A Flexible Mesh-Generation Strategy for Image Representation Based on Data-Dependent Triangulation

by

Ping Li
B.Sc., Xi'an Jiaotong University, 2009

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

A Flexible Mesh-Generation Strategy for Image Representation Based on Data-Dependent Triangulation

by

Ping Li
B.Sc., Xi'an Jiaotong University, 2009

Supervisory Committee

Dr. Michael D. Adams, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Panajotis Agathoklis, Departmental Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Michael D. Adams, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Panajotis Agathoklis, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Data-dependent triangulation (DDT) based mesh-generation schemes for image representation are studied. A flexible mesh-generation framework and a highly effective mesh-generation method that employs this framework are proposed.

The proposed framework is derived from frameworks proposed by Rippa and Garland and Heckbert by making a number of key modifications to facilitate the development of much more effective mesh-generation methods. As the proposed framework has several free parameters, the effects of different choices of these parameters on mesh quality (both in terms of squared error and subjectively) are studied, leading to the recommendation of a particular set of choices for these parameters. A new mesh-generation method is then introduced that employs the proposed framework with these best parameter choices.

Experimental results show our proposed mesh-generation method outperforms several competing approaches, namely, the DDT-based incremental scheme proposed by Garland and Heckbert, the COMPRESS scheme proposed by Rippa, and the adaptive thinning scheme proposed by Demaret and Iske. More specifically, in terms of PSNR, our proposed method was found to outperform these three schemes by median margins of 4.1 dB, 10.76 dB, and 0.83 dB, respectively. The subjective qualities of reconstructed images were also found to be correspondingly better. In terms of computational cost, our proposed method was found to be comparable to the schemes proposed by Garland and Heckbert and Rippa. Moreover, our proposed method requires only about 5 to 10% of the time of the scheme proposed by Demaret and Iske. In terms of memory cost, our proposed method was shown to require essentially same amount of memory as the schemes proposed by Garland

and Heckbert and Rippa, and orders of magnitude (33 to 800 times) less memory than the scheme proposed by Demaret and Iske.

Contents

List of Tables

List of Figures

List of Acronyms

DDT	data-dependent triangulation
LOP	Lawson's local optimization procedure
PSNR	peak signal-to-noise ratio
MSE	mean squared error
MMSODD	maximum magnitude second-order directional derivative
PAE	peak absolute error
PWAE	peak weighted absolute error
ABN	angle between normals
JND	jump in normal derivative
DP	derivations from linear polynominals
DLP	distances from planes
YMS	Yu-Mores-Sederberg cost
ELABN	edge-length-weighted ABN
ELJND	edge-length-weighted JND
D	Delaunay
SE	squared error
GHH	Garland and Heckbert hybrid
SQSE	shape-quality-weighted SE
JNDSE	JND-weighted SE
EL	edge length
LIFO	last-in first-out

ACKNOWLEDGEMENTS

This thesis would never have been possible without the help and support of numerous kind people who in one way or another contributed and extended their valuable assistance along the way. I would like to take this opportunity to express my thanks to these individuals.

First and foremost I offer my sincerest gratitude to my supervisor, Dr. Michael Adams, who has guided and supported me throughout my graduate studies with his knowledge and patience. I thank Michael for giving me the opportunity to work on research topics that really interest me deeply, and for this reason, I have enjoyed my studies at UVic very much. Without his help, this thesis would never have been written. One simply could not wish for a better or friendlier supervisor.

Next, I would like to thank my supervisory committee member Dr. Panajotis Agathoklis for serving on my committee and providing constructive comments. I also want to express my gratitude to the course instructors during my graduate studies, Dr. Andreas Antoniou, Dr. Pan Agathoklis, Dr. Wu-Sheng Lu, and Dr. Alexandra Branzan Albu, for their fantastic lessons and inspiration.

I wish to thank my best friend, Chenyuan Wang, for being such a delightful companion. I thank you from the bottom of my heart for every little thing you did for me, every concern, every laugh, every tear, and most of all for every moment that we spent together.

I am also indebted to many friends and colleagues who have helped me during my studies: Dan Li, Xi Tu, Jie Yan, Chamira Edussooriya, Yang Song, Li Ji, Binyan Zhao, Teng Ge, and Congzi Liu. I am also very grateful for the help and assistance that I received from the staff in the Department of Electrical and Computer Engineering: Vicky Smith, Moneca Bracken, Janice Closson, Dan Mai, and Erik Laxdal.

I would like to acknowledge the financial support that I received from the Natural Sciences and Engineering Research Council of Canada and University of Victoria. The research grant and fellowship they provided helped me resolve the financial burden so that I can focus on my research.

Last but not the least, I want to thank my entire family for their unconditional love, understanding and constant support, my parents for working so hard and putting me through school, Zhichao for being such a sweet little brother, my grandpa for believing in me, and my aunt Qian for being so giving, loving and caring.

DEDICATION

To my family, who offered me unconditional love and support!

Chapter 1

Introduction

1.1 Mesh Representation of Images

Digital images can be represented in various ways. One of the most straightforward and commonly used approaches is a lattice based representation, e.g., images are uniformly sampled at each point on a rectangular grid. Due to the nonstationary nature of most images, such sampling is far from optimal. When uniform sampling is employed, the sampling density will inevitably be too high in regions where the image is changing slowly and too low in regions where the image is changing rapidly. Besides that, storing and transmitting uniformly sampled images often requires large amounts of memory or high bandwidths. Thus, nonuniform sampling (i.e., sampling at a subset of points from a lattice) for image representations has been considered as a means to overcome these drawbacks.

With nonuniform sampling, the sample points are wisely chosen so that their spatial density varies in relation to the degree of local image detail. That is, more sample points are placed in regions containing high frequency features and fewer sample points are placed in regions containing predominantly low frequency components. Nonuniform sampling often leads to much more compact representations because with nonuniform sampling, an image can be represented using far fewer sample points than uniform sampling. Image representations based on nonuniform sampling also have the ability to better capture characteristic features inherent in images, such as sharp edges. Thus, such representations have been utilized and proven beneficial for many applications, such as feature detection [1], pattern recognition [2], computer vision [3], restoration [4], tomographic reconstruction [5], filtering [6], interpolation [7], and image/video coding [8, 9, 10, 11, 12, 13, 14].

Although many classes of image representations based on nonuniform sampling have

been proposed, the class based on triangle meshes has become quite popular. The process of constructing meshes to represent images is called mesh modelling, and the meshes are referred as mesh models for images. With a triangle mesh model of an image, the image domain is partitioned by a triangulation into a set of (triangle) faces and then over each face, an approximating function is constructed. A triangle mesh is characterized by its sample points and their connectivity. Sample points correspond to geometry of the triangle mesh, which comprise coordinate information. Connectivity corresponds to the topology of the mesh, which captures the incidence relation between the triangles in the mesh.

Triangle meshes are advantageous since a single triangle can often well approximate many sample values on a rectangular grid. For example, a large white rectangular region, which may comprise thousands of pixels in an image, could be well represented by two triangles instead. Additionally, triangle meshes are ideally suited for modeling images with sharp edges and corners. Furthermore, the geometry of such meshes is mathematically simple and quick to calculate, which makes them particularly useful in real-time environments where speed is important. Considering the above benefits, researchers have proposed numerous representation methods based on triangle meshes [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 7, 32, 33].

From the foregoing, we know that for image representations, triangle meshes have many attractive benefits; but how do we generate triangle meshes to represent images? The answer to the above question is known as the mesh-generation problem. The mesh-generation problem that we address in this thesis can be succinctly stated as follows: Given an image and a desired number of sample points, find the triangle mesh model that minimizes the measure of the difference between the given image and the mesh-model approximation. This problem can be separated into two distinct (but related) sub-problems:

1. the selection of the sample points (i.e., the vertices of a triangulation), and

2. the selection of the connectivity of the triangulation.

Finding good *computationally-efficient* methods for solving the above-stated mesh-generation problem is quite challenging, since problems like this are known to be NP hard [34].

1.2 Historical Perspective

As stated earlier, a great many mesh-generation methods have been developed over the years. One way to categorize these methods depends on whether they are iterative or non-iterative during the sample points selection process. Noniterative methods determine all

of the sample points in only *one* step. For instance, in [35], the classical Floyd-Steinberg error-diffusion algorithm [36] was employed to choose all of the sample points in the image domain according to the local image content. Then Delaunay triangulation is used to connect all of the sample points. More similar non-iterative approaches can be found in [4, 5, 37, 38]. Iterative schemes, on the other hand, choose the set of sample points in many iterations. Such schemes are more complicated, since they can either add and/or remove sample points during each iteration [16, 17, 39, 40, 21].

Typically, non-iterative methods are faster than iterative ones. The quality of image approximations produced by noniterative methods, however, are often lower than those generated by iterative schemes.

Two classes of iterative schemes are quite popular: refinement schemes and simplification schemes. A simplification strategy works by removing vertices from a fine initial triangulation. For instance, the works [16, 39] proposed similar iterative point removal schemes, called adaptive thinning algorithms. Such schemes remove sample points one by one, based on a certain anticipated error, until a subset of most significant points is generated. During the sample point removing process, they make sure that the piecewise linear interpolants over the Delaunay triangulation of these subsets approximate progressively the function values sampled at the original scattered points. These schemes perform very well, however, they require a significant amount of memory space and computational cost.

Refinement schemes, on the other hand, operate by adding vertices to a coarse triangulation, while minimizing the error in every step. They are simple and fast relative to simplification schemes, but may fail to find a good approximation because of local minima. For instance, the papers [21, 17] proposed two similar DDT-based refinement mesh-generation schemes that iteratively add new sample points, based on local error measure, into the coarse mesh until a target number of sample points or a prescribed error tolerance has been reached.

One particularly popular subclass of triangle mesh representations is those based on Delaunay triangulations) [41]. The connectivity of Delaunay triangulations (i.e., how the points in the triangulation are connected by edges) is determined solely by the geometry (i.e., position) of the points being triangulated. Delaunay triangulations have numerous properties that make them attractive choices for approximation purposes. First, to whatever extent is possible, Delaunay triangulations avoid long thin (i.e., sliver) triangles which can lead to very poor approximations *if not well chosen*, by maximizing the minimum interior angle of the triangles in the triangulations. Another advantage of Delaunay triangulations is that they are unique, if certain degeneracies are handled by a technique such as the preferred

directions scheme of [42]. Delaunay triangulations play an important role in computational geometry, and there are many mesh-generation schemes based on Delaunay triangulations proposed to date [15, 43, 19, 16, 39, 17, 20, 3, 18, 40].

For a long time, long and thin (i.e., sliver) triangles were considered bad for approximation and avoided whenever possible. Thus, Delaunay triangulation based methods were commonly employed, since they try to choose triangulations so as to contain many nice looking, nearly equiangular, triangles. Studies and numerical examples [44, 30, 17, 18, 45], however, showed that sliver triangles are not always bad choices for approximation and that great improvement in the quality of approximation can be obtained if sliver triangles are well chosen. In particular it was demonstrated that long and thin triangles are very suitable for the reconstruction of areas where a function has high second-order derivatives in one direction as compared to others [45, 18]. In images, such areas often correspond to edges. For this reason, Delaunay triangulation is frequently suboptimal for image approximation.

Another subclass of triangle mesh representations is those based on data-dependent triangulations (DDTs). It is this particular subclass that is of interest herein. In the case of DDTs, the connectivity of the triangulation is chosen in a way that depends on the data set from which the points to be triangulated originated (and not just the geometry of those points). Since, unlike the Delaunay case, DDTs can have their connectivity chosen arbitrarily, DDTs offer vastly greater flexibility, and theoretically have the potential to perform much better than their Delaunay counterparts *if well chosen* [44]. In practice, however, due to this increased flexibility, it is much more difficult to develop highly-effective *computationally-efficient* mesh-generation schemes that are based on DDTs, as compared to the Delaunay case.

To date, numerous DDT-based mesh-generation methods have been proposed [17, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 18, 31, 7, 32, 46, 33], however, many of them (e.g. [24, 22, 26, 29, 28, 27, 23, 25, 7]) concern themselves with only the problem of triangulation-connectivity selection (i.e., the second sub-problem of mesh generation mentioned on page 2). That is, they assume that the sample points are given or have already been chosen through some unspecified means. Such an assumption is not very realistic in many applications. For this reason, methods that select both the sample points and triangulation connectivity are of great practical interest.

One difficulty with DDT-based mesh-generation methods is that they can often have very high computational cost. For example, some DDT-based schemes [47, 48, 32] are based on simulated annealing, which is very computationally expensive. Another DDT-based method proposed in [27] takes several iterations and requires between 0.5 and 5

seconds per iteration for an 80×80 image. Furthermore, this method only considers triangulation connectivity. Therefore, as one can well imagine, in the case of the more difficult problem of choosing both the sample points and triangulation connectivity, mesh-generation methods can potentially become very computationally complex.

Of those schemes that are relatively fast and choose both the sample points and triangulation connectivity, a particularly good one was proposed by Garland and Heckbert [17, Algorithm IV] (with the quality threshold parameter `qthresh` chosen as 0.5 and an L^2 error measure), which we henceforth refer to as the **Garland-Heckbert (GH)** method. The GH method is associated with a basic framework for mesh generation, which is very similar to a framework proposed even earlier by Rippa [21]. These two frameworks served as foundation for the work in this thesis.

1.3 Overview and Contribution of the Thesis

This thesis is primary concerned with image representations based on triangle meshes. In particular, we studied the choices of mesh models, and examined how to select the parameters of mesh model (i.e., mesh generation). In passing, we note that some of the work presented in this thesis has also been partly described in the author's papers [49, 50].

This thesis makes two main contributions. The first contribution is that it proposes a new framework for mesh-generation algorithms by adding a number of key improvements to the earlier frameworks from Rippa and Garland and Heckbert. As the proposed framework has several free parameters, we also studied the framework to determine how different choices for these free parameters affect mesh quality, leading to the recommendation of a particular set of choices for these parameters. The second contribution of our thesis is that it proposes a highly effective mesh-generation method based on the proposed framework, with these best parameter choices. As we shall see, our proposed mesh-generation method produces meshes with significantly lower approximation errors than those generated by other competing schemes. At the same time, the computational and memory costs of the proposed method are relatively modest.

Structurally, the remainder of this thesis is organized into three chapters and one appendix. The first chapter provides the background information necessary to place this work in context and facilitate the understanding of the research results presented herein. The remaining two chapters present the main research work and results. The appendix provides supplemental information about the research work in this thesis, but such details are not strictly necessary for an understanding of the main thesis content.

Chapter 2 provides some essential background information necessary to understand the work in this thesis. Some of the notation and terminology used herein are presented first, followed by some image processing fundamentals. Then some basic concepts from computational geometry are introduced, such as triangulation, Delaunay triangulation, and DDT. After that, the mesh model of images used in our framework is presented. Lastly, we comment on a grid-point to face mapping strategy used herein.

Chapter 3 presents a flexible mesh-generation framework and a new DDT-based mesh-generation scheme derived from our framework. The development of our framework and the proposed method, together with the results and analysis, are also discussed.

The chapter begins with the introduction of the local optimization procedure (LOP) algorithm. Next, we propose our new mesh-generation framework based on DDT, which has several free parameters that must be chosen in order to produce a fully-specified mesh-generation method. After that, we study how different choices of each free parameter affect the performance of the resulting method. Careful analysis is performed on these results, leading to the recommendation of a particular choice for each parameter.

Finally, in Chapter 3, we evaluate the performance of our proposed mesh-generation method by comparing it to three previously proposed mesh-generation methods regarding mesh quality, time and memory complexity. The first of the methods is a DDT-based refinement mesh-generation scheme proposed by Garland and Heckbert [17, Algorithm IV] with the quality threshold parameter `qthresh` chosen as 0.5 and an L^2 error measure, which we henceforth refer to by the name "GH". The second of the methods is also a DDT-based refinment mesh-generation scheme proposed by Rippa [21] with the least-squares edge cost (in the interpolating case), which we henceforth refer to by the name "R". The third of the methods is a Delaunay-based adaptive thinning scheme proposed by Demaret and Iske [16], which will be referred as "AT" henceforth. Based on the results collected, the proposed method is demonstrated to outperform the GH, R, and AT schemes, by median margins of 4.1 dB, 10.76 dB, and 0.83 dB, respectively. The subjective qualities of reconstructed images correlate reasonably well with PSNR. In terms of computational cost, our proposed method was found to be comparable to the GH and R schemes, with about 6 to 12% and 36 to 52% increase, respectively. Moreover, our proposed method requires only about 5 to 10% of the time of the AT scheme. In terms of memory cost, our proposed method was shown require essentially same amount of memory as the GH and R schemes and orders of magnitude (33 to 800 times) less memory than the AT scheme.

Chapter 4 summarizes the key results presented in this thesis. Finally, it concludes with suggestions of related topics for future research.

The appendix provides a brief description of the software used to collect experimental results in our research. This software was developed by the author with guidance from her supervisor. The code is fairly long and complex, as was not trivial to write. Amongst other things, the appendix includes a short tutorial on how to use our software.

Chapter 2

Preliminaries

2.1 Overview

Some essential background information is introduced in this chapter to promote a better understanding of the work presented in this thesis. We begin with an introduction to the basic notation and terminology used herein. Then, some image processing background is presented. Next, we explain some concepts from computational geometry. This is then followed by a discussion of mesh models of images and mesh generation. We conclude this chapter by some comments regarding a grid-point to face mapping strategy.

2.2 Notation and Terminology

In this thesis, the sets of integers and real numbers are denoted $\mathbb{Z}$ and $\mathbb{R}$, respectively. For $a, b \in \mathbb{R}$, the expressions (a, b), $[a, b]$, $(a, b]$, and $[a, b)$ denote the intervals $\{x : a < x < b\}$, $\{x : a \leq x \leq b\}$, $\{x : a < x \leq b\}$, and $\{x : a \leq x < b\}$, respectively. For a set S, its cardinality is denoted $|S|$. The empty set is denoted $\emptyset$. For a set S, $\overline{S}$ denotes the closure of S.

For a vector $x = (x_1, x_2, \ldots, x_n)$ in $\mathbb{R}^n$, its 2-norm is defined as

$$||x|| = \sqrt{{x_1}^2 + {x_2}^2 + \ldots + {x_n}^2}.$$

The gradient of a function f, denoted as ∇f, is defined as the vector field whose components are the partial derivatives of f. That is,

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right).$$

2.3 Image Processing

Binomial filters are lowpass filters with simple and efficient structures based on the binomial coefficients for implementing Gaussian filtering [51]. The transfer function H_n of the nth-order one-dimensional (1-D) binomial filter with zero-phase and unity DC gain is given by

$$H_n(z) = z^{(n-1)/2}\left(\frac{1}{2} + \frac{1}{2}z^{-1}\right)^{n-1},$$

where n is an odd integer. Two-dimensional binomial filters can be generated by using two one-dimensional binomial filters in a separable fashion (i.e., a tensor-product construction).

The Laplacian of a function f, denoted Δf, is defined as

$$\Delta f(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

For a function f, its **maximum magnitude second-order directional derivative (MMSODD)** [35] $d(x,y)$ at (x,y) can be computed by

$$d(x,y) = \max\left\{|\alpha(x,y) + \beta(x,y)|, |\alpha(x,y) - \beta(x,y)|\right\}, \tag{2.1}$$

where

$$\alpha(x,y) = \frac{1}{2}\left[\frac{\partial^2}{\partial x^2}f(x,y) + \frac{\partial^2}{\partial y^2}f(x,y)\right], \text{ and}$$

$$\beta(x,y) = \sqrt{\frac{1}{4}\left[\frac{\partial^2}{\partial x^2}f(x,y) - \frac{\partial^2}{\partial y^2}f(x,y)\right]^2 + \left[\frac{\partial^2}{\partial x \partial y}f(x,y)\right]^2}.$$

2.4 Computational Geometry

Next, we introduce some computational geometry concepts used in this thesis. This includes concepts such as a triangulation, Delaunay triangulation, and DDT.

In order to fully define the concept of triangulation, two basic computational geometry concepts, convex set and convex hull, need to be introduced first.

Definition 2.1 (Convex set)**.** *A set V of points is **convex** if for every pair of points a and b in V, every point on the line segment $\overline{ab}$ is in V.*

The definition of convex set is illustrated in Figure 2.1. As we can see, the set V of points, denoted by the shaded area in Figure 2.1(a), is convex since every point on a line segment

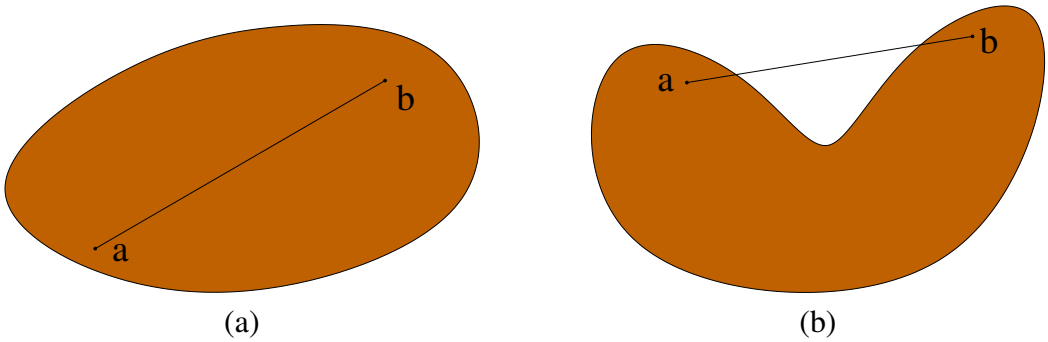


Figure 2.1: Examples of a (a) convex, and (b) non-convex sets .

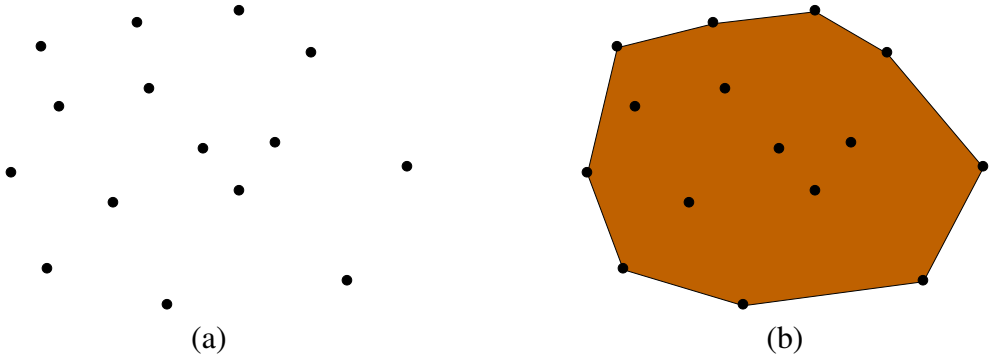


Figure 2.2: Convex hull example. (a) A set V of points, and (b) the convex hull of V .

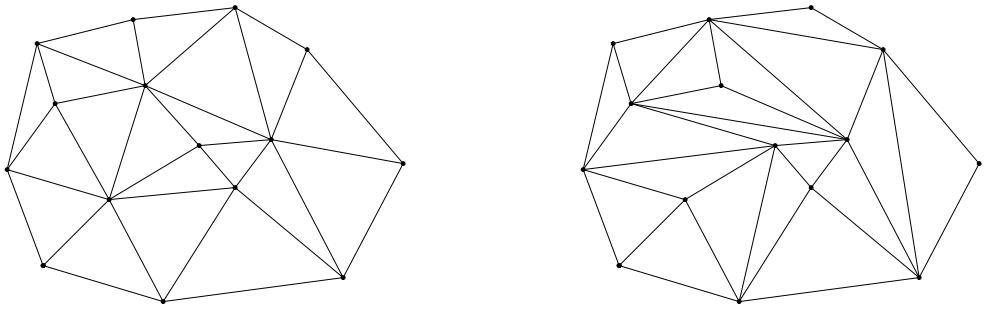
formed by a pair of points a and b from V is still in V. In contrast, the set V of points, denoted by the shaded area in Figure 2.1(b), is not convex, because part of the line segment $\overline{ab}$ is not in V.

Definition 2.2 (Convex hull)**.** *Given a finite set $V = \{p_1, p_2, \ldots, p_n\}$ of points in R^2, the **convex hull** of V, denoted $H(V)$, is the intersection of all convex sets that contain V (i.e., the "smallest" convex set that contains all of the points of V).*

An example of a convex hull is shown in Figure 2.2. Given a set V of points as shown in Figure 2.2(a), the convex hull of V is the shaded area shown in Figure 2.2(b). One way to visualize a convex hull of V is to imagine letting a rubber band snap tight around all the points in V. The resultant polygon formed by the rubber band is the boundary of the convex hull of V.

