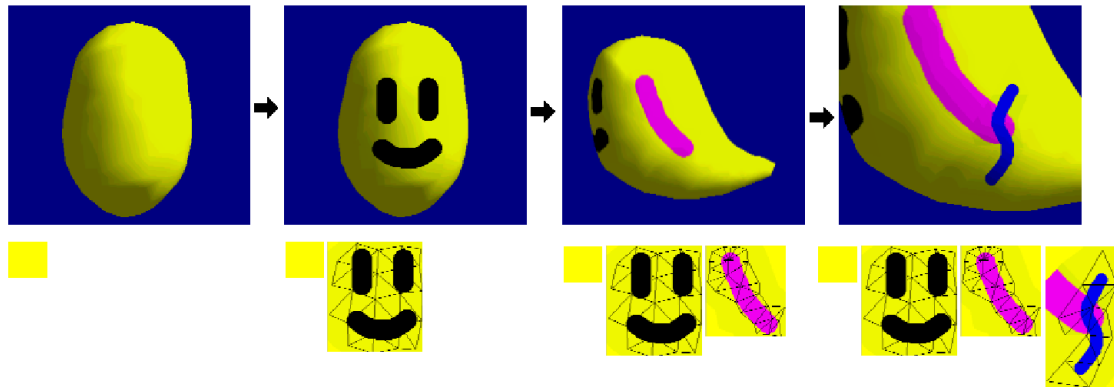


Figure 2.11: Each stroke is stored in an individual atlas [13].

either minification or magnification. Another issue is that the mapping function itself could be complicated. The essential idea of procedural texturing is that the color of a pixel is based functionally on the three dimensional coordinates of its corresponding vertex.

In general, the result of procedural texturing is stored via texture atlases. Some methods for constructing mesh atlas have been proposed, for example, uniform mesh atlas (Figure 2.12), area-weighted mesh atlas (Figure 2.13) and length-weighted mesh atlas. There are two major drawbacks among these methods. The first is that the texture sample space is not completely used. As shown in Figure 2.13, much texture space are wasted at the right-side of the atlas. The second drawback is that these methods do not take the triangles' shape and area into account. As shown in Figure 2.12, each triangle has the same sample space no matter how large it is.

Carr and Hart made use of the Metis algorithm [14] to recursively partition the mesh into several charts; see Figure 2.14. A quaternary tree hierarchy, called MMA, is then constructed. The construction is based on the clustering of charts via a top-down or a bottom-up approach. The MMA framework takes the size of mesh triangles into account such that each triangle could be allocated suitable texture sample space.

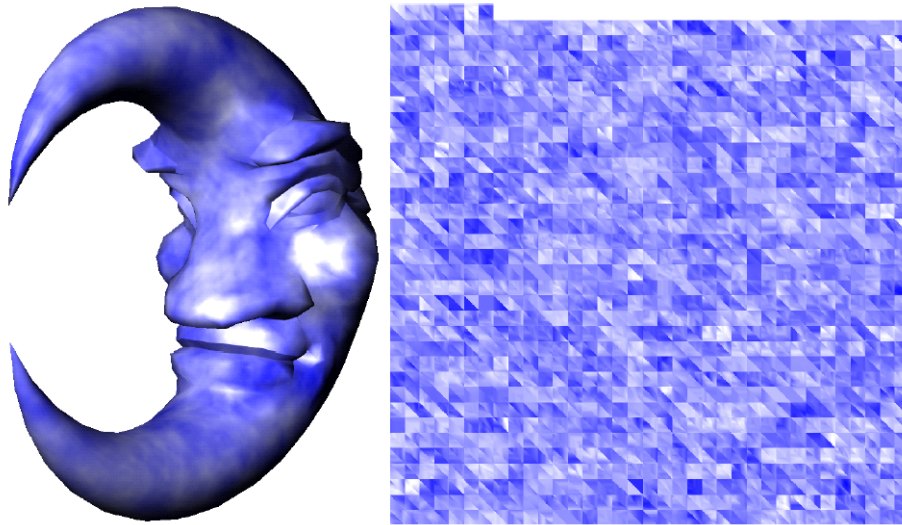


Figure 2.12: Uniform mesh atlas for a cloud textured moon [1].

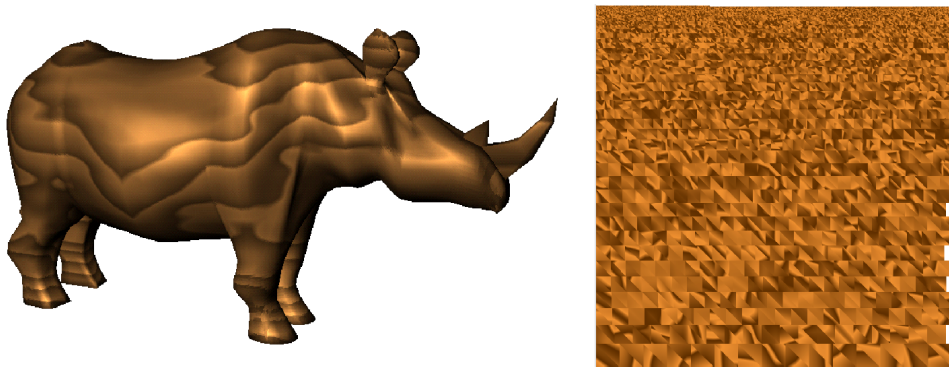


Figure 2.13: Rhino sculpted from wood and its area-weighted mesh atlas [1].

Based on the MMA framework, Carr and Hart proposed a method aiming to derive a parameterization that is sensitive to signal distribution [2]. The method consists of three steps. Firstly, the mesh is divided into several charts based on the method proposed by Sander et al. [19]. Several triangles are chosen as seed of the charts and from each of which faces are joined into chart based on the geometric distance and normal difference between face and chart. Secondly, each chart is parameterized into a two

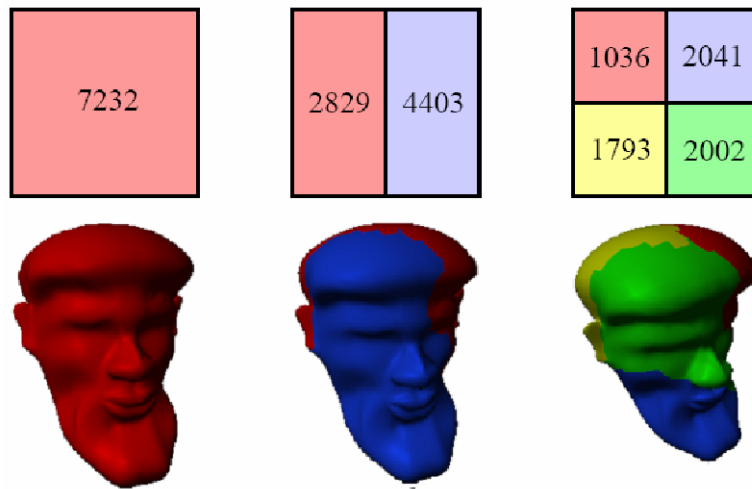


Figure 2.14: Recursively partition the mesh using Metis algorithm [1].

dimensional domain based on Sander et al's geometry stretch metric [17]. Finally, an MMA hierarchy is constructed for all charts built in step one. Each chart has the weight as its L^2 stretch computed in step two. The MMA hierarchy can be presented by a quaternary tree, in which tree nodes at the same level have the same weight, and, in consequence, have equivalent texture space. As shown in Figure 2.15, each of the four child nodes occupies one-fourth texture space of its parent node.

During the surface painting process, all painting strokes are rendered into texture and then the stroke frequency distributed on the texture is analyzed using graphics hardware. After the analysis, an importance value computed from frequency analysis for previous strokes is attached to each chart and the MMA hierarchy (quaternary tree) is re-balanced to generate a new parameterization. All of the charts are placed in a priority queue based on the importance value. The MMA hierarchy is reconstructed via the priority queue. Each chart should consist of a quite large number of faces in order to reduce distortion introduced by the parameterization. Moreover, the re-balancing is more significant with more number of charts. Therefore this method is suitable for meshes with large number of triangles.

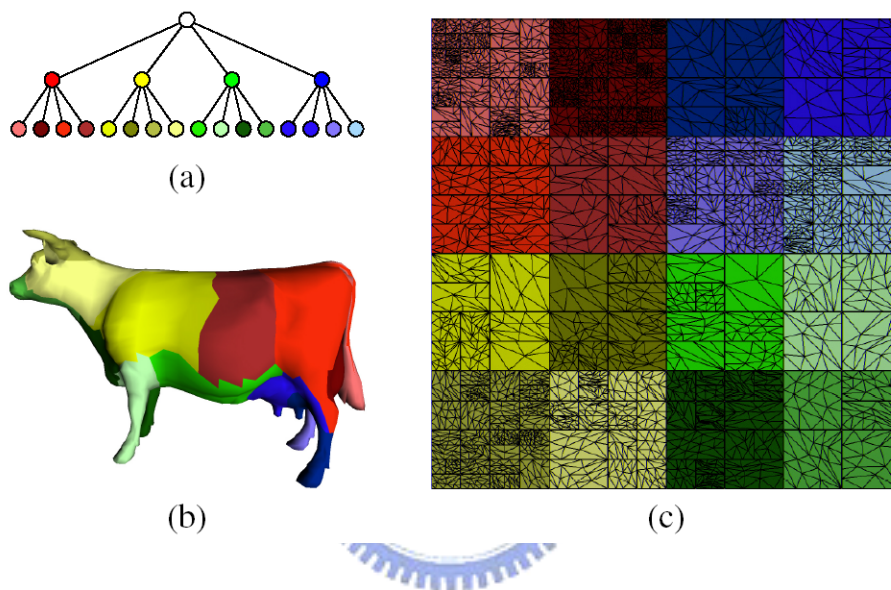


Figure 2.15: A multiresolution meshed atlas of a cow. Each node in the tree corresponds to (a) a cluster of triangles (b) and a square region in the texture domain. (c) Each node's cluster is the union of its children's clusters [2].

A Re-parameterization Framework for Surface Painting

3.1 Approach overview



To optimize the sampling resolution in parametric space, our basic idea is to increase the resolution of regions with high signal variation while decreasing the resolution of other regions.

Our parameterization optimization framework for surface painting comprises the following steps as shown in Figure 3.1:

1. Transform the closed-surface Ω_T^* into an open-surface Ω'_T using topological surgery, construct a global initial parameterization for the surface mesh, and generate a base texture based on the parameterization.
2. Resample painting strokes into the base texture. Analyze the signal frequency on base texture using graphics hardware to generate the importance map. Another

map called geometry stretch map is also computed using graphics hardware.

3. Apply a uniform grid G underlying the parameterization domain, in which each point of G is assigned a L_s^2 stretch value derived from importance map and geometry stretch map, and then apply a two-stage optimization to get an optimized uniform grid G_{opt} .
4. Re-parameterize Ω'_T according to the optimized uniform grid G_{opt} , and resample the painting strokes according to the new parameterization.

Our proposed parameterization optimization framework has the following characteristics:

- Topological surgery is used to automatically transform the closed-surface into an open-surface which is topologically equivalent to a disk.
- A method which analyzes the surface signal variation by using graphics hardware.
- A re-parameterization derived based on optimized uniform grid approach is fast enough for interactive applications.

3.2 Topological surgery

To parameterize Ω_T onto a planar domain, Ω_T should be topologically equivalent to a disk. If Ω_T is a closed-surface we want to transform Ω_T to an open-surface Ω'_T that is equivalent to a topological disk. To achieve this goal we use the topological surgery proposed in [9].

