

# 國立交通大學

應用數學系

數學建模與科學計算碩士班

碩士論文

一個對於變動曲面的網格重構之研究

A remeshing study for evolving surfaces

研究生：關湘源

指導教授：賴明治 教授

中華民國一百零二年九月

一個對於變動曲面的網格重構之研究

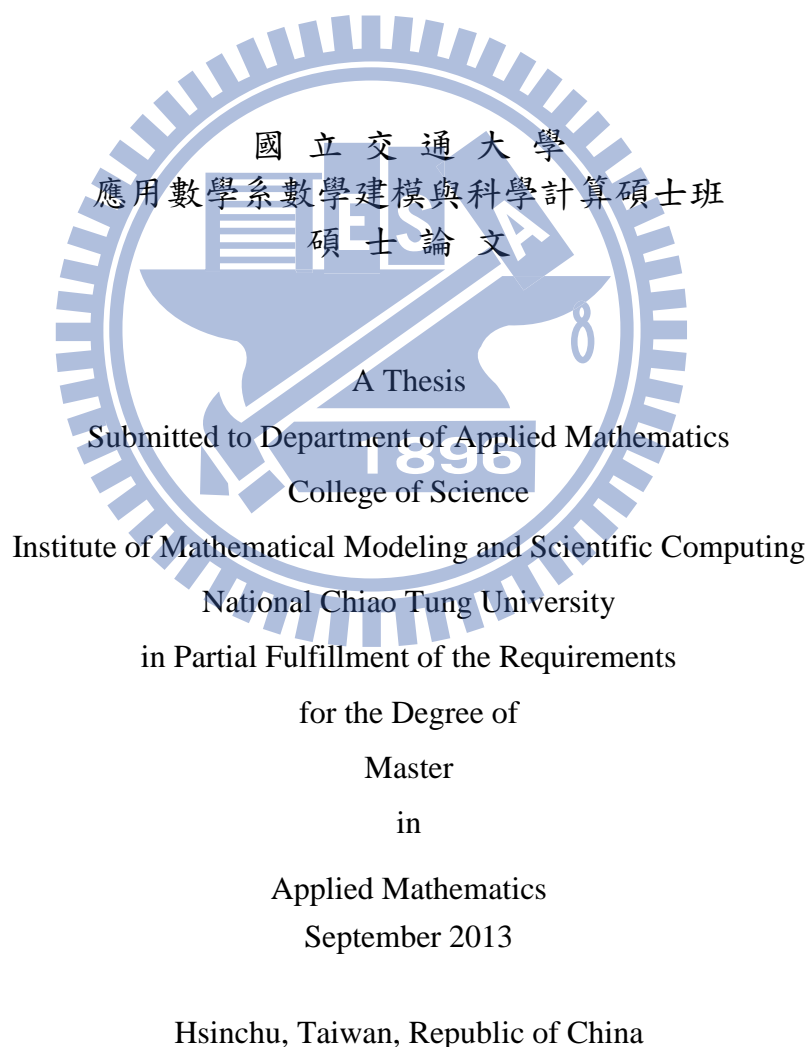
A remeshing study for evolving surfaces

研究生：關湘源

Student : Hsiang-Yuan Kuan

指導教授：賴明治 教授

Advisor : Ming-Chih Lai



中華民國一百零二年九月

# 一個對於變動曲面的網格重構的研究

學生：關湘源

指導教授：賴明治 教授

國立交通大學應用數學系數學建模與科學計算碩士班



我們提出了一個以提高網格質量的理念為基礎的重新網格化方法，利用一連串區域的運算來改進網格的幾何性質。其核心技術為 Area-based smoothing，此方法能建造一個非均勻網格，並且能有效地降低鈍角三角形的數目。

# A remehsing study for evolving surfaces

Student : Hsiang-Yuan Kuan

Advisor : Ming-Chih Lai

Institute of Mathematical Modeling and Scientific Computing  
Department of Applied Mathematics  
National Chiao Tung University



## ABSTRACT

We present a remeshing scheme based on the idea of improving mesh quality by a series of local modifications of the mesh geometry and connectivity. The central local modification technique is an area-based smoothing technique. Area-based smoothing allows the control of both triangle quality and vertex sampling over the mesh, as a function of some criteria, e.g. the mesh curvature. The algorithm is able to create an unstructured mesh, and reduce irregular vertices efficiently.

## 誌謝

首先誠摯的感謝指導教授賴明治教授，老師悉心的教導使我得以了解應用數學與計算數學的應用，不時的討論並指點我正確的方向，使我在這些年中獲益匪淺。

兩年多的日子裡，研究室裡共同的生活點滴，學術上的討論、趕作業的革命情感、一起出遊的回憶…，感謝眾位學長姐、同學、學弟妹的共同砥礪，你們的陪伴讓兩年的研究生活變得絢麗多彩。

特別感謝陳冠羽、胡偉帆學長們不厭其煩的指出我研究中的缺失，且總能在我迷惘時為我解惑，也感謝張毓倫同學與我共同討論並給予意見。

女朋友在背後的默默支持更是我前進的動力，沒有的體諒、包容，相信這兩年的生活將是很不一樣的光景。

最後，謹以此文獻給我摯愛的雙親，沒有你們就沒有今天的我。

關湘源 謹致於

交通大學數學建模與科學計算所

中華民國 102 年 9 月

# 目 錄

中文提要 .....	i
英文提要 .....	ii
誌謝 .....	iii
目錄 .....	iv
一、 Introduction .....	1
二、 Contribution and Overview .....	2
三、 Surface Reconstruction .....	2
四、 Remeshing .....	6
五、 Connectivity Regularization .....	11
六、 Numerical Results .....	13
七、 Conclusion .....	20
Reference .....	21

# 1 Introduction

Triangular meshes are used to approximate surfaces or flow fields for rendering and simulation. Triangles with large angles are also found to hamper the efficiency of some iterative algebraic solvers. This implies that obtuse triangles should be avoided as much as possible. The control of the maximum angle is difficult, however. Especially for the triangular mesh under flow, it always produce meshes that contain a large number of obtuse triangles. There are a number of methods for generating triangular meshes [3, 10, 11]. However, none of these methods consider avoiding obtuse triangles.

In this paper, we present an effective method for reducing obtuse triangles efficiently.

## 1.1 Related Works

Recent remeshing algorithms, e.g. [2, 6, 9] are based on global parameterization of the original mesh, and then a resampling of the parameter domain. Following this, the new triangulation is projected back into 3D space, resulting in an improved version of the original model. The main drawback of the global parameterization methods is the sensitivity of the result to the specific parameterization used, and to the cut used to force models that are not isomorphic to a disk to be so. Embedding a non-trivial 3D structure in the parameter plane severely distorts this structure, and important information, which is not specified explicitly, may be lost on the way. Even if the parameterization minimizes the metric distortion of the 3D original in some reasonable sense, it is impossible to eliminate it completely. Moreover, methods finding a global parameterization are slow, usually involving the solution of a large set of (sometimes nonlinear) equations. Recent progress may accelerate the process to almost linear time even for large meshes, using multi-resolutional approaches, e.g. [15], inspired by multi-grid methods together with good preconditioning. Unfortunately, when dealing with extremely large meshes, or meshes with severe isoperimetric distortion (like sock-shaped regions) numerical precision issues may arise. In such cases, a global parameterization is almost impossible to perform without using multi-scale or precise arithmetic representation of the parametric domain.