With convex hull defined, we are ready to introduce the concept of a triangulation, which is of fundamental importance herein.

Definition 2.3 (Triangulation)**.** *A **triangulation** of a finite set V of points is a set $T = \{f_i\}_{i=0}^{|T|-1}$ of nondegenerate open triangles satisfying the following conditions:*



(a) (b)

Figure 2.3: Example of triangulation of a set of points. (a) a triangulation of V, and (b) another triangulation of V .

1. *the set of all vertices of triangles in T is V;*
2. *every edge of a triangle in T contains only two points from V;*
3. $\cup_{i=0}^{|T|-1} \overline{f_i}$ *is the convex hull of V; and*
4. $f_i \cap f_j = \emptyset$ *for* $i \neq j$.

In other words, a triangulation T of a finite set V of points is a subdivision of the convex hull of V into a set of triangles such that any two triangles in T never intersect, and the set of points that are vertices of T coincides with V. For a triangulation T, the vertices, edges, and faces of T are denoted as $\mathcal{V}(T)$, $\mathcal{E}(T)$, and $\mathcal{F}(T)$, respectively.

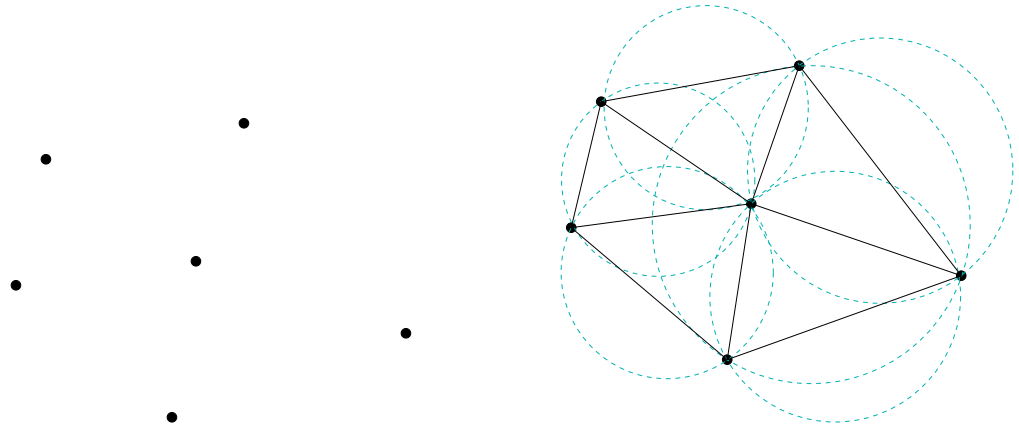
There are many ways to subdivide the convex hull of a finite set V of points in order to produce a valid triangulation. Figure 2.3 provides two different triangulations of a set of points. We can see that, although the triangulations in Figures 2.3(a) and (b) share the same set of vertices, the connectivities of the triangulations (i.e., the edges in the triangulations) are different.

The two commonly employed types of triangulations are the Delaunay triangulation and the DDT. The Delaunay triangulation was described by Delaunay in 1934 [41]. Before defining Delaunay triangulation, we first need to introduce the concept of a circumcircle.

Definition 2.4. (Circumcircle) *The unique circle that passes through all three vertices of a triangle T is called the* ***circumcircle*** *of T.*

With the definition of circumcircle in place, we can now present the definition of Delaunay triangulation.

Definition 2.5. (Delaunay triangulation) *A triangulation T of a set V of points in a plane is said to be* ***Delaunay*** *if no point in V is inside the circumcircle of any triangle in T.*



(a) (b)

Figure 2.4: Example of a Delaunay triangulation of a set of points. (a) A set V of points, and (b) a Delaunay triangulation of V .

Delaunay triangulations have the property that they maximize the minimum interior angle of all triangles in the triangulation. In this sense, Delaunay triangulations avoid silver triangles to whatever extent is possible. The Delaunay triangulation of a set V of points is only guaranteed to be unique if no four points in V are cocircular. In many applications, however, it is desirable that the Delaunay triangulation be uniquely determined. Take image representation, for example. If the Delaunay triangulation of a set V of points is unique, only the positions and values of the sample points are needed to generate a unique mesh representation for an image. On the other hand, if the uniqueness cannot be ensured, the connectivity of the sample points in V is also needed to fully determine the triangulation for an image. Thus, the uniqueness of the Delaunay triangulation is desirable for more compact representations. Several schemes have been proposed to resolve this nonuniqueness issue. One simple strategy, called preferred directions, was proposed by Dyken in 2006 [42]. This scheme provides a means to uniquely choose one out of all possible Delaunay triangulations of a set of points.

An example of a Delaunay triangulation is shown in Figure 2.4. A set V of points is given by Figure 2.4(a) and the Delaunay triangulation of V is given by Figure 2.4(b). In Figure 2.4(b), the circumcircle of each face is drawn using dashed lines. As we can see, no vertex falls inside any circumcircle, hence, this triangulation is Delaunay.

Another type of triangulation is a DDT, which is less restrictive than a Delaunay triangulation. In the case of DDTs, the connectivity of the triangulation is chosen in a way that depends on the data set from which the points to be triangulated originated (and not just the geometry of those points). Unlike the Delaunay case, DDTs can have their con-

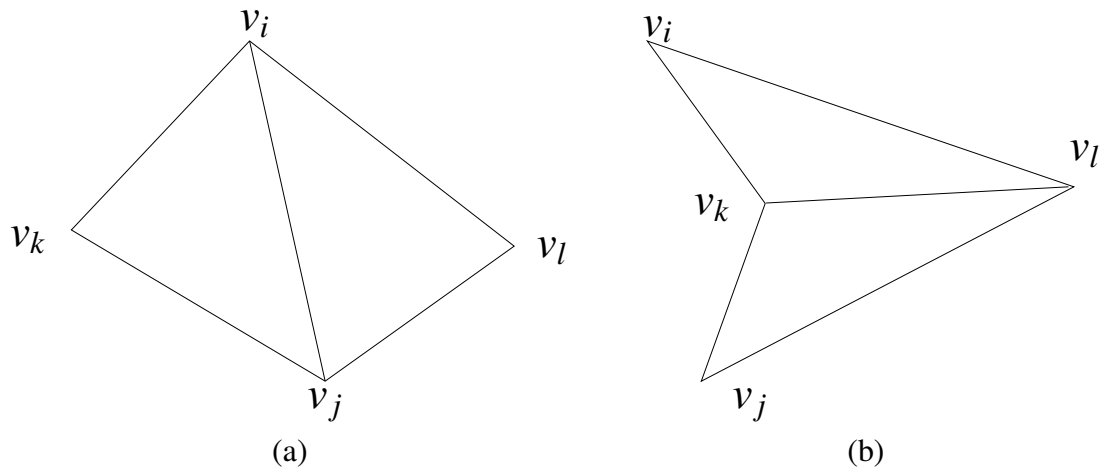


Figure 2.5: Flippable and non-flippable edge example. (a) Flippable edge $\overline{v_i v_j}$ and (b) non-flippable edge $\overline{v_k v_l}$.

nectivity chosen arbitrarily. Thus, DDTs offer vastly greater flexibility, and theoretically have the potential to perform much better than their Delaunay counterparts *if well chosen* [44]. In practice, however, due to this increased flexibility, it is much more difficult to develop highly-effective *computationally-efficient* mesh-generation schemes that are based on DDTs, as compared to the Delaunay case. Next, we introduce the concepts of flippable edges and edge flips, which are of fundamental importance in DDTs.

Definition 2.6 (Flippable edge)**.** *An edge e in a triangulation is said to be **flippable** if it has two incident faces (i.e., is not on the triangulation boundary) and the union of these two faces is a strictly convex quadrilateral q.*

Figure 2.5(a) shows an example of a flippable edge $\overline{v_i v_j}$ in a convex quadrilateral $v_i v_k v_j v_l$. Edge $\overline{v_k v_l}$ in Figure 2.5(b) is not flippable because the the quadrilateral $v_i v_k v_j v_l$ is not convex.

If an edge e is flippable with its two incident faces forming the strictly convex quadrilateral q, a valid triangulation is obtained if e is deleted from the triangulation and replaced by the other diagonal of q. This transformation is known as an **edge flip**. An example of edge flip is shown in Figure 2.6, where a flippable edge $\overline{v_i v_j}$ in Figure 2.6(a) is transformed to the edge $\overline{v_k v_l}$ in Figure 2.6(b) by an edge flip. Any triangulation of a set of points can be obtained from any other triangulation of the same set of points by a finite sequence of edge flips [52, 53]. Moreover, since an edge flip does not change the number of edges in a triangulation, this further implies that every triangulation of a set of points has the same number of edges.

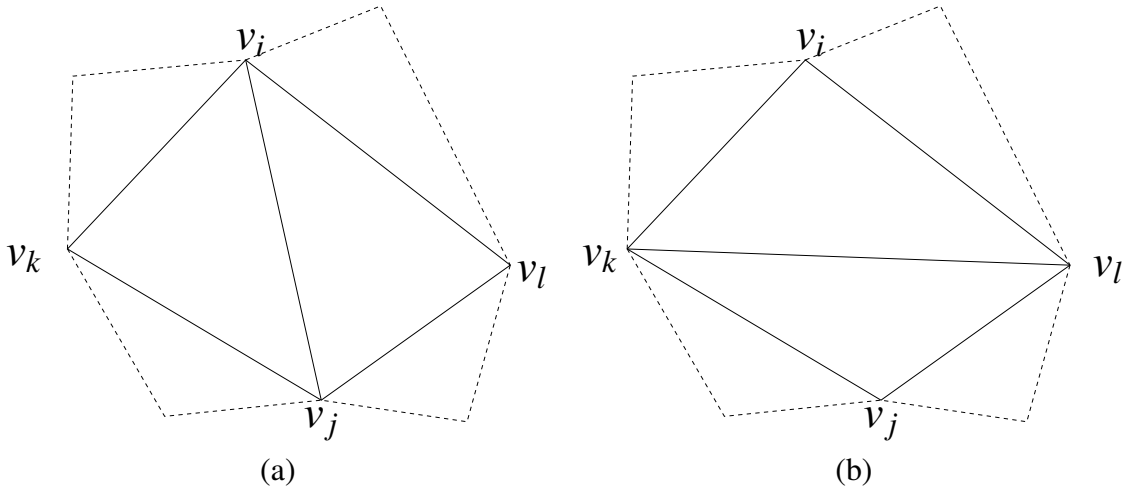


Figure 2.6: Edge flip example. Part of (a) a triangulation with an edge $\overline{v_i v_j}$ and (b) the new triangulation obtained after the edge $\overline{v_i v_j}$ is transformed to the edge $\overline{v_k v_l}$ by an edge flip.

2.5 Mesh Models of Images

As mentioned before, triangle mesh models are quite beneficial for image representation purposes. There are many ways to represent images using meshes, and each has different advantages and drawbacks. For example, Tu and Adams proposed a scheme [54], based on constrained Delaunay triangulation, that explicitly represents discontinuities (i.e., image edges). This scheme , while flexible, is conceptually complex.

In this thesis, we chose to employ a mesh model for images based on DDTs, as described below. It offers great flexibility and efficiency in dynamically changing the geometry and topology of the mesh.

Consider an integer-valued image function ϕ defined on $\Lambda = \{0, 1, \ldots, W-1\} \times \{0, 1, \ldots, H-1\}$ (i.e., a rectangular grid of width W and height H). With a triangle mesh model of an image, the image domain is partitioned by a triangulation into a set of (triangle) faces and then over each face of the triangulation an approximating function is constructed. Figure 2.7 shows an example of mesh modelling using triangle meshes. The image in Figure 2.7(a) can be viewed as a surface in 3-D as shown in Figure 2.7(b) with the surface height corresponds to the image values of sample points. A triangulation is formed by partitioning the image domain into a set of triangles, as illustrated in Figure 2.7(c). The resulting triangle mesh is shown in Figure 2.7(d) This process is known as mesh modelling. Then a reconstructed image can be generated by sampling the grid points of the image, which is shown in Figure 2.7(e).

For convenience in what follows, we let $\mathcal{V}(T)$ and $\mathcal{F}(T)$ denote the vertices and faces

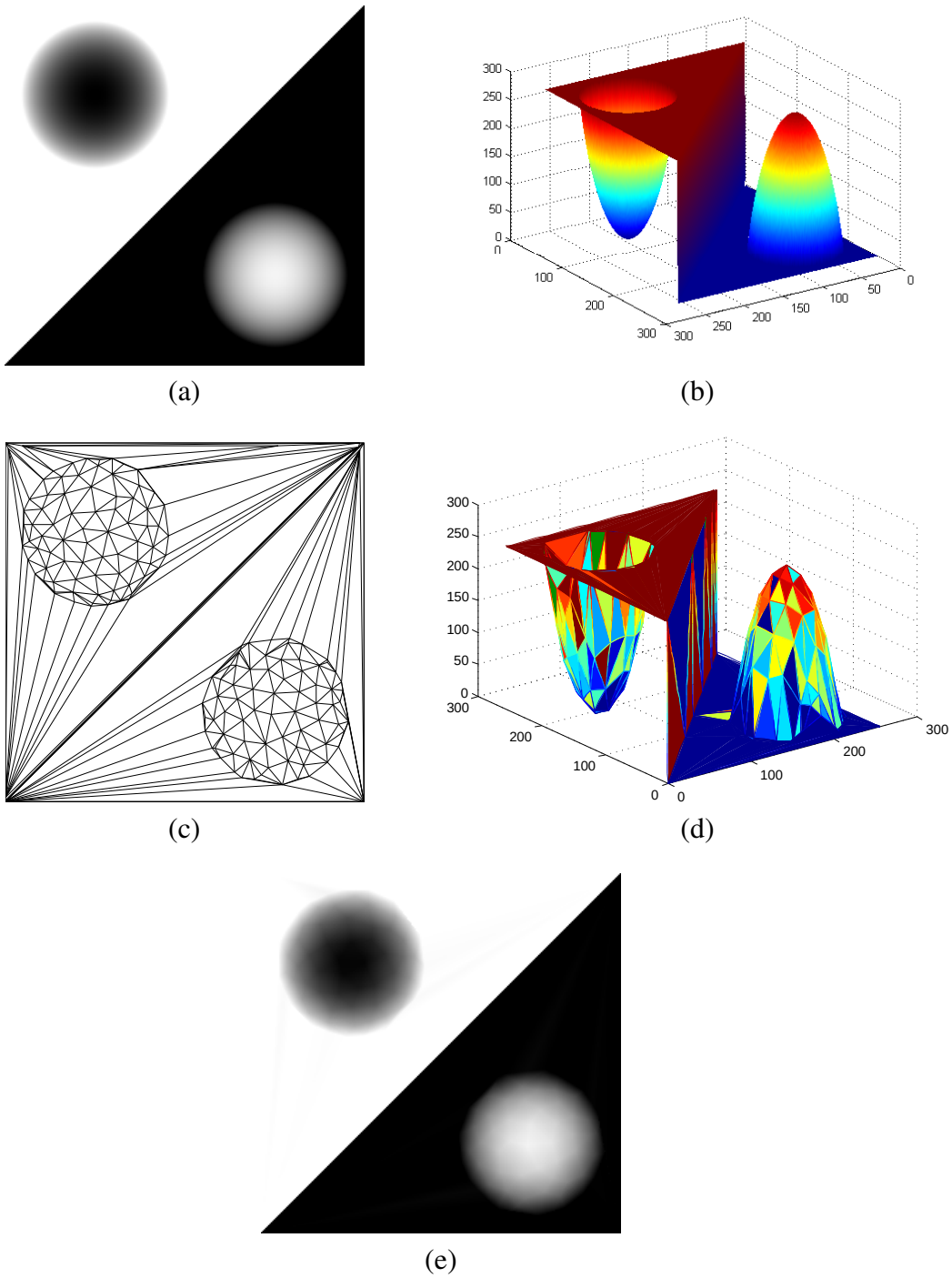


Figure 2.7: Mesh model of an image. (a) Original image, (b) image modelled as surface, (c) triangulation of image domain, (d) resulting triangle mesh , and (e) reconstructed image.

of a triangulation T as previously defined (in Section 2.4 on page 12). A triangle mesh model is completely characterized by a triangulation T covering the image domain Λ as

well as the values of ϕ for each point $p \in \mathcal{V}(T)$. We refer to each element of $\mathcal{V}(T)$ as a **sample point**. Given T, a function $\hat{\phi}_T$ that interpolates ϕ at the points in $\mathcal{V}(T)$ is constructed as follows. First, we form a continuous piecewise-linear function $\tilde{\phi}_T$. For each (triangle) face $f \in \mathcal{F}(T)$, $\tilde{\phi}_T$ is chosen as the unique linear function that interpolates ϕ at the three vertices of f. To ensure that $\hat{\phi}_T$ is integer valued (just like ϕ), we choose $\hat{\phi}_T$ as $\hat{\phi}_T(p) = \text{round}(\tilde{\phi}_T(p))$ for all $p \in \Lambda$, where round denotes an operator that rounds to an integer value. The set $\mathcal{V}(T)$ must always include the extreme convex-hull points of Λ (i.e., the four corners of the image bounding box) so that the triangulation of $\mathcal{V}(T)$ covers all points in Λ. As a matter of terminology, the **size** and **sampling density** of the mesh model T are defined as $|\mathcal{V}(T)|$ (i.e., the number of vertices in T) and $|\mathcal{V}(T)| \,/\, |\Lambda|$, respectively.

With the mesh model introduced, the process to generate such models is known as mesh generation. The mesh-generation problem that we address in this thesis can be succinctly stated as follows: Given ϕ and a desired number N of sample points, find the mesh model T of ϕ with $|\mathcal{V}(T)| = N$ that minimizes the measure ϵ_T of the difference between ϕ and the approximation $\hat{\phi}_T$. In our work, the **mean squared error (MSE)** is used as the error measure, so that

$$\epsilon_T = |\Lambda|^{-1} \sum_{p \in \Lambda} \left(\hat{\phi}_T(p) - \phi(p) \right)^2 . \tag{2.2}$$

Herein, the MSE is typically expressed in terms of the **peak signal-to-noise ratio (PSNR)**, which is defined as $\text{PSNR} = 20 \log_{10} \left(\frac{2^\rho - 1}{\sqrt{\epsilon}} \right)$, where ρ is the number of bits/sample in the image ϕ. Essentially, the PSNR measures the MSE relative to the dynamic range of the data.

2.6 Grid-Point to Face Mapping

In the process of mesh generation, we usually have an image defined on a rectangular grid and a triangulation superimposed on it, as shown in Figure 2.8(a), with "·" represents grid point. For reasons that will become clear later, we need a convenient scheme to define map from grid point to face. A slightly modified scheme from Fleischer [55], is introduced below.

Suppose we have an image ϕ defined on a rectangular grid, and a triangulation T of image domain, the grid-point to face mapping strategy assign each grid point p to exactly one face in T as follows:

1. If p is strictly inside a face f, map p to the face f;

2. If p is on a horizontal edge e excluding its endpoints, map p to the face below e unless there is no face below, in which case p is mapped to the face above e;

3. If p belongs to a non-horizontal edge e excluding its endpoints, map p to the face to the left of e unless there is no face to the left of e, in which case p is mapped to the face to the right of e;

4. If p is the right endpoint of a horizontal edge e, map p to the same face to which e belongs;

5. If p is a vertex in T but not a right endpoint of a horizontal edge e, map p to the face to the left of e;

6. If p is the top-left or bottom-left corner vertices of T, map it to the same face to which the top or bottom horizontal edge belong, respectively.

Figure 2.8 is provided to illustrate the above grid-point to face mapping scheme. Figure 2.8(a) shows an image ϕ defined on a rectangular grid $\{0, 1, \ldots, 15\} \times \{0, 1, \ldots, 9\}$, and a triangulation T superimposed on the grid including the four extreme points of the image. In order to understand the above mapping strategy more clearly, the grid points in Figure 2.8(b) are all marked with different symbols. The grid points belong to each face share the same symbol. Examples will be given for each type of grid points listed above.

Consider the point $p_1 = (2, 5)$. The point p_1 is strictly inside face f_4. Therefore, according to case 1 above, p_1 is mapped to face f_4. Now consider points $p_2 = (8, 9)$ and $p_3 = (8, 1)$, which are both on horizontal edges. According to case 2, p_2 is mapped to face f_2 (i.e. the face below edge v_4v_3), while p_3 is mapped to the face f_2, since there is no face below edge v_1v_2. For points $p_4 = (13, 7)$ and $p_5 = (0, 5)$ that belong to non-horizontal edges v_3v_6 and v_0v_1, respectively. According to case 3, p_4 will be mapped to f_3 as f_3 is the face to the left of v_3v_6. On the other hand, p_5 will be mapped to face f_4, which is to the right of v_0v_1, because there is no face to the left of v_0v_1. Case 4 and 5 map the vertices in T. Consider points $p_6 = (4, 5)$ and $p_7 = (15, 0)$ for instance. p_6 is mapped to f_4, which is the face to the left of edge v_0v_5, and p_7 is mapped to f_7, the same face to which edge v_1v_2 belongs. Case 6 applies to the top-left vertex and bottom-left vertex in the triangulation. $p_8 = (0, 9)$, and $p_9 = (0, 0)$ are mapped to face f_1 and f_7, respectively.

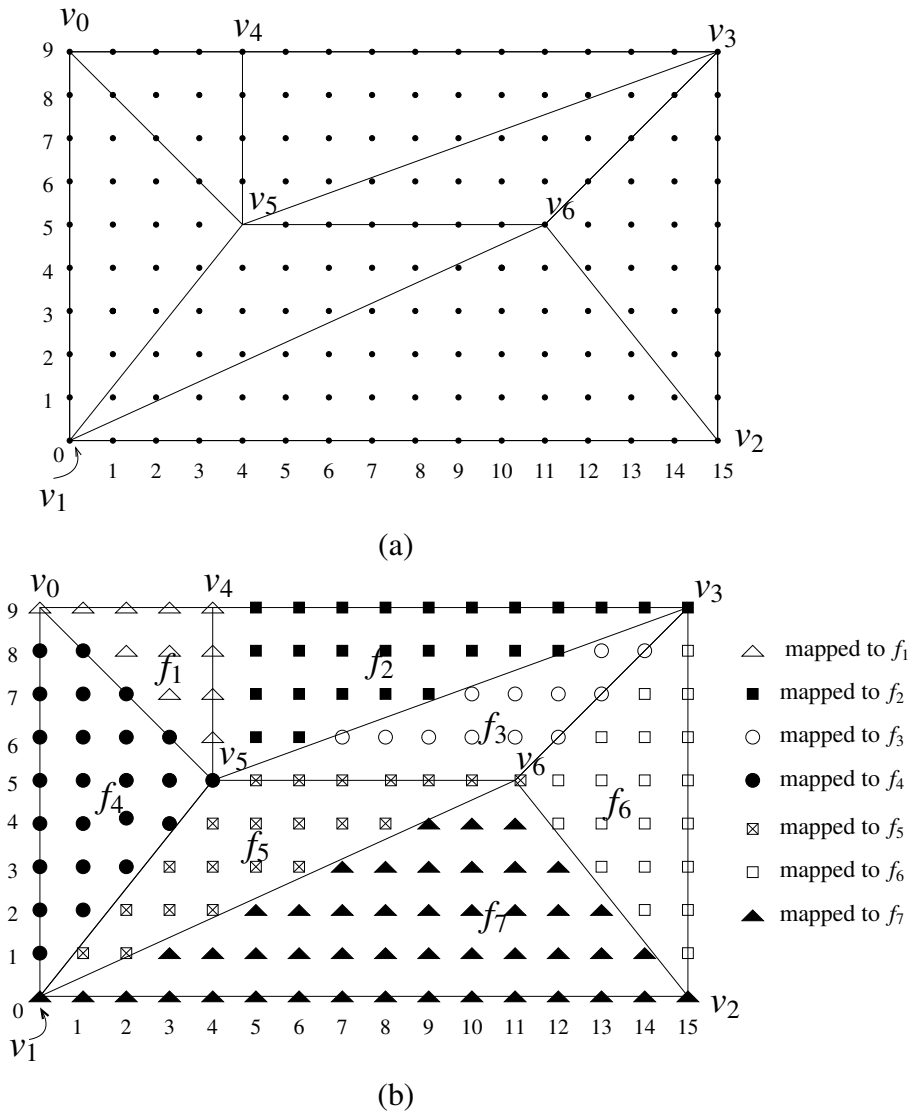


Figure 2.8: An example of how points are assigned to only one face. (a) A triangulation on a rectangular grid, and (b) a triangulation with only vertices and points on edges shown.