The topological surgery consists of two steps, as depicted in Figure 3.2. In step 1, we find a good cut ρ to reduce the potential distortions of the parameterization. Such a cutting may introduce parametric discontinuity. Several automatic solutions for finding

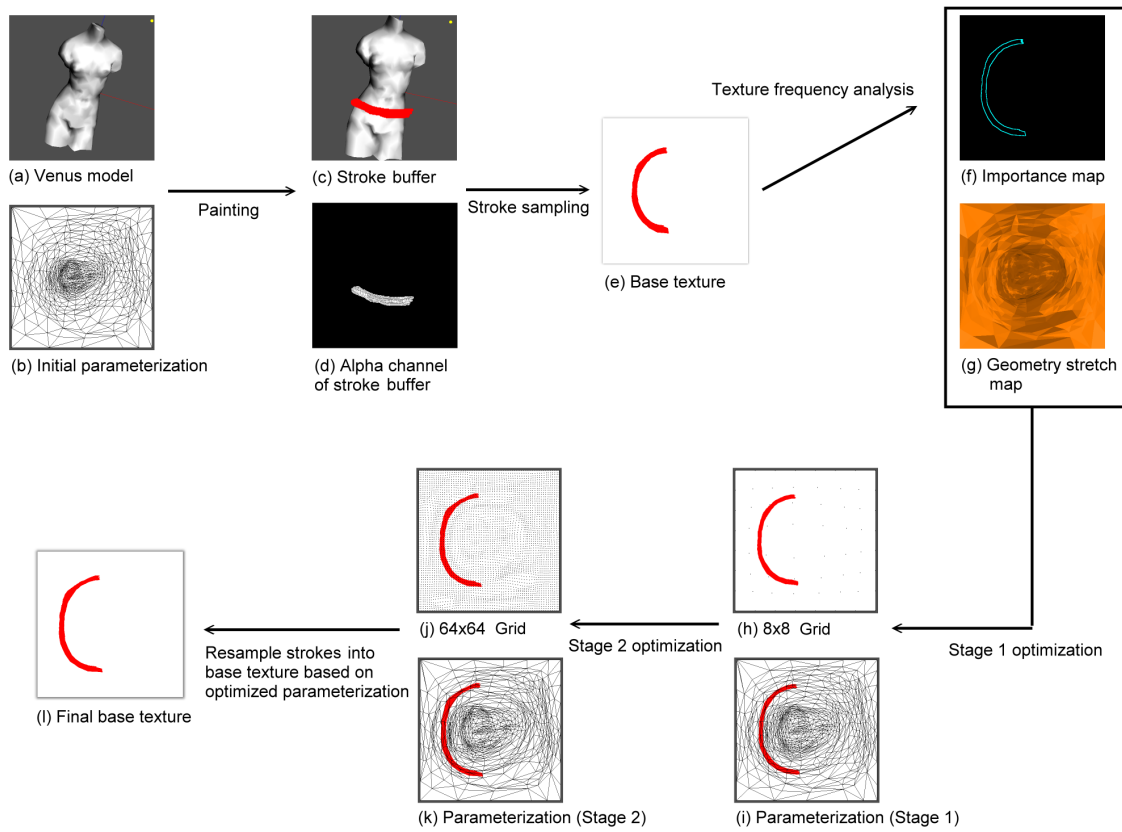


Figure 3.1: The overall process of our method.

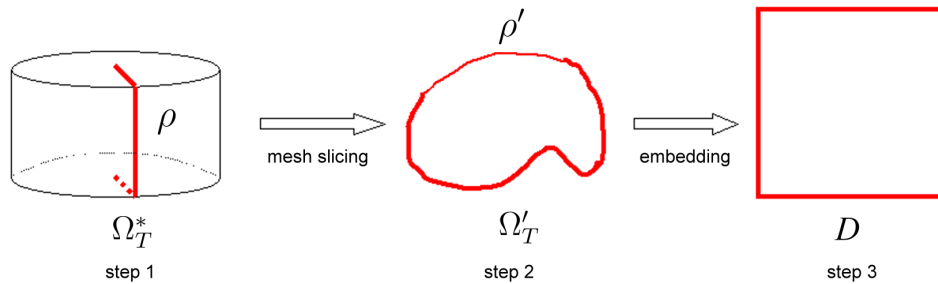


Figure 3.2: Procedure of parameterizing closed-surface [3].

such a cutting while reducing parametric discontinuity have been proposed. To this end, the total length of ρ should be as short as possible [8, 9, 20, 23]. In this thesis, we

slightly modify Gu's cutting algorithm [9]. The algorithm begins by finding an initial cut, and followed by iteratively augmenting the cut in parametric domain D to reduce the potential distortion of the embedding. For each iteration, instead of geometry stretch parameterization [17], a faster parameterization proposed by Yoshizawa [26] is applied to speed up the cutting process.

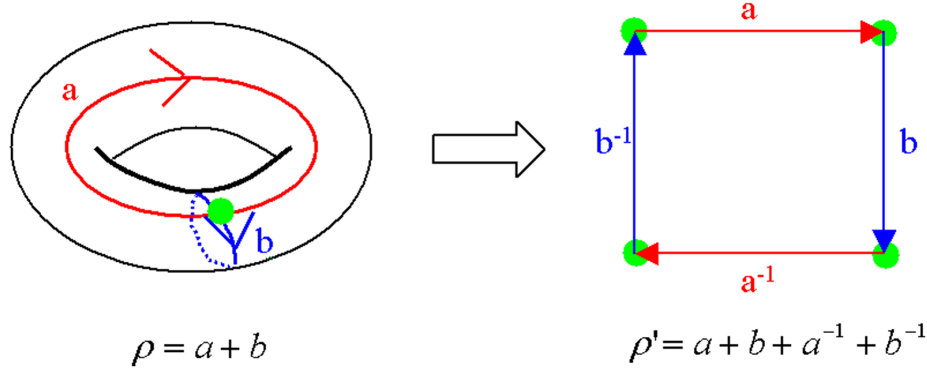


Figure 3.3: Slice the closed-mesh along the cut [3].

After finding a ρ , Ω_T is cut into Ω'_T , which is topologically equivalent to a disk. Each non-boundary edge of ρ is split into two boundary edges to form an open cut ρ' as shown in Figure 3.3. This directed loop of edges ρ' is then the boundary edges of Ω'_T .

We say that two edges in ρ' are mates if they result from the splitting of an edge in ρ . A vertex v with valence k in ρ is replicated as k vertices in ρ' . Vertices in ρ that have valence $k \neq 2$ in the cut are called *cut-nodes* of ρ and ρ' . A *cut-path* is the set of boundary edges and vertices between two ordered cut-nodes in the loop ρ' .

The slicing algorithm starts from a cut node in ρ , and recursively traces the path along the cut until all cut vertices and edges have been produced. Note that, whenever the algorithm traces to a cut node and there are more than two paths to trace, the paths should be traced clockwise to avoid path overlapping.

3.3 Initial parameterization

Although many parameterization techniques are adequate to derive a global initial parameterization, the one aiming to guarantee uniform sampling and preserve conformality structure of the input mesh is most preferable. Here, we use the method proposed by Yoshizawa et al. [26] because it meets the preferable properties and requires solving a simple, sparse linear system, which is usually handled in a matter of seconds using Conjugate Gradient solver with good preconditioning.

3.4 Stroke sampling

To sample painting stroke into texture, we use the method proposed by Carr et al [2], in which each paint stroke applied in the same object pose (i.e. modelview coordinates of the model) is rendered directly into base texture map using graphics hardware. For this task, we need a stroke buffer for storing the painting data and a depth buffer for the depth of current object pose.

The resampling is done by a vertex shader and a fragment shader. The vertex shader transforms the world space position into model view coordinates and then swaps each vertex's model view coordinates with its texture coordinates. The fragment shader is applied to render the new base texture by taking the stroke buffer, depth buffer, and the original base texture as input. The alpha channel in stroke buffer represents the existence of paint strokes to ensure that only the strokes can overwrite the existing base texture. The depth buffer is used to prevent paint being applied to invisible portions of the model.

This process is performed for the stroke painted at each pose. The following pseudo code gives an overview of this method.

Pseudo code 1*// Vertex Shader .***Procedure** vertexShader()*// take input texture coordinates as output vertex coordinates**OUT.Pos ← IN.texCoord**// take model view coordinates as output texture coordinates**OUT.Tex ← mul(mvp, IN.position)**// Fragment Shader .***Procedure** fragmentShader()*// oldColor got from original base texture**oldColor ← tex2D(baseMap, t1)**// newColor got from stroke buffer**newColor ← tex2D(strokeMap, t0)**// depth value got from depth buffer**shadowCoeff ← tex2D(shadowMap, t0)**// alpha channel of stroke buffer stores the existence of stroke color**test ← newColor.alpha * shadowCoeff***if** *test* > 0 **then***result ← newTexColor***else***result ← oldTexColor***end if**

3.5 Importance map and geometry stretch map

The signal stretch derived by Sander et al. [18] is define as:

$$E_h(s, t) = \|h(s, t) - \tilde{h}_{ij}(s, t)\|^2 = \|h(s_i + \hat{s}, t_j + \hat{t}) - h(s_i, t_j)\|^2$$

where $(s, t) = (s_i + \hat{s}, t_j + \hat{t})$, as shown in Figure 3.4, h is the function that maps (s, t) from texture domain to signal domain and \tilde{h} is its reconstruction from a discrete sampling of the texture domain given by $\tilde{h}_{ij}(s, t) = h(s_i, t_j)$. Therefore $E_h(s, t)$ estimates the difference between $(s_i + \hat{s}, t_j + \hat{t})$ and (s_i, t_j) , i.e. $E_h(s, t)$ represents the *gradient* of $h(s, t)$. Therefore $E_h(s, t)$ can be re-written as

$$E_h(s, t) = \frac{\partial h}{\partial \mathbf{s}} \cdot \frac{\partial h}{\partial \mathbf{s}} + \frac{\partial h}{\partial \mathbf{t}} \cdot \frac{\partial h}{\partial \mathbf{t}}$$

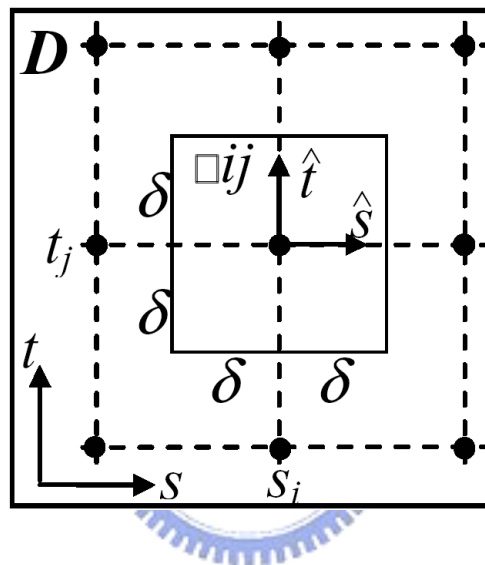


Figure 3.4: Texture domain.

To analyze the base texture for finding regions that requires additional samples, a four-tap gradient magnitude filter is used in [2] to find undersampled regions. The four-tap filter fetches four samples from the input texture, and outputs the result in half resolution. For the four-tap gradient magnitude filter, some gradient features will be missed. For example, as shown in Figure 3.5, each red rectangle represents 4 pixels on the texture, and we detected the gradient in s -direction of paint (a), but not in paint (b).

Here we modify previous four-tap filter. For each pixel on the base texture, we calculated its magnitude of the gradient using fragment shader arithmetic by central difference, as shown in Figure 3.6. Actually, this is the *Sobel Filter* in the field of image

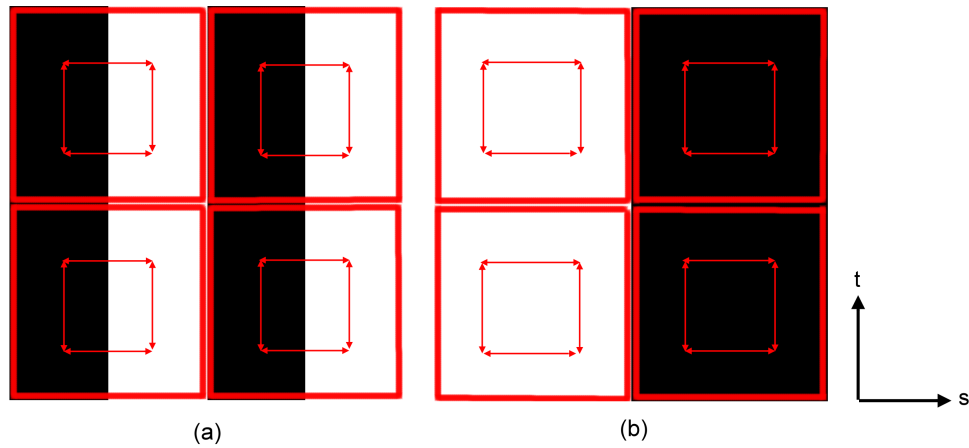


Figure 3.5: Problems of four-tap filter.

processing. The filter is applied for each pixel of the base texture, therefore the output image is the same resolution as the base texture. Figure 3.7 demonstrates the result of two filters which shows that our modified filter is more accurate than four-tap filter.



g_{00}	g_{01}	g_{02}
g_{10}	g_{11}	g_{12}
g_{20}	g_{21}	g_{22}

$$\frac{\partial S}{\partial u} = g_{22} + g_{02} - g_{00} - g_{20} + 2 \cdot (g_{12} - g_{10})$$

$$\frac{\partial S}{\partial v} = g_{20} + g_{22} - g_{00} - g_{02} + 2 \cdot (g_{21} - g_{01})$$

Figure 3.6: Our filter.