The main alternative to global parameterization is to work directly on the surface and perform a series of local modifications on the mesh. This approach is also known as the mesh adaptation process and is the one we use in this work. Remeshing algorithms using this approach [1, 5, 7, 8, 14, 19] usually involve computationally expensive optimizations in 3D or more efficient but less accurate optimization in the tangent plane. In Section 3.2, we use PN triangles as a good tradeoff between accuracy and efficiency.

## 2 Contribution and Overview

In this paper we present a remeshing method which we call *explicit*. The word *explicit* we mean that we operate directly on the mesh surface and apply local modifications to it, instead of working on some indirect representation of the surface, e.g. on a global parametric domain. The components of our remeshing algorithm are natural and straightforward ways to improve a mesh. Our scheme is based on the work that Vitaly Surazhsky and Craig Gostman presented [17]. We use local parameterization to reduce the problem of local mesh optimizations to 2D.

Our remeshing algorithm incorporates a number of novel techniques, each of independent interest. The central technique is the area-based mesh optimization described in Section 4, which manipulates the areas of triangulation in order to achieve a uniform or otherwise specified vertex sampling. Section 5 presents a novel regularization technique that performs local modifications to the mesh connectivity, resulting in a mesh whose connectivity is much regular. When combined, our techniques provide an accurate and robust remeshing algorithm that can be applied to meshes of arbitrary genus. Our remeshing scheme is very efficient and is easy to coding.

## 3 Surface Reconstruction

We perform an estimate of the smooth surface in the vicinity of a mesh triangle. This may be obtained by reconstructing an approximation of the surface using triangular cubic Bézier patches for every face of the mesh. Vlachos et al. [20] presented a simple and efficient yet robust and accurate method to construct such curved patches called *PN triangles*. The triangle vertex normals together with vertex coordinates are used to construct a PN triangle. If the normals at the mesh vertices are not given, we use a method similar to [13] to define them, and we will introduce the algorithm later. The normal of any point within a PN triangle is defined as an efficient quadratic interpolation of the normals at the triangle vertices. We use PN triangles as a good tradeoff between accuracy and efficiency.

### 3.1 Normals

The definition of cubic Bézier triangle requires normals to be provided or approximated at each vertices. Defining the normal,  $N_P$  at a point as the weighted average of the adjacent faces normal,  $N_j$ , where the weight,  $w_j$  is the normalized angle at the point. Equations (1), (2) and Figure 1 describe this procedure.

$$N_P = \sum_{j=1}^n N_j w_j \quad (1)$$



$$w_j = \frac{\alpha_j}{\sum_{j=1}^n \alpha_j} \quad (2)$$

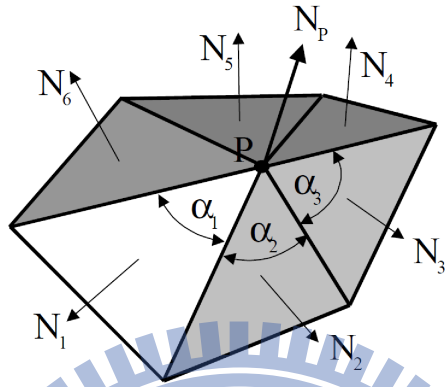
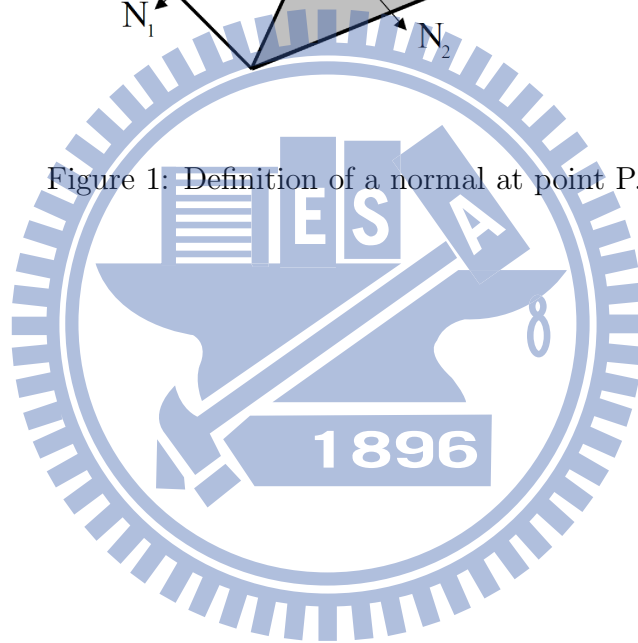
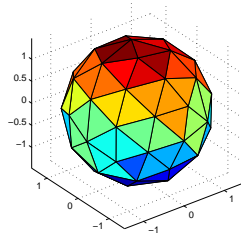


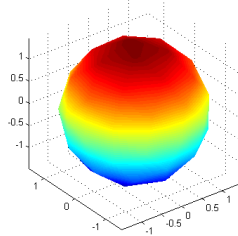
Figure 1: Definition of a normal at point P.



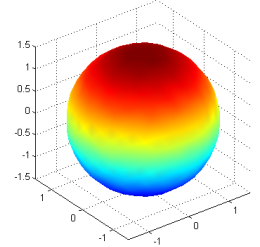
### 3.2 PN triangle



(a) Input triangulation



(b) Gouraud shaded input triangulation



(c) Using PN triangles

Figure 2: Which rendering would you prefer?

A cubic Bézier triangle is a surface with the equation

$$b: R^2 \rightarrow R^3, \text{ for } w = 1 - u - v, \quad u, v, w \geq 0$$

$$\begin{aligned} b(u, v) &= \sum_{i+j+k=3} b_{ijk} \frac{3!}{i!j!k!} u^i v^j w^k \\ &= b_{300}w^3 + b_{030}u^3 + b_{003}v^3 \\ &\quad + b_{210}3w^2u + b_{120}3wu^2 + b_{201}3w^2v \\ &\quad + b_{021}3u^2v + b_{102}3wv^2 + b_{012}3uv^2 \\ &\quad + b_{111}6wuv \end{aligned}$$

We group the  $b_{ijk}$  together as

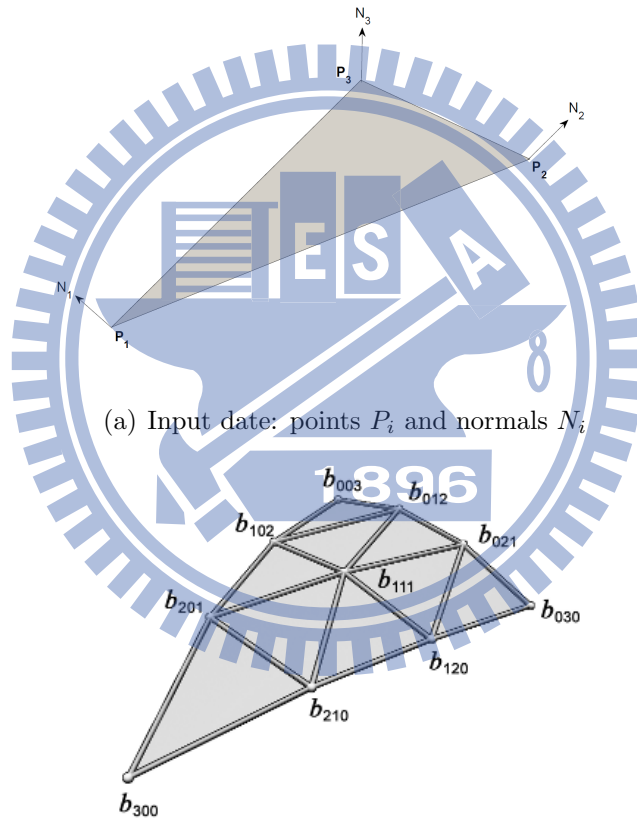
$$\begin{aligned} \text{vertex coefficients} &: b_{300}, b_{030}, b_{003} \\ \text{tangent coefficients} &: b_{210}, b_{120}, b_{021}, b_{012}, b_{102}, b_{201} \\ \text{center coefficient} &: b_{111} \end{aligned}$$

Coefficients are also often called control points and are connected to form a control net(see Figure 3(b)).