Chapter 3

Proposed Mesh-Generation Framework and Method

3.1 Overview

In this chapter, we propose a flexible DDT-based mesh-generation framework and a new highly-effective mesh-generation method derived from this framework. We begin with the introduction of Lawson's local optimization procedure (LOP). Then, our new mesh-generation framework is presented, which has several free parameters that must be chosen in order to produce a fully-specified mesh-generation method. Next, we present a summary of our analysis, which leads to the recommendation of a particular choice for each parameter. Finally, we propose a specific mesh-generation method, based on these recommended choices. The performance of our proposed mesh-generation method is evaluated by comparing to several competing schemes, with our method proving to be superior.

3.2 Local Optimization Procedure (LOP)

As mentioned earlier in Section 1.2 on page 5, the mesh-generation framework proposed in this thesis was inspired by the frameworks of Rippa [21] (known by the name "COMPRESS" therein) and Garland and Heckbert [17] (known as "Algorithm IV" therein). Both the Rippa and Garland-Heckbert frameworks employ the well-known **local optimization procedure (LOP)** [29] of Lawson. Our proposed framework also utilizes a variant of the LOP. Since knowledge of the LOP is essential to the understanding of our framework, we will first describe the LOP below.

The fact that every triangulation is reachable from every other triangulation via edge flips [52, 53] motivated Lawson to propose the so called LOP [29], an algorithm for finding an optimal triangulation of a set of points via edge flips. (Recall that edge flips were discussed in Section 2.4 on page 14.) To cast the triangulation problem as an optimization, we define a rule, called an **edge-flip criterion**, that determines, for a flippable edge e, if the triangulation with the edge e is preferred over the triangulation obtained if e were transformed to e' by an edge flip. The edge-flip criterion is specified as a binary-valued decision function, denoted isPreferred, where isPreferred(e) is one if e is preferred to e', and zero otherwise. As a matter of terminology, an edge e is said to be **optimal** if:

1. it is not flippable; or
2. it is flippable and isPreferred(e) = 1.

A triangulation is said to be (locally) **optimal** if each one of its edges is optimal. An edge whose optimality is uncertain is said to be **suspect**. Note that, by definition, an edge that is not flippable cannot be suspect. In short, the LOP tests any suspect edges for optimality, and performs edge flips to eliminate any suspect edge that is not optimal. Letting S denote the current set of suspect edges, the variant of the LOP used herein consists of the following steps:

1. Initialize S to contain all suspect edges (i.e., edges that are flippable but whose optimality has not yet been tested).
2. If $|S| = 0$ (i.e., S is empty), then stop.
3. Remove an edge e from S.
4. If e is not flippable or e has been visited more than 5 times since the LOP started, go to step 2.
5. Let q denote the (strictly convex) quadrilateral formed by the union of the two faces incident on e, and let e' denote the edge obtained by flipping e (i.e., the other diagonal of q). If isPreferred(e) = 0 (i.e., e is not optimal according to whatever edge-flip criterion is in effect), apply an edge flip to e (to produce e'), and add to S any newly suspect edges resulting from the edge flip. Which edges become newly suspect as a result of the edge flip depend on the specific choice of isPreferred, and will be addressed in more detail later in Section 3.3.2, after we have introduced the various edge-flip criteria considered herein. In practice, however, these edges are in a small neighbourhood about e.

6. Go to step 2.

In passing, we note that the LOP is only guaranteed to produce a locally (as opposed to globally) optimal triangulation, and the locally optimal triangulation produced depends on the order in which edges are flipped. The algorithm presented above differs slightly from the LOP as proposed by Lawson. In particular, a limit is placed on the number of times that an edge can be tested for optimality during the LOP process via the second condition appearing in step 4. This extra condition is necessary in order to ensure that the algorithm does not become trapped in a cycle, repeating the same sequence of edge flips indefinitely. In the context of our work, cycles can arise for two reasons. First, we allow for the use of edge-flip criteria that are not guaranteed to always be well behaved, and when such criteria are used, cycles can sometimes occur. Second, due to the effects of finite-precision arithmetic (i.e., roundoff error), decisions regarding the optimality of an edge can occasionally be made in an inconsistent manner, leading to cycles. A more detailed treatment of the cycle problem will be presented in Section 3.4.4 after we have introduced the edge flip criteria used in our work.

3.3 Proposed Mesh-Generation Framework

Having introduced the (slightly modified) LOP used in our work, we are now ready to present our proposed mesh-generation framework. With our framework, for a given triangulation T, each point in the image domain Λ is assigned to *exactly one* face in T. This is achieved by applying the grid-point to face mapping scheme described earlier in Section 2.6. The set of all points in Λ belonging to the face f in T is denoted $\mathcal{P}_T(f)$.

As input, our framework takes an image ϕ (defined on Λ) and a target number N of sample points for the mesh to be generated. Our proposed framework is iterative in nature. It starts with a nearly empty mesh and adds points to the mesh until the desired sampling density is achieved. Let T denote the triangulation in the current iteration. $\tilde{\phi}_T$ is the unique linear interpolant of ϕ and $\hat{\phi}_T$ is the integer valued interpolant of ϕ defined before in Section 2.5. $\mathcal{V}(T)$, $\mathcal{E}(T)$, and $\mathcal{F}(T)$ denote the vertices, edges and faces of T as previously defined on page 12. Our framework then consists of the following steps (in order):

1. **Initial triangulation.** Initially choose the triangulation T as a triangulation of the extreme convex-hull points of the image domain Λ (i.e., the four corner points of the image bounding box).

2. **Initial connectivity adjustment.** Adjust the connectivity of the triangulation by applying the LOP (described earlier) choosing the edge-flip criterion isPreferred as isPreferred = $\text{isPreferred}_{\text{main}}$, where $\text{isPreferred}_{\text{main}}$ is a free parameter of our framework. Initially, when the LOP is invoked, all flippable edges in T are marked as suspect.

3. If the target number of sample points has been reached (i.e., $|\mathcal{V}(T)| \geq N$), go to step 8.

4. **Point selection.** Select a new point $p^* \in \Lambda \setminus \mathcal{V}(T)$ to add to the triangulation T. This is accomplished in two steps. First, select a face f^* in T into which a new point is to be inserted, as given by

$$f^* = \text{selFace},$$

where selFace is a function (implicitly depending on T) that embodies the face-selection process and is a free parameter of our framework. Second, having chosen the face f^*, select a candidate point p^* belonging to f^* and not currently in T (i.e., $p^* \in \mathcal{P}_T(f^*) \setminus \mathcal{V}(T)$) for insertion, as given by

$$p^* = \text{selCand}(f^*), \tag{3.1}$$

where selCand is a function that embodies the candidate-selection process and is a free parameter of our framework.

5. **Point insertion.** Insert p^* into the triangulation T. If p^* is strictly inside a face in T, say face $v_i v_j v_k$, we connect p^* by new edges to each vertex of its containing face, as shown in Figure 3.1(a). If p^* is on an edge in T, say on edge $\overline{v_i v_k}$, we split this edge at p^* and, for each face f incident on $\overline{v_i v_k}$, we connect p^* with a new edge to the vertex in f that is opposite the edge $\overline{v_i v_k}$, as shown in Figure 3.1(b).

6. **Main connectivity adjustment.** Adjust the connectivity of the triangulation by applying the LOP, with the edge-flip criterion isPreferred chosen as isPreferred = $\text{isPreferred}_{\text{main}}$ and the suspect edges initially chosen as all edges whose optimality could have been changed by the insertion of p^* in the previous step (i.e., step 5). The edges whose optimality can be affected by the insertion of p^* depend on the particular choice of $\text{isPreferred}_{\text{main}}$, and will be discussed in more detail later in Sec-

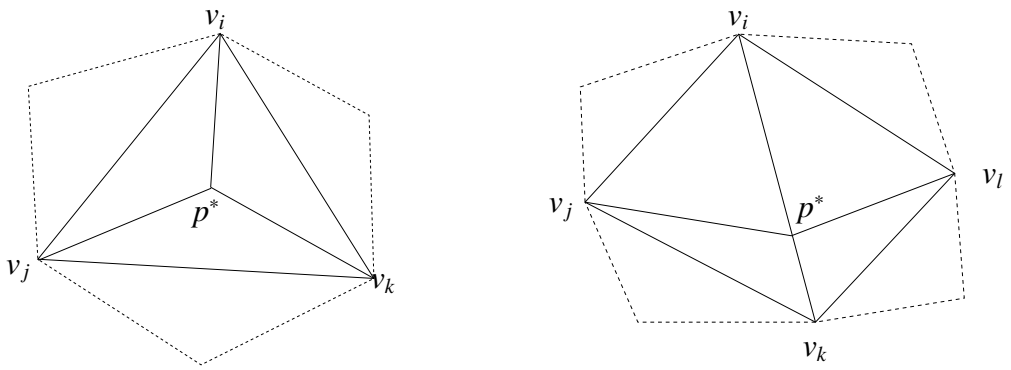


Figure 3.1: Point insertion example. Part of a triangulation showing how the new vertex p^* is inserted (a) inside a triangle $v_i v_j v_k$ and (b) on an edge $v_i v_k$.

tion 3.3.2, after we have introduced the various edge-flip criteria considered herein. In practice, these edges are in a relatively small neighbourhood about p^*.

7. Go to step 3.

8. **Final connectivity adjustment (optional).** If the optional final connectivity-adjustment step is enabled, continue; otherwise, stop. Adjust the connectivity of the triangulation by applying the LOP, with the edge-flip criterion isPreferred chosen as isPreferred = $\text{isPreferred}_{\text{final}}$, where $\text{isPreferred}_{\text{final}}$ is a free parameter of our framework. Initially, when the LOP is invoked, all flippable edges in the triangulation are marked as suspect.

From above, one can see that our framework requires the choice of several free parameters in order to arrive at a completely-specified method for mesh generation. In particular, we must choose:

1. a point-selection strategy that consists of a face-selection policy selFace together with a candidate-selection policy selCand;

2. an edge-flip criterion $\text{isPreferred}_{\text{main}}$ to be used in the initial and main connectivity adjustment, called the **main edge-flip criterion**;

3. a choice of whether to perform the optional final connectivity adjustment;

4. if final connectivity adjustment is to be performed, an edge-flip criterion $\text{isPreferred}_{\text{final}}$ to be used for this purpose, called the **final edge-flip criterion**.

Note that it is the intention of our framework that $\text{isPreferred}_{\text{main}}$ and $\text{isPreferred}_{\text{final}}$ be chosen differently. As for how the above parameters might be chosen, we defer this discussion until later.

Although, as mentioned above, our proposed framework was inspired by the frameworks of Rippa [21] and Garland and Heckbert [17], a number of key differences exist between our framework and these other ones. In particular, the two most fundamental differences are as follows. First, in the Rippa and Garland-Heckbert frameworks, the point-selection process (corresponding to step 4 in our framework) is not *logically* viewed as being split into two smaller steps, with the first choosing a face and the second choosing a point within the face. Both approaches always choose p^* as the point $p \in \Lambda \setminus \mathcal{V}(T)$ for which $\left|\hat{\phi}_T(p) - \phi(p)\right|$ is greatest. That is, the Rippa and Garland-Heckbert frameworks essentially replace (3.1) in our framework with the fixed choice

$$p^* = \underset{p \in \Lambda \setminus \mathcal{V}(T)}{\arg\max} \left|\hat{\phi}_T(p) - \phi(p)\right|, \tag{3.2}$$

in which case there is no explicit face-selection process (selFace) per se. As we shall see later, the point-selection scheme given by (3.2) leaves much to be desired. The second key difference is that neither the Rippa nor Garland-Heckbert framework has an equivalent to the final connectivity-adjustment step (i.e., step 8) in our framework. As we shall see later, final connectivity adjustment plays a crucial role in allowing highly effective mesh-generation methods to be synthesized.

In passing, we note that mesh-generation methods derived from our proposed framework (as well as the Rippa and Garland-Heckbert frameworks) have a number of desirable characteristics. In particular, because our framework is based strictly on the refinement of an initial coarse mesh (with four vertices), the current mesh size never exceeds the target mesh size (i.e., N vertices) at any point during mesh generation. This is important in order to minimize memory usage (and often computational complexity as well). In contrast, frameworks or methods based on mesh simplification typically have a much greater peak mesh size. Another desirable characteristic of our proposed framework is that it can easily accommodate a stopping criterion for the mesh-generation process that is based on either sampling density or a prescribed error tolerance (in step 3 of our framework).

As seen above, our proposed framework requires the choice of several free parameters in order to arrive at a completely-specified method for mesh generation. For example, we must specify face-selection and candidate-selection policies as well as various edge-flip criteria. So far, we have not made any suggestions as to how these items might be chosen.

In what follows, we introduce a variety of possible choices for these items as considered in our work.

3.3.1 Face- and Candidate-Selection Policies

In step 4 of our framework (i.e., point selection), we must choose a face f^* in which to select a new point for insertion. This face-selection policy is embodied by the function selFace. In our work, we considered two face-selection policies.

The first policy, called **greatest absolute error (GAE)**, chooses selFace as

$$\text{selFace}_{\mathsf{GAE}} = \operatorname*{arg\,max}_{f \in U(T)} \max_{p \in \mathcal{P}_T(f)} \left| \hat{\phi}_T(p) - \phi(p) \right|,$$

where $U(T)$ is the set of all faces $f \in \mathcal{F}(T)$ such that $\mathcal{P}_T(f) \neq \emptyset$. That is, selFace is defined to choose the face in T that contains the point where the original image and approximation differ most.

The second policy, called **greatest squared error (GSE)**, selects selFace as

$$\text{selFace}_{\mathsf{GSE}} = \operatorname*{arg\,max}_{f \in U(T)} \sum_{p \in \mathcal{P}_T(f)} \left(\hat{\phi}_T(p) - \phi(p) \right)^2,$$

where $U(T)$ is the set of all faces $f \in \mathcal{F}(T)$ such that $\mathcal{P}_T(f) \neq \emptyset$. That is, selFace is defined to choose the face in T with the largest squared error.

In step 4 of our framework (i.e., point selection), once a face f^* has been chosen, we must select a candidate point p^* within that face for insertion. This candidate-selection policy is embodied by the function selCand. In our work, we considered five candidate-selection policies.

The first policy, called **peak absolute error (PAE)**, selects selCand as

$$\text{selCand}_{\mathsf{PAE}}(f) = \operatorname*{arg\,max}_{p \in \mathcal{P}_T(f) \setminus \mathcal{V}(T)} \left| \hat{\phi}_T(p) - \phi(p) \right|.$$

That is, of all candidate points in the face, this policy selects the point at which the absolute error is greatest.

The second policy, called **peak weighted absolute error (PWAE)**, selects selCand as

$$\text{selCand}_{\mathsf{PWAE}}(f) = \operatorname*{arg\,max}_{p \in \mathcal{P}_T(f) \setminus \mathcal{V}(T)} w(p) \left| \hat{\phi}_T(p) - \phi(p) \right|,$$

where $w(p)$ denotes the MMSODD (a second-order derivative defined in Section 2.3) of a given image ϕ. That is, of all candidate points in the face, this policy selects the point at which the weighted absolute error is greatest. Note the MMSODD of an image ϕ can be computed using (2.1) at each grid point. The partial-derivative operator in (2.1) are formed from the tensor product of one-dimensional derivative operators, where the discrete-time approximations of the one-dimensional first-order and second-order derivative operator are computed using the filters with transfer functions $\frac{1}{2}z - \frac{1}{2}z^{-1}$ and $z - 2 + z^{-1}$, respectively. Furthermore, binomial filters with order of nine are employed as smoothing operator for image ϕ before the derivative computation.

The third policy, called **approximate minimum squared error based on PAE (AMSE-PAE)**, chooses p^* in two steps. First, we choose a set Ω of test points to consider as candidates for insertion. We then choose p^* from Ω. If $|\mathcal{P}_T(f^*) \setminus \mathcal{V}(T)| > 8$, Ω is chosen as the eight points p in $\mathcal{P}_T(f^*) \setminus \mathcal{V}(T)$ for which $\left|\hat{\phi}_T(p) - \phi(p)\right|$ is greatest; otherwise, we choose $\Omega = \mathcal{P}_T(f^*) \setminus \mathcal{V}(T)$. (As an aside, we note that the value of eight here was chosen based on experimentation.) Given Ω, we then choose selCand as

$$\text{selCand}_{\text{AMSE-PAE}}(f) = \arg\min_{t \in \Omega} \sum_{p \in \mathcal{P}_T(f)} \left(\hat{\phi}_{\Upsilon(t)}(p) - \phi(p)\right)^2,$$

where $\Upsilon(t)$ denotes the triangulation that would be obtained if the point t were inserted into T. That is, selCand is defined to select the point in Ω whose insertion would result in the least squared error over the face f^*.

The fourth policy, called **approximate minimum squared error based on PWAE (AMSE-PWAE)**, chooses p^* in an identical manner as AMSE-PAE, except that weighted absolute error instead of absolute error is used when choose the set Ω of test points. That is, if $\mathcal{P}_T(f^*) \setminus \mathcal{V}(T) > 8$, Ω is chosen as the eight points p in $\mathcal{P}_T(f^*) \setminus \mathcal{V}(T)$ for which $w(p)\left|\hat{\phi}_T(p) - \phi(p)\right|$ is greatest; otherwise, we choose $\Omega = \mathcal{P}_T(f^*) \setminus \mathcal{V}(T)$. Then, selCand is chosen as

$$\text{selCand}_{\text{AMSE-PWAE}}(f) = \arg\min_{t \in \Omega} \sum_{p \in \mathcal{P}_T(f)} \left(\hat{\phi}_{\Upsilon(t)}(p) - \phi(p)\right)^2,$$

The fifth policy, called **hybrid**, simply employs the PAE policy until the number of sample points in the mesh reaches 25% of the desired number, with the AMSE-PAE policy being used thereafter. This policy is motivated largely by the desire to save computation. By using the less-computationally costly PAE policy initially, computational cost can be significantly reduced (relative to the AMSE-PAE policy).

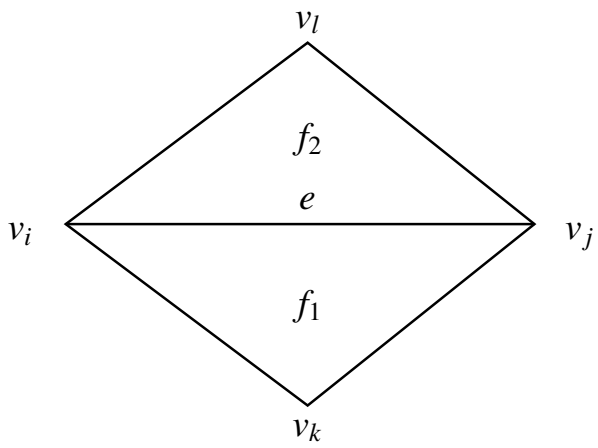


Figure 3.2: An edge $e = \overline{v_i v_j}$ that is a diagonal of the quadrilateral $v_i v_k v_j v_l$ in the triangulation T.

Lastly, we note that choosing the face-selection and candidate-selection policies as GAE and PAE, respectively, is mathematically equivalent to the point-selection scheme used in the Rippa and Garland-Heckbert frameworks, given earlier by (3.2).

3.3.2 Edge-Flip Criteria

In our framework, steps 2, 6, and 8 each employ the LOP, and the LOP requires the specification of an edge-flip criterion. So far, we have not commented on how this edge-flip criterion might be chosen. In what follows, we introduce twelve edge-flip criteria that were considered in our work. Herein, we employ two different classes of edge-flip criterion. Both are based on the idea of assigning costs to edges, and then making the decision of whether to flip an edge based on these costs. The main difference between these two classes is in how they use these edge-cost functions in order to make decisions.

Before we can introduce any of the edge-flip criteria, we must first introduce the edge-cost functions that these criteria employ. Each edge-cost function assigns a cost to an edge e in the triangulation T. Let $v_n = (x_n, y_n)$ denote the nth vertex in T and let $z_n = \phi(x_n, y_n)$ be the corresponding sample value. Furthermore, let $e = \overline{v_i v_j}$. In what follows, in the case that e is not a boundary edge (and must therefore have two incident faces), its two incident faces are denoted as f_1 and f_2, where f_1 is triangle $v_i v_k v_j$ and f_2 is triangle $v_i v_j v_l$. The preceding definitions are illustrated in Figure 3.2. Furthermore, let P_1 and P_2 respectively denote the

linear interpolant over f_1 and f_2 (i.e., $\tilde{\phi}_T$ restricted to f_1 and f_2), where

$$P_1(x, y) = a_1 x + b_1 y + c_1, \text{ and}$$

$$P_2(x, y) = a_2 x + b_2 y + c_2.$$

Lastly, we introduce a few additional definitions needed in what follows. The **approximate diameter** of the face f, denoted $\text{diam}(f)$, is defined as the length of the longest side of the (smallest axis-aligned) bounding box of the face f. The **shape quality** of a face f, denoted $\text{sq}(f)$, is defined as

$$\text{sq}(f) = \text{area}(f)/\,\text{diam}(f),$$

where $\text{area}(f)$ denotes the area of the face f.

The first class of edge-flip criterion is associated with edge-cost functions that assign a cost to *every* edge in the triangulation T. This class makes use of the following seven edge-cost functions:

1. $\text{edgeCost}_{\text{ABN}}$, the **angle between normals (ABN)** from [22];
2. $\text{edgeCost}_{\text{JND}}$, the **jump in normal derivatives (JND)** from [22];
3. $\text{edgeCost}_{\text{DLP}}$, the **deviations from linear polynomials (DLP)** from [22];
4. $\text{edgeCost}_{\text{DP}}$, the **distances from planes (DP)** from [22];
5. $\text{edgeCost}_{\text{YMS}}$, the **Yu-Morse-Sederberg (YMS)** cost from [27];
6. $\text{edgeCost}_{\text{ELABN}}$, the **edge-length-weighted ABN (ELABN)** from [25, 26];
7. $\text{edgeCost}_{\text{ELJND}}$, the **edge-length-weighted JND (ELJND)** which is newly proposed herein.

The definitions of these functions are as follows. For a nonboundary edge e (which must have two incident faces) in the triangulation T, we define

$$\text{edgeCost}_{\text{ABN}}(T, e) = \arccos\left[\frac{(a_1, b_1, -1) \cdot (a_2, b_2, -1)}{\|(a_1, b_1, -1)\|\,\|(a_2, b_2, -1)\|}\right], \tag{3.3a}$$

$$\text{edgeCost}_{\text{JND}}(T, e) = \left|(n_x, n_y) \cdot [(a_1, b_1) - (a_2, b_2)]\right|, \tag{3.3b}$$

$$\text{edgeCost}_{\text{DLP}}(T, e) = \|(|P_1(x_l, y_l) - z_l|\,, |P_2(x_k, y_k) - z_k|)\|\,, \tag{3.3c}$$

$$\text{edgeCost}_{\text{DP}}(T,e) = \|(\text{dist}(P_1,(x_l,y_l,z_l)),\text{dist}(P_2,(x_k,y_k,z_k)))\|\,, \tag{3.3d}$$

$$\text{edgeCost}_{\text{ELABN}}(T,e) = \left\|v_i - v_j\right\| \text{edgeCost}_{\text{ABN}}(T,e), \tag{3.3e}$$

$$\text{edgeCost}_{\text{YMS}}(T,e) = \|(a_1,b_1)\|\,\|(a_2,b_2)\| - (a_1,b_1)\cdot(a_2,b_2), \quad \text{and} \tag{3.3f}$$

$$\text{edgeCost}_{\text{ELJND}}(T,e) = \left\|v_i - v_j\right\| \text{edgeCost}_{\text{JND}}(T,e), \tag{3.3g}$$

where $(n_{\text{x}}, n_{\text{y}})$ is a unit vector normal to e and

$$\text{dist}(P_\alpha,(x,y,z)) = \frac{|P_\alpha(x,y) - z|}{\|(a_\alpha,b_\alpha,-1)\|}.$$

For a boundary edge e (which has only one incident face), we simply define

$$\text{edgeCost}_{\text{ABN}}(T,e) = 0,$$

$$\text{edgeCost}_{\text{JND}}(T,e) = 0,$$

$$\text{edgeCost}_{\text{DLP}}(T,e) = 0,$$

$$\text{edgeCost}_{\text{DP}}(T,e) = 0,$$

$$\text{edgeCost}_{\text{YMS}}(T,e) = 0,$$

$$\text{edgeCost}_{\text{ELABN}}(T,e) = 0, \quad \text{and}$$

$$\text{edgeCost}_{\text{ELJND}}(T,e) = 0.$$

The second class of edge-flip criterion is associated with edge-cost functions that assign a cost to every *flippable* edge in the triangulation. This class makes use of the following five edge-cost functions:

1. $\text{edgeCost}_{\text{D}}$, the (preferred-directions) **Delaunay (D)** cost [42];

2. $\text{edgeCost}_{\text{SE}}$, the **squared error (SE)** from [24];

3. $\text{edgeCost}_{\text{GHH}}$, the **GH hybrid (GHH)** from [17];

4. $\text{edgeCost}_{\text{SQSE}}$, the **shape-quality-weighted SE (SQSE)**, which is newly proposed herein; and

5. $\text{edgeCost}_{\text{JNDSE}}$, the **JND-weighted SE (JNDSE)**, which is newly proposed herein.