Besides the importance map, another map called geometry stretch map is also computed. This geometric stretch map stores L^2 stretch value for each face on parametric domain as shown in Figure 3.8. We normalize the value of geometric stretch of each face to lie between 0 and 1. Next, we render the mesh on parametric domain using the normalized geometric stretch value as the color of the face.

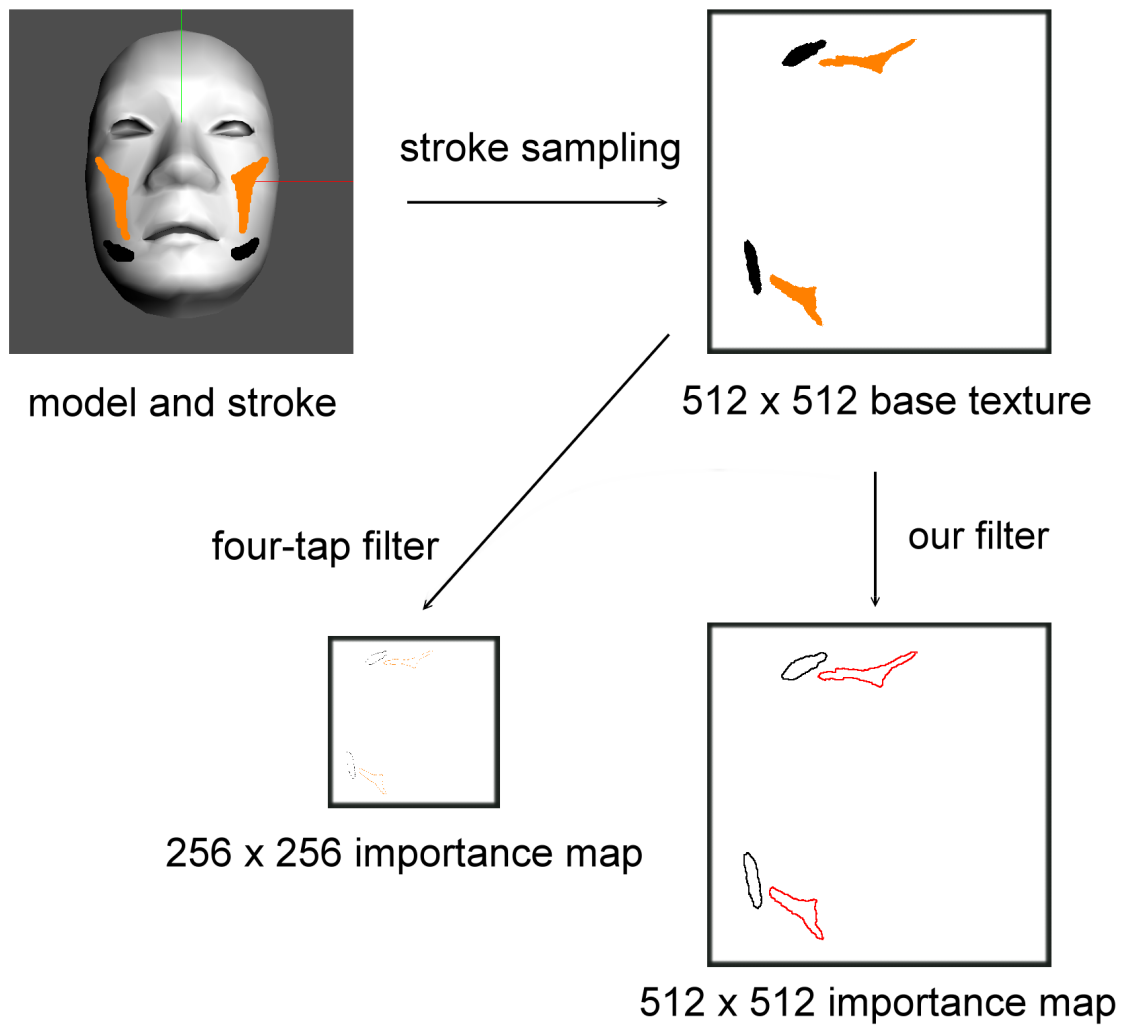


Figure 3.7: Four-tap filter and our filter.

3.6 The L_s^2 stretch

After the generation of importance map and geometry stretch map, the L_s^2 stretch is derived from these two maps. As mentioned in [18], the signal stretch can have zero gradient since the signal may be locally constant on a region of the surface. Therefore, a tiny fraction of geometry stretch is added into the energy function to be minimized.

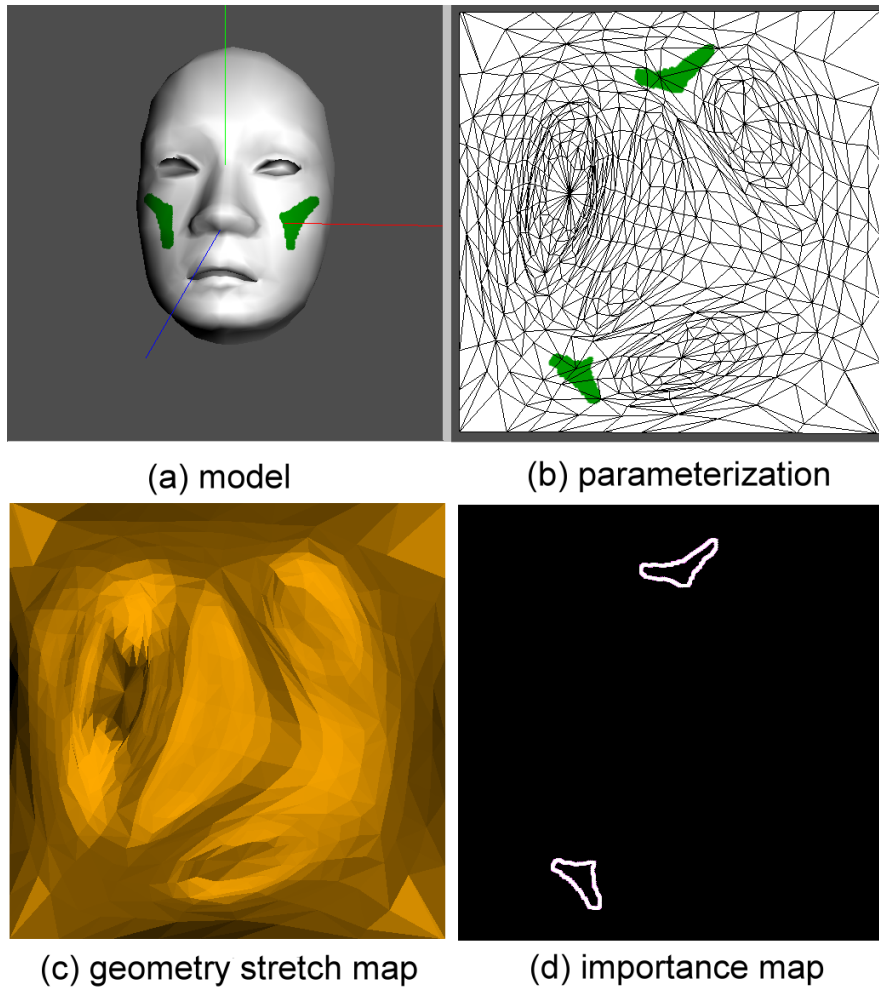


Figure 3.8: Face model with its parameterization, geometry stretch map and importance map.

The L_s^2 stretch is defined as follows:

$$L_s^2(s, t) = \begin{cases} 1 - L^2(s, t) & , \text{if } E_h(s, t) = 0 \\ 1 - (\alpha \cdot L^2(s, t) + \beta \cdot E_h(s, t)) & , \text{otherwise.} \end{cases}$$

where $E_h(s, t)$ is the signal stretch proposed by Sander et al. [18], and $L^2(s, t)$ is the geometry stretch [17]. The two values are obtained from importance map and geometry stretch map, respectively. In the region with signal variation, we use the weighted

geometric stretch and signal stretch as in [18]. Otherwise, in the region without signal variation, we purely take the geometry stretch into account to prevent undersampling in regions with no signal variation. The L_s^2 stretch could be considered as the extension of signal stretch.

3.7 Rapid re-parameterization

This section describes the main contribution of this thesis, an iterative optimization framework for the re-parameterization procedure. After constructing an initial parameterization, we re-parameterize Ω'_T in response to the strokes painted by the user. The objective of the re-parameterization is to assign more texture samples to the regions with high signal variation.

3.7.1 Iterative optimization based on uniform grid

For interactive applications, the parameterization proposed by Sander et al. [18] has two major problems when it is applied to surface painting systems. First, since the signal introduced by painting strokes is not constant over the triangle, numerical integration is used to compute the signal stretch on each triangle. All the mesh triangles are subdivided into 64 sub-triangles and the signal stretch are evaluated at all the vertices. The second drawback is that the optimization process proposed by Sander et al. is a non-linear, global optimization. As a result, the parameterization is expensive and therefore not suitable for interactive surface painting applications.

To reduce the cost of computing signal stretch, instead of subdividing each triangle, we derive the signal stretch on parametric domain. We apply an $N \times N$ uniform grid G to the parametric domain in which the initial parameterization lies as shown in Figure 3.9. These grid points, rather than the mapping of mesh vertices, are used to sample L_s^2 stretch on parametric domain, that is, we compute L_s^2 stretch for each grid point. Such

an approach allows us to control the sampling resolution. Moreover, the grid is used to be the target for stretch optimization. By doing this, the computational complexity of performing optimization will be dependent on the resolution of the grid, rather than the mesh.

We then optimize G by the following steps:

1. For each point $N \in G$, derive $L_s^2(N)$ from importance map and geometry stretch map by graphics hardware.
2. For each interior point $N_i \in G$ in turn,

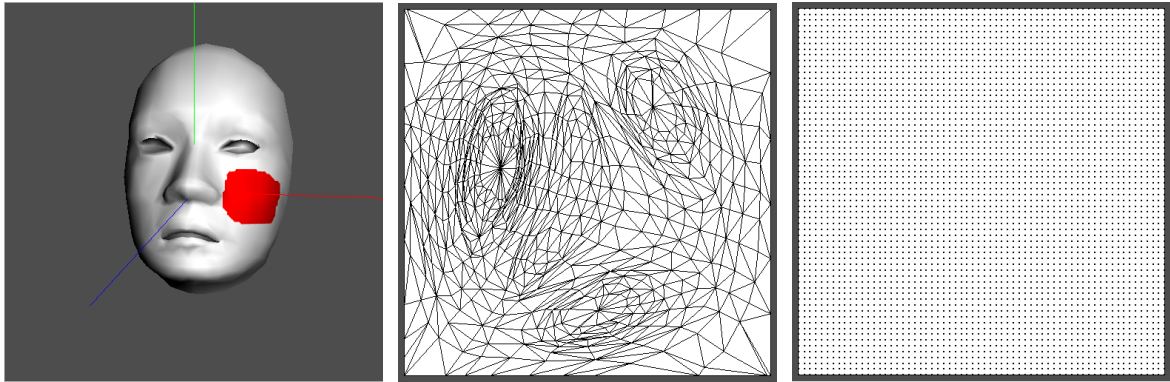
$$\text{compute } \widetilde{N}_i = \frac{\sum_{N' \in 1\text{-ring of } N_i} L_s^2(N') \cdot N'}{\sum_{N' \in 1\text{-ring of } N_i} L_s^2(N')},$$

$$\text{set } N_i = \widetilde{N}_i.$$

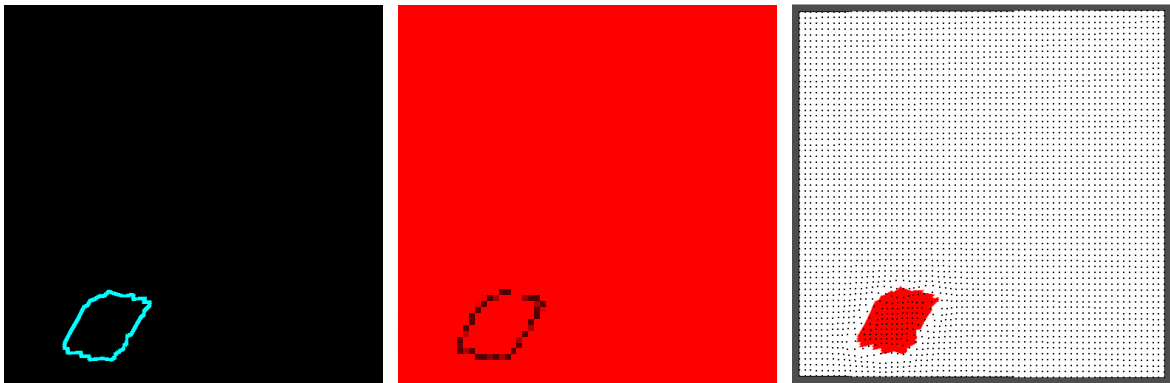
3. Repeat 1 and 2 until $\|\widetilde{N}_i - N_i\| < \epsilon$ for every i .