We are given the positions  $P_1, P_2, P_3 \in R^3$  and normal  $N_1, N_2, N_3 \in R^3$  of the triangle corners as shown in Figure 3(a). In formulas for implementation, coefficients of the PN triangle are defined as follows:

$$\begin{aligned} b_{300} &= P_1, \\ b_{030} &= P_2, \\ b_{003} &= P_3, \\ w_{ij} &= (P_j - P_i) \cdot N_i \end{aligned}$$

$$\begin{aligned}
b_{210} &= (2P_1 + P_2 - w_{12}N_1)/3, \\
b_{120} &= (2P_2 + P_1 - w_{21}N_2)/3, \\
b_{021} &= (2P_2 + P_3 - w_{23}N_2)/3, \\
b_{012} &= (2P_3 + P_2 - w_{32}N_3)/3, \\
b_{102} &= (2P_3 + P_1 - w_{31}N_3)/3, \\
b_{201} &= (2P_1 + P_3 - w_{13}N_1)/3, \\
E &= (b_{210} + b_{120} + b_{021} + b_{012} + b_{102} + b_{201})/6, \\
V &= (P_1 + P_2 + P_3)/3, \\
b_{111} &= E + (E - V)/2.
\end{aligned}$$



(a) Input data: points  $P_i$  and normals  $N_i$

(b) The coefficients or control points of a triangular Bézier patch arranged to form a control net

Figure 3

## 4 Remeshing

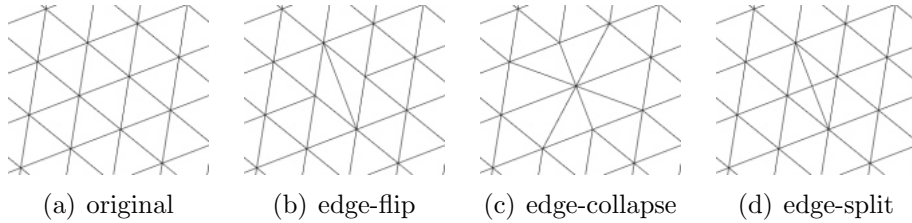


Figure 4: Types of edge operation

The focus of our remeshing scheme is on maximizing the angles of all triangles of the mesh. Remeshing of the given mesh  $M$  is performed by a series of local modifications. The most well-known and commonly used local modifications are *edge – flip*, *edge – collapse*, *edge – split* and *vertex relocation*.

The main stages of our remeshing scheme are as follows:

- Adjust the number of vertices of  $M$
- Apply the area-based remeshing procedure on  $M$
- Regularize  $M$  using the algorithm of Section 5
- Apply the angle-based smoothing procedure on  $M$

Edge-collapse and edge-split are used to change the number of mesh vertices. Edge-flip and vertex relocation improve the quality of the mesh triangles. The area-based remeshing procedure is the heart of our remeshing scheme and produces a mesh with high quality triangles and the required vertex sampling. Another two stages improve the mesh quality further. The regularization stage improves the regularity of the mesh connectivity. The angle-based smoothing then polishes the mesh to obtain the optimal mesh geometry without changing its connectivity.

The area-based remeshing and the angle-based smoothing involve the most critical and difficult operation – vertex relocation.

### 4.1 Vertex Relocation

Let  $v$  be a vertex having location  $\mathbf{x}(v)$  and whose neighbors are  $v_1, \dots, v_k$  with locations  $\mathbf{x}(v_1), \dots, \mathbf{x}(v_k)$ , respectively, and  $k$  is the vertex degree. We want to find a new location  $\mathbf{x}_{new}(v)$  satisfying some condition, e.g. improving the angles of the triangles incident on  $v$ . A solution to this problem that directly finds  $\mathbf{x}_{new}(v)$  in 3D usually involves solving a difficult optimization problem, which may have non-linear constraints. We overcome this problem by mapping faces incident on  $v$  into the 2D

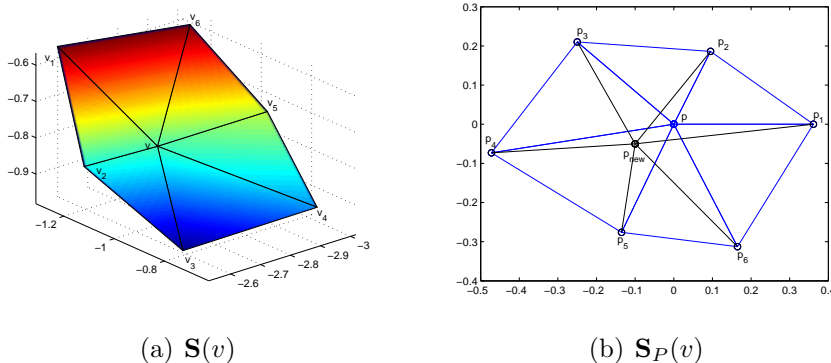


Figure 5: Vertex relocation

plane and solving the problem there. The location computed in the plane is then brought back to the original surface.

Let  $\mathbf{S}(v)$  be a sub-mesh of  $M$  containing only  $v, v_1, \dots, v_k$  and faces incident on  $v$ . (See Figure 5(a)) We map  $\mathbf{S}(v)$  into the plane using a natural and simple method approximating the geodesic polar map [16], as described, for example, by Welch and Witkin [21] and Floater [4]. Let  $p, p_1, \dots, p_k$  be the positions of vertices  $v, v_1, \dots, v_k$  within the resulting mapping  $\mathbf{S}_P(v)$ .  $p$  is mapped to the origin.  $p_1, \dots, p_k$  satisfy the following conditions: the distances between  $p$  and its neighbors are the same as the corresponding distances in  $M$ , namely,  $\|p - p_i\| = \|\mathbf{x}(v) - \mathbf{x}(v_i)\|$  for  $1 \leq i \leq k$ . The angles of all triangles at  $p$  are proportional to the corresponding angles in  $M$  and sum to  $2\pi$ .

The next step is to find a new location  $p_{new}$  to satisfies some condition, we will introduce it later. (See Figure 5(b))

## 4.2 Back to the Original Surface

After the new vertex location  $p_{new}$  has been found, we need to find its corresponding location on the original surface, namely, to find  $\mathbf{x}_{new}(v)$ . Existing remeshing methods, e.g. [1, 8, 14] solve this problem by finding the vertex projection onto the original surface. Projecting the vertex involves a computationally expensive and not always accurate computation that without special care may even lead to topological errors during the remeshing process.