The preceding functions are defined as follows. The first function $\text{edgeCost}_{\mathsf{D}}(T, e)$ is defined to be 0 if e passes the preferred-directions-augmented in-circle (Delaunay) test in [42], and 1 otherwise. The remaining functions are then given by

$$\text{edgeCost}_{\mathsf{SE}}(T, e) = \beta(T, e),$$

$$\text{edgeCost}_{\mathsf{GHH}}(T, e) = \begin{cases} [\text{sq}(f_1)\,\text{sq}(f_2)]^{-1} & \tau \leq \frac{1}{2} \\ \beta(T, e) & \text{otherwise,} \end{cases}$$

$$\text{edgeCost}_{\mathsf{SQSE}}(T, e) = [\text{sq}(f_1)\,\text{sq}(f_2)]^{-1}\,\beta(T, e), \quad \text{and}$$

$$\text{edgeCost}_{\mathsf{JNDSE}}(T, e) = \text{edgeCost}_{\mathsf{JND}}(T, e)\beta(T, e), \quad \text{where}$$

$$\beta(T, e) = \sum_{p \in \mathcal{P}_T(f_1) \cup \mathcal{P}_T(f_2)} \left(\hat{\phi}_T(p) - \phi(p)\right)^2,$$

$$\tau = \frac{\min\{\sigma, \sigma'\}}{\max\{\sigma, \sigma'\}}, \quad \sigma = \text{sq}(f_1)\,\text{sq}(f_2), \quad \sigma' = \text{sq}(f_1')\,\text{sq}(f_2'),$$

and f_1' and f_2' are the two faces incident on the edge e', with e' denoting the edge obtained by applying an edge flip to e.

With the above edge-cost functions having been defined, we are now ready to introduce the twelve edge-flip criteria considered in our work, known by the names ABN, JND, DLP, DP, YMS, ELABN, ELJND, Delaunay, SE, GHH, SQSE, and JNDSE. Let e denote the edge to be tested for optimality in the triangulation T, where $e = \overline{v_i v_j}$. Let e' denote the new edge obtained by applying an edge-flip transformation to e, and let T' be the triangulation obtained if e is flipped. As mentioned above, we employ two different classes of edge-flip criterion in our work. We consider each in turn below.

The first class of edge-clip criterion assigns a cost to each edge in triangulation, and then assigns a cost to the triangulation, which corresponds to the sum of the edge costs. The edge e is then preferred (over e') if the cost of triangulation T does not exceed the cost of triangulation T'. That is, the first class of edge-flip criterion chooses isPreferred to be of the form

$$\text{isPreferred}(e) = \begin{cases} 1 & \text{triCost}(T) \leq \text{triCost}(T') \\ 0 & \text{otherwise,} \end{cases} \quad \text{where} \tag{3.4a}$$

$$\text{triCost}(T) = \sum_{e \in \mathcal{E}(T)} \text{edgeCost}(T, e). \tag{3.4b}$$

The **ABN, JND, DLP, DP, YMS, ELABN, and ELJND edge-flip criteria** are obtained by

choosing isPreferred as given in (3.4) with edgeCost selected as $\text{edgeCost}_{\text{ABN}}$, $\text{edgeCost}_{\text{JND}}$, $\text{edgeCost}_{\text{DLP}}$, $\text{edgeCost}_{\text{DP}}$, $\text{edgeCost}_{\text{YMS}}$, $\text{edgeCost}_{\text{ELABN}}$, and $\text{edgeCost}_{\text{ELJND}}$, respectively. Note that, although the summation in (3.4b) is taken over all edges in the triangulation, only a small number of terms differ between $\text{triCost}(T)$ and $\text{triCost}(T')$ for all of the edge-cost functions introduced above. So, as a practical matter, in order to compare $\text{triCost}(T)$ and $\text{triCost}(T')$, it is only necessary to compute the relatively small number of terms that can actually differ between the two summations.

The second class of edge-flip criterion simply uses the edge-cost function directly in order to decide whether to flip an edge. If the edge e has a strictly higher cost then its flipped counterpart e', the decision is made to flip the edge. That is, the second class of edge-flip criterion chooses isPreferred to be of the form

$$\text{isPreferred}(e) = \begin{cases} 1 & \text{edgeCost}(T, e) \leq \text{edgeCost}(T', e') \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

The **Delaunay, SE, GHH, SQSE, JNDSE edge-flip criteria** are obtained by choosing isPreferred as given in (3.5) with edgeCost selected as $\text{edgeCost}_{\text{D}}$, $\text{edgeCost}_{\text{SE}}$, $\text{edgeCost}_{\text{GHH}}$, $\text{edgeCost}_{\text{SQSE}}$, and $\text{edgeCost}_{\text{JNDSE}}$, respectively. Observe that, the edge-flip criteria JNDSE and SQSE, which are newly proposed herein, employ an underlying edge-cost function that weighs the squared error by some measure related to triangle shape (namely, shape quality or JND). Also, note that the use of the Delaunay edge-flip criterion in the LOP produces the unique preferred-directions Delaunay triangulation.

At this point, we would like to make one brief but important comment regarding the SE edge-flip criterion. From its definition above, observe that the SE edge-flip criterion will only choose to flip an edge if the squared error, as defined in (2.2), is *strictly* reduced by the edge flip. Since the objective herein is to minimize this squared error, the reader might have the initial suspicion that consistently using the SE criterion everywhere in our framework must trivially lead to the best results (i.e., the lowest squared error). As we shall see later, however, this suspicion is, in fact, wrong.

Determination of suspect edges. Earlier, in the discussion of our mesh-generation framework and the LOP, the question arose as to which edges can become suspect when a new point is inserted in the triangulation or an edge is flipped. Although a general and unavoidably vague answer to this question was given at that time, we are now in a position to provide a much more precise answer to this question, which we do in what follows.

Recall that, in step 6 of our proposed mesh-generation framework (i.e., main connec-

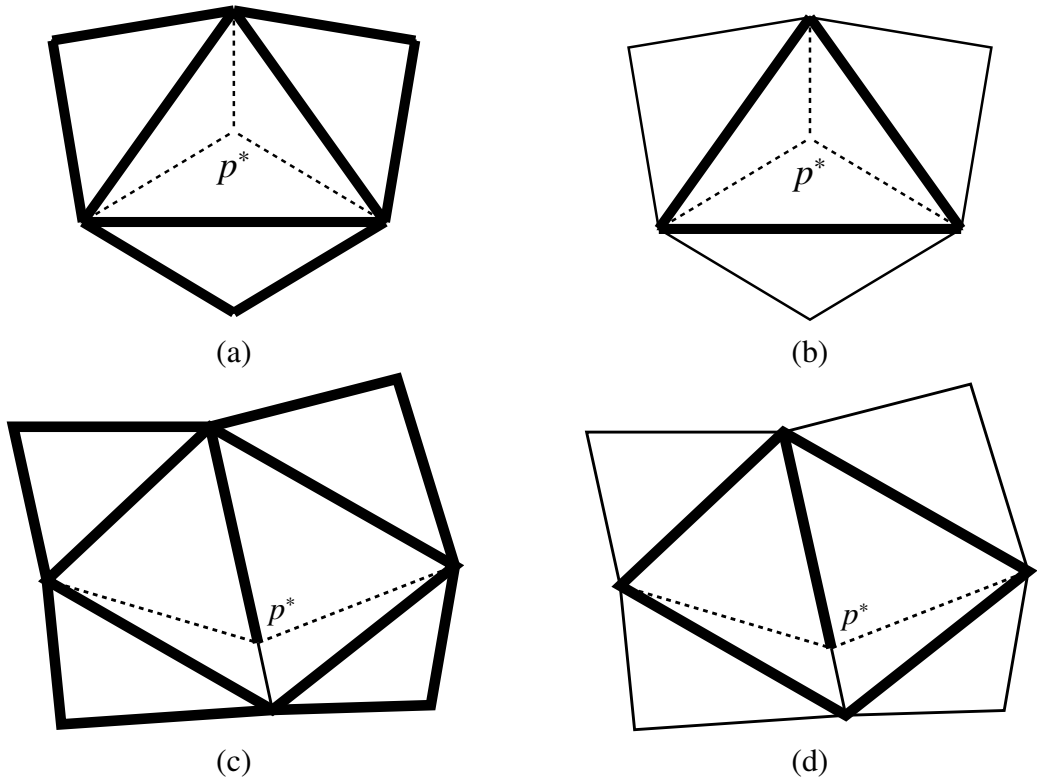


Figure 3.3: Example of determining the newly suspect edges after the insertion of the point p^* inside a face: (a) first case, (b) second case, and on the edge: (c) first case, (d) second case.

tivity adjustment), the LOP is performed after having inserted a new point p^* in the triangulation. When the LOP is invoked, we need to determine which edges should be initially marked as suspect. This, however, depends on the specific edge-flip criterion being employed. Let $\mho_0$ denote the set of all faces incident on the new vertex p^* and let $\mho_1$ denote the set of all faces that share at least one edge with a face in $\mho_0$. In the cases of the ABN, JND, DLP, DP, ELABN, ELJND, and YMS edge-flip criteria, the edges that should be marked as suspect are the flippable edges of all faces in $\mho_0 \cup \mho_1$. For example, after inserting the point p^* inside a face, as shown in Figure 3.3(a), the edges drawn using a thicker line would be marked as suspect. When the newly inserted point p^* is on an edge, the suspect edges are those marked thicker in Figure 3.3(c). In the cases of the Delaunay, SE, GHH, SQSE, and JNDSE edge-flip criteria, the edges that should be marked as suspect are all flippable edges of all faces in $\mho_0$. For example, after inserting the point p^* in a face, as shown in Figure 3.3(b), the edges drawn with a thicker line would be marked as suspect. In the case that the newly inserted point p^* is on an edge, the suspect edges are the thicker lines as shown in Figure 3.3(d).

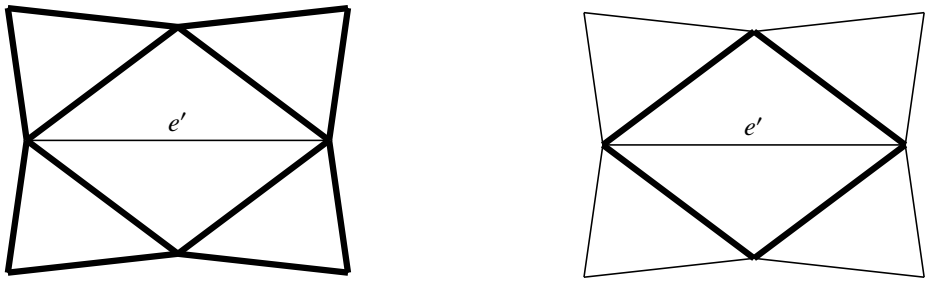


Figure 3.4: Example of determining the newly suspect edges after an edge flip that produces the edge e'. (a) First case. (b) Second case.

Also, recall that, in step 5 of the LOP, whenever an edge is flipped, we must determine which edges can become newly suspect as a result of the edge flip. Again, this depends on the particular edge-flip criterion being used. Let e denote the edge being flipped; let q denote the quadrilateral formed by the union of the two faces incident on e; and let e' denote the edge obtained by applying an edge flip to e. In the cases of the ABN, JND, DLP, DP, ELABN, ELJND, and YMS edge-flip criteria, the edges that should be marked as suspect are all flippable edges belonging to q or belonging to faces incident on edges of q. For example, if the edge e' as shown in Figure 3.4(a) was just produced as a result of an edge flip, we would need to mark all of the edges drawn with a thicker line as suspect (presuming that each edge is flippable). In the case of the Delaunay, SE, GHH, SQSE, and JNDSE edge-flip criteria, the edges that should be marked as suspect are all flippable edges belonging to q. For example, if the edge e' as shown in Figure 3.4(b) was just produced as a result of an edge flip, we would need to mark all of the edges drawn with a thicker line as suspect.

3.4 Proposed Mesh-Generation Method and Its Development

So far, we have introduced our proposed mesh-generation framework, which has several free parameters, and suggested a number of possible choices for each of these parameters. As one might suspect, however, not all of these choices are equally good. In our work, we studied how different choices of each parameter affects mesh quality. In what follows, we present a summary of our analysis, which leads to the recommendation of a particular choice for each parameter. Then, finally, we propose a specific mesh-generation method,

Table 3.1: Test images

Image	Size, Bits/Sample	Description
`bull`	1024×768, 8	computer-generated bull [59]
`ct`	512×512, 12	CT scan of head [56]
`glasses`	1024×768, 8	raytraced glasses [59]
`lena`	512×512, 8	woman [58]
`mri`	256×256, 11	MRI scan of head [56]
`peppers`	512×512, 8	collection of peppers [58]

based on these recommended choices.

Before proceeding further, a brief digression is in order regarding the image data that we used for analysis and evaluation purposes. In our work, we employed 42 images as test data, many of which were taken from standard image sets, such as [56], [57] and [58]. Herein, we present results for a representative subset of these images, namely, those listed in Table 3.1. This subset was deliberately chosen to contain a variety of image types including photographic, medical, and computer-generated imagery.

3.4.1 Choice of Face- and Candidate-Selection Policies

First, we examine how the choice of the face-selection policy selFace affects mesh quality. To do this, we fix the candidate-selection policy selCand as PAE, the main-edge flip criterion $\text{isPreferred}_{\text{main}}$ as JNDSE, and the final connectivity adjustment as enabled with the final edge-flip criterion $\text{isPreferred}_{\text{final}}$ as SE. Then, we select from amongst the two face-selection policies under consideration, namely, GAE and GSE. For all 42 images in our test set and several sampling densities, we generated a mesh using each of the face-selection policies, and then measured the resulting approximation error in terms of PSNR. A representative subset of the results (namely, for the six images in Table 3.1) is given in Table 3.2, with the best result in each case (i.e., each row in the table) shown in **boldface**.

Since the goal in our work is to minimize the squared error as defined in (2.2), we would suspect that it is probably beneficial to choose the face with the greatest squared error into which to insert a new point (i.e., the GSE policy). Examining Table 3.2, we can confirm this suspicion to be correct. As the results show, the GSE policy beats the GAE policy in all cases by a margin of 0.03 to 6.80 dB (with a median of 1.945 dB). Clearly, the GSE policy is superior.

With the PSNR evaluated above, we next show the subjective quality of reconstructed images, which is also an important aspect of mesh quality. Three images with various statis-

Table 3.2: Comparison of the mesh quality obtained with the various face-selection policies

Image	Samp. Density (%)	PSNR (dB) GAE	PSNR (dB) GSE
bull	0.125	28.58	**35.38**
	0.250	35.70	**39.40**
	0.500	39.24	**41.42**
	1.000	41.85	**43.11**
ct	0.250	31.45	**33.66**
	0.500	36.85	**38.69**
	1.000	41.68	**43.09**
	2.000	46.36	**47.70**
glasses	0.250	18.99	**22.75**
	0.500	22.70	**26.05**
	1.000	26.55	**29.99**
	2.000	31.40	**34.51**
lena	0.500	23.45	**26.51**
	1.000	27.37	**29.41**
	2.000	30.76	**32.03**
	3.000	32.84	**33.55**
mri	0.500	27.11	**31.23**
	1.000	33.44	**34.29**
	2.000	36.77	**37.56**
	3.000	39.18	**39.21**
peppers	0.250	21.70	**23.55**
	0.500	24.57	**26.83**
	1.000	28.47	**29.58**
	2.000	31.27	**31.70**

tical characteristics, `lena` (photographic), `ct` (medical), and `glasses` (computer-generated) images, are deliberately chosen for a more representative comparison. The three sets of reconstructed images from Table 3.2 are shown in Figures 3.5, 3.6 and 3.7. We can see that reconstructed images obtained with GSE policy represent the contour/edge information much more precisely than those obtained with GAE policy. In particular, the pitcher and the glass in the back of Figure 3.5 is a good example to show the difference between these two policies. Reconstructed image obtained with GSE policy drew the contour of these two objects quite clearly, while the image obtained with GAE did a poor job in that region. More differences can be found in Figures 3.6 and 3.7, such as the face and hat area of `lena` image and the shape of the holes in the middle region of `ct` image. As we can see, GSE policy provides overall better approximations than GAE. Hence, the subjective

(a) (b)

Figure 3.5: Comparison of the subjective mesh quality obtained for the `glasses` image at sampling density of 1% with the (a) GAE (26.55 dB), and (b) GSE (29.99 dB).

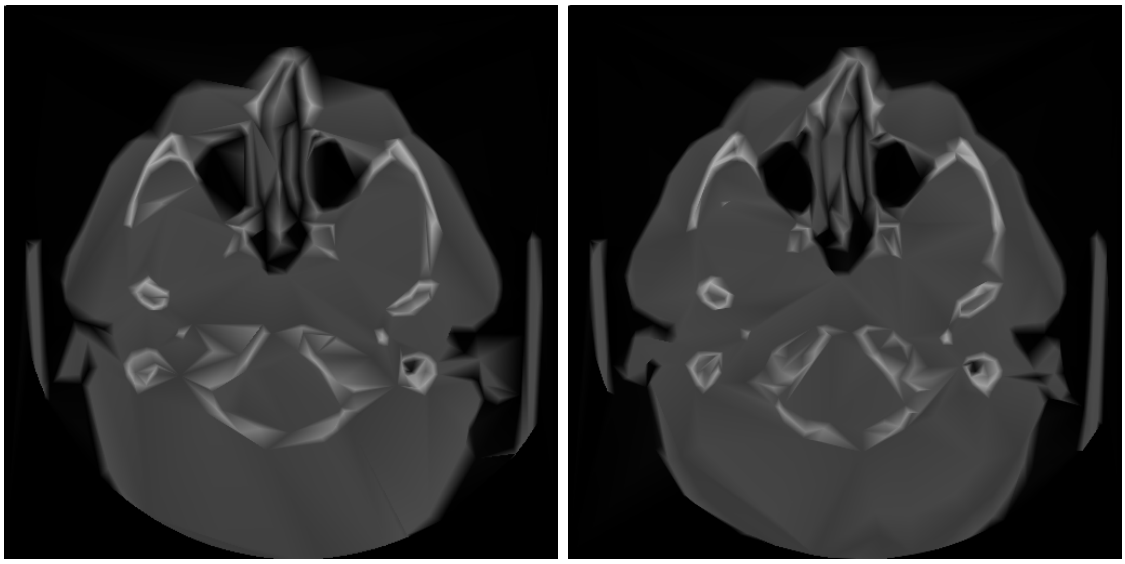
(c) (d)

Figure 3.6: Comparison of the subjective mesh quality obtained for the `lena` image at sampling density of 0.5% with the (a) GAE (23.45 dB), and (b) GSE (26.51 dB).

qualities we observed correlate well with the PSNR listed in Table 3.2.

Although we have only shown results above for one specific choice of the fixed parameters (i.e., the parameters other than the face-selection policy), we generally found similar results with other choices. In passing, we also note that the computational complexity of the GSE and GAE policies are quite comparable. For the above reasons, we deem the GSE policy as best and recommend its use in our framework.

Next, we examine how the choice of the candidate-selection policy selCand affects



(e) (f)

Figure 3.7: Comparison of the subjective mesh quality obtained for the `ct` image at sampling density of 0.25% with the (a) GAE (31.45 dB), and (b) GSE (33.66 dB).

mesh quality. To do this, we fix the face-selection policy selFace as GSE, the main edge-flip criterion $\text{isPreferred}_{\text{main}}$ as JNDSE, and the final connectivity adjustment as enabled with the final edge-flip criterion $\text{isPreferred}_{\text{final}}$ as SE. The candidate-selection policy is then chosen from amongst the five under consideration, namely, PAE, PWAE, AMSE-PAE, AMSE-PWAE and hybrid. For all 42 images in our test set and several sampling densities, we generated a mesh using each of the various candidate-selection policies and measured the resulting approximation error in terms of PSNR. A representative subset of the results (namely, for the six images in Table 3.1) is given in Table 3.3, with the best and worst values in each test case indicated by **boldface** and gray, respectively.

Examining Table 3.3, we can see that the AMSE-PAE and hybrid polices outperform the PAE, PWAE, and AMSE-PWAE policies in the vast majority of cases. In particular, the AMSE-PAE and hybrid policies beat the other three policies in 19/24 and 22/24 of the test cases, respectively. Among the three policies PAE, PWAE, and AMSE-PWAE, the PWAE policy is clearly the worst (lowest PSNRs in 23/24 of the test cases comparing to the other four policies), and the AMSE-PWAE policy works better than the PAE policy in 16/24 of cases. When comparing the best two policies AMSE-PAE and hybrid, we can also see that the hybrid policy outperforms the AMSE-PAE policy in 15/24 of the test cases. Furthermore, the hybrid policy typically wins by a much larger margin than it loses. That is, of the 9 cases where the AMSE-PAE policy beats the hybrid policy, only 2 are by more

Table 3.3: Comparison of the mesh quality obtained with the various candidate-selection policies

Image	Samp. Density (%)	PSNR (dB)				
		PAE	PWAE	AMSE-PAE	AMSE-PWAE	Hybrid
bull	0.125	35.38	33.42	36.41	35.63	**36.48**
	0.250	39.40	38.05	40.25	39.61	**40.45**
	0.500	41.42	40.79	42.63	42.17	**42.68**
	1.000	43.11	42.76	**44.32**	44.04	44.29
ct	0.250	33.66	32.39	**33.87**	33.16	33.85
	0.500	38.69	37.54	38.38	38.20	**38.70**
	1.000	43.09	42.02	42.70	42.54	**43.16**
	2.000	47.70	46.82	47.50	47.31	**47.82**
glasses	0.250	22.75	22.05	**23.68**	23.25	23.56
	0.500	26.05	25.61	26.76	26.58	**26.87**
	1.000	29.99	29.35	30.08	29.89	**30.27**
	2.000	**34.51**	33.47	34.07	33.77	34.40
lena	0.500	26.51	26.35	27.45	**27.57**	27.37
	1.000	29.41	29.32	**30.22**	30.06	30.16
	2.000	32.03	31.82	32.66	32.54	**32.80**
	3.000	33.55	33.27	34.09	33.96	**34.21**
mri	0.500	31.23	30.81	31.89	31.20	**32.07**
	1.000	34.29	34.30	34.78	34.66	**35.17**
	2.000	37.56	37.09	37.69	37.55	**37.88**
	3.000	39.21	38.87	39.36	39.35	**39.59**
peppers	0.250	23.55	23.07	**24.71**	24.00	24.56
	0.500	26.83	26.58	**27.81**	27.45	27.73
	1.000	29.58	29.38	**30.56**	30.36	30.52
	2.000	31.70	31.59	**32.64**	32.53	32.63

than a margin of 0.1 dB, whereas of the cases where the hybrid policy beats the AMSE-PAE policy, 13 involve a margin of more than 0.1 dB. From the above observations, it is clear that the hybrid policy is best.