Figure 3.9(a) shows the face model with a red painting stroke and the resulting importance map is shown in Figure 3.9(d). A 64×64 uniform grid G is applied to the parametric domain, where each sample point is assigned a L_s^2 stretch value as shown in Figure 3.9(e) (we take only signal stretch into account in this case). Figure 3.9(f) shows the optimized grid G_{opt} , where the sample points are more sparse in the regions with signal variation. The optimization procedure on the grid points is illustrated in Figure 3.10. Figure 3.10(a) depicts the grid points and the corresponding parametric domain with signal distributed. Since the L_s^2 stretch values of p_2 , p_7 and p_{12} are smaller than that of p_0 , p_5 and p_{10} , p_1 , p_6 and p_{11} are moved toward p_0 , p_5 and p_{10} . Similarly, p_3 , p_8 and p_{13} are moved toward p_4 , p_9 and p_{14} ; as shown in Figure 3.10(b)(c).

The optimization procedure is an iterative optimization process, in which the local optimization optimizes a grid point in one iteration. After the optimization, we will get an optimized uniform grid G_{opt} . On G_{opt} , grid points will become dense in the regions

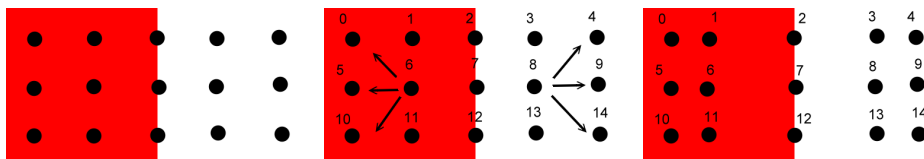


(a) The face model with a painting stroke. (b) Initial parameterization. (c) Initial uniform grid G .



(d) Importance map. (e) Stretch value on each sample points (64×64). (f) Optimized uniform sample points G_{opt} .

Figure 3.9: The optimization result base on one iteration.



(a) Initial grid. (b) The optimization process. (c) The optimized grid.

Figure 3.10: Chart diagram of the optimization process.

with high L_s^2 stretch (lower signal variation), and sparse otherwise. After the optimization process, the underlying parameterization will be re-computed by barycentric

interpolation according to the optimized grid points as described in next section.

3.7.2 Re-parameterization

After optimizing the initial uniform sample points G , we re-parameterize the parameterization by the barycentric interpolation based on the optimized uniform sample points G_{opt} . For each vertex $v_j \in V_I$, let N^{j_0} , N^{j_1} , N^{j_2} and N^{j_3} be the sample points of the cell that contains v_j . Barycentric coordinates w_0 , w_1 , w_2 and w_3 are derived such that

$$v_j = \sum_{i=0}^3 w_i \cdot N^{j_i}.$$

The new position of v_j will be

$$v_j^{opt} = \sum_{i=0}^3 w_i \cdot N_{opt}^{j_i},$$

where $N_{opt}^{j_0}$, $N_{opt}^{j_1}$, $N_{opt}^{j_2}$ and $N_{opt}^{j_3}$ are the homologous points of N^{j_0} , N^{j_1} , N^{j_2} and N^{j_3} in G_{opt} .

Figure 3.11 demonstrates the re-parameterization process for venus model with no painting strokes. Figure 3.11(c) shows that the geometry stretch is high in the center of parametric domain. Therefore, the central region should have more texture space to minimize the overall geometry stretch. After the process, Figure 3.11(d) shows the optimized uniform grid G_{opt} in which the grid points become dense in high stretch areas and sparse otherwise. After the barycentric interpolation, the central region on parametric domain will be assigned more texture space. Figure 3.11(e) shows the optimized parameterization, where geometry stretch is reduced from 0.075163 to 0.069868.

3.7.3 Stroke resampling over optimized parameterization

Finally, we resample the base texture based on the optimized parameterization. The sampling process is similar to the method mentioned in section 3.4. The only difference

is that now we have two texture coordinates, i.e. parameter values t_{prev} and t_{opt} for each vertex, which are derived from initial parameterization ϕ and optimized parameterization ϕ_{opt} , respectively. As described in section 3.4, we first swap each vertex's model view coordinates with its current texture coordinates t_{opt} in vertex shader, and then we resample painting strokes to form a new base texture. The resampling procedure here consists of two step. The first step resamples current painting strokes stored in stroke buffer; step two resamples previous painting strokes stored in the previous base texture. Therefore current model view coordinates and t_{opt} are used to sample current stroke from stroke buffer and t_{prev} is used to sample previous stroke from previous base texture.

3.8 Two-stage re-parameterization framework

Compared to Sander's signal-specialized parameterization [18], the proposed framework tends to be a local optimization process. Figure 3.12 shows the re-parameterization result using a 256x256 uniform sample points. We can see that the relaxation of sample points is bounded inside the cell it lies. As shown by the red arrow in Figure 3.12, there should be less sample space in these regions with lower signal gradient. However, the movement of the sample points in these regions is not much due to the fact that the L_s^2 stretch of these points are almost the same. Therefore the relaxation works well in the regions with high gradient, but may not work well in the other regions. To solve this problem, a two stage optimization framework is used instead of the single stage optimization.

In the two-stage optimization, we expect that the first stage diminishes the texture sample space in region with lower signal gradient and the second stage magnifies the texture space in regions with high signal gradient. To achieve this goal, a lower resolution uniform grid is used in the first stage and a high resolution grid in the second stage.

Figure 3.13 illustrates the optimization result using a high resolution grid. As shown in Figure 3.13(c), only these sample points near the regions of low L_s^2 stretch value (high signal variation) will be moved after the iterative optimization. Other sample points will remain fixed in other regions where neighboring points have the same L_s^2 stretch value. Figure 3.13(d) shows the final result of the optimization. The regions with lower signal stretch are expected to obtain less texture space. Apparently, optimization using a high resolution grid does not work well for this purpose, see the comparison highlighted by the blue circle in Figure 3.13(a) and Figure 3.13(d).

The optimization resulting from using a lower resolution grid will have more convergence effect in the regions of high L_s^2 stretch (lower signal variation), and allocate less texture space in these regions. See the comparison shown in Figure 3.14(a) and Figure 3.14(d).

Compare to Figure 3.11, Figure 3.15 shows the two-stage optimization result of the venus model. A 8×8 uniform grid is used in the first stage and a 64×64 uniform grid is used in the second stage. The geometry stretch is reduced from 0.075163 to 0.059678, which is better than that of single stage optimization.

Figure 3.16(b) shows that the sample points of 16×16 resolution in the first stage and Figure 3.16(d) is the result of using the sample points of 256×256 resolution in the second stage. We see that the texture space in regions of lower signal gradient is diminished in stage one; as shown in Figure 3.16(c), while in stage two, more texture space in the regions of high signal gradient are allocated; see Figure 3.16(e). Figure 3.17 shows the result of single stage optimization and two stage optimization for comparison. Obviously, the texture space is used more efficiently using the two stage optimization method, especially in the regions of lower signal gradient, see the comparison highlighted by the red arrows in Figure 3.17.

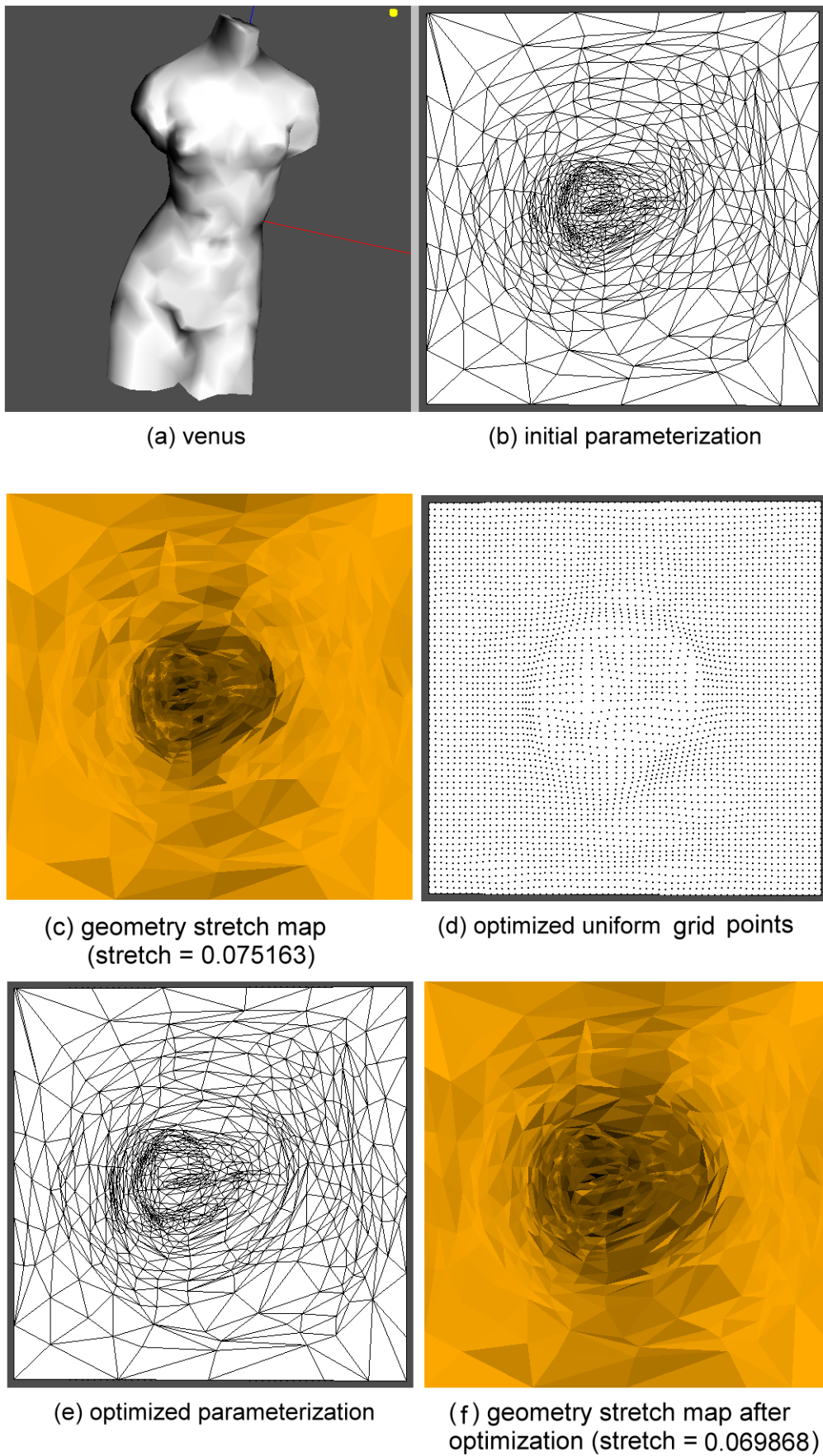


Figure 3.11: Venus model and optimized parameterization.