In our case,  $\mathbf{x}_{new}(v)$  is computed by a barycentric coordinate scheme. Given a point  $p$  and that the triangle  $(q_1, q_2, q_3)$  contains  $p$ , we can compute its barycentric coordinate  $b' = (b^1, b^2, b^3)$  using (3). Then we apply the method PN triangle in Section 3.2 to find it.

$$b^1 = \frac{A(p, q_2, q_3)}{A(q_1, q_2, q_3)}, \quad b^2 = \frac{A(p, q_3, q_1)}{A(q_1, q_2, q_3)}, \quad b^3 = \frac{A(p, q_1, q_2)}{A(q_1, q_2, q_3)} \quad (3)$$

where  $A(q_i, q_j, q_k)$  is the signed area of triangle  $(i, j, k)$ .

### 4.3 Area-based Remeshing

The concept of triangle areas has never been used as a central factor in mesh generation. Triangle areas are usually used to assist, analyze or control meshing. The reason for this is that by using triangle areas alone we cannot obtain meshes of reasonable quality. A mesh optimization that equalizes the areas of the mesh triangles or brings triangle areas to specified (absolute or relative) values will, in most cases, result in many long and skinny triangles. Nevertheless, Surazhsky and Gotsman [18] discovered that a 2D triangulation having triangles with equal (or close to equal) areas has globally uniform spatial vertex sampling. They presented the following remeshing scheme that exploits this: Alternate between area equalization and a series of angle-improving (Delaunay) edge-flips. Applying this simple scheme results in a 2D mesh with a very uniform sampling and well-shaped triangles.

It is important to mention that this alternation process does not usually converge. After a uniform sampling rate is obtained, the process begins to oscillate, producing different but similar uniform vertex distributions. However, this oscillation is not a problem for our remeshing algorithm, since subsequent steps of our remeshing algorithm improve quality of the mesh further by regularizing and smoothing it.

### 4.4 Area-based Vertex Relocation

Area equalization is done iteratively by relocating every vertex such that the areas of the triangles incident on the vertex are as equal as possible. In this work we extend this method to relocating vertices such that the ratios between the areas are as close as possible to some specified values. To define this formally, we return to the definitions of point  $p$  and its neighbors  $p_1, \dots, p_k$  from Section 4.1. Let  $(x_i, y_i)$  be the coordinates of  $p_i$ . Our goal is to find  $p = (x, y)$  such that the ratios of the triangle areas are as close as possible to  $\mu_1, \dots, \mu_k$ .  $\mu_i$  is all positive and sum to unity. Denote  $A_i(x, y)$  be the area of triangle  $p, p_i, p_{i+1}$ :

$$A_i(x, y) = \frac{1}{2} \begin{vmatrix} x_i & y_i & 1 \\ x_{i+1} & y_{i+1} & 1 \\ x & y & 1 \end{vmatrix}$$

Let  $A$  be the area of polygon  $(p_1, \dots, p_k)$ , which may be computed as  $\sum_{i=1}^k A_i(0, 0)$ . Now the location of  $p$  is defined as follows:

$$(x, y) = \arg \min_{(x, y)} \sum_{i=1}^k (A_i(x, y) - \mu_i A)^2$$

This reduces to solving a system of two linear equations in  $x$  and  $y$ , which has a unique solution. Thus, area-based relocation is almost as efficient as Laplacian smoothing.

## 4.5 Curvature Sensitive Remeshing

We now show how to use area-based vertex relocation to produce a mesh reflecting the curvature of the original mesh. Intuitively, more curved regions of  $M$  will contain small triangles and a dense vertex sampling, while almost flat regions will have large triangles with more sparse vertices. The idea is to specify ratios between triangle areas depending on curvature.

Let  $\Psi$  be a density function defined over  $M$ . For every vertex  $v$  of  $M$ , we define  $\Psi(v)$  as  $1/(\alpha|K(v)| + \beta H^2(v))$ , where  $H(v)$  and  $K(v)$  are approximated discrete Gaussian and mean curvatures, respectively.  $\alpha$  and  $\beta$  are user-defined values, which are positive and sum to unity. Usually we use  $\alpha = \beta = 0.5$ . We compute  $H(v)$  and  $K(v)$  using the method described in [12].

The next step is to define the triangle area ratios that we use in vertex relocation. We return to the notations of Section 4.4. Define  $\mu'_i$  for  $1 \leq i \leq k$  as the average between  $\Psi(v_i)$  and  $\Psi(v_{i+1})$ . The value  $\mu'_i$  describes the required density of the corresponding triangle  $(v, v_i, v_{i+1})$ . Then  $\mu'_1, \dots, \mu'_k$  are normalized to obtain valid  $\mu_1, \dots, \mu_k$ , namely,  $\mu_i = \mu'_i / \sum_{j=1}^k \mu'_j$ .

## 4.6 Weight-angle-based smoothing

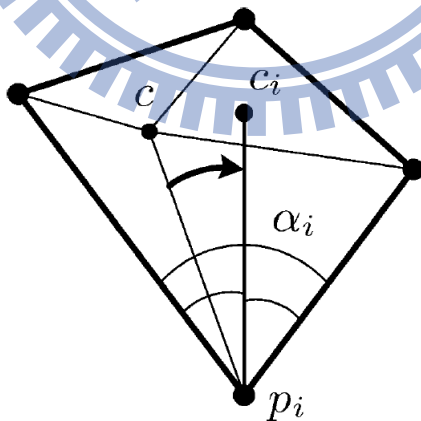


Figure 6: Weighted angle-based smoothing

Zhou and Shimada [22] presented an effective and easy to implement angle-based mesh smoothing scheme. They show that the quality of the mesh after angle-based smoothing is much better than after Laplacian smoothing. Moreover, the chance that the scheme will produce inverted (invalid) faces is much less than that in Laplacian smoothing. Unfortunately, this is true mostly for meshes whose vertices have degrees close to the average degree, namely, the mesh connectivity is close to regular. When the mesh has more irregular connectivity, the scheme may fail. In applications involving meshes with very distorted (long and skinny) triangles, a more robust smoothing scheme is critical. We propose a very simple improvement to the original angle-smoothing scheme, which significantly reduces the chances of inverted triangles and improves the quality of the resulting mesh. Furthermore, it has almost the same computational cost per iteration and a lower total computational cost due to better convergence in practice.