With the PSNR evaluated above, we now show the subjective quality of the reconstructed images to see how well the subjective quality correlates with the PSNR. The reconstructed images of bull at sampling density of 0.125% from Table 3.3 are chosen to demonstrate the performance of the five candidate-selection methods. Since the differences between some reconstructed images obtained by these policies are not always easy for the human eye to distinguish, we zoomed in a region in each reconstructed bull image and included them in Figure 3.8. From the part of the reconstructed images in Figure 3.8, we can

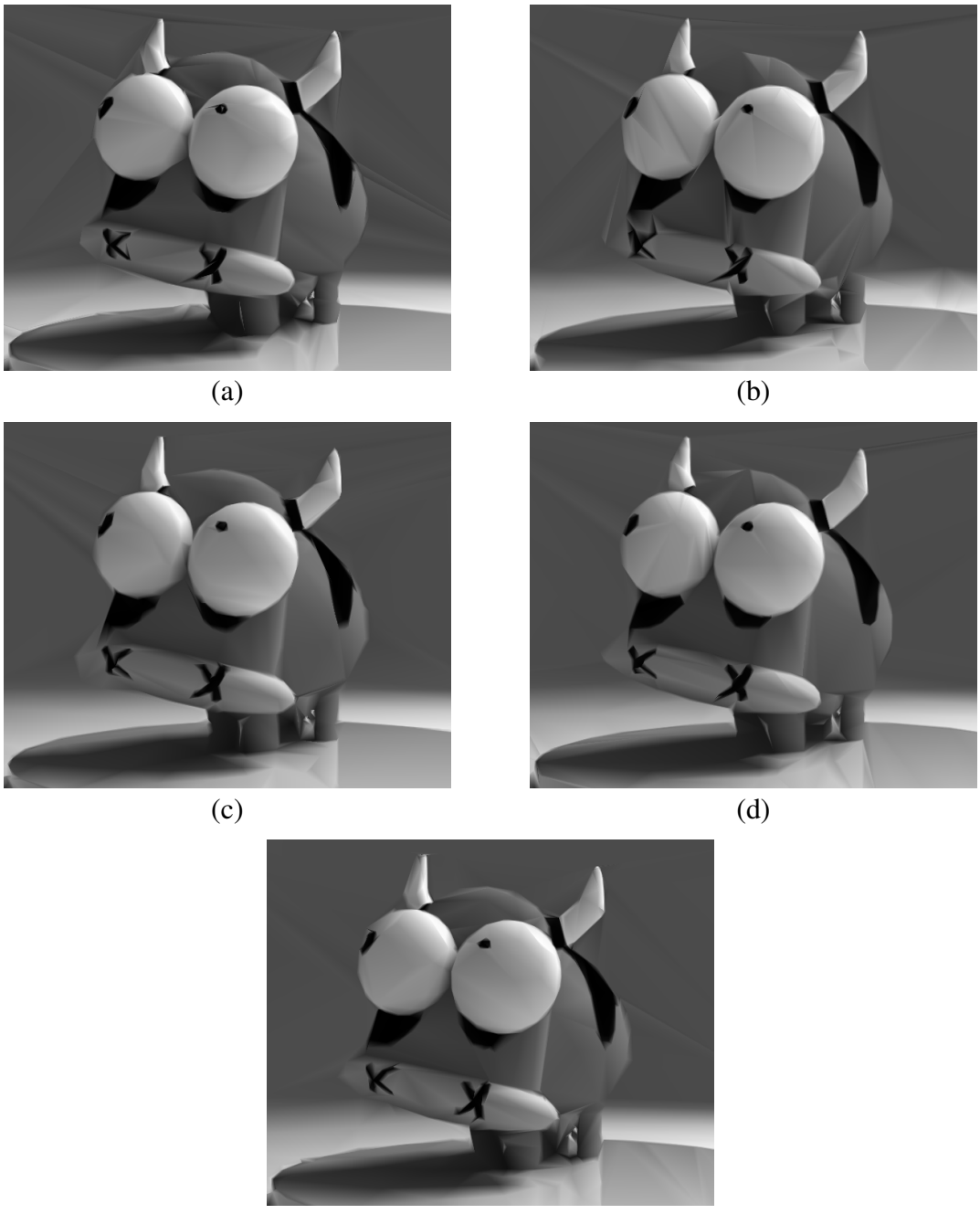
see that the hybrid, AMSE-PAE and AMSE-PWAE policies produce meshes with better quality than those produced by PAE and PWAE. Among the top three best policies, hybrid results in the least distortion. These observations correlate well with the PSNR results in Table 3.3.

Although we have only shown results above for one particular choice of the fixed parameters (i.e., the parameters other than the candidate-selection policy), we have generally found similar results with other choices. In passing, we note that, since the hybrid policy requires significantly less computation than the AMSE-PAE policy, selecting the former over the latter also happens to be beneficial in terms of computational complexity. For the reasons above, we deem the hybrid policy to be most effective and recommend its use in our framework.

3.4.2 Choice of Main Edge-Flip Criterion

Next, we consider how mesh quality is affected by the choice of the main edge-flip criterion $\text{isPreferred}_{\text{main}}$ used in initial and main connectivity adjustment. To do this, we fix the face-selection policy selFace as GSE, the candidate-selection policy selCand as PAE, and the final connectivity adjustment as enabled with the final edge-flip criterion $\text{isPreferred}_{\text{final}}$ as SE. Then we select $\text{isPreferred}_{\text{main}}$ from amongst the various criteria under consideration, namely, Delaunay, ABN, JND, DLP, DP, ELABN, ELJND, YMS, SE, GHH, SQSE, and JNDSE. For all 42 images in our test set and several sampling densities, we generated a mesh using each of the various main edge-flip criteria under consideration, and then measured the resulting approximation error in terms of PSNR. A representative subset of the results obtained (namely, for the six images in Table 3.1) is given in Table 3.4, where the best and worst values in each test case are again shown in **boldface** and gray, respectively. Note "D" in Table 3.4 represents the Delaunay edge-flip criterion.

Let us now examine Table 3.4. Rather than attempting to individually rank each of the criteria from best to worst (which may not be possible to do in general), we instead observe that there are some clear winners and losers here. In particular, the best performers are the JNDSE, SQSE, and GHH criteria. Of these three, the JNDSE criterion (which is newly proposed herein) is the overall winner, yielding the best result in 20/24 of the test cases and never losing by a margin of more than 0.04 dB in the remaining cases. At the other extreme, the worst performers in decreasing order of badness are the DP, SE, and ABN criteria, which produce the three poorest results in every case. The DP criterion is the worst performer overall, yielding the poorest result in 22/24 of the test cases. The



(a) (b) (c) (d) (e)

Figure 3.8: Part of the image approximation obtained for the `bull` image at sampling density of 0.125% with the various candidate-selection policies (a) PAE (35.38 dB), (b) PWAE (33.42 dB), (c) AMSE-PAE (36.41 dB), (d) AMSE-PWAE (35.63 dB), and (e) hybrid (36.48 dB).

SE criterion is worst or second worst in 16/24 of the test cases. The ABN criterion is third worst or second worst in every case. The poor performance of the SE criterion is

Table 3.4: Comparison of the mesh quality obtained with the various main edge-flip criteria

Image	Samp. Density (%)	PSNR (dB)											
		D	ABN	JND	DLP	DP	ELABN	ELJND	YMS	SE	GHH	SQSE	JNDSE
bull	0.125	33.15	29.49	34.91	33.97	25.37	32.88	34.52	31.26	25.83	34.03	35.24	**35.38**
	0.250	37.72	33.80	38.93	38.80	29.20	37.75	38.69	36.40	30.59	39.01	39.31	**39.40**
	0.500	40.96	38.27	41.09	41.08	33.45	40.79	41.14	40.18	35.81	41.28	41.38	**41.42**
	1.000	42.76	41.18	42.74	42.60	37.74	42.63	42.85	42.25	40.13	42.96	**43.11**	43.11
ct	0.250	31.22	25.65	31.82	30.56	24.74	29.58	32.51	29.37	26.02	32.77	33.16	**33.66**
	0.500	36.85	28.94	38.02	36.21	27.03	34.25	37.71	34.85	29.61	38.21	38.44	**38.69**
	1.000	41.52	32.47	42.55	41.43	29.66	39.99	42.37	40.23	33.91	42.54	43.07	**43.09**
	2.000	45.91	37.07	47.19	46.47	33.24	45.12	47.08	45.24	39.12	47.10	**47.70**	47.70
glasses	0.250	22.26	19.23	22.06	21.06	17.78	20.80	22.27	20.46	17.51	22.40	22.58	**22.75**
	0.500	25.29	21.06	25.24	24.29	19.21	23.76	25.33	23.17	19.19	25.51	**26.06**	26.05
	1.000	28.85	23.49	29.15	28.02	21.06	27.14	29.11	26.57	21.38	29.40	29.93	**29.99**
	2.000	32.87	26.55	33.65	32.64	23.24	31.28	33.42	30.95	24.26	33.73	34.36	**34.51**
lena	0.500	25.75	22.27	25.74	25.34	20.83	24.57	25.85	24.64	21.93	25.84	26.39	**26.51**
	1.000	28.55	24.54	28.82	28.22	22.75	27.17	28.58	27.19	24.20	28.93	29.41	**29.41**
	2.000	31.36	27.01	31.54	31.00	24.93	30.13	31.45	30.19	27.03	31.69	**32.07**	32.03
	3.000	32.85	28.76	32.99	32.51	26.35	31.90	33.03	31.88	28.77	33.21	33.48	**33.55**
mri	0.500	30.40	25.81	30.03	29.69	25.43	28.65	30.64	28.71	25.51	30.96	31.19	**31.23**
	1.000	33.92	28.66	33.64	32.97	27.64	32.14	33.67	32.09	28.34	34.17	34.28	**34.29**
	2.000	36.76	31.28	36.69	36.00	30.33	35.36	36.95	35.23	31.71	37.15	37.44	**37.56**
	3.000	38.58	33.17	38.61	37.73	32.15	37.18	38.72	37.24	33.80	38.81	39.15	**39.21**
peppers	0.250	23.02	19.50	22.72	22.21	17.83	21.50	22.76	21.01	19.20	23.03	23.41	**23.55**
	0.500	26.13	22.28	25.88	25.23	19.73	24.61	26.08	24.09	21.66	26.28	26.72	**26.83**
	1.000	28.86	25.09	28.82	28.33	21.85	27.85	29.01	27.38	24.60	29.18	29.51	**29.58**
	2.000	31.18	27.99	31.12	30.81	24.37	30.50	31.28	30.21	27.73	31.49	31.70	**31.70**

extremely important to note. It is because the SE edge-flip criterion performs so badly that the intuitively "obvious" solution of simply always using the SE edge-flip criterion in our framework is not a good one.

Since the subjective quality of meshes is as important as the PSNR, we next show some subjective quality comparisons using one of the test cases in Table 3.4. The reconstructed images obtained using `lena` image at sampling density of 1% from Table 3.4 are shown in in Figure 3.9, 3.10 and 3.11. From these figures, we can see that the quality of reconstructed images varies largely among the twelve main edge-flip criteria. The performance differences can be easily seen through many regions in the reconstructed `lena` image, such as her face, hat, shoulder, and objects with clear edges at the back of the image. By examining them carefully, we observe that the subjective quality correlates well with the PSNRs of reconstructed images, i.e., JNDSE, SQSE, GHH criteria perform the best among all of them, while SE, ABN, and DP criteria are the worst ones.

Although we have only shown results above for one particular choice of the fixed parameters (i.e., the parameters other than the main edge-flip criterion), we have generally found similar results with other choices. Since the JNDSE criterion is the clear winner, we recommend its use as the main edge-flip criterion is$\text{Preferred}_{\text{main}}$ in our framework.

A more careful examination of the results shows that the DP, SE, and ABN criteria perform very poorly due to their propensity to lead to triangulations with a very large

(a) (b)

(c) (d)

Figure 3.9: Comparison of the mesh quality obtained for the `lena` image at sampling density of 1% with various main edge-flip criteria (a) Delaunay (28.55 dB), (b) ABN (24.54 dB), (c) JND (28.82 dB), (d) DLP (28.22 dB),

number of *poorly chosen* sliver triangles. For example, for one of the test cases from Table 3.4, Figure 3.12 shows the triangulations that were obtained in the cases of the ABN, DP, and SE criteria, which perform poorly, as well as the DLP and JNDSE criteria, which perform relatively better. Notice how the results for the ABN, DP, and SE criteria are dominated by many poorly chosen sliver triangles, whereas the DLP and JNDSE results

(a) (b)

(c) (d)

Figure 3.10: Comparison of the mesh quality obtained for the `lena` image at sampling density of 1% with various main edge-flip criteria (Cont'd) (a) DP (22.75 dB), (b) ELABN (27.17 dB), (c) ELJND (28.58 dB), and (d) YMS (27.19 dB).

are not. The good performance of the JNDSE, SQSE, and GHH edge-flip criteria can be explained by the fact that these three criteria take into account not only squared error but triangle shape as well (via shape quality in the GHH and SQSE cases and JND in the JNDSE case). This allows these criteria to avoid triangulations with many poorly chosen sliver triangles.

(a) (b)

(c) (d)

Figure 3.11: Comparison of the mesh quality obtained for the `lena` image at sampling density of 1% with various main edge-flip criteria (Cont'd) (c) SE (24.20 dB), (d) GHH (28.93 dB), (e) SQSE (29.41 dB), and (f) JNDSE (29.41 dB).

Although the interplay between edge-flip criteria and sliver triangles is often extremely complex, we are able to offer some explanation as to why the DP and SE edge-flip criteria (i.e., the two worst performers) tend to produce many sliver triangles. First, let us consider the SE criterion. Since a sliver triangle has a very small area and therefore contains few or no points from the image domain Λ, the squared error calculated over a sliver triangle

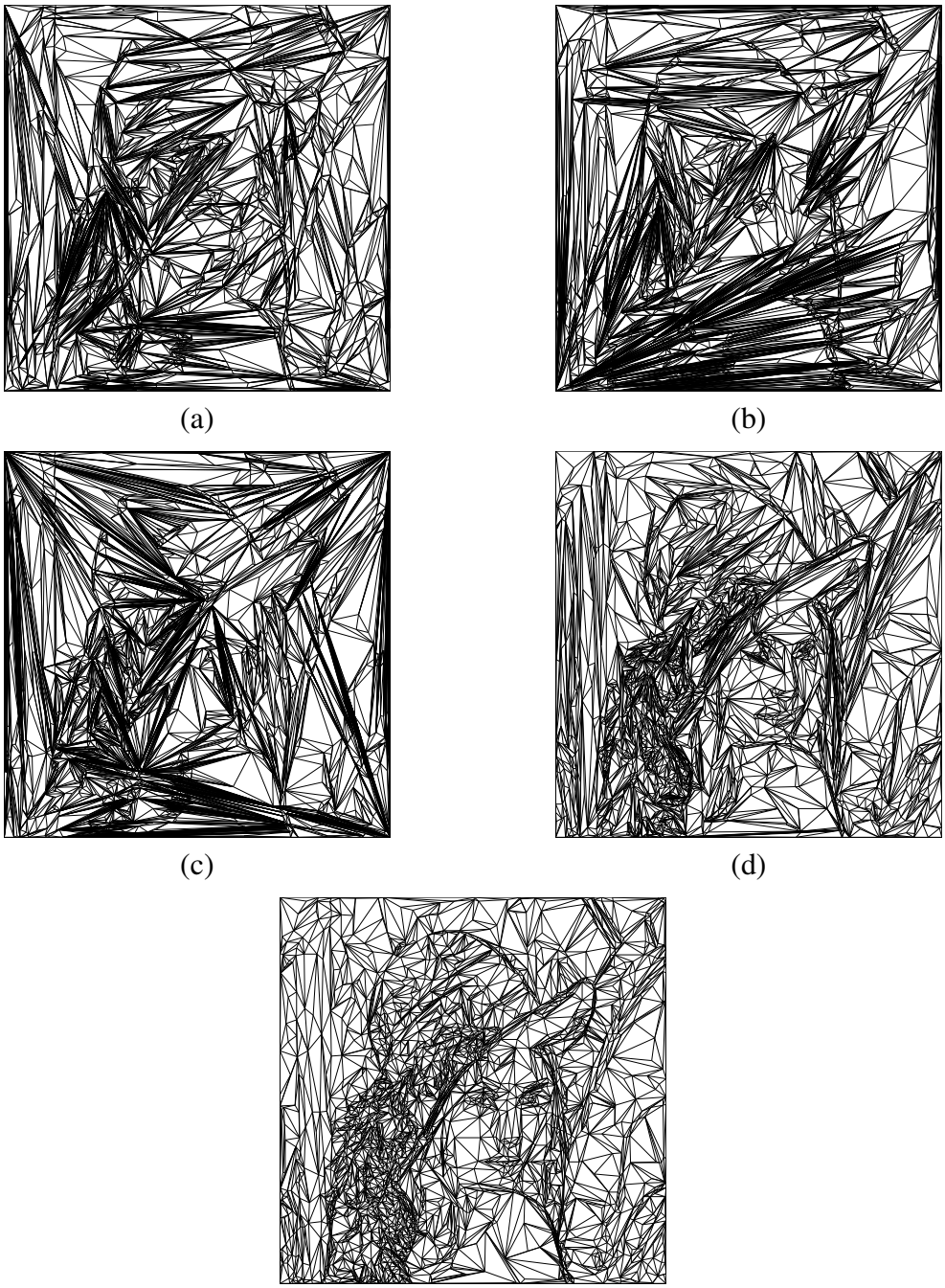


Figure 3.12: The triangulations obtained for the `lena` image at a sampling density of 1% with the main edge-flip criterion chosen as (a) ABN (28.55 dB), (b) DP (22.75 dB), (c) SE (24.20 dB), (d) DLP (28.22 dB), and (e) JNDSE (29.41 dB), respectively.

will always be very small and often zero. So, when viewed strictly from the standpoint of squared error, sliver triangles seem quite good, as they have very low squared error.

Unfortunately, because of this fact, many undesirable situations can arise when the SE criterion is employed. For example, consider an edge e whose two incident faces f_1 and f_2 are each sliver triangles that contain no points from Λ. The SE criterion will never flip such an edge (i.e., isPreferred(e) will always be one). Such behavior can, in turn, increase the likelihood that the two sliver triangles f_1 and f_2 remain in the triangulation. This problem is further exacerbated by the fact that, since f_1 and f_2 do not contain any points from Λ, no new point can be inserted into f_1 or f_2.

Next, let us contemplate why the DP criterion can lead to many sliver triangles. Consider an edge e with two incident faces f_1 and f_2 whose union forms the (strictly convex) quadrilateral q, and let e' denote the edge obtained by flipping e. Suppose that e and e' differ greatly in length, say, by a factor of more than 64, so that q is long and thin and f_1 and f_2 are sliver triangles. Let e_{short} and e_{long} denote the shorter and longer of the edges e and e', respectively. Due to the manner in which the DP edge-cost function $\text{edgeCost}_{\text{DP}}$ is defined in (3.3d), it is statistically very likely that $\text{edgeCost}_{\text{DP}}(e_{\text{long}})$ is quite small and $\text{edgeCost}_{\text{DP}}(e_{\text{long}}) < \text{edgeCost}_{\text{DP}}(e_{\text{short}})$. Thus, the DP edge-cost function tends to strongly favor longer edges that are associated with sliver triangles. Because the DP edge-flip criterion is based on this edge-cost function, all others things being equal, the DP edge-flip criterion will tend to prefer longer edges when they are associated with sliver triangles, and this in turn tends to lead to triangulations with many sliver triangles.

3.4.3 Choice of Final Edge-Flip Criterion

Next, we examine how the choice of the final edge-flip criterion $\text{isPreferred}_{\text{final}}$ affects mesh quality. To do this, we fix the face-selection policy selFace as GSE, the candidate-selection policy selCand as PAE, and the main edge-flip criterion $\text{isPreferred}_{\text{main}}$ as Delaunay. Then, we select from amongst the cases of no final connectivity adjustment as well as final connectivity adjustment with the final edge-flip criterion $\text{isPreferred}_{\text{final}}$ chosen as each of Delaunay, ABN, JND, DLP, DP, ELABN, ELJND, YMS, SE, GHH, SQSE, and JNDSE.

For all 42 images in our test set and several sampling densities, we generated a mesh using each of the schemes under consideration, and then measured the resulting approximation error in terms of PSNR. A representative subset of the results obtained is given in Table 3.5, with the best result in each test case shown in **boldface**. The case of no final connectivity adjustment is labelled as "None" in the table. Note that, we exclude the case of choosing $\text{isPreferred}_{\text{final}}$ as Delaunay since this is equivalent to no final connectivity adjustment when $\text{isPreferred}_{\text{main}}$ is Delaunay.

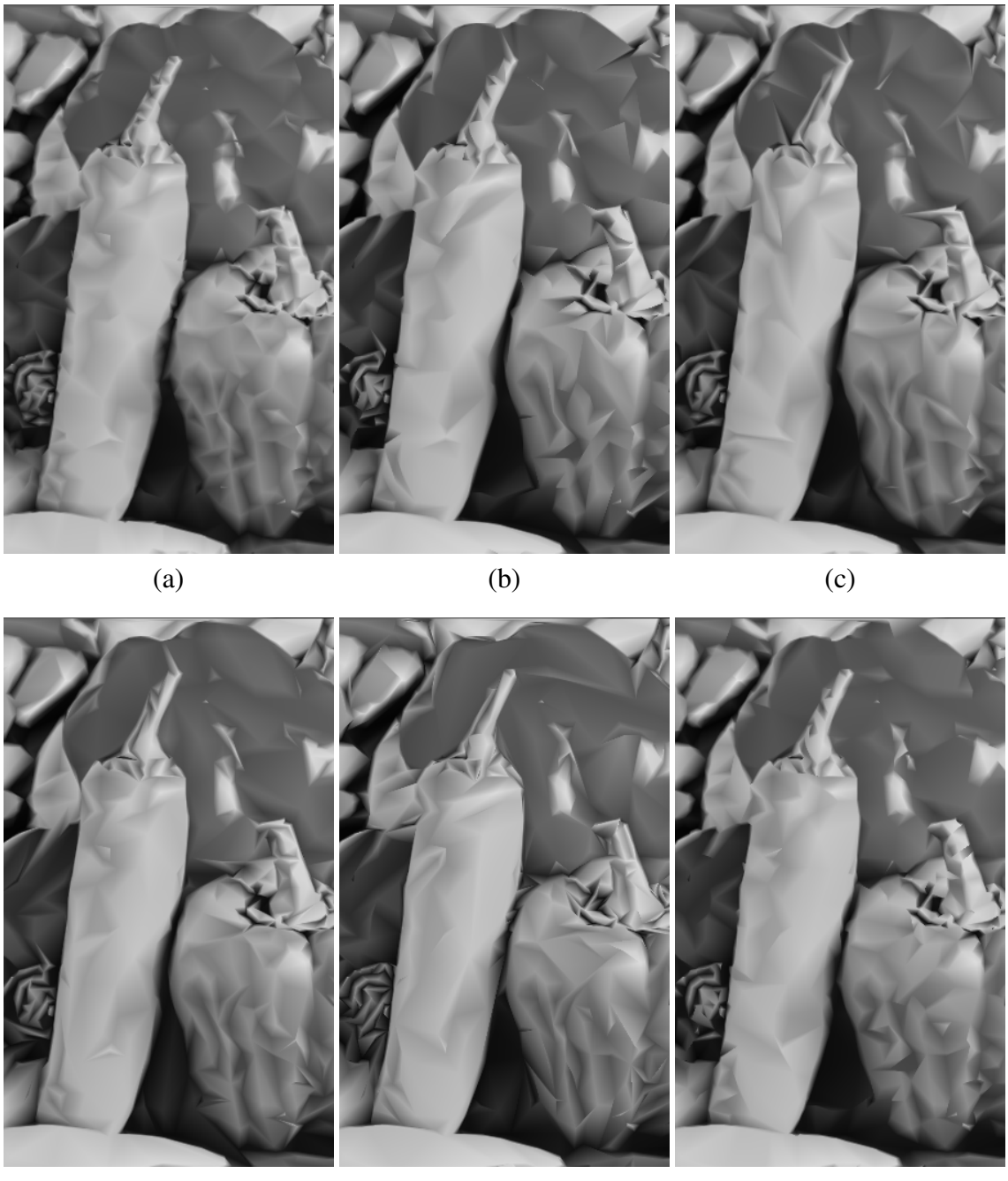
Table 3.5: Comparison of the mesh quality obtained with the various final edge-flip criteria

Image	Samp. Density (%)	PSNR (dB)											
		None	ABN	JND	DLP	DP	ELABN	ELJND	YMS	SE	GHH	SQSE	JNDSE
bull	0.125	30.86	29.70	27.79	29.77	29.89	30.35	30.81	27.89	**33.15**	31.29	32.10	32.07
	0.250	35.16	33.62	34.36	34.98	33.73	34.73	35.33	32.64	**37.72**	35.84	36.51	36.54
	0.500	38.91	37.72	37.96	38.52	37.60	38.24	38.76	37.04	**40.96**	39.50	39.89	39.98
	1.000	41.13	39.59	40.11	40.60	39.99	39.89	40.77	39.23	**42.76**	41.85	41.97	42.09
ct	0.250	29.73	27.27	28.05	27.98	25.82	28.08	29.65	26.61	**31.22**	29.80	30.66	30.69
	0.500	35.20	32.48	33.60	33.79	30.20	33.33	35.01	32.96	**36.85**	33.55	36.20	36.32
	1.000	39.62	37.15	37.33	38.51	35.07	37.60	39.72	37.14	**41.52**	38.64	40.72	40.83
	2.000	43.74	41.45	41.07	42.05	40.68	42.24	43.88	39.77	**45.91**	40.69	45.01	45.09
glasses	0.250	20.74	18.86	19.39	19.65	18.70	19.35	20.22	18.24	**22.26**	21.62	21.70	21.82
	0.500	23.77	21.71	22.36	22.49	21.17	22.29	23.33	20.84	**25.29**	24.59	24.69	24.81
	1.000	27.07	25.22	26.04	26.10	24.55	25.70	26.78	24.83	**28.85**	27.70	28.12	28.25
	2.000	30.92	28.95	30.21	30.05	28.25	29.53	30.72	29.05	**32.87**	31.45	32.07	32.22
lena	0.500	24.21	22.66	23.55	23.45	22.17	22.81	23.96	22.56	**25.75**	25.15	25.07	25.17
	1.000	26.88	24.86	26.12	26.09	24.85	25.37	26.55	24.99	**28.55**	27.90	27.89	27.96
	2.000	29.82	27.94	29.15	29.08	27.85	28.39	29.54	28.28	**31.36**	30.77	30.71	30.81
	3.000	31.37	29.45	30.71	30.57	29.23	30.05	31.06	29.83	**32.85**	32.24	32.24	32.34
mri	0.500	28.84	26.64	27.85	26.67	27.08	27.57	28.40	25.99	**30.40**	29.71	29.77	29.82
	1.000	32.32	30.44	31.13	31.24	30.25	30.81	32.19	30.14	**33.92**	33.19	33.21	33.38
	2.000	35.37	33.25	33.99	34.13	32.55	33.52	34.90	32.99	**36.76**	36.22	36.24	36.29
	3.000	37.16	34.40	36.42	36.17	34.12	34.99	36.86	35.35	**38.58**	38.09	38.10	38.19
peppers	0.250	21.38	19.51	20.49	20.35	18.65	20.11	20.94	19.18	**23.02**	22.38	22.35	22.45
	0.500	24.71	22.74	23.68	23.48	22.31	23.18	24.31	22.25	**26.13**	25.61	25.58	25.63
	1.000	27.36	25.46	26.34	26.25	25.06	25.96	27.04	25.42	**28.86**	28.14	28.25	28.30
	2.000	29.96	28.10	28.91	28.95	28.03	28.68	29.60	28.13	**31.18**	30.72	30.69	30.78

Examining Table 3.5, we can see that the SE criterion performs best in every test case, followed by the JNDSE and SQSE criteria (in that approximate order). Moreover, the use of final connectivity adjustment with the SE criterion beats the case of no final connectivity adjustment (i.e., the "None" column) by a margin of 1.48 to 2.56 dB. So, clearly the use of final connectivity adjustment with the SE criterion is extremely beneficial.