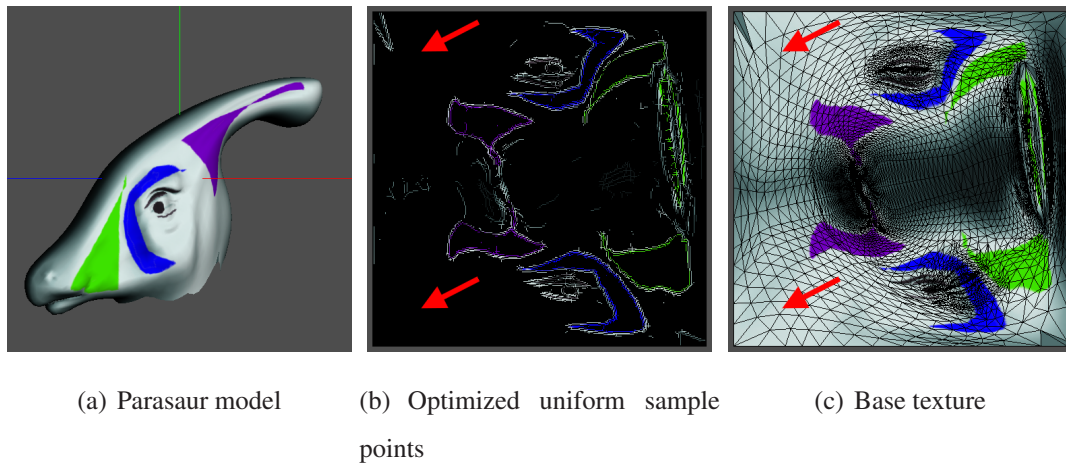


Figure 3.12: Parasaur model : single stage optimization using 256x256 uniform sample points

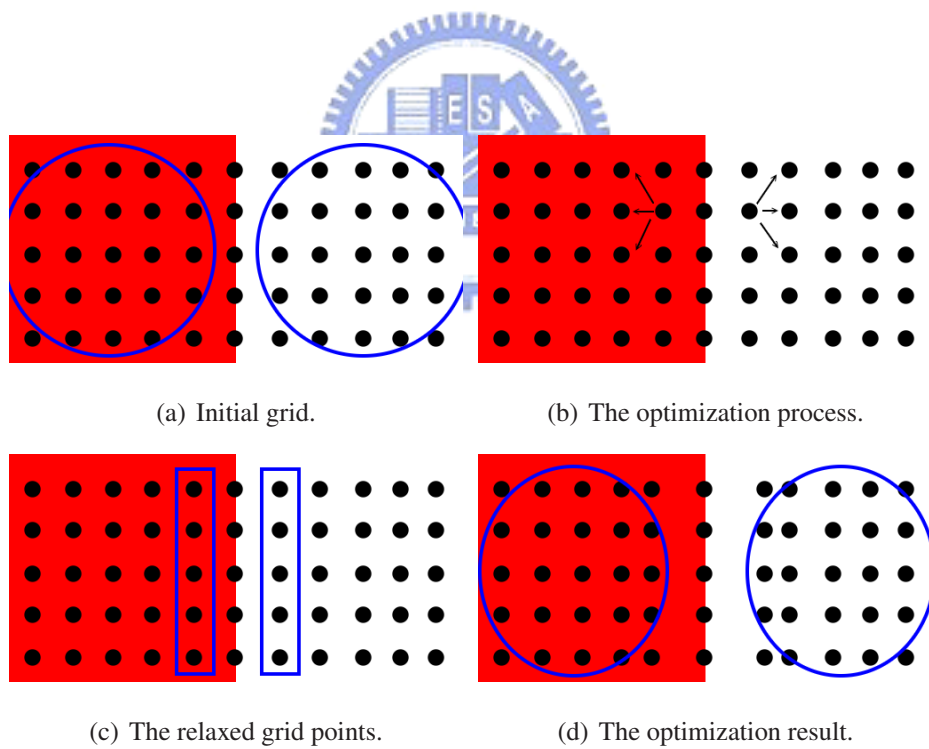


Figure 3.13: High resolution uniform grid points.

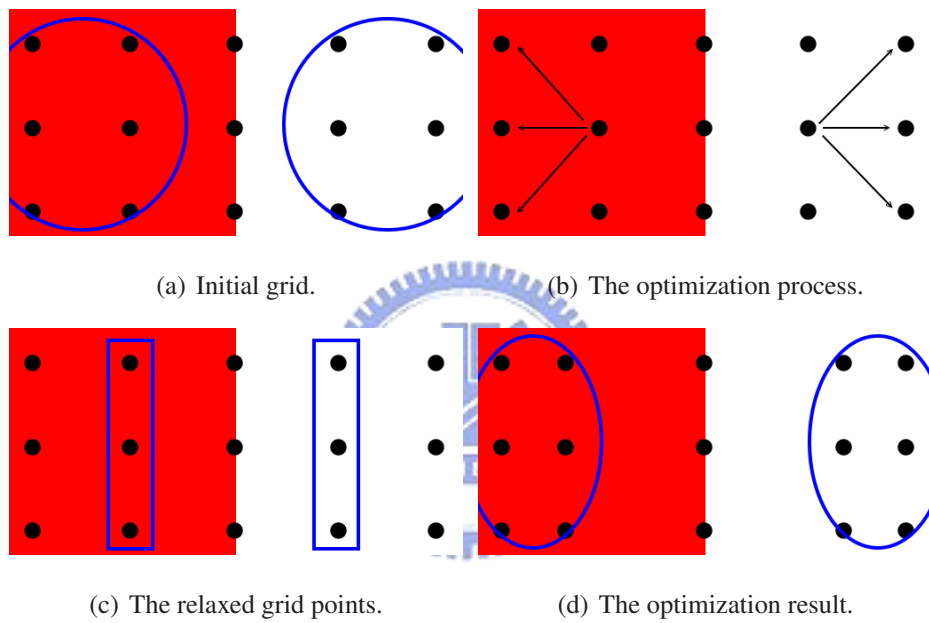


Figure 3.14: Low resolution uniform grid points.

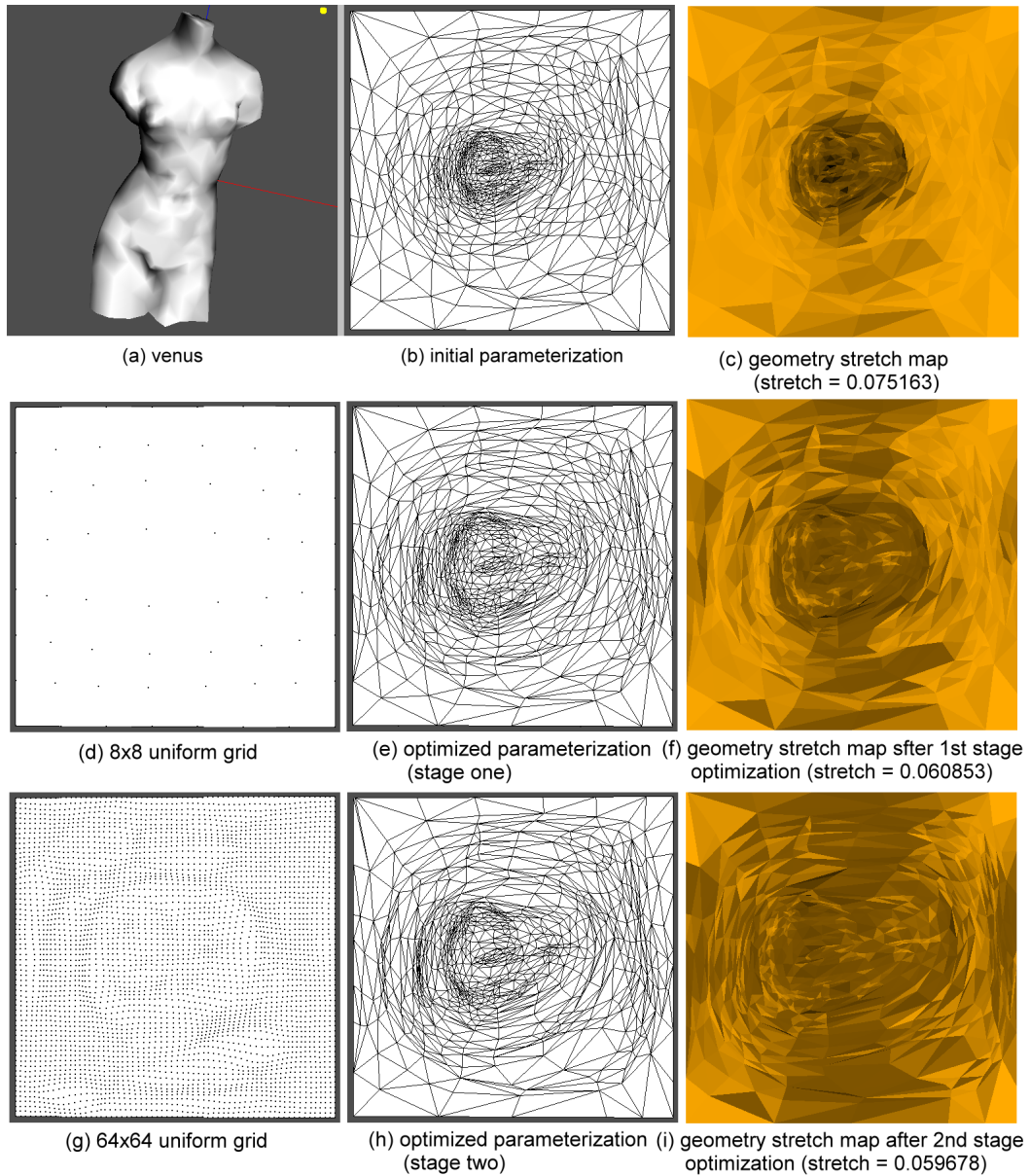
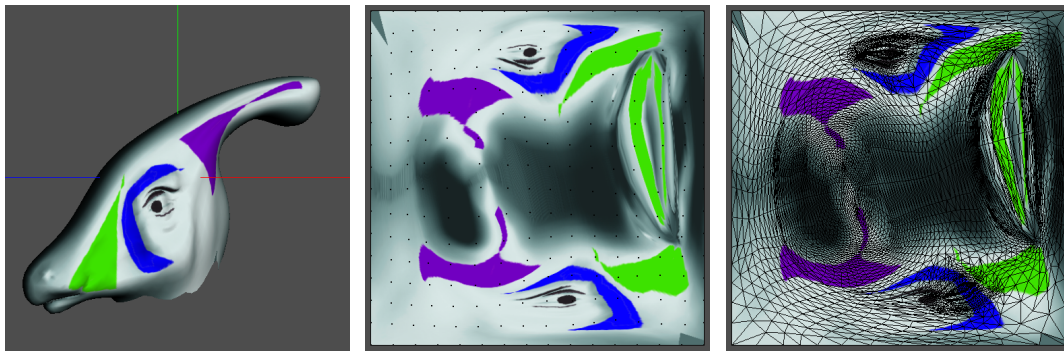
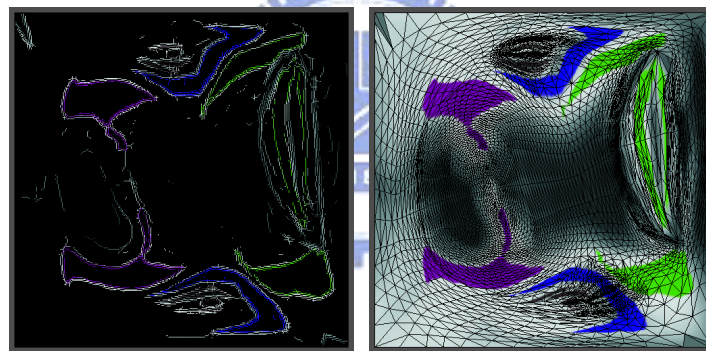


Figure 3.15: Venus model and two-stage optimized parameterization.

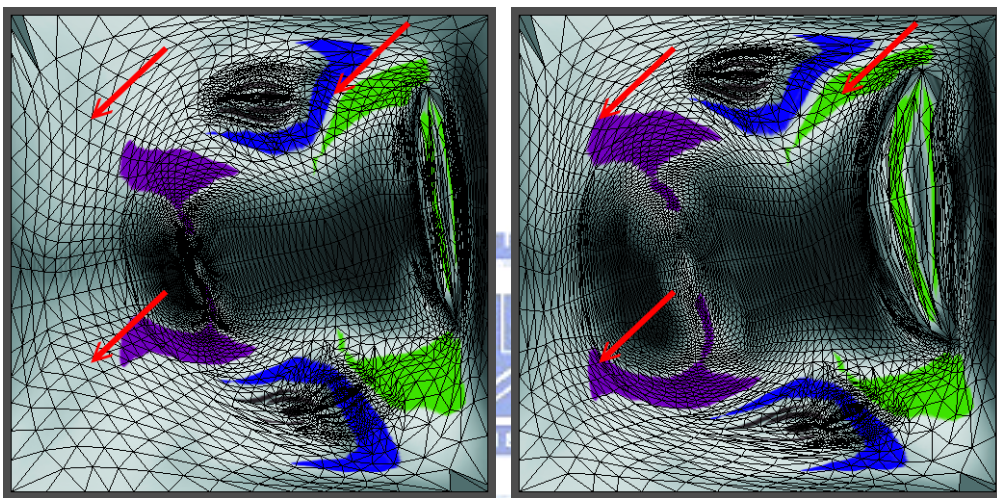


(a) Parasaur model. (b) Optimized 16×16 grid in the first stage. (c) Base texture (Stage 1).



(d) Optimized 256×256 grid in the second stage. (e) Base texture (Stage 2).

Figure 3.16: Parasaur model : Two-stage optimization using 16×16 and 256×256 grids.



(a) Single stage optimization

(b) Two stage optimization

Figure 3.17: Comparison of single and two stage optimization

Results and Performance Analysis

All results are performed with a AMD Athlon64 3000+ PC, 512 MB RAM and an NVIDIA GeForce 6800 graphics card. It is running Windows XP with NVIDIA Cg 1.3 compiler, vp40 vertex shader profile and fp40 fragment shader profile. We use the pBuffer extension for efficient texture rendering. We demonstrate the result of our method applied to surface painting in section 4.1. We compare our result with that based on static parameterization in current surface painting systems. In section 4.2, we analyze the performance of our method for interactive use. In section 4.3, we compare the texturing result of our parameterization with signal-specialized parameterization[18] proposed by Sander et al. Finally, a simple comparison between *Painting Detail* [2] and our method is given in section 4.4.