The original scheme attempts to make each pair of adjacent angles equal. Given a vertex  $c$  and its neighbours  $p_1, \dots, p_k$ , we want to move  $c$  in order to improve the angles of the triangles incident on  $c$ . Let  $\alpha_i$  be the angle adjacent to  $p_i$  in the polygon  $p_1, \dots, p_k$ . We define  $c_i$  to be the point lying on the bisector of  $\alpha_i$  such that  $\|p_i - c_i\| = \|p_i - c\|$ , namely, the edge  $(p_i, c)$  is rotated around  $p_i$  to coincide with the bisector of  $\alpha_i$  (see Figure 4). The new position of  $c$  is defined as the average of all  $c_i$  for all the neighbours, namely:

$$c_{new} = \frac{1}{k} \sum_{i=1}^k c_i \quad (4)$$

We improve this scheme by introducing weights into Eq.4. For a small angle  $\alpha_i$  it is difficult to guarantee that the resulting  $c_{new}$  will be placed relatively close to the bisector of  $\alpha_i$ . Since  $\alpha_i$  is itself small, a large deviation of  $c_{new}$  from the bisector of  $\alpha_i$  will create angles not only much smaller than  $\alpha_i/2$  but even negative (invalid) ones. Thus, the resulting mesh will have poor quality. To prevent this, we modify Eq.4 in the following way:

$$c_{new} = \frac{1}{\sum_{i=1}^k 1/\alpha_i^2} \cdot \sum_{i=1}^k \frac{1}{\alpha_i^2} \cdot c_i$$

Namely, the  $c_i$  for small angles  $\alpha_i$  will carry more weight than for large angles.

## 4.7 Implementation Notes

### Area-based remeshing

The procedure is controlled by two parameters:  $n_{step}$  and  $n_{area}$ . We alternate  $n_{step}$  times between curvature sensitive area equalization and a series of Delaunay edge-flips. Area equalization consists of  $n_{area}$  iterations of applying area-based



vertex relocation for every vertex of  $M$ . Edge-flips are performed until a Delaunay edge-flip can no longer be applied.  $n_{step}$  and  $n_{area}$  are usually small. There is no need to bring the triangle areas as close as possible to some required ratios to change vertex sampling. A very small number (1 to 3) of  $n_{area}$  iterations is enough to move vertices in the proper direction towards the required vertex sampling.  $n_{step}$  is usually between 5 and 10 and is sufficient to produce a mesh with vertex sampling very close to the required one.

### Angle-based smoothing

The parameter  $m_{step}$  of this procedure defines how many iterations of weighted angle-based smoothing are performed. Each iteration relocates once every vertex of  $M$ .  $m_{step}$  is also small and usually somewhere between 5 and 10.

### Adjusting the number of the mesh vertices

To obtain a mesh with the number of vertices specified by the user, we apply local refinement or simplification operations to the mesh. Until the required number of vertices is achieved we perform a series of edge-collapse or edge-split modifications (depending on the required size) such that the edges affected by the modifications are an independent edge set. Edges whose faces have minimal/maximal angle are simplified/refined first. Before every series of modifications we apply the area-based remeshing procedure with  $n_{step}=1$  to maintain a fair vertex sampling.

## 5 Connectivity Regularization

Another component of our remeshing scheme is an effective yet simple and efficient algorithm to improve the mesh quality by regularizing its connectivity. The algorithm performs a series of local operations that modify the mesh connectivity, namely, edge-flips, edge-collapses and edge-splits. Formally, improving regularity means minimizing the following function:

$$R(M) = \sum_{v \in M} (d(v) - d_{opt}(v))^2$$

where  $d(v)$  is the degree of vertex  $v$  and  $d_{opt}(v)$  its optimal degree, which is equal to 6. We do not define this formally, but during the mesh regularization we allow only a small change in the total number of mesh vertices and vertex sampling along the mesh.

We call an edge-flip *basic* if it decreases  $R(M)$ . In their elegant work, Alliez et al. [2] proposed to randomly apply basic edge-flips to regularize mesh connectivity. This straightforward method results in some improvement. However, it still leaves

too many irregular vertices even when basic edge-flips can no longer be applied. The reason for this is that  $R$  has many local minima with respect to basic edge-flips. The intriguing question is how to continue from such a local minimum.

We pose this problem as a puzzle. The player can click an edge to flip, collapse or split it. We call an edge *easy* if we can apply a basic edge-flip on it. The game starts with a mesh without easy edges, namely, when  $R(M)$  is at a local minimum. The goal of the game is to minimize  $R(M)$  further with a small number of mesh modifications. To visualize, we color vertices according to their degree. A vertex  $v$  is black if  $d(v) < d_{opt}(v)$  and white when  $d(v) > d_{opt}(v)$ . The vertex is not colored when  $d(v) = d_{opt}(v)$ , namely, the player has solved the puzzle for  $v$ . See Figure 7. When solving the puzzle we discovered that there are three types of edges that are actually interesting, and for every type there is only one specific local modification to apply.

### Long edges

An edge is a long edge if both its vertices are white. The definition of a long edge is based on the connectivity alone. However, optimizing the mesh geometry using

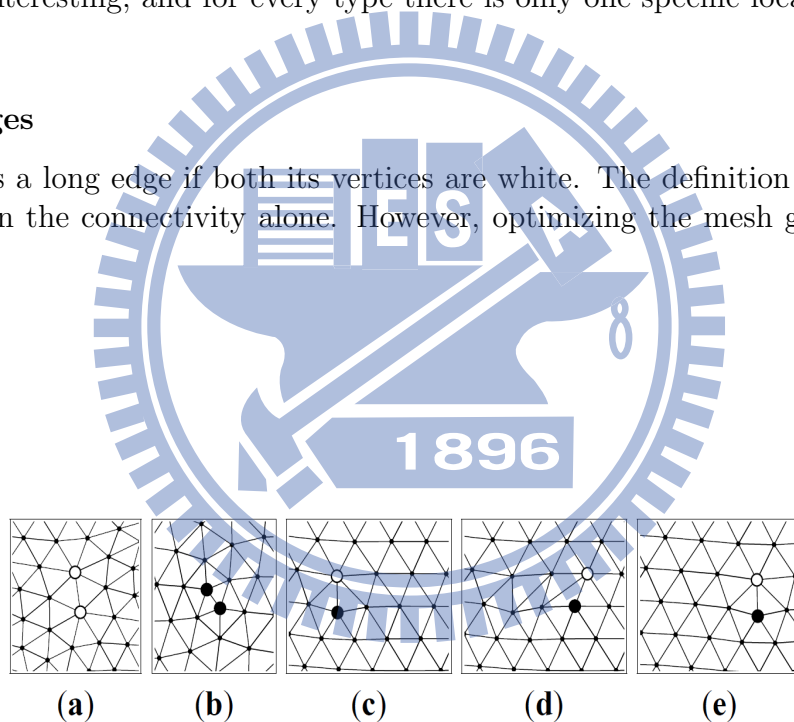


Figure 7: Types of edges: (a) A long edge. (b) A short edge. (c) A drifting edge. (d) The drifting edge moved two steps to the right. (e) After angle-based smoothing.

the angle-based smoothing reveals that long edges are actually geometrically longer than their nearby edges. Thus, the natural modification for this edge is to refine it. See Figure 5(a).

### Short edges

An edge is a short edge if both of its vertices are black. Short edges are actually shorter than other nearby edges if the mesh has been optimized using the anglebased smoothing procedure. Thus, we collapse short edges. See Figure 5(b).

### Drifting edges

An edge is a drifting edge if one of its endpoints is a white vertex and the other is black. Every drifting edge  $e$  has the following nice property: If we flip an edge  $e'$  incident on the white vertex that belongs to one of the faces adjacent to  $e$ , then  $e$  disappears (loses its drifting property) and reappears as the opposite to  $e$  within the quad defined by  $e'$ . Thus, we say that we have moved a drifting edge. This allows us to move a pair of white and black vertices of a drifting edge across a regular region of the mesh; see Figure 5(c-e).