After examining the PSNRs, now we want to check whether the subjective quality corresponds well with the PSNRs. The reconstructed `peppers` images with sampling density of 1% is chosen from Table 3.5 for comparison. Since the differences between some reconstructed images obtained by these edge-flip criteria are not always easy for the human eye to distinguish, zoomed-in parts of the reconstructed `peppers` images are shown in Figure 3.13 and Figure 3.14. Examining the reconstructed image more closely, we note that the performance differences between those final edge-flip criteria can be seen around the contour of the objects in the images. Take "None" and SE criteria for example, the contour of the long pepper in Figure 3.13(a) is quite unclear, while in Figure 3.14(c), the contour around the same pepper are more smooth and clear. Thus the general trend we observed from the PSNRs of reconstructed images correlates well with the subjective quality of reconstructed images.

Although we have only presented results for one set of choices for the fixed parameters (i.e., the parameters other than the final edge-flip criterion), the preceding trends were found

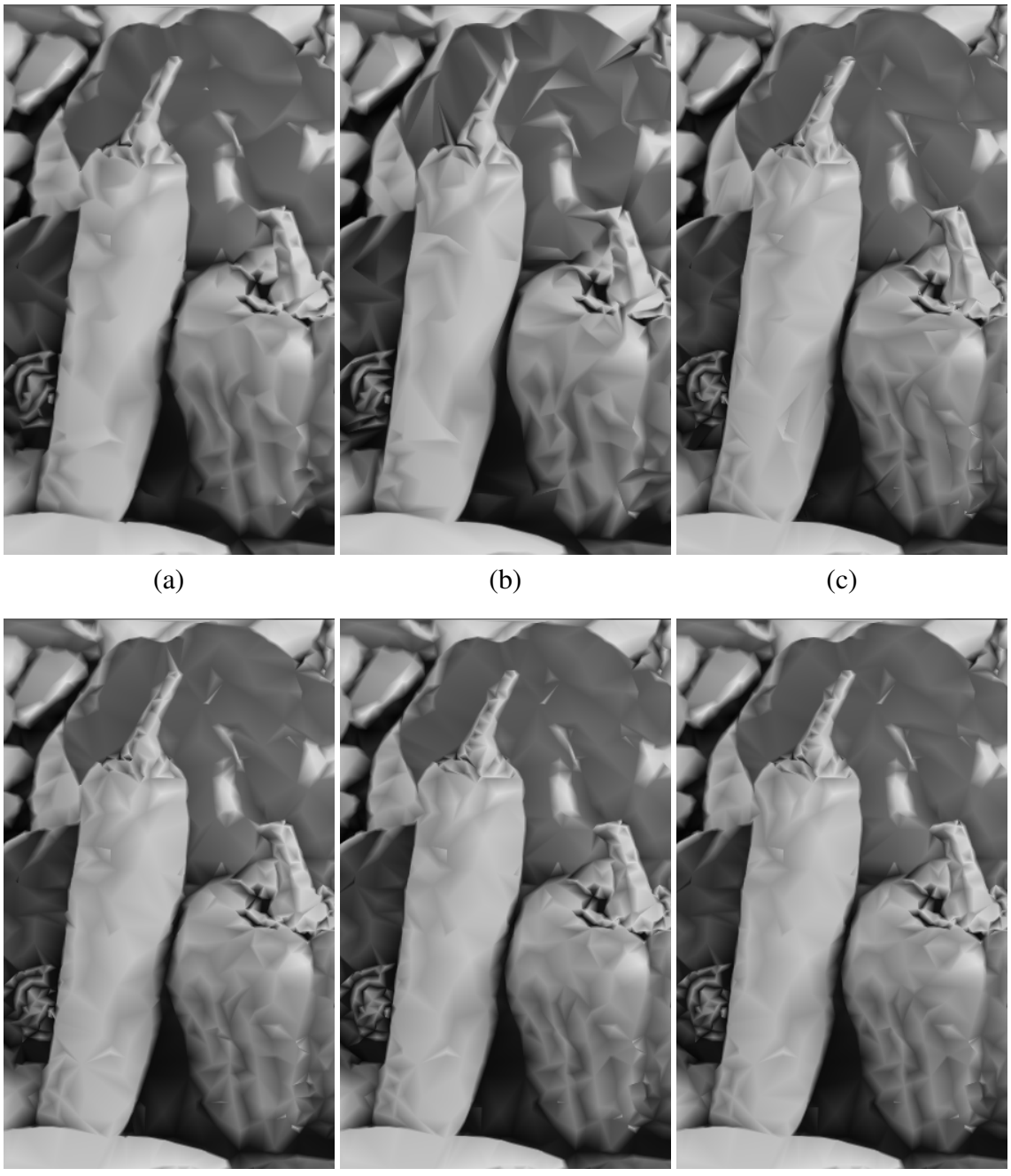


(a) (b) (c)

(d) (e) (f)

Figure 3.13: Comparison of the mesh quality obtained for the `peppers` image at sampling density of 1% with the various final edge-flip criteria (a) None (27.36 dB), (b) ABN (25.46 dB), (c) JND (26.34 dB), (d) DLP (26.25 dB), (e) DP (25.06 dB), and (f) ELABN (25.96 dB).

to be followed for other choices, provided that the main edge-flip criterion was chosen to produce a reasonably good triangulation as input to final connectivity adjustment. In cases



(a) (b) (c)

(d) (e) (f)

Figure 3.14: Comparison of the mesh quality obtained for the `peppers` image at sampling density of 1% with the various final edge-flip criteria (Cont'd) (a) ELJND (27.04 dB), (b) YMS (25.42 dB), (c) SE (28.86 dB), (d) GHH (28.14 dB), (e) SQSE (28.25 dB), and (f) JNDSE (28.30 dB).

where the main edge-flip criterion produces very poor triangulations (e.g., the DP, SE, and ABN cases), it actually turns out that the JNDSE and SQSE criteria perform better than the

SE criterion.

The above behavior can be explained as follows. Nominally, we would expect the SE criterion to perform best, as it directly minimizes the squared error in (2.2). In cases where the triangulation that is input to the final connectivity adjustment process is reasonably good, this is the exactly the behavior observed. If, however, the main edge-flip criterion is chosen as one of the criteria (i.e., DP, SE, and ABN) that leads to very poor triangulations, the behavior changes with the JNDSE and SQSE criteria becoming much more effective than the SE criterion. This change in behavior is due to the fact that the SE criterion does not take triangle shape into account. If the input triangulation (for final connectivity adjustment) has many badly shaped triangles, the SE criterion cannot improve much upon the situation since it is effectively blind to triangle shape, which will ultimately lead to a poor quality mesh. In contrast, the JNDSE and SQSE criteria both consider triangle shape (in addition to squared error). For this reason, they are able to partially mitigate the effects of a poorly chosen triangulation and produce a better result than is possible with the SE criterion.

Since the main edge-flip criterion $\text{isPreferred}_{\text{main}}$ was previously recommended to be chosen as JNDSE, which generally produces good triangulations with very few poorly-chosen sliver triangles, we conclude that we should use final connectivity adjustment with a final edge-flip criterion that works well for good triangulations. As we saw above, the clear choice in this case is the SE criterion. Therefore, in our framework, we recommend always using final connectivity adjustment and selecting the final edge-flip criterion $\text{isPreferred}_{\text{final}}$ as the SE criterion.

3.4.4 Extra Experiments

Thus far, many free parameters in our framework are presented, and different choices of each parameter are discussed. During the development of our framework, we have actually explored more parameters and more choices for each parameter than those listed above. Since some of them do not produce very competitive results or some of them may add too much computational complexity to the method, we decided to not include them in the earlier description of our proposed framework. Instead, we will briefly discuss them below for readers' interest.

As introduced earlier in Section 3.4.1, the AMSE-PAE and AMSE-PWAE candidate-selection policies select candidate point in two steps: first choose a set Ω of test points and then choose the candidate point from Ω. We noted that the size of set Ω (i.e., eight) was

chosen based on experimentation. During the development of our framework and methods, various sizes of set Ω have been tested, from 1 to 50. We noticed that when the size of set Ω increases, the PSNR of the reconstructed images have a general trend of increasing, but the increase is diminishing. For example, when we increase the size of set Ω from 8 to 9, the PSNR of the reconstructed images increases, however, the increase in PSNR is not as larger as that when the size of set Ω increase from 7 to 8. The computation complexity, however, increases linearly with the size of set Ω. We also noticed that when the size of the test point set increases from 20 to 50, the PSNR of the reconstructed images start to oscillate in a small range (less than 0.01 dB). Trying to balance between the mesh quality and computation complexity, we settle down with a compromise number of eight.

Another area that we explored during the development of our framework and mesh-generation methods was candidate-selection methods. For example, one other method we have tried is to use a 3 by 3 window in the PAE and PWAE candidate-selection methods. Instead of just calculating the difference between the image and its approximation in one particular point p, we average the differences of nine points in a 3 by 3 window region, with p as the center point of the window. Figure 3.15 gives a simple example to illustrate this strategy. With PAE and PWAE policies, we only compute the difference between the image and its approximation in point p, while for the window strategy, we compute the differences for all the nine points in the window, average them and then assign that to point p. Based on the experimental results, we noticed that this strategy helps to improve the performance of PWAE candidate selection method, however, it actually weakens the PAE candidate selection method. Since the best candidate selection methods are based on PAE, and employing this window scheme requires extra computation, we decided to not include this strategy in the earlier description of the candidate-selection methods.

Recall that the proposed candidate selection method is a hybrid scheme that employs the PAE method before the number of sample points reaches 25% of the target number, then employs the AMSE-PAE method for the rest of the sample points selection. Actually we have experimented with different percentages for the hybrid scheme. For instance, we tried a method that employs the PAE policy until the number of sample points reaches 50% of the desired number, with the AMSE-PAE being used afterward. Experimental results show that the hybrid scheme (i.e., candidate selection method with 25% as switching percentage) outperforms the method with 50% as the switching percentage in a majority of cases. Besides, the hybrid scheme (with the lower 25% threshold) requires less computational cost.

When we employ the LOP algorithm in Step 2 and Step 6 of our proposed framework,

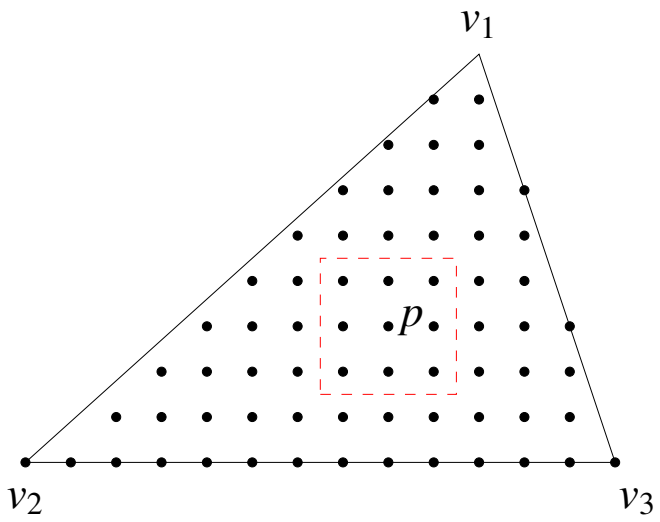


Figure 3.15: An example of candidate-selection method using window.

we need to remove one edge at a time from the suspect edge set for optimality test. If edges are not optimal, we need to flip them. The order in which edges are flipped affects the final mesh and therefore the quality of the reconstructed image [28]. The order we used in our framework is **last-in first-out (LIFO)** order. That is, remove the suspect edges in the order opposite from which they were inserted into the suspect edge set. A few other orders were also examined during the development of our framework, such as first-in first-out (FIFO), longest-edge first, random, and so on. Experimental results show that amongst all of the orders we considered, there are no obvious winners. Thus, we decided to employ the LIFO order in our work, since it is easy to implement efficiently.

One heavily explored area in our framework was edge-flip criteria. Although we have already listed twelve edge-flip criteria above, there are still several criteria that were omitted in our earlier discussions due to their non-impressive performance. One type of edge flip criterion we tried combine certain shape-quality criteria with squared error, such as the product of ABN and SE, the product of EL, SQ, and SE, and the product of EL, JND, and SE. None of these hybrid schemes were found work very well. Thus they were not included in the list of edge-flip criteria considered earlier.

In our framework, we have two classes of edge flip criteria, one associates with edge-cost functions that assign a cost to *every* edge in the triangulation T and the other one associates edge-cost functions that assign a cost to every *flippable* edge in the triangulation. As we mentioned in Section 3.2, during the LOP, cycles can occur when the second class of edge-flip (non-well-behaved) criteria are employed. For the first type

of edge-flip criteria, theoretically, cycle cannot occur except due to the roundoff error. Take DLP for example, we will flip one edge only if it causes the triCost(T) to decrease. where triCost(T) = $\sum_{e\in\mathcal{E}(T)}$ edgeCost$_{\mathsf{DLP}}(T, e)$, and edgeCost$_{\mathsf{DLP}}(T, e)$ is a 2-norm defined in (3.3c). Thus, for the DLP case, the triCost(T) only decreases in each edge flip, and the edge flip will eventually stops when triCost(T) = 0. On the other hand, for the second class of edge-flip criteria, cycles could theoretically happen because we are only trying to minimize the cost for only one edge at a time, not the cost of the whole triangulation. Hence, it is quite possible that while one edge flip causes the edge cost to decrease while it actually increases the cost of the whole triangulation. Thus, the edge-flip process using the non-well-behaved edge-flip criterion, is not a strict optimization process and potential can cause cycles.

An example of an cycle extracted from the `bull` image at sampling density of 0.125% with face-selection policy selFace as GSE, the candidate-selection policy selCand as AMSE-PAE, the main edge-flip criterion isPreferred$_{\mathsf{main}}$ as JNDSE. and the final connectivity adjustment as disabled, is given in Figure 3.16. In Figure 3.16, the dashed line denotes the new edge will be shown in the triangulation after flipping an old non-optimal edge; dark line denotes the next suspect edge that will be tested for optimality; other lines denote the edges in the current triangulation. Assume during the mesh generation process, we have a triangulation as shown in Figure 3.16(a) (excluding the dashed line) and edge v_iv_k is just tested as a non-optimal edge. So we flip edge v_iv_k to the other diagonal v_jv_l of the convex quadrilateral $v_iv_jv_kv_l$, and add suspect edges (i.e., the four edges of quadrilateral $v_iv_jv_kv_l$) to the suspect edge set. After the edge flip, the triangulation becomes the one shown in Figure 3.16(b) (excluding the dashed line). Next edge v_iv_l is tested as non-optimal, then we flip edge v_iv_l to v_jv_m as shown in Figure 3.16(b). We continue the LOP process, and after five edge flips we end up with a triangulation shown in Figure 3.16(f) , which is the same triangulation as shown in Figure 3.16(a). During the above example, edges are flipped only if it cause the cost of the edge under consideration to decrease. We, however, did not consider the cost for the whole triangulation. Hence, it is possible that for the five edge flips in the above example, certain edge flips cause the cost of the triangulation to increase, while the rest cause the cost of the triangulation to decrease, then the conflicts lead to a cycle. With careful examination in each step in the above cycle, we acknowledge that the order in which suspect edges are flipped can potentially affect the cycles' occurrence.

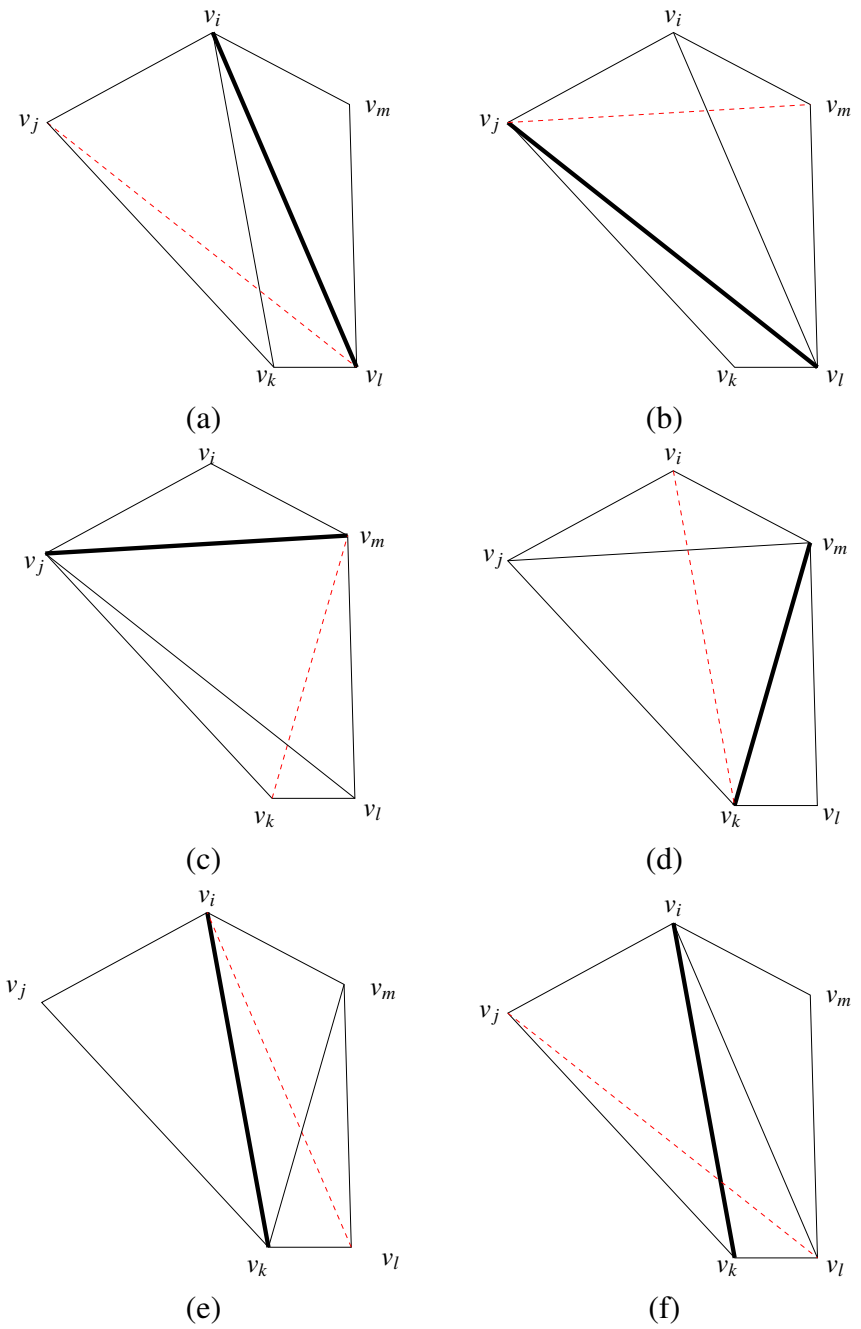


Figure 3.16: An example of cycle in mesh generation. In each triangulation, dashed line denotes the new edge from flipping an old non-optimal edge; dark line denotes the next suspect edge that will be tested for optimality; the regular lines are the edges in the current triangulation.

3.4.5 Proposed Mesh-Generation Method

Above, we have studied how various choices of the free parameters of our proposed framework affect mesh quality. This led us to recommend a particular choice for each of these parameters. The mesh-generation method that we propose in this thesis simply corresponds to our framework with the parameters selected as recommended earlier, namely, with the face-selection policy selFace as GSE, the candidate-selection policy selCand as hybrid, the main edge-flip criterion $\text{isPreferred}_{\text{main}}$ as JNDSE, and final connectivity adjustment enabled with the final edge-flip criterion $\text{isPreferred}_{\text{final}}$ as SE.

3.5 Evaluation of Proposed Mesh-Generation Method

Having introduced our proposed method, we now evaluate its performance by comparing it to several other competing schemes in terms of mesh quality and computational/memory complexity. The first of the methods that we consider for comparison purposes is the one proposed by Garland and Heckbert [17, Algorithm IV] with the quality threshold parameter `qthresh` chosen as 0.5 and an L^2 error measure, which we henceforth refer to by the name "GH". The GH method is essentially a special case of our framework with no final connectivity adjustment and the face-selection policy, candidate-selection policy, and main edge-flip criterion chosen as GAE, PAE, and GHH, respectively. The second of the methods that we consider for comparison purposes is the one proposed by Rippa in [21] with the least-squares edge cost (in the interpolating case), which we henceforth refer to by the name "R". (As an aside, we note that we elected to consider the least-squares edge-cost function from [21], as it is the only edge-cost function therein that incorporates squared error in some way.) The R method is a special case of our framework with no final connectivity adjustment and the face-selection policy, candidate-selection policy, and main edge-flip criterion chosen as GAE, PAE, and SE, respectively. The third of the methods that we consider for comparison purposes is the adaptive-thinning (AT) scheme of Demaret and Iske[16]. The AT method is one of the very best state-of-the-art Delaunay-based methods. It produces extremely high-quality meshes, even better than many DDT-based schemes, but has extremely high computational and memory requirements.