4.1 Results of surface painting

In current surface painting systems, the underlying surface parameterization is fixed during surface painting process. Therefore some texturing artifacts appear in the regions where texture samples are insufficient. In this section, we demonstrate the effect of our optimization framework used in surface painting system.

We paint stroke onto the mesh surface directly and the strokes are stored in strokes buffer. The strokes are rendered by OpenGL “*GL_POINTS*” and “*GL_POINT_SMOOTH*” procedure.

We paint the wear and a flag on the back of the venus body. Figure 4.1 and Figure 4.2 show the painting result of our surface painting system. The left columns show the result of current surface painting systems, i.e. with fixed underlying parameterization. The right columns show the result of our two-stage optimization process. Our method depicts better texturing quality than that for current surface painting systems.

Figure 4.3, Figure 4.4 and Figure 4.5 demonstrate other painting results. Aliasing occurs in undersampling regions and our method alleviate this problem efficiently.

The strokes of all the results shown in Figure 4.1 to Figure 4.5 are painted manually. Figure 4.6 shows the result where an image is texture mapped to simulate painting strokes. The artifact, blur, occurs due to the fact that texture is undersampled using fixed parameterization as shown in the left column of Figure 4.6. The right column shows that the result of two-stage optimization is much more better. Figure 4.7 demonstrates another result. Four images is texture mapped to simulate painting strokes for the parasaur model.

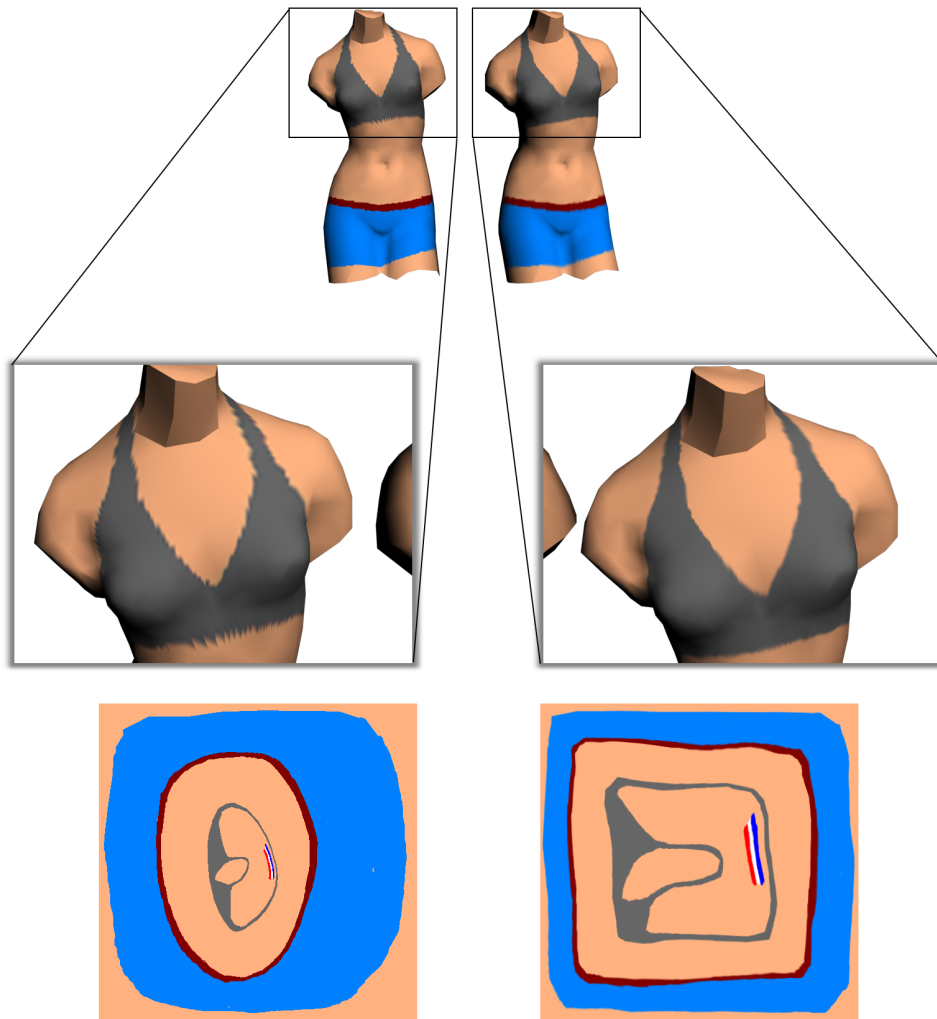


Figure 4.1: Painting results of the venus model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).

4.2 Analysis of interactive application

The performance of re-parameterization is an important issue for surface painting system. Table 4.1 shows the computation time for initial parameterization, two-stage optimization and re-parameterization occurs during surface painting process. Because the optimization procedure is done on parametric domain, the computation cost of two-stage

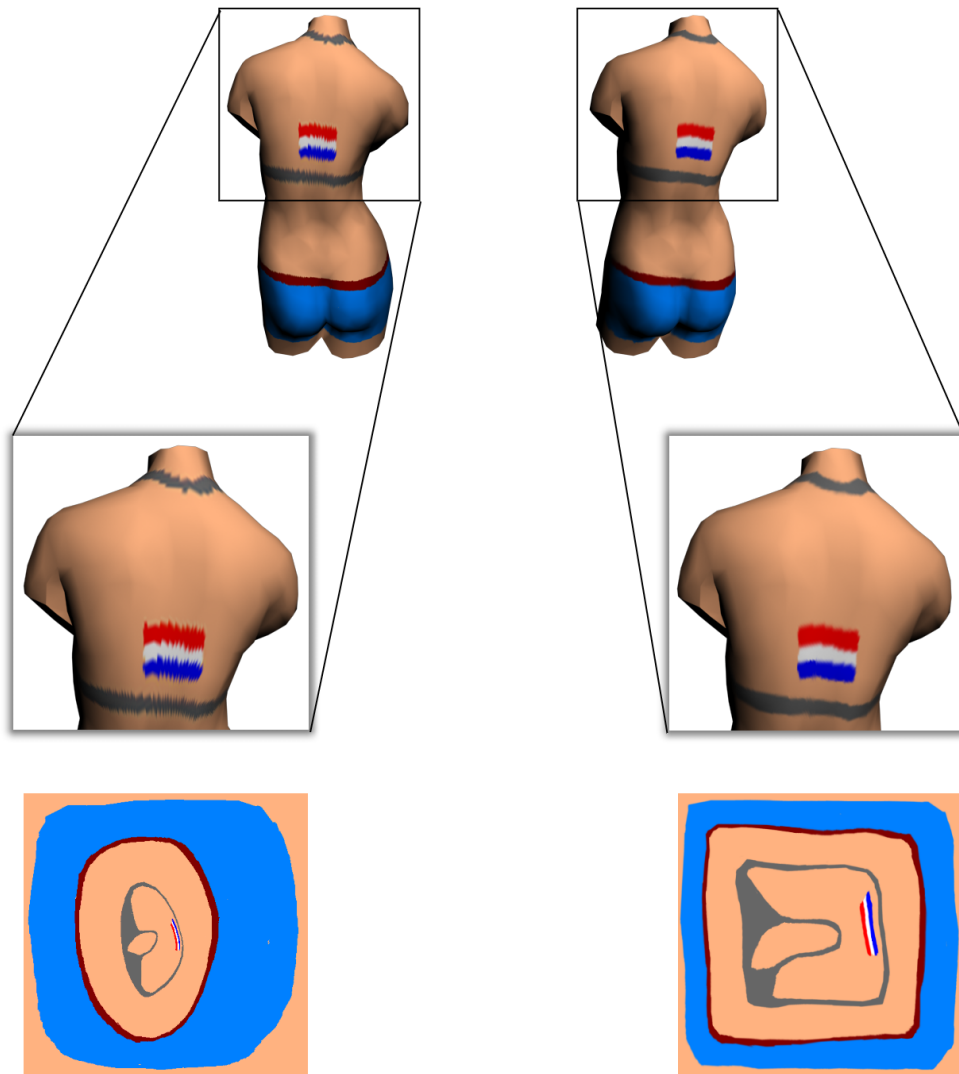


Figure 4.2: Back-view of the painting results of the venus model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).

optimization is independent on the face number of input model. The timing required by the two-stage optimization is reasonable for the interactive application of surface painting systems. Figure 4.8 shows the optimization time after each stroke is applied on the triceratops model. Since the geometry stretch is minimized in the first optimization

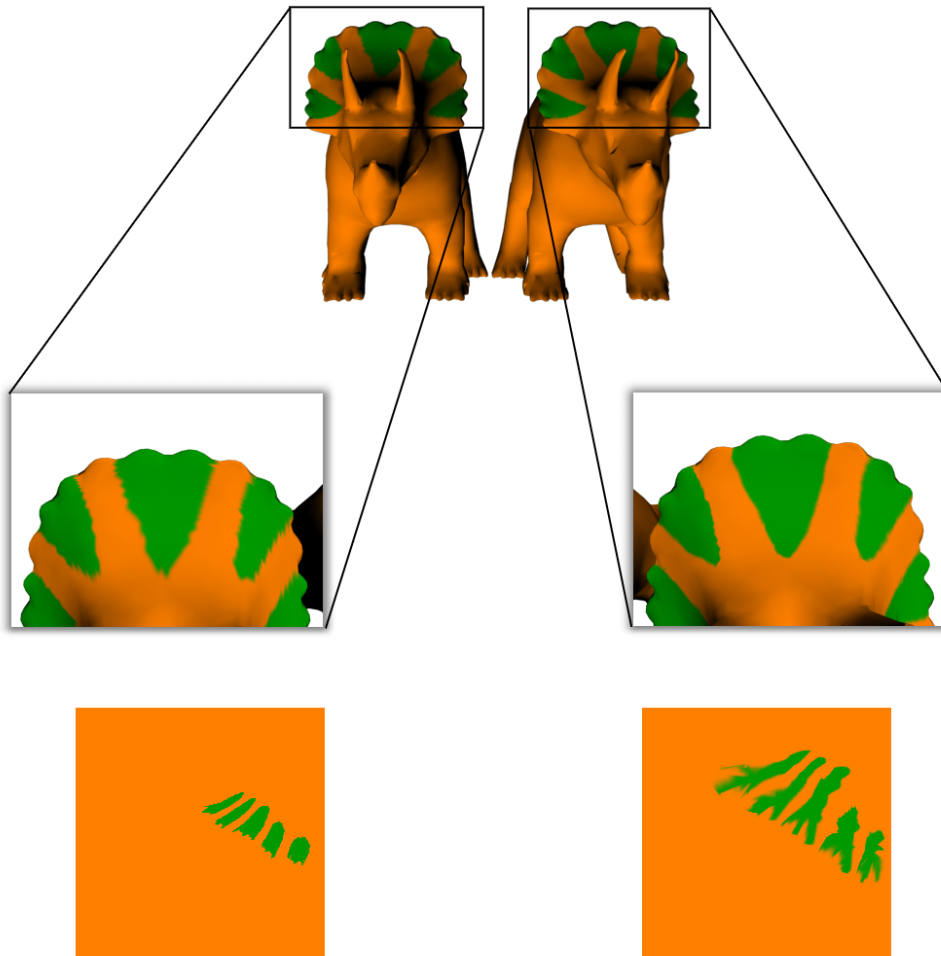


Figure 4.3: Painting results of the triceratops model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).

process, the timing is higher than succeeding optimizations.