## 5.1 Solving the Puzzle

To every edge among long, short or drifting edges, we apply only its corresponding operation. The central idea in solving the puzzle is to cause a drifting edge to migrate until it meets irregular vertices, and thus, an easy, long or short edge may appear. We perform operations on these edges, until only drifting edges are left. Then we choose an arbitrary drifting edge as the next edge to move. We proceed this way, until no drifting edge is left. This condition means that there are no easy, long and short edges as well, and the algorithm terminates. Consequently, the algorithm results in a mesh that has all irregular vertices surrounded by regular vertices. The number of such isolated irregular vertices is usually very small.

## 6 Numerical Results

First, we will show the remeshing method can reduce obtuse triangle and improve the angle in the mesh. Second, we compute the mean curvature on the original mesh, mesh after remesh, and after mesh refinement, to show that both of them are more accurate than the original mesh.

# 1

## Example 1

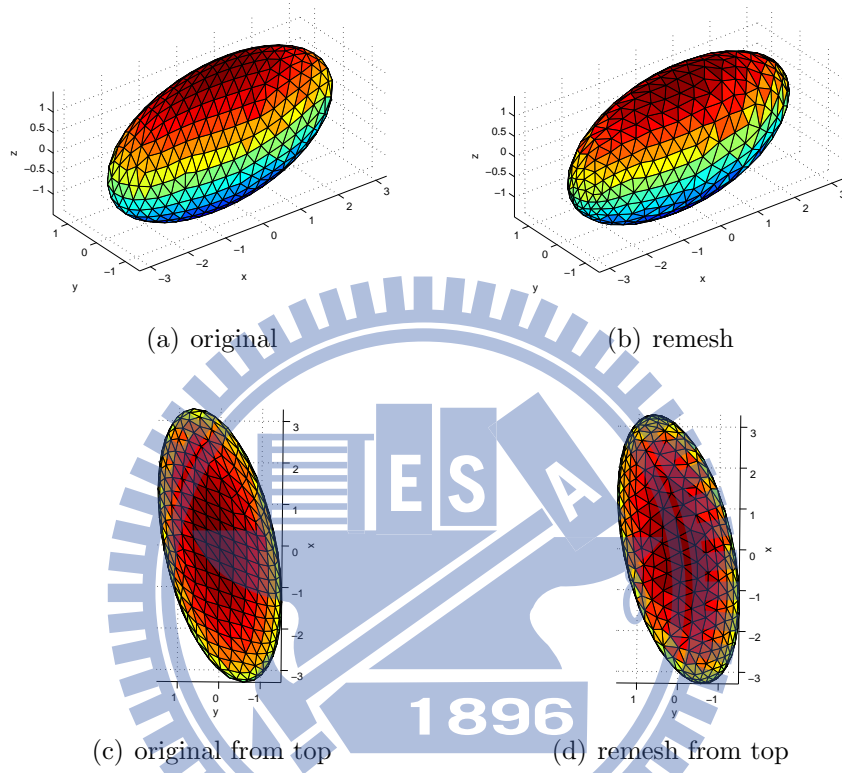


Figure 8: Example 1 for the ellipsoid under shear flow at  $T=1$

Table 1

	number of triangles	$\theta_{min}$	$\theta_{avg}$	obtuse triangles
origin	888	$23^\circ$	$40.2^\circ$	43.4%
remesh	874	$27.5^\circ$	$46.2^\circ$	1.2%
mesh refinement	7866	$27^\circ$	$46.2^\circ$	1.5%

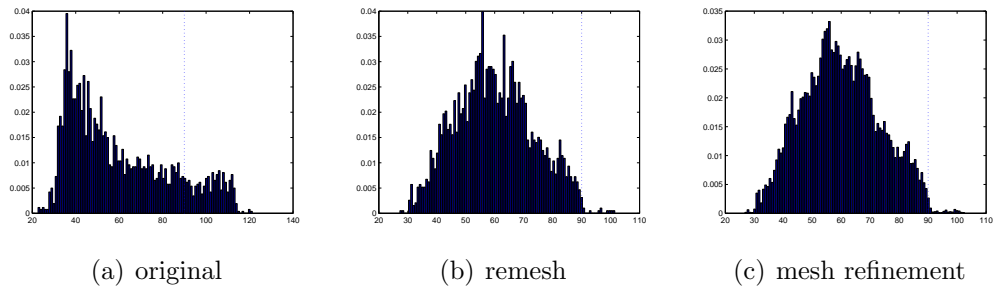
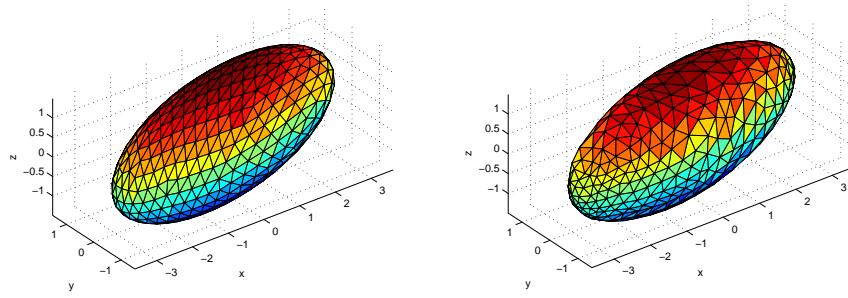


Figure 9: angle distribution

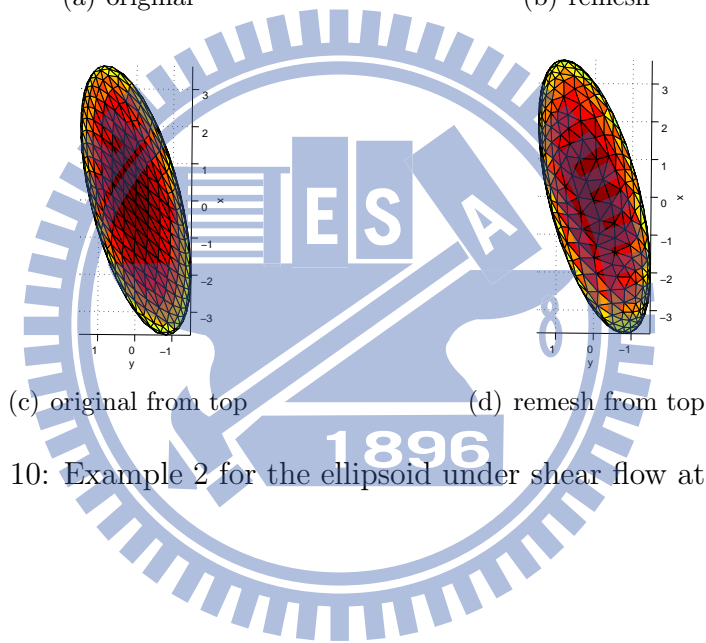
**Example 2**

	number of triangles	$\theta_{min}$	$\theta_{avg}$	obtuse triangles
origin	888	16.9°	33°	65.3%
remesh	852	24.4°	44.8°	3.5%
mesh refinement	7668	24.3°	44.8°	4%