Mesh quality. For all 42 images in our test set and several sampling densities, we used each of the methods under consideration to generate a mesh and then measured the resulting approximation error in terms of PSNR. A representative subset of the results obtained (namely, for the six images in Table 3.1) is shown in Table 3.6, with the best result

Table 3.6: Comparison of the mesh quality obtained with the various mesh-generation methods

Image	Samp. Density (%)	PSNR (dB)					
		Proposed	GH	R	GH2	R2	AT
`bull`	0.125	**36.48**	26.55	19.26	33.12	25.83	33.12
	0.250	**40.45**	31.99	23.08	38.28	30.59	38.23
	0.500	**42.68**	37.64	26.46	40.72	35.81	41.87
	1.000	**44.29**	40.56	32.40	42.48	40.13	43.99
`ct`	0.250	**33.85**	28.69	20.75	32.22	26.02	32.15
	0.500	**38.70**	35.23	25.92	37.68	29.61	37.22
	1.000	**43.16**	39.43	30.08	42.01	33.91	41.35
	2.000	**47.82**	44.99	36.16	46.61	39.12	45.33
`glasses`	0.250	**23.56**	18.38	14.07	21.94	17.51	23.36
	0.500	**26.87**	20.95	15.29	25.07	19.19	25.84
	1.000	**30.27**	25.27	17.82	28.87	21.38	28.88
	2.000	**34.40**	29.94	20.99	33.11	24.26	32.73
`lena`	0.500	**27.37**	22.39	16.66	25.37	21.93	26.66
	1.000	**30.16**	25.27	19.34	28.51	24.20	29.12
	2.000	**32.80**	29.80	22.41	31.26	27.03	31.82
	3.000	**34.21**	31.84	24.11	32.77	28.77	33.37
`mri`	0.500	**32.07**	26.00	22.29	30.57	25.51	31.48
	1.000	**35.17**	32.38	24.48	33.77	28.34	34.39
	2.000	**37.88**	35.35	28.87	36.80	31.71	37.25
	3.000	**39.59**	38.01	31.73	38.42	33.80	39.01
`peppers`	0.250	**24.56**	19.84	14.78	22.58	19.20	23.93
	0.500	**27.73**	23.98	17.36	25.95	21.66	27.09
	1.000	**30.52**	27.49	20.32	28.77	24.60	30.05
	2.000	**32.63**	30.36	24.53	31.14	27.73	32.39

in each case (i.e., each row in the table) highlighted in **boldface**. In order to show that the improvements offered by our proposed method are not solely due to its use of the GSE face-selection scheme, we also consider our own improved versions of the GH and R methods, called GH2 and R2, respectively, where the face-selection policy of GAE is replaced by GSE.

Examining Table 3.6, we can see that our proposed method is the clear winner, producing the best result in all 24 test cases. Inspecting the results more closely, we find that our proposed method outperforms the GH, R, GH2, R2, and AT schemes by margins of 1.58 to 9.93 dB (with median 4.1050 dB), 7.86 to 17.37 dB (with median 10.765 dB), 1.02 to 3.36 dB (with median 1.5800 dB), 4.16 to 10.65 dB (with median 6.3650 dB), and 0.20 to

3.36 dB (with median 0.8250 dB), respectively. The fact that our proposed method beats the GH2 and R2 schemes (i.e., our improved versions of the GH and R schemes, respectively) by significant margins demonstrates that the excellent results from our method are not simply due to our different point-selection strategy alone. The fact that our method can outperform the AT scheme is extremely impressive, given that the AT scheme produces very high quality meshes and (as we shall see shortly) requires over 10 times more computation and orders of magnitude more memory. Again, examining Table 3.6, we observe that the R and R2 methods are the clear losers, yielding the two worst results in every case.

In the above evaluation, PSNR was found to correlate reasonably well with subjective quality. For the benefit of the reader, however, we provide three examples illustrating the subjective quality achieved by the various methods. These three examples (`lena`, `ct`, and `bull`) are chosen to contain a variety of image types including photographic, medical, and computer-generated imagery. For each example (from Table 3.6), a small part of each image reconstruction is shown under magnification in Figures 3.17, 3.18, and 3.19, along with the corresponding triangulations. Examining the figures, we can see that our proposed method produces approximations with better subjective quality as compared to the other methods. So, our proposed method not only yields approximations with low squared error, but good subjective quality as well. Observe that the R and R2 methods perform very poorly, due to the large number of poorly chosen sliver triangles in their triangulations.

Computational complexity. Next, we compare the various mesh-generation methods in terms of their computational complexity (i.e., execution time). To do this, we provide a representative subset of some timing results collected on very modest hardware (namely, a seven year old notebook computer with a 3.4 GHz Intel Pentium 4 and 1 GB of RAM). For the `lena` image and several sampling densities, the time required for mesh generation for each of the methods under consideration is shown in Table 3.7. In the case of the GH and R methods, a second set of numbers is given in parentheses. These numbers correspond to the time required if the method in question terminates mesh generation not when the target sampling density is achieved, but rather when the PSNR mesh quality matches that of the corresponding result from our proposed method.

Examining Table 3.7 (excluding the results in parentheses), we observe the following. Our proposed method has very substantially lower complexity than the AT scheme, requiring only about 5 to 10% of the time of the AT scheme. This is particularly impressive when one considers that our method produces significantly higher quality meshes than the AT scheme. Also, our proposed method requires about 6 to 12% and 36 to 52% more computation than the GH and R schemes, respectively. The increase in time relative to the GH

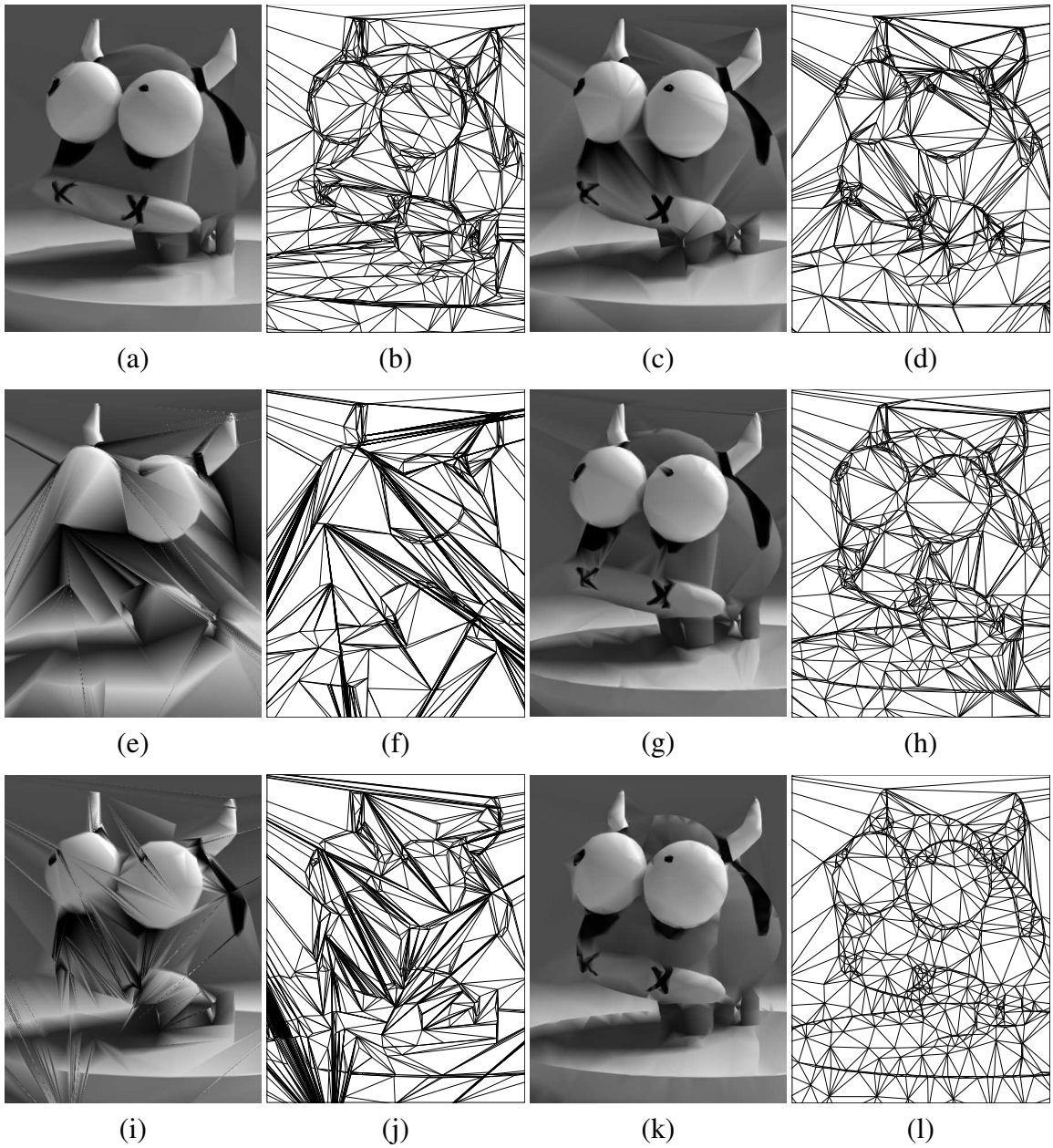


Figure 3.17: Part of the image approximation obtained for the `bull` image at a sampling density of 0.125% with the (a) proposed (36.48 dB), (c) GH (26.55 dB), (e) R (19.26 dB), (g) GH2 (33.12 dB), (i) R2 (25.83 dB), and (k) AT (33.12 dB), methods and (b), (d), (f), (h), (j), and (l) their corresponding triangulations.

and R schemes is due primarily to the additional final connectivity adjustment step and the use of a more computationally-expensive point-selection strategy (which often requires the computation of $\text{selCand}_{\text{AMSE-PAE}}$). When interpreting these computational-cost results, it is important to keep in mind that the GH and R schemes produce meshes of very significantly

Table 3.7: Comparison of the computational complexity for the various methods

Samp. Density (%)	Time (s)			
	Proposed	GH	R	AT
0.5	2.31	2.10 (2.85)	1.52 (4.22)	43.03
1.0	2.77	2.59 (3.33)	1.93 (5.45)	43.03
2.0	3.54	3.26 (4.10)	2.54 (7.25)	42.40
3.0	4.19	3.76 (4.67)	3.06 (8.53)	42.12

lower quality than the proposed method. If, in the case of the GH and R methods, we instead let the mesh-generation process terminate when it achieves the same PSNR mesh quality as the proposed method, the execution times in parenthesis in Table 3.7 apply. Viewing the situation from this perspective, the GH and R methods are actually much slower, by factors of 1.11 to 1.24 and 1.82 to 2.05, respectively. With the preceding observation in mind (as well as others made above), we deem the computational complexity of our scheme to be at least comparable to (and arguably better than) the GH and R methods. So overall, our proposed method performs quite well in terms of computational complexity.

The curious reader might wonder why the execution times of the GH and R schemes are not closer. As it turns out, the main reason for this difference is that the R scheme requires many fewer edge flips than the GH scheme, typically about 2.4 to 2.9 times less. Recall that the R scheme produces a relatively large number of poorly chosen sliver triangles. This tends to significantly reduce the number of flippable edges, causing the LOP to converge much more quickly with fewer edge flips. So, essentially, the only reason that the R scheme is faster is because of the poor choice of triangulation that it yields.

Memory complexity. For all of the methods under consideration, memory usage is dominated by the triangulation data structure and a few auxiliary data structures (such as priority queues) whose size grows (approximately) proportionally to mesh size. Therefore, memory usage is essentially determined by mesh size, and correspondingly, the peak memory usage is determined by the peak mesh size. For an image of width W, height H, and a given sampling density D, the peak mesh size for each of the proposed, GH, and R methods is DWH, whereas the peak mesh size for the AT scheme is WH (since the AT scheme begins with a mesh containing all WH sample points from the original image). So, the peak memory usage for the proposed, GH, and R methods is essentially the same, while, for values of D of practical interest, say $D \in [0.125\%, 3\%]$, the AT method requires 33 to 800 times more memory than the proposed, GH, and R methods. So, in terms of memory usage, our proposed method compares very favorably, being tied for best (i.e., lowest

peak memory usage) with the GH and R schemes and requiring orders of magnitude less memory than the AT scheme.

Summary. To summarize, through experimental results, our proposed mesh-generation method was shown to produce much higher quality meshes (both in terms of PNSR and subjective quality) than those generated by the three competing schemes, namely, the GH, R and AT schemes. Furthermore, our proposed method was found to require very substantially less computation and memory than the AT scheme. Relative to the GH and R schemes, our proposed method was seen to have the same memory cost and comparable computational cost.

Figure 3.18: Part of the image approximation obtained for the `lena` image at a sampling density of 1% with the (a) proposed (30.16 dB), (c) GH (25.27 dB), (e) R (19.34 dB), (g) GH2 (28.51 dB), (i) R2 (24.20 dB), and (k) AT (29.12 dB), methods and (b), (d), (f), (h), (j), and (l) their corresponding triangulations.

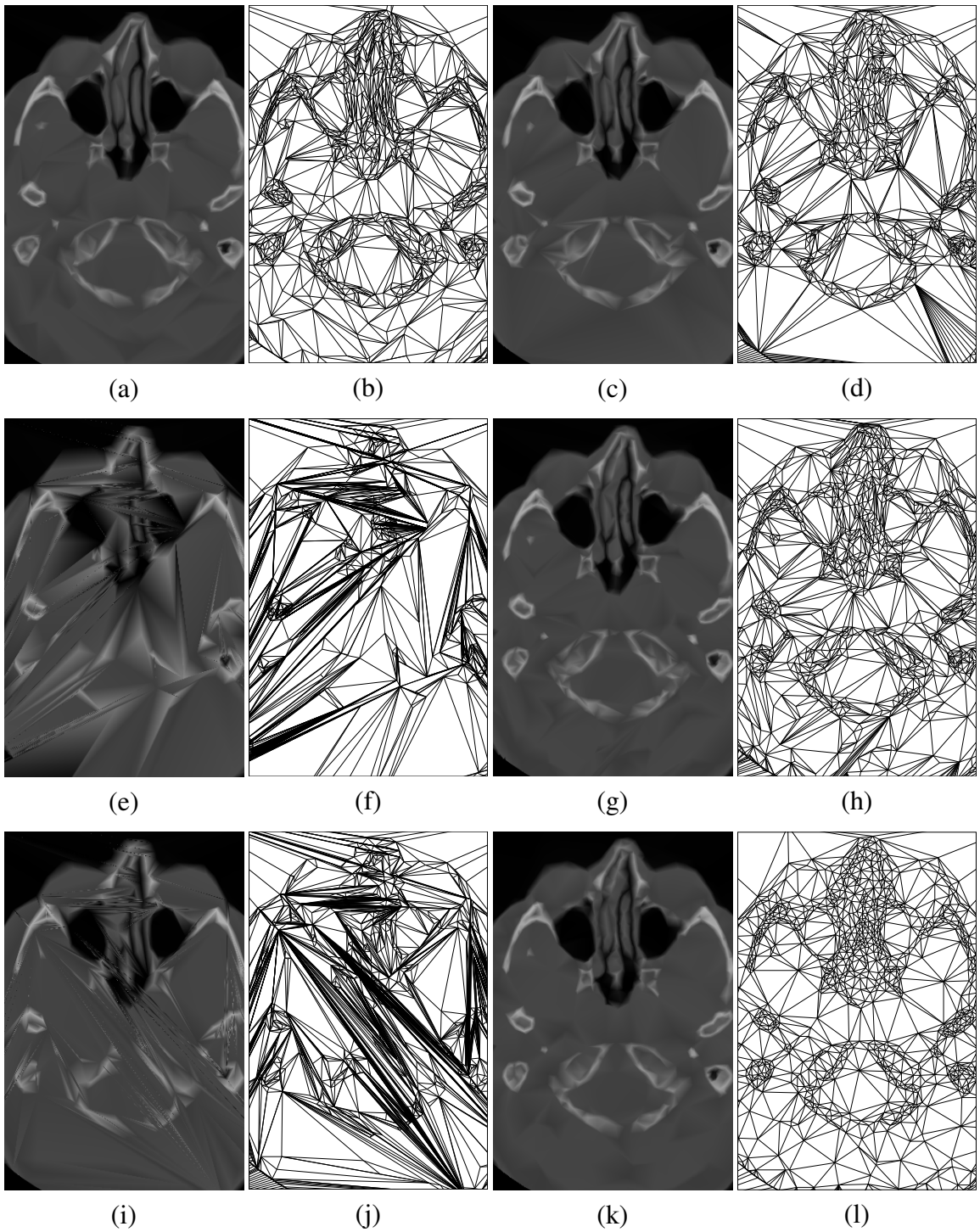


Figure 3.19: Part of the image approximation obtained for the `ct` image at a sampling density of 0.5% with the (a) proposed (38.70 dB), (c) GH (35.23 dB), (e) R (25.92 dB), (g) GH2 (37.68 dB), (i) R2 (29.61 dB), and (k) AT (37.22 dB), methods and (b), (d), (f), (h), (j), and (l) their corresponding triangulations.

Chapter 4

Conclusions and Future Research

4.1 Conclusions

In this thesis, we have studied DDT-based mesh-generation schemes for image representation. In particular, we have proposed a new DDT-based mesh-generation framework for image representation, derived by making a number of key modifications to the Rippa and Garland-Heckbert frameworks, including

1. the addition of a final connectivity-adjustment step,

2. the development of more effective edge-flip criteria for the LOP, and

3. the introduction of a better point-selection strategy.

As the proposed framework consists of several free parameters, the effects of different choices of each parameter on mesh quality (both in terms of PSNR and subjectively) are studied, leading to the recommendation of a particular set of choices for these parameters. Then a highly-effective mesh-generation method, obtained by choosing the best choice for each free parameter within our framework, was proposed.

Through experimental results, our proposed mesh-generation method was shown to produce meshes of significantly higher quality than those generated by the three competing schemes, namely, the GH, R, and AT schemes. In terms of computational cost, our proposed method was found to be comparable (for the reasons given earlier) to the GH and R schemes and to require over an order of magnitude less computation time than the AT method. In terms of memory cost, our proposed method was shown to require essentially same amount of memory as the GH and R schemes and orders of magnitude less memory

than the AT scheme. In short, our proposed method yields meshes of very high quality at reasonably low computational and memory costs. Consequently, our method is of great benefit to the many applications that employ mesh models of images. Moreover, by further exploring the algorithmic possibilities afforded by our proposed framework, we are optimistic that even more effective mesh-generation schemes can be synthesized.

4.2 Future Research

Although this thesis has made significant contributions on DDT-based mesh-generation schemes, further work in this area would still be beneficial. Some future research areas are discussed in what follows.

As mentioned earlier, when certain non-well-behaved edge-flip criteria are employed in our framework, sometimes cycles can occur in the LOP. A naive approach has been employed to avoid cycles in some mesh-generation methods. In particular, we set a limit (five) for the number of times each edge can be tested for optimality in each invocation of the LOP. Then if this limit is exceeded (i.e., a certain edge has been tested for optimality for more than 5 times), our algorithm will drop the edge under consideration and never test it again in the same LOP process. Perhaps, a more elegant way can be found to avoid cycles.

In the LOP used in our framework, the order of testing or flipping suspect edges is set to be LIFO. Although several other schemes were explored during the development of our framework, none of them performs constantly better than others. As Dyn, Levin and Rippa stated in [28], mesh quality might potentially be improved with better edge-testing orders for the LOP. The author of this thesis believes that this is worth of further experimentation. That is, we can assign a certain priority to each suspect edge, and then test the edges for optimality in decreasing order of priority. For example, the priority for an edge e could chosen to be related to the $\text{edgeCost}_{\mathsf{SE}}$ of e or the shape qualities of the two faces sharing e.

Yu, Morse, and Sederberg proposed a mesh-generation method that employs a so called "look-ahead" scheme in [27]. Both our proposed method and theirs are greedy schemes that employ the LOP to optimize the connectivity of a triangulation. One major difference between our method and theirs is that our method only considers impact on current iteration, while their method considers the impact on the current and the next iteration. Thus their method has the potential to achieve meshes with higher quality since the look-ahead scheme can reduce the possibility of being trapped in a local minimum. As we can foresee, the look-ahead scheme will be computational expensive, but if we can implement it in a

more intelligent fashion, it would still be potentially valuable.

Appendix A

Software User Manual

A.1 Introduction

During the course of the research presented in this thesis, software that implements the proposed mesh-generation framework and methods based on DDTs was developed by the author with guidance from her supervisor. This software was written in C++, and consists of more than 8500 lines of code (approximately 130 pages if printed). It involves some fairly complicated algorithms and data structures, and it also utilizes two libraries, namely the Computational Geometry Algorithm Library (CGAL) and Signal Processing Library (SPL).

A large number of mesh generation methods can be derived from our framework, since the framework contains several free parameters, and each one has many choices to consider. The software takes an image, a target number of sample points, and a specific mesh-generation method, which is represented by a set of sub-parameters, as basic inputs, and produces a triangle mesh for the input image, an image approximation reconstructed from the mesh, and mesh quality information (e.g., PSNR).

In the remainder of this appendix, we will introduce the software in more detail including such information as

1. how to build the software;
2. the functionality of the software;
3. the organization of the source code; and
4. how to use the software.

A.2 Building the Software

In order to use our software, the first thing one needs to do is to build the software. The term software build refers to the process of converting source code files into standalone software artifacts that can be run on a computer. One of the most important steps of a software build is the compilation process where source code files are converted into executable code, which is also the step we will talk about herein.

This software was developed under Linux using C++. The building method herein is based on the well known build tool `make` provided by most UNIX (or UNIX-like) systems. The `make` command can automatically build executable programs and libraries from the source code by reading files called makefiles which specify how to derive the target program. As mentioned earlier, this software requires some particular libraries, namely, CGAL and SPL. Thus in order to build the software, CGAL and SPL must first be installed. The CGAL version we have been using for this software is 3.5.1, and the SPL version we have been using is 1.0.27.

In order to compile all the source files and link the object files, you need to set the directory to the top level directory for the software. Then, to delete all the object files and executable files that generated during the previous building process, type:

```
make clean
```

To compile all the source files and link the object files, and type:

```
make
```

A.3 Software Functionality

Generally speaking, this software can accomplish three tasks:

1. generate a mesh for a given input image, a target number of sample points and a specific mesh-generation method;
2. reconstruct an image approximation from the mesh; and
3. output some information regarding the quality of the mesh.

The information this software can produce in step 3 above includes:

- PSNR value, mean square error, and peak absolute error of the reconstructed image;

- time cost of the program;
- a polyline format data file that shows all the non-Delaunay edges in the mesh;
- a polyline format data file that shows all the flippable edges in the mesh;
- the number of non-Delaunay edges, the number of total edges in the mesh;
- an file, similar to OFF format, that shows all the vertices in 3D; and
- a text file that records every change to the mesh during the mesh-generation process.

A.4 Organization of Source Code

The source code consists of a number of files of various functionalities. In the list that follows, each source file of the software is briefly described.

`Array2.hpp`	Contains definition of a 2-D array class, `Array2`, which is used to represent images in our software.
`elements.hpp`	Contains modified vertex and face classes that derived from CGAL library, with extra information inside them.
`globalHeadFile.hpp`	Contains various macro definitions.
`makemesh.cpp`	Contains the main `MeshGenerator` class and the main function.
`makemesh.hpp`	Contains some types that specify all the available options for face-selection method, candidate-selection method, and edge-flip criterion.
`mathUtil.hpp`	Contains some basic mathematical functions used in our software.
`outputTri.cpp`	Contains functions that output information regarding the mesh quality.
`PriQue.hpp`	Contains a heap-based priority queue template class.
`read_write_PGM.cpp`	Contains functions that read and write images as inputs and outputs. Our software only takes binary PGM format images as input, and the maximum pixel value of the image is 65535(16 bits).

`read_write_PGM.hpp`	Contains declarations of all the functions in `read_write_PGM.cpp`.
`scanConvert.cpp`	Contains functions that scan-convert the mesh; it also contains several functions that calculate the derivatives of input images and shape quality of triangles.
`scanConvert.hpp`	Contains declarations of all the functions in `scanConvert.cpp`.
`Util.cpp`	Contains definitions of several functions that output extra information for debugging
`Util.hpp`	Contains declarations of all the functions in `Util.cpp`.