4.3 Signal-specialized parameterization versus our method

The signal-specialized parameterization proposed by Sander et al.[18] is thought to be the state-of-art work in mesh parameterization which is sensitive to surface signal. We

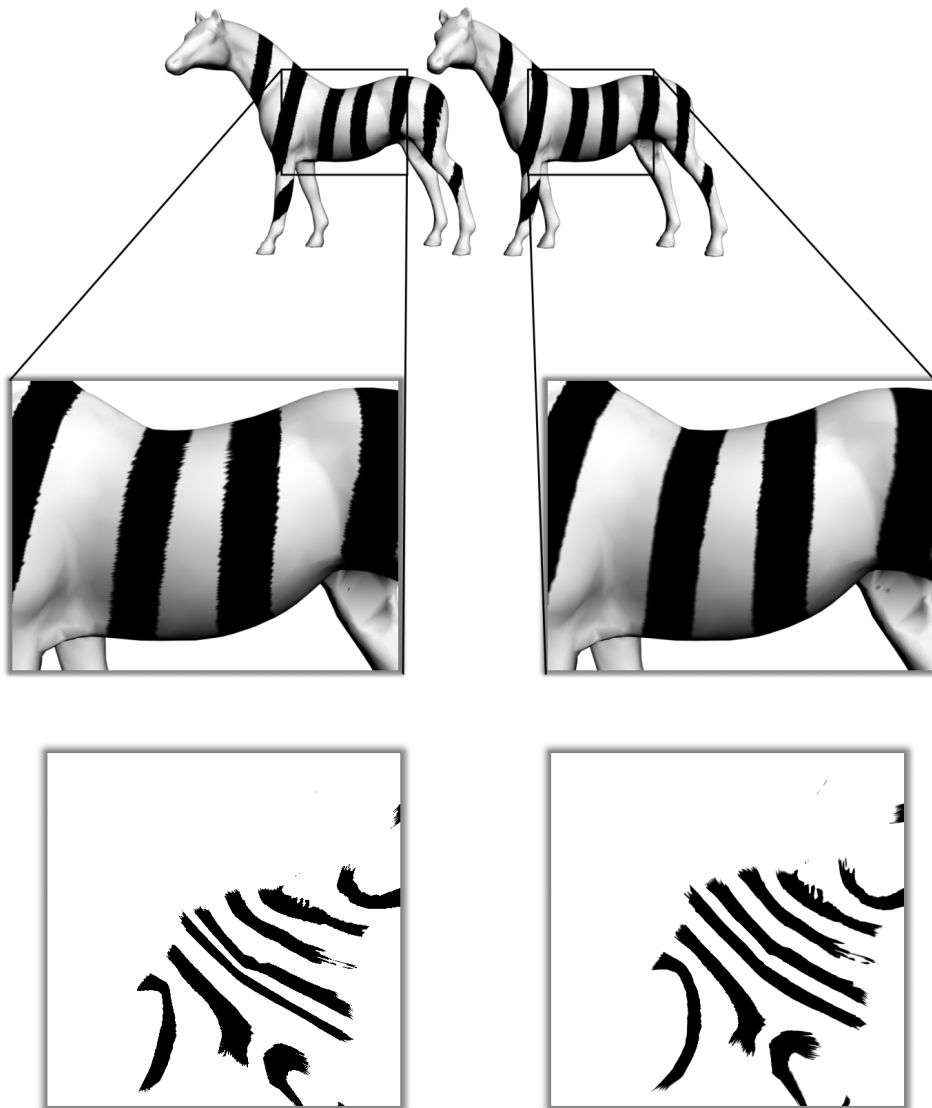


Figure 4.4: Painting results of the face model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).

compare the parameterization performance between our two-stage optimization framework and signal-specialized parameterization. The comparison is done as follows: Firstly, the parasaur model with its signal-specialized parameterization ϕ_{sig} and a high resolution texture(2048x2048) based on ϕ_{sig} are given. Then we load the parasaur model

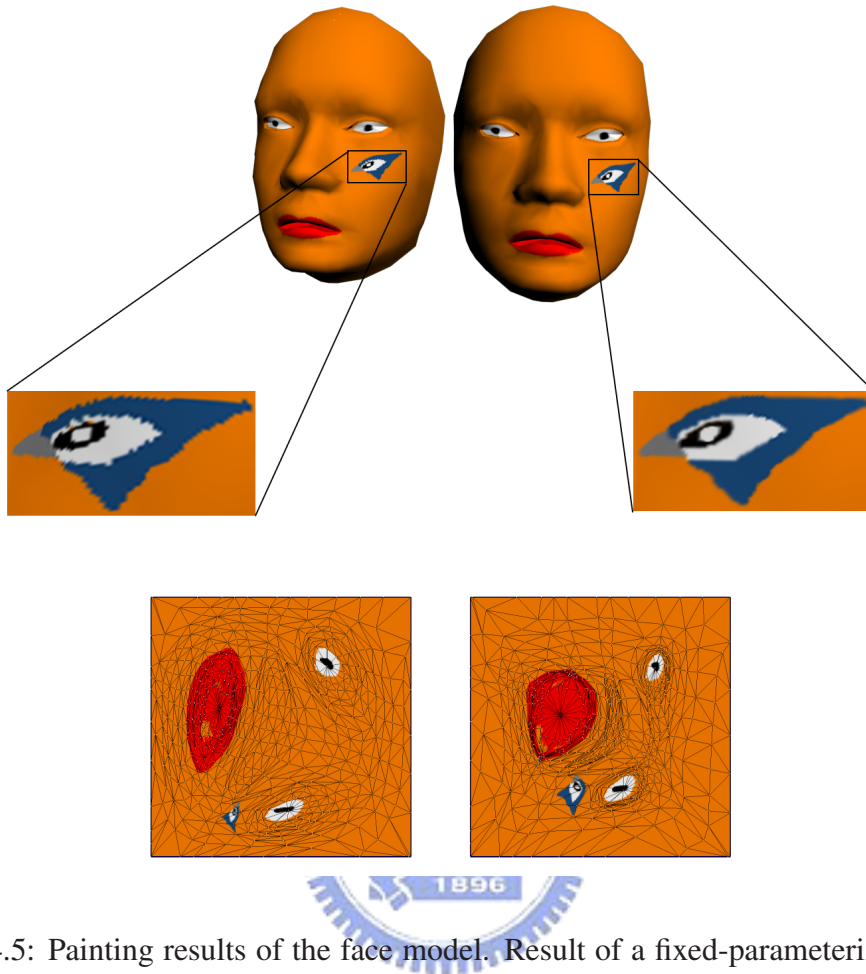


Figure 4.5: Painting results of the face model. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).

and form its initial parameterization ϕ_{init} as described in section 3.3. Next, for each two-stage optimization process, we resample the color from the high resolution texture into our lower resolution base texture. After resampling, texture frequency analysis is processed then the two-stage optimization process comes next. After the two-stage optimization, we obtain the optimized parameterization. Finally the resampling procedure is executed again to output the resulted based texture.

Figure 4.9(a)(b) show the result of signal-specialized parameterization using 2048×2048 and 128×128 texture maps, respectively. Figure 4.9(c)(d) shows the result of our

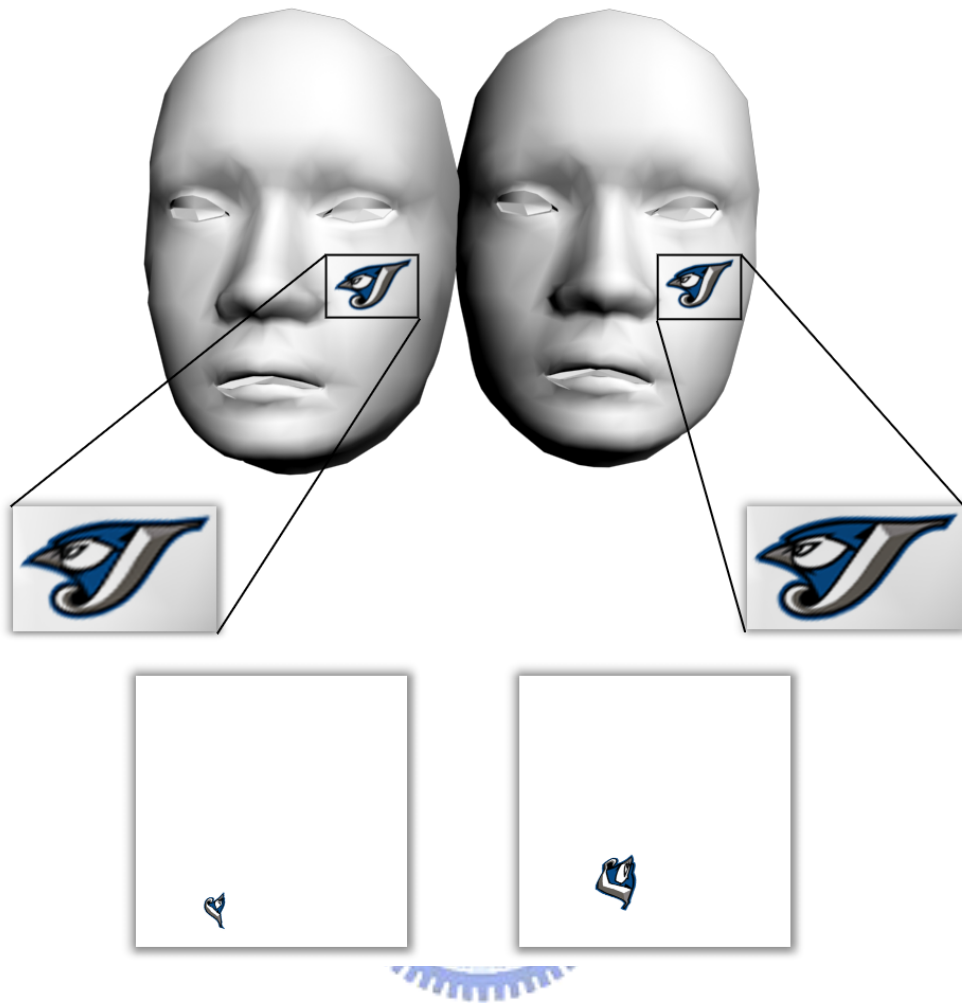


Figure 4.6: An image is texture mapped to simulate painting strokes. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).

two-stage optimization under two different texture map resolutions. Our result is pretty good under resolution of 256x256 and still fine under resolution of 128x128.

Model	face	Init-param.	Two-stage Optimization		Re-param.
			range	avg.	
venus	1396	0.625	0.718 - 3.843	1.784	0.016
triceratops	5660	3.234	0.625 - 3.156	1.739	0.063
face	1162	0.5	1.531 - 3.828	1.690	0.016
horse	7500	4.906	1.031 - 3.125	1.375	0.078

Table 4.1: Statistics of initial parameterization, two-stage optimization and re-parameterization time (sec.) for four different models.

4.4 Comparison to painting detail

Finally, we compare our method to *Painting Detail* proposed by Carr et al [2]. The re-parameterization for *Painting Detail* is based on the re-balancing of the MMA tree hierarchy. To get significant re-balancing effect, a quite deep tree hierarchy is expected. Therefore, *Painting Detail* works better on models with large polygon count. On the contrary, our optimization procedure is performed on parametric domain, hence works for models of high and low polygon count.

The common drawback of the methods based on texture atlases, such as *Painting Detail*, is the problem of mip-mapping. In *Painting Detail*, a quaternary MMA tree is used to alleviate the problem. For our methods, the based texture is not constructed by atlases. There is no mip-mapping problem in our scheme.