(a) original

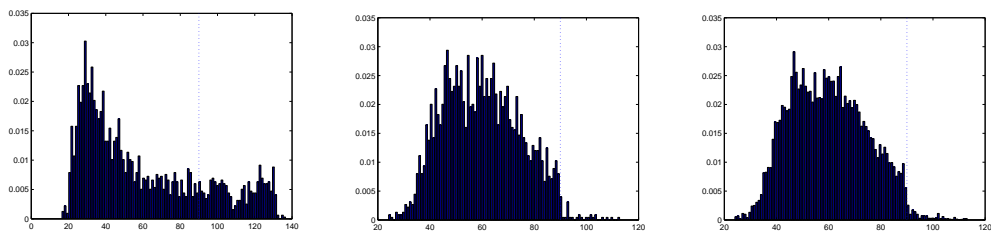
(b) remesh



(c) original from top

(d) remesh from top

Figure 10: Example 2 for the ellipsoid under shear flow at  $T=1.5$

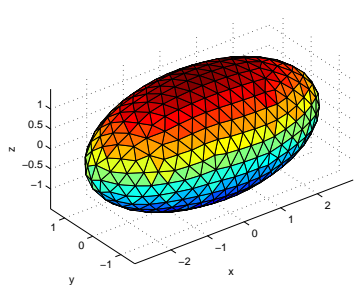


(a) original

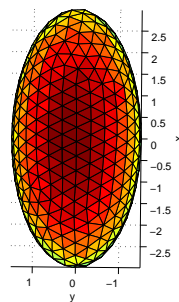
(b) remesh

(c) mesh refinement

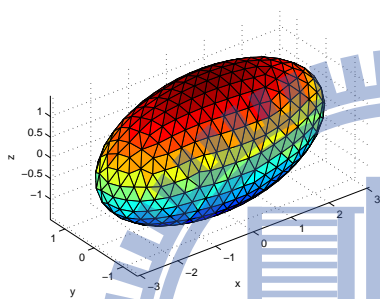
Figure 11: angle distribution



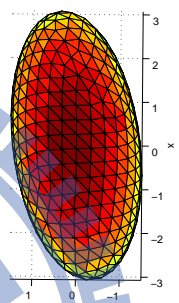
(a)  $T=0$



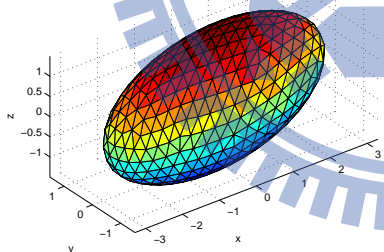
(b) top



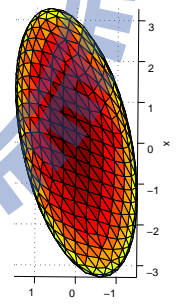
(c)  $T=0.5$



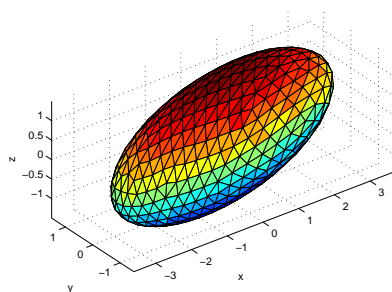
(d) top



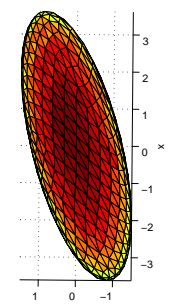
(e)  $T=1$



(f) top

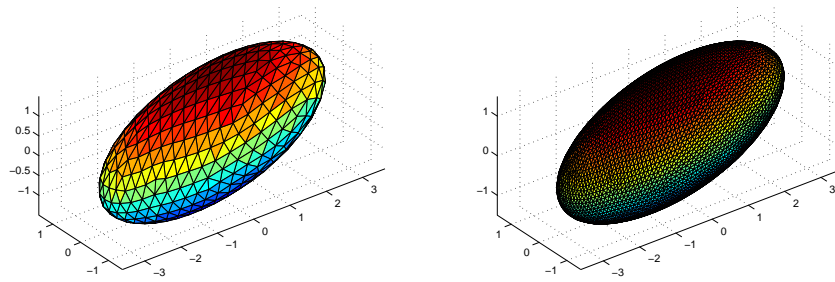


(g)  $T=1.5$



(h) top

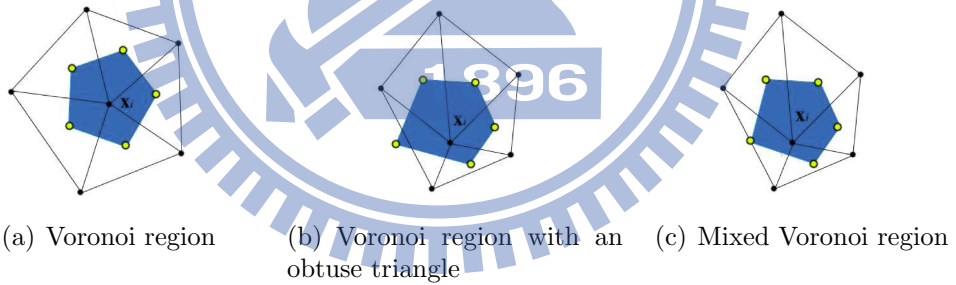
Figure 12: Ellipsoid is evolving under shear flow



(a) original

(b) mesh refinement

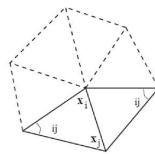
Figure 13: Explain what is mesh refinement



(a) Voronoi region

(b) Voronoi region with an obtuse triangle

(c) Mixed Voronoi region



(d) 1-ring neighbors and angles opposite to an edge

Figure 14



## 2

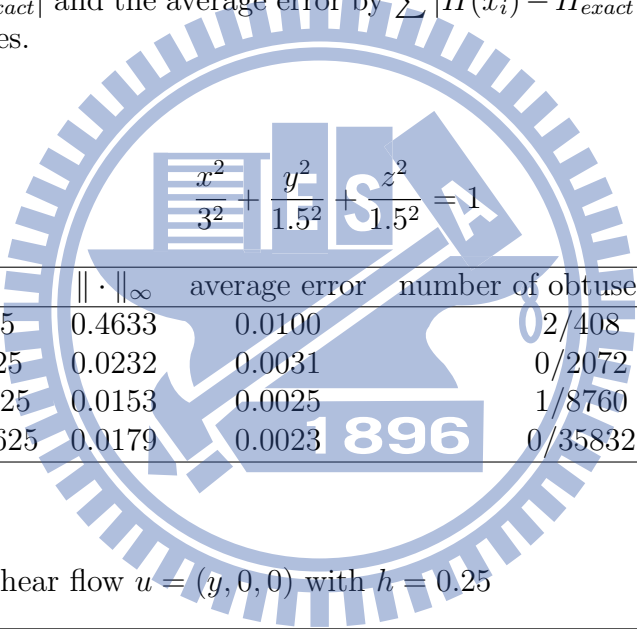
To compute the mean curvature numerically, we have already known that the equation  $\Delta_s \mathbf{X} = 2H\mathbf{n}$ , where  $H$  is the mean curvature and  $\mathbf{n}$  is the outward unit normal. The mean curvature is defined on the vertex of triangle:

$$\Delta_s \mathbf{X}_i = \frac{1}{A_M(\mathbf{X}_i)} \sum_{j \in N_1(\mathbf{X}_i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} (\mathbf{X}_i - \mathbf{X}_j)$$

The  $N_1(\mathbf{X}_i)$  means the 1-ring neighbors of vertex  $\mathbf{X}_i$ ,  $A_M(\mathbf{X}_i)$  is the area of Mixed Voronoi region (see Figure 14). The angles  $\alpha_{ij}$  and  $\beta_{ij}$  are shown in Figure 14(d).