A.5 Application Programs

In order to demonstrate how to use our software, first, we will briefly introduce the `makemesh` command. Then, we will introduce the software from three aspects: the inputs of the software, the outputs of the software, and some examples of mesh-generation schemes together with their corresponding commands.

A.5.1 The `makemesh` Command

Synopsis

```
makemesh -i inputImage -n sampleDensity -k faceSelMethod \
  -c edgeFlipCri -p candPointSel  [options]
makemesh -i inputImage -N targetNumber -k faceSelMethod  \
  -c edgeFlipCri -p candPointSel  [options]
```

Description

The `makemesh` command runs the program to generate a mesh for a given image based on a given sampling density or a target number of sample points, a face-selection method, a main edge-flip criterion, and a candidate-point-selection method. Besides the mesh output, the command can also produce a reconstructed image and other information regarding the mesh quality. by adding output options to the command.

Table A.1: Options for Potential Cycle Problem

Number	Description
0	continue the program without any warning(Default setting)
1	print out the iteration number and the two end points of the edge that causes the cycle, then continue the program
2	print out the iteration number and the two end points of the edge that causes the cycle, then exit the program

Options

The `makemesh` program accepts the following options:

`-A loopAlert` Sets the response mode when a potential loop is detected to `loopAlert`. The program can get stuck in a cycle because we allow for the use of some non-well-behaved edge-flip criteria. There are 3 options available herein and they are listed in Table A.1.

`-b binaryFile` Outputs an image file, `binaryFile`, that shows the locations of all the sample points in the output mesh.

`-B dumpTri` Outputs a `dumpTri` file that contains the triangulation generated by the main mesh-generation process (i.e., without the post processing).

`-c edgeFlipCri` Sets the edge-flip criterion, `edgeFlipCri`, in the main mesh-generation process. This option has to be provided, and a number of criteria are available for this option. They are listed in Table A.2.

Table A.2: Edge Flip Criteria

Number	Criterion Description
0	Delaunay criterion (D) described in Section 3.3.2.
1	Squared error criterion (SE) described in Section 3.3.2.
2	Shape-quality-weighted SE (SQSE) criterion described in Section 3.3.2.
3	Garland-Heckbert hybrid (GHH) criterion described in Section 3.3.2. This criterion involves another command line option, `-s`, which is used to set the threshold for GHH criterion.

Continued on next page

Table A.2 – continued from previous page

Number	Criterion Description
10	JND-weighted SE (JNDSE) criterion described in Section 3.3.2.
12	ABN-weighted SE (ABNSE) criterion. It uses the product of ABN and SE of an edge as the edge-cost function, and this criterion is associated with edge-cost functions that assign a cost to every flippable edge in the triangulation.
13	Edge-length-weighted JND (ELJND) criterion. It uses the product of JND and the length of the edge as edge-cost function. This criterion is associated with edge-cost functions that assign a cost to every flippable edge in the triangulation.
20	Angle between normal(ABN) criterion described in Section 3.3.2.
21	Jump in normal derivatives(JND) criterion described in Section 3.3.2.
22	Distances between planes(DP) criterion described in Section 3.3.2.
23	Deviation from linear polynomial(DLP) criterion described in Section 3.3.2.
24	Edge-length-weighted ABN (ELABN) criterion described in Section 3.3.2.
25	Edge-length-weighted JND (ELJND) criterion described in Section 3.3.2.
26	Edge length (EL). It uses the edge length of a edge e as the edge-cost function that decides whether to flip the edge e or not. If edge e has a strictly higher cost then it is flipped to counterpart e'.
28	Yu-Morse-Sederberg cost function (YMS) criterion described in Section 3.3.2.

`-C postFlipOption` Chooses the edge flip criteria (`postFlipOption`) used for the post edge processing. The available choices are the same as for the `-c`

option, which are listed in Table A.2.

`-d nonDelaunayFile` Outputs a polyline format data file (`nonDelaunayFile`) that contains all the non-Delaunay edges.

`-f postFlag` Use `postFlag` to indicates whether the mesh-generation method has a final edge connectivity adjustment step. There are two choices for this flag:

- 0–disable the final edge connectivity adjustment (default setting);
- 1–enable the final edge connectivity adjustment;

`-F flippableEdge` Outputs a polyline format data file (`flippableEdge`) that contains all the flippable edges in the output mesh.

`-g propagatingFlag` Use `propagatingFlag` to indicates whether the final edge connectivity adjustment will be propagating or not. The two available choices are:

- 0–disable the propagating property (default setting);
- 1–enable the propagating property.

`-G globalOpt` Use `globalOpt` to indicates whether to do a global edge connectivity adjustment or a local edge one after each vertex got inserted into the mesh. The two available choices are:

- 0–local edge connectivity adjustment (default setting for this option);
- 1– global edge connectivity adjustment.

`-H meshHistoryFile` If enabled, our software will output a text file (`meshHistoryFile`) that contains every changes of the mesh during the mesh-generation process, regarding point insertion, edge flip-ability test, edge optimality test and edge flip.

`-i inputImage` Takes the input image for the program. The image format for our program is portable graymap PGM with binary encoding.

Table A.3: Candidate Point Selection Choices

Number	Choice Description
0	Peak absolute error (`PAE`) described in Section 3.3.1.
1	Peak weighted absolute error(`PWAE`) described in Section 3.3.1.
2	Approximate minimum squared error based on `PWAE` (`AMSE-PWAE`) described in Section 3.3.1.
3	Approximate minimum squared error based on `PAE` (`AMSE-PAE`) described in Section 3.3.1.
5	Hybrid policy (`hybrid`) described in Section 3.3.1.

`-k faceSelMethod` Provides with two methods to select a face from the triangulation to insert a candidate point. The two available choices are:

- 0–select face with maximum squared error (GAE);
- 1–select face with point of maximum absolute error (GSE).

`-n sampleDensity` Sets the sampling density for mesh-generation process. The range can be any number between [0, 1]. Either this option or `-N` option has to be provided.

`-N pointsNumber` Sets the target (integer) number of sample points to be inserted into the mesh. Either this option or `-n` option has to be provided.

`-o ouputImage` Outputs the reconstructed image.

`-O offFile` Outputs a file (`offFile`), that contains all the vertices in the mesh and their connectivity in 3D.

`-p candPointSel` Sets the candidate-point-selection method. Refer to Table A.3 for more detail about each choice.

`-r infoResults` Outputs a text file (`infoResults`) contains information about the quality of the output mesh.

`-s threshold` Sets the threshold for GHH criterion. This threshold is used to switch the edge flip criterion between SE and shape quality. If `-s` is set to 1, then it uses only shape quality as edge flip criterion; while `-s` is set to 0, it only uses SE as edge flip criterion. 0.5 is

	the default value and also the value that generates the meshes with highest PSNR [17].
`-t triFile`	Outputs a file (`triFile`) contains the triangulation generated by the method.
`-T iterTri`	Outputs two triangulations files (`iterTri*a.tri` and `iterTri*b.tri`) in each vertex-insertion iteration, one before edge adjustment and one after.
`-W useWindow`	Use flag `useWindow` to indicates whether to use a 3 by 3 window during the candidate-point-selection process. There are two choices for this flag: • 0–disable the window functionality (default setting); • 1–enable the window functionality;

A.5.2 Inputs of the Software

With all the command line options introduced above, we now divide them into two categories (input and output of the software) for the readers to better understand how to use those options. Generally speaking, our software takes three main inputs:

1. an input image,
2. a sampling density (or a target number of sample points), and
3. a specific mesh-generation method.

We will describe each of them in the following.

Our software only takes portable graymap format images with binary encoding as input images, that is, PGM images with magic number of `P5`. The command line option for input image is `-i`, which has to be provided. Our software can handle up to 16 bits/sample images. The P2 (ASCII encoding) format PGM images are not included, since this format is less commonly used.

The second input for our software is the sampling density or the target number of sample points. The command line option for sampling density is `-n` and we use a real number to indicate the density. The command line option for the target number of sample points is

`-N` and it takes positive integer numbers. The commonly used range of sampling density is between 0.125% and 0.3%. Our software needs one and only one of these two options.

The mesh-generation method is the last input parameter of our software. Each method includes several sub-parameters as follows:

- the face-selection method, which corresponds to the `-k` option;
- the candidate-point-selection method, which corresponds to the `-p` option;
- the main edge-flip criterion, which corresponds to the `-c` option;
- whether to employ the final edge-connectivity adjustment or not, which corresponds to the `-f` option;
- the edge-flip criterion for the final edge-connectivity adjustment if it is enabled, which corresponds to the `-C` option;
- whether the final edge-flip adjustment should be edge propagating or non-propagating, which corresponds to the `-g` option;

Some additional input options are `-A`, `-G`, `-W`, and `-s`.

A.5.3 Outputs of the Software

Now let us talk about the outputs of our software. All the outputs are optional, meaning we can omit all of the output options. We, however, usually want to output three main items:

1. the mesh, which corresponds to the `-t` option;
2. the reconstructed image, which corresponds to the `-o` option; and
3. a text file that includes mesh quality information, which corresponds to the `-r` option.

The output triangulation data file with `tri` as extension consists of the following fields (in order), with fields being separated by white space:

- The total number of vertices in the triangulation.
- The total number of faces in the triangulation.
- For each vertex, its x and y coordinates.

- For each face, the indices of the three vertices, where the first vertex has index 0, the second vertex has index 1, and so on.

The output image format corresponds with that of the input image (P5). All the information regarding the mesh quality that our program generates will be stored in a `txt` file. This file contains the following information:

- PSNR, mean square error, and the peak absolute error of the reconstructed image;
- the number of vertices that have the different function value as the input image;
- time cost of the program;
- the number of non-Delaunay edges in the mesh;
- the total number of edges in the mesh;
- the number of edges that can be validly flipped in the mesh; and
- the memory usage.

One needs to note that the number of total edges and non-Delaunay edges are only calculated when `-d` option is enabled and the number of flippable edges is computed only when `-F` option is enabled. If the corresponding option are not enabled, then these three numbers will be initialized to -1.

Other available output options are `-b`, `-B`, `-H`, `-O`, and `-T`. The ploy line format data file has `dat` as file extension, and it is compatible with the `iviewer` program developed by Dr. Michael Adams. The `iviewer` program is an application that takes `pnm` format images, triangulation files (with `tri` extension) and polyline format files (with `dat` extension) as inputs and displays them for visualization purposes.

The `-O` option will generate a file that has a similar format as an OFF file as follows. This format can be used to view the triangulation mesh in 3D.

> the number of vertex the number of faces the number of edges
>
> the x, y, z coordinates (x y z) for each vertex form 0, 1, ..., to vertex_count-1 (one line for each vertex)
>
> the indices of three vertices (i j k) of each face in the triangulation (one line for each triangle face)

A.5.4 Examples of Mesh-Generation Schemes

Here in this section, some mesh-generation methods and their corresponding commands will be provided as examples to show how to use our software.

Example A:

Suppose we want to generate a mesh for the `lena` image with the following requirements:

- target sampling density sets to 0.1%;
- use GSE face-selection method;
- use AMSE-PAE as candidate-selection method;
- use SQSE as main edge-flip criterion;
- employ propagating final edge connectivity adjustment with SE edge-flip criterion;
- output the reconstructed image and mesh quality information, such as the PSNR values, the time cost for this program.

We can type the following command line to obtain the mesh that satisfies the above requirements.

```
makemesh -i lena.pnm -n 0.001 -k 1 -p 3 -c 2 -f 1 -g 1 -C 1 \
 -t triMesh.tri -o reconsImage.pnm  -r stat.txt
```

Example B:

Suppose we want to output more information about the changes made to the mesh during the mesh-generation process. Take the `peppers` image as the input, and generate a mesh with the following requirements:

- insert 2500 sample points;
- use GAE face-selection method;
- use PAE candidate-selection method;
- use ABN as main edge-flip criterion;
- no final edge connectivity adjustment;

- output a reconstructed image together with some mesh quality information, such as the PSNR values, the time cost for this program.
- output a file that contains all the flippable edges in the output mesh.

In order to accomplish the above, we can type:

```
makemesh -i peppers.pnm -N 2500 -k 0 -p 0 -c 20 -f 0 -t \
 triMesh.tri -o reconsImage.pnm -r stat.txt \
 -F flippableEdgeFile.txt
```

Example C:

Suppose we want to use the proposed mesh-generation method to generate a mesh for the `bull` image with sampling density sets to 0.5%. As we know, the proposed mesh-generation method set the parameters of our framework as follows:

- use GSE face selection method;
- use hybrid as candidate-selection method;
- use JNDSE as main edge-flip criterion;
- enable the propagating final edge-connectivity adjustment with SE edge-flip criterion;

Besides the mesh, we also want to output a reconstructed image, a file contains some mesh quality information, such as the PSNR values and the time cost for this program. The command to generate such a mesh using the proposed method is as follows:

```
makemesh -i bull.pnm -n 0.005 -k 1 -c 10 -p 5 -f 1 -g 1 -C 1 \
  -o lena_0.01.pnm -t lena_0.01.tri -r lena_0.01_stat.txt
```

Bibliography

[1] S. A. Coleman, B. W. Scotney, and M. G. Herron, "Image feature detection on content-based meshes," in *Proc. of IEEE International Conference on Image Processing*, vol. 1, 2002, pp. 844–847.

[2] M. Petrou, R. Piroddi, and A. Talebpour, "Texture recognition from sparsely and irregularly sampled data," *Computer Vision and Image Understanding*, vol. 102, pp. 95–104, 2006.

[3] M. Sarkis and K. Diepold, "A fast solution to the approximation of 3-D scattered point data from stereo images using triangular meshes," in *Proc. of IEEE-RAS International Conference on Humanoid Robots*, Pittsburgh, PA, USA, Nov. 2007, pp. 235 – 241.

[4] J. G. Brankov, Y. Yang, and N. P. Galatsanos, "Image restoration using content-adaptive mesh modeling," in *Proc. of IEEE International Conference on Image Processing*, vol. 2, 2003, pp. 997–1000.

[5] J. G. Brankov, Y. Yang, and M. N. Wernick, "Tomographic image reconstruction based on a content-adaptive mesh model," *IEEE Trans. on Medical Imaging*, vol. 23, no. 2, pp. 202–212, Feb. 2004.

[6] M. A. Garcia and B. X. Vintimilla, "Acceleration of filtering and enhancement operations through geometric processing of gray-level images," in *Proc. of IEEE International Conference on Image Processing*, vol. 1, Vancouver, BC, Canada, 2000, pp. 97–100.

[7] D. Su and P. Willis, "Image interpolation by pixel-level data-dependent triangulation," *Computer Graphics Forum*, vol. 23, no. 2, pp. 189–201, 2004.

[8] M. D. Adams, "Progressive lossy-to-lossless coding of arbitrarily-sampled image data using the modified scattered data coding method," in *Proc. of IEEE International*

Conference on Acoustics, Speech, and Signal Processing, Taipei, Taiwan, Apr. 2009, pp. 1017–1020.

[9] G. Ramponi and S. Carrato, "An adaptive irregular sampling algorithm and its application to image coding," *Image and Vision Computing*, vol. 19, pp. 451–460, 2001.

[10] P. Lechat, H. Sanson, and L. Labelle, "Image approximation by minimization of a geometric distance applied to a 3D finite elements based model," in *Proc. of IEEE International Conference on Image Processing*, vol. 2, 1997, pp. 724–727.

[11] Y. Wang, O. Lee, and A. Vetro, "Use of two-dimensional deformable mesh structures for video coding, part II–the analysis problem and a region-based coder employing an active mesh representation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, no. 6, pp. 647–659, Dec. 1996.

[12] F. Davoine, M. Antonini, J.-M. Chassery, and M. Barlaud, "Fractal image compression based on Delaunay triangulation and vector quantization," *IEEE Trans. on Image Processing*, vol. 5, no. 2, pp. 338–346, Feb. 1996.

[13] K.-L. Hung and C.-C. Chang, "New irregular sampling coding method for transmitting images progressively," *IEE Proceedings Vision, Image and Signal Processing*, vol. 150, no. 1, pp. 44–50, Feb. 2003.

[14] M. D. Adams, "An efficient progressive coding method for arbitrarily-sampled image data," *IEEE Signal Processing Letters*, vol. 15, pp. 629–632, 2008.

[15] ——, "An evaluation of several mesh-generation methods using a simple mesh-based image coder," in *Proc. of IEEE International Conference on Image Processing*, San Diego, CA, USA, Oct. 2008, pp. 1041–1044.

[16] L. Demaret and A. Iske, "Advances in digital image compression by adaptive thinning," in *Annals of the Marie-Curie Fellowship Association*. Marie Curie Fellowship Association, Feb. 2004, vol. 3, pp. 105–109.

[17] M. Garland and P. S. Heckbert, "Fast polygonal approximation of terrains and height fields," School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU-CS-95-181, Sep. 1995.

[18] K. Wang, C.-P. Lo, G. A. Brook, and H. R. Arabnia, "Comparison of existing triangulation methods for regularly and irregularly spaced height fields," *International Journal of Geographical Information Science*, vol. 15, no. 8, pp. 743–762, 2001.

[19] M. D. Adams, "An incremental/decremental delaunay mesh-generation framework for image representation," in *Proc. of IEEE International Conference on Image Processing*, 2011, pp. 189–192.

[20] M. Garland and P. S. Heckbert, "Fast triangular approximation of terrains and height fields," draft manuscript, 1997, dated May 2, 1997 (19 pages).

[21] S. Rippa, "Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 5, pp. 1123–1141, 1992.

[22] N. Dyn, D. Levin, and S. Rippa, "Data dependent triangulations for piecewise linear interpolation," *IMA Journal of Numerical Analysis*, vol. 10, pp. 137–154, 1990.

[23] N. Dyn, "Data-dependent triangulations for scattered data interpolation and finite element approximation," *Applied Numerical Mathematics*, vol. 12, pp. 89–105, 1993.

[24] E. Quak and L. L. Schumaker, "Least squares fitting by linear splines on data dependent triangulations," in *Curves and Surfaces*, P. J. Laurent, A. L. Mehaute, and L. L. Schumaker, Eds. Boston, MA, USA: Academic Press, 1991, pp. 387–390.

[25] L. Alboul, G. Kloosterman, C. Traas, and R. van Damme, "Best data-dependent triangulations," *Journal of Computational and Applied Mathematics*, vol. 119, pp. 1–12, 2000.

[26] J. Weisz and R. Bodnar, "A refined "angle between normals" criterion for scattered data interpolation," *Computers and Mathematics with Applications*, vol. 41, pp. 531–534, 2001.

[27] X. Yu, B. S. Morse, and T. W. Sederberg, "Image reconstruction using data-dependent triangulation," *IEEE Computer Graphics and Applications*, vol. 21, no. 3, pp. 62–68, May 2001.

[28] N. Dyn, D. Levin, and S. Rippa, "Algorithms for the construction of data dependent triangulations," in *Algorithms for Approximation II*, J. C. Mason and M. G. Cox, Eds. London: Chapman and Hall, 1990, pp. 185–192.

[29] C. L. Lawson, "Software for C^1 surface interpolation," in *Mathematical Software III*, J. R. Rice, Ed. New York, NY, USA: Academic Press, 1977, pp. 161–194.

[30] J. L. Brown, "Vertex based data dependent triangulations," *Computer Aided Geometric Design*, vol. 8, pp. 239–251, 1991.

[31] D. Su and P. Willis, "Demosaicing of colour images using pixel level data-dependent triangulation," in *Proc. of the Theory and Practice of Computer Graphics*, 2003, pp. 16–23.

[32] B. Lehner, G. Umlauf, and B. Hamann, "Image compression using data-dependent triangulations," *Lecture Notes in Computer Science*, vol. 4841, pp. 351–362, 2007.

[33] M. Bertram, J. C. Barnes, B. Hamann, K. I. Joy, H. Pottmann, and D. Wushour, "Piecewise optimal triangulation for the approximation of scattered data in the plane," *Computer Aided Geometric Design*, vol. 17, pp. 767–787, 2000.

[34] P. K. Agarwal and S. Suri, "Surface approximation and geometric partitions," in *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, Jan. 1994, pp. 24–33.

[35] Y. Yang, M. Wernick, and J. Brankov, "A fast approach for accurate content-adaptive mesh generation," *Image Processing, IEEE Transactions on*, vol. 12, no. 8, pp. 866 – 881, aug. 2003.

[36] R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial greyscale," *Proceedings of the Society for Information Display*, vol. 17, no. 2, pp. 75–77, 1976.

[37] Y. Yang, J. Brankov, and M. Wernick, "Content-adaptive mesh modeling for fully-3d tomographic image reconstruction," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 2, 2002, pp. II–621 – II–624 vol.2.

[38] M. A. Garcia and A. D. Sappa, "Efficient generation of discontinuity-preserving adaptive triangulations from range images," *IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 34, no. 5, pp. 2003–2014, Oct. 2004.

[39] N. Dyn, M. S. Floater, and A. Iske, "Adaptive thinning for bivariate scattered data," *Journal of Computational and Applied Mathematics*, vol. 145, pp. 505–517, 2002.

[40] H. Weimer and J. Warren, "Fast approximating triangulation of large scattered datasets," *Advances in Engineering Software*, vol. 30, pp. 389–400, 1999.

[41] B. Delaunay, "Sur la sphere vide," *Bulletin of the Academy of Sciences of the USSR, Classe des Sciences Mathematiques et Naturelle*, vol. 7, no. 6, pp. 793–800, 1934.

[42] C. Dyken and M. S. Floater, "Preferred directions for resolving the non-uniqueness of Delaunay triangulations," *Computational Geometry—Theory and Applications*, vol. 34, pp. 96–101, 2006.

[43] M. D. Adams, "A flexible content-adaptive mesh-generation strategy for image representation," *IEEE Transactions on Image Processing*, vol. 20, no. 9, pp. 2414–2427, 2011.

[44] S. Rippa, "Long and thin triangles can be good for linear interpolation," *SIAM Journal on Numerical Analysis*, vol. 29, no. 1, pp. 257–270, 1992.

[45] Z. Toth, I. Viola, A. Ferko, and E. Groller, "*n*-dimensional data-dependent reconstruction using topological changes," in *Topology-Based Methods in Visualization*, ser. Mathematics and Visualization, H. Hauser, H. Hagen, and H. Theisel, Eds. New York: Springer, 2007, pp. 183–198.

[46] L. Rila and A. G. Constantinides, "Image coding using data-dependent triangulation," in *Proc. of International Conference on Digital Signal Processing*, vol. 2, 1997, pp. 531–534.

[47] L. L. Schumaker, "Computing optimal triangulations using simulated annealing," *Computer Aided Geometric Design*, vol. 10, pp. 329–345, 1993.

[48] O. Kreylos and B. Hamann, "On simulated annealing and the construction of linear spline approximations for scattered data," *IEEE Trans. on Visualization and Computer Graphics*, vol. 7, no. 1, pp. 17–31, Jan. 2001.

[49] P. Li and M. D. Adams, "An effective mesh-generation strategy for image representation using data-dependent triangulation," under review.

[50] ——, "An improved mesh-generation strategy for image representation base on data-dependent triangulation," under review.

[51] M. Aubury and W. Luk, "Binomial filters," *Journal of VLSI Signal Processing*, vol. 12, pp. 35–50, 1996.

[52] C. L. Lawson, “Transforming triangulations,” *Discrete Mathematics*, vol. 3, pp. 365–372, 1972.

[53] E. Osherovich and A. M. Bruckstein, “All triangulations are reachable via sequences of edge-flips: an elementary proof,” *Computer Aided Geometric Design*, vol. 25, pp. 157–161, 2008.

[54] X. Tu and M. D. Adams, “Image representation using triangle meshes with explicit discontinuities,” Victoria, BC, Canada, Aug. 2011, pp. 97–101.

[55] K. Fleischer and D. Salesin, “Accurate polygon scan conversion using half-open intervals,” in *Graphics Gems III*, 1995, pp. 362–365.

[56] “JPEG-2000 test images,” ISO/IEC JTC 1/SC 29/WG 1 N 545, Jul. 1997.

[57] “Kodak lossless true color image suite,” 2011. [Online]. Available: http://r0k.us/graphics/kodak

[58] “USC-SIPI image database,” 2011. [Online]. Available: http://sipi.usc.edu/database

[59] “Michael Adams’ research datasets,” 2011. [Online]. Available: http://www.ece.uvic.ca/~mdadams/datasets