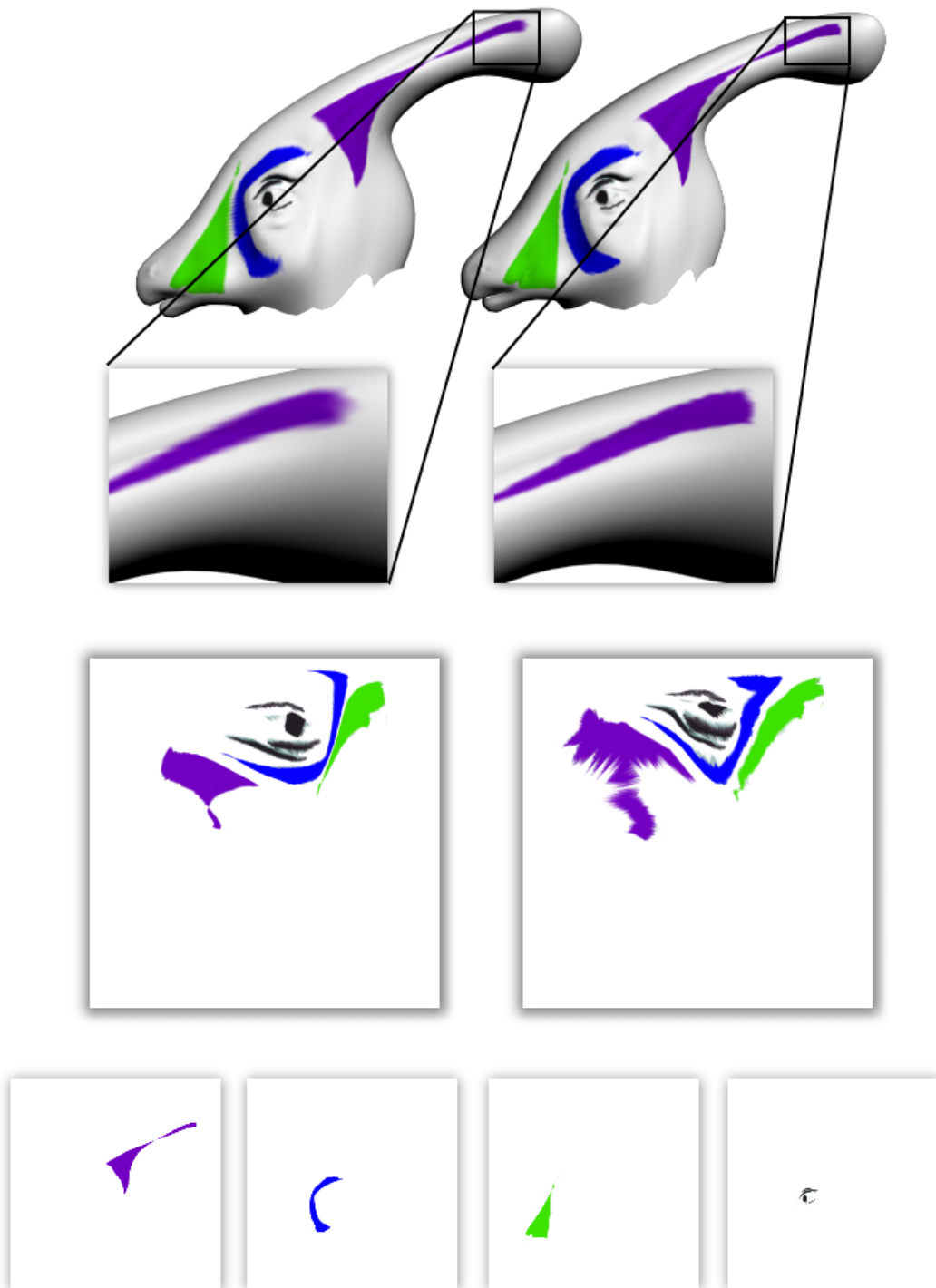


Figure 4.7: Four images is texture mapped to simulate painting strokes. Result of a fixed-parameterization (left column) and the result of our two-stage optimized parameterization (right column).

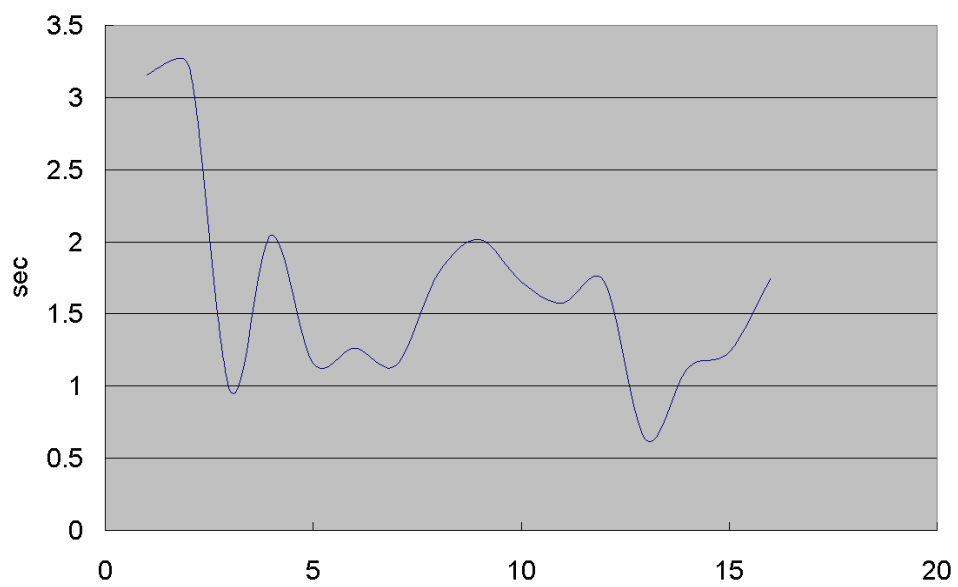


Figure 4.8: The optimization process of triceratops model.

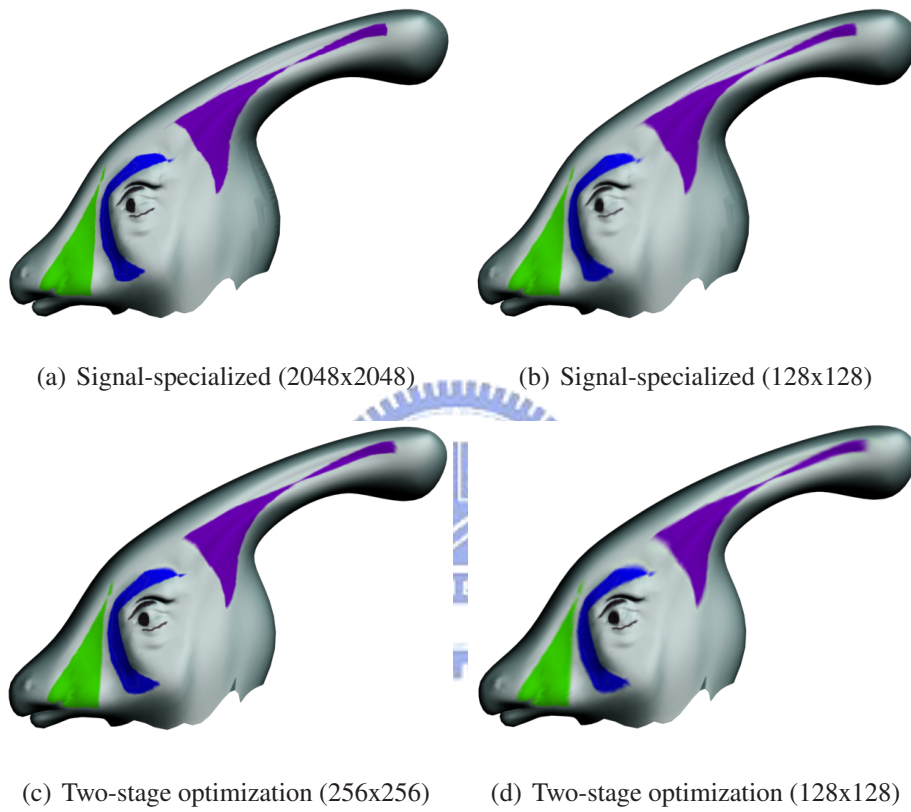
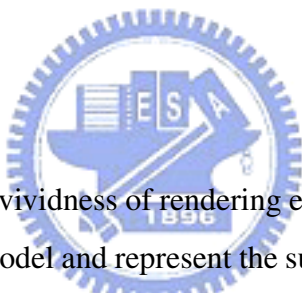


Figure 4.9: Comparison of our result with signal specialized parameterization under different texture map resolutions.

Conclusion and Future Work

5.1 Conclusion



Texture mapping increases the vividness of rendering effects in computer graphics. It is a simple and efficient way to model and represent the surface's details. Surface painting is a technique that allows a user to paint directly onto three dimensional surface. In general, the result is stored in parametric space as texture. Therefore a fine parameterization is required to construct a good texture when the regions with high signal variation will obtain more samples in parametric space. Furthermore, the signal varies during surface painting process, hence a re-parameterization framework that can interactively respond to the signal variation is strongly desirable.

We have proposed a rapidly re-parameterization framework for surface painting which redistributes texture sample space according to the surface signal variation. We proposed a two stage uniform grid optimization framework which diminished sample space in lower gradient regions in stage one and magnifies sample space for high gradi-

ent regions in stage two. In addition, this two stage optimization framework is suitable for interactive use required by surface painting. For the optimization process, we derived the modified L^2 metric denoted as L_s^2 . The L_s^2 metric takes signal stretch into account in the regions of signal variation and combines geometry stretch in the regions without signal variation.

5.2 Future work

Some potential future work are listed as follows:

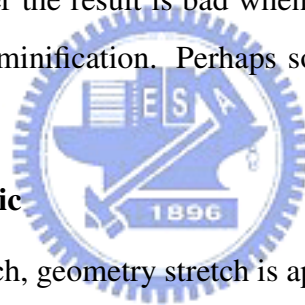
- **Better stroke sampling method**

The stroke sampling method[2] based on graphics hardware is simple and fast for interactive use. However the result is bad when the resampling was done under either magnification or minification. Perhaps some filter on image space could alleviate this problem.

- **Parameterization metric**

In the proposed L_s^2 stretch, geometry stretch is applied in the regions without signal gradient to prevent the excessively undersampling in the un-painted regions. The major issue is that the same value of geometry and signal stretch does not imply the equal significance. Therefore, a study on the weighted relationship between geometry and signal stretch will enhance the theoretical background of our method. Though our L_s^2 metric works well for a surface painting system, but it is a little heuristic in some measure. We look for a better metric, especially the one which is more sensitive for the anisotropical distribution of surface signal on parametric domain.

- **Hierarchical optimization**



Optimization based on adaptive sample points can be utilized to improve the performance. In our two-stage optimization framework, the sample points are uniformly distributed on parametric domain at each step. To use the sample points more efficiently, we distribute more sample points on the regions of high signal gradient to accurately grab the signal variation. Less sample points are distributed on the regions of lower signal gradient, thus these regions will be converged more quickly. To achieve the goal, a hierarchy architecture of uniform grid is required to maintain the different resolution of grid points. For sampling, there are two major problems of the hierarchical method. The first one is the determination of high gradient region and lower gradient region. A two-pass method will be practical to accomplish this. The second problem is that a theoretical and efficient method to propagate the L_s^2 stretch from high resolution grid points to lower resolution grid points is required. In addition to the problems of sampling, an efficient optimization algorithm for the hierarchical grid architecture is also required.

- **Dynamic cutting**

Topological surgery is used to transform the closed surface into an open-one. Current method [9] only takes the geometric information into account. A signal sensitive topological surgery will be a novel and great contribution for current surface painting system. The main issue is the time complexity of the cutting algorithm.

Bibliography

- [1] N. A. Carr and J. C. Hart. Meshed atlases for real-time procedural solid texturing. *ACM Transactions on Graphics*, 21(2):106–131, 2002.
- [2] N. A. Carr and J. C. Hart. Painting detail. *ACM Transactions on Graphics. Special issue for SIGGRAPH conference*, 23, 3:845–852, 2004.
- [3] W.-Z. Dai. User-assisted parameterization of polygonal meshes. Master’s thesis, National Chiao Tung University, 2004.
- [4] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. In *Eurographics conference proceedings*, pages 209–218, 2002.
- [5] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Conference Proceedings*, pages 173–182, Aug. 1995.
- [6] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.
- [7] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20:19–27, 2003.

- [8] X. Gu. *Parameterization for Surfaces with Arbitrary Topologies*. PhD thesis, Department of Computer Science, Harvard University, 2002.
- [9] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. In *SIGGRAPH 2002 Conference Proceedings*, pages 335–361, 2002.
- [10] P. Hanrahan and P. Haeberli. Direct wysiwyg painting and texturing on 3d shapes. In *International Conference on Computer Graphics and Interactive Techniques*, pages 215–223, 1990.
- [11] K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. Vanderbilt University Press, 2000.
- [12] K. Hormann, G. Greiner, and S. Campagna. Hierarchical parametrization of triangulated surfaces. In *Proceedings of Vision, Modeling, and Visualization 1999*, pages 219–226, Nov. 1999.
- [13] T. Igarashi and D. Cosgrove. Adaptive unwrapping for interactive texture painting. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 209–216, 2001.
- [14] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, pages 1–13, 1998.
- [15] A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics. Special issue for SIGGRAPH conference*, 22, 3:350–357, 2003.
- [16] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *Proceedings of the 29th Conference on*

- Computer Graphics and Interactive Techniques (SIGGRAPH-02)*, volume 21, 3 of *ACM Transactions on Graphics*, pages 362–371, July 21–25 2002.
- [17] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *SIGGRAPH 2001 Conference Proceedings*, pages 409–416, 2001.
- [18] P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering (RENDERING TECHNIQUES-02)*, pages 87–98, June 26–28 2002.
- [19] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 146–155, 2003.
- [20] A. Sheffer. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Shape Modelling International*, pages 61–66, 2002.
- [21] A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17, 3:326–337, 2001.
- [22] A. Sheffer and E. de Sturler. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Transactions on Graphics*, 21(4):874–890, Oct. 2002.
- [23] A. Sheffer and J. C. Hart. Seamster: Inconspicuous low-distortion texture seam layout. In *IEEE Visualization (Vis02)*, pages 291–298, 2002.
- [24] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization '02*, pages 355–362, June7–11 2002.
- [25] W. T. Tutte. How to draw a graph. *Proc. London Mathematical Society*, 13:743–768, 1963.

- [26] S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel. A fast and simple stretch-minimizing mesh parameterization. In *International Conference on Shape Modeling and Applications (SMI 2004)*, pages 200–208, June 7–11 2004.