To measure the error, we compute the maximum error in the notation  $\|\cdot\|_\infty$  by  $\max |H(x_i) - H_{exact}|$  and the average error by  $\sum |H(x_i) - H_{exact}|/N$ , where  $N$  is the number of vertices.

### Original mesh



	$\ \cdot\ _\infty$	average error	number of obtuse triangles
h=0.5	0.4633	0.0100	2/408
h=0.25	0.0232	0.0031	0/2072
h=0.125	0.0153	0.0025	1/8760
h=0.0625	0.0179	0.0023	0/35832

### Under flow

Evolving under shear flow  $u = (y, 0, 0)$  with  $h = 0.25$

	$\ \cdot\ _\infty$	average error	number of obtuse triangles
t=0	0.0232	0.0031	0/2072
t=0.5	0.1384	0.0069	112/2072
t=1	0.4807	0.1027	984/2072
t=1.5	0.8136	0.2491	1461/2072

### Remesh for t=1

	$\ \cdot\ _\infty$	average error	number of obtuse triangles
origin	0.4807	0.1027	984/2072
remesh	0.3032	0.0141	70/2024
mesh refinement	0.3856	0.0080	293/8096

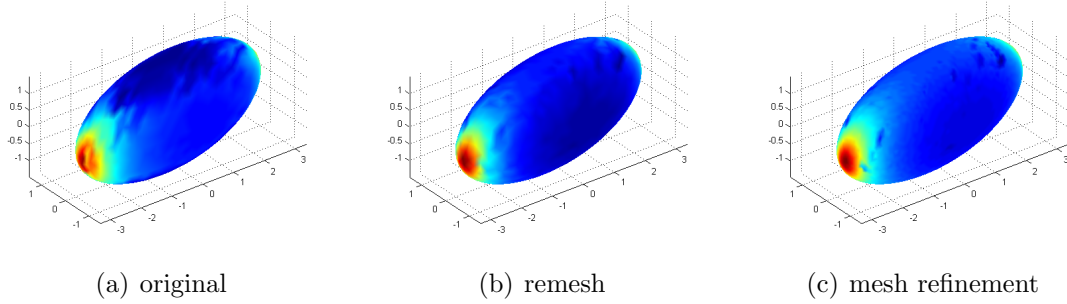


Figure 15: mean curvature for  $t=1$

### Remesh for $t=1.5$

	$\ \cdot\ _\infty$	average error	number of obtuse triangles
origin	0.8136	0.2491	1461/2072
remesh	0.3468	0.0163	52/2024
mesh refinement	0.6159	0.0078	297/8048

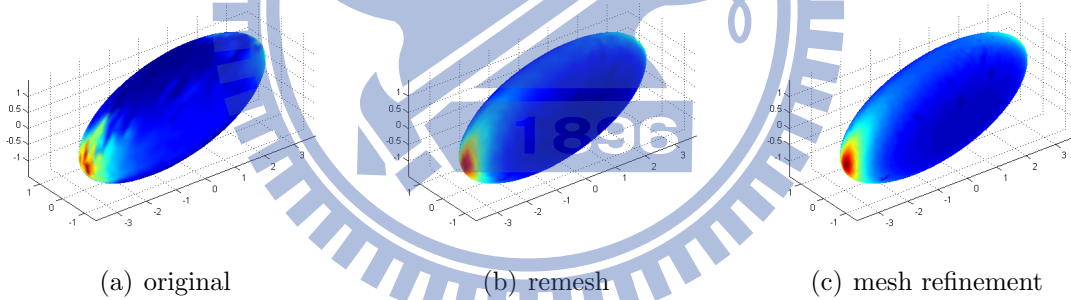


Figure 16: mean curvature for  $t=1.5$

## 7 Conclusion

Our method can greatly reduce the number of obtuse triangles in the given triangular mesh. However, it still cannot remove obtuse triangles completely. The reason is the regularity. In Section 5, we improve the mesh quality by regularizing its connectivity, but it still exists irregular vertices. The irregular vertices usually generate obtuse triangles. Hence, further research is needed to improve the result of the regularity.

## Reference

- [1] J. M. Saviat O. Stab. A. Rassineux, P. Villon. Surface remeshing by local hermite diffuse interpolation. *International Journal for Numerical Methods in Engineering*, 49:31–49, 2000.
- [2] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 347–354, 2002.
- [3] Siu-Wing Cheng, Tamal K. Dey, Edgar A. Ramos, and Rephael Wenger. Anisotropic surface meshing. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 202–211, 2006.
- [4] Michael S. Floater. Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geom. Des.*, 14(3):231–250, 1997.
- [5] Pascal J. Frey. About surface remeshing. In *IMR*, pages 123–136, 2000.
- [6] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 355–361, 2002.
- [7] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 99–108, 1996.
- [8] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 19–26, 1993.
- [9] Kai Hormann, Ulf Labsik, and Günther Greiner. Remeshing triangulated surfaces with optimal parameterizations. *Computer-Aided Design*, 33(11):779–788, 2001.
- [10] Francois Labelle and Jonathan Richard Shewchuk. Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Proceedings of the nineteenth annual symposium on Computational geometry*, SCG '03, pages 191–200, 2003.
- [11] Greg Leibon and David Letscher. Delaunay triangulations and voronoi diagrams for riemannian manifolds. In *Proceedings of the sixteenth annual symposium on Computational geometry*, SCG '00, pages 341–349, 2000.

- [12] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Proc. VisMath*, pages 35–57. 2002.
- [13] Steven J. Owen, David R. White, and Timothy J. Tautges. Facet-based surfaces for 3d mesh generation. In *IMR*, pages 297–311, 2002.
- [14] Houman Borouchaki Pascal J. Frey. Geometric surface mesh optimization. *Computing and Visualization in Science*, 1:113–121, 1998.
- [15] Pedro V. Sander, Steven J. Gortler, John Snyder, and Hugues Hoppe. Signal-specialized parametrization. In *Proceedings of the 13th Eurographics workshop on Rendering, EGRW '02*, pages 87–98, 2002.
- [16] D.J. Struik. *Lectures on Classical Differential Geometry: Second Edition*. Addison-Wesley series in mathematics. Addison-Wesley Publishing Company, 1961.
- [17] Vitaly Surazhsky and Craig Gotsman. Explicit surface remeshing. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, SGP '03*, pages 20–30, 2003.
- [18] Vitaly Surazhsky and Craig Gotsman. High quality compatible triangulations. *Eng. with Comput.*, 20(2):147–156, 2004.
- [19] Greg Turk. Re-tiling polygonal surfaces. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques, SIGGRAPH '92*, pages 55–64, 1992.
- [20] Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L. Mitchell. Curved PN triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics, I3D '01*, pages 159–166, 2001.
- [21] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94*, pages 247–256, 1994.
- [22] Tian Zhou and Kenji Shimada. An angle-based approach to two-dimensional mesh smoothing. In *IMR*, pages 373–384, 2000.