Effective Techniques for Generating Delaunay Mesh Models of Single- and Multi-Component Images

by

Jun Luo
B.Eng., North China University of Technology, 2014

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

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Effective Techniques for Generating Delaunay Mesh Models of Single- and Multi-Component Images

by

Jun Luo
B.Eng., North China University of Technology, 2014

ABSTRACT

In this thesis, we propose a general computational framework for generating mesh models of single-component (e.g., grayscale) and multi-component (e.g., RGB color) images. This framework builds on ideas from the previously-proposed GPRFSED method for single-component images to produce a framework that can handle images with any arbitrary number of components. The key ideas embodied in our framework are Floyd-Steinberg error diffusion and greedy-point removal. Our framework has several free parameters and the effect of the choices of these parameters is studied. Based on experimentation, we recommend two specific sets of parameter choices, yielding two highly effective single/multi-component mesh-generation methods, known as MED and MGPRFS. These two methods make different trade offs between mesh quality and computational cost. The MGPRFS method is able to produce high quality meshes at a reasonable computational cost, while the MED method trades off some mesh quality for a reduction in computational cost relative to the MGPRFS method.

To evaluate the performance of our proposed methods, we compared them to three highly-effective previously-proposed single-component mesh generators for both grayscale and color images. In particular, our evaluation considered the following previously-proposed methods: the error diffusion (ED) method of Yang et al., the greedy-point-removal from-subset (GPRFSED) method of Adams, and the greedy-point removal (GPR) method of Demaret and Iske. Since these methods cannot directly handle color images, color images were handled through conversion to grayscale as a preprocessing step, and then as a postprocessing step after mesh generation, the grayscale sample values in the generated mesh were replaced by their corresponding color values. These color-capable versions of ED, GPRFSED, and GPR are henceforth referred to as CED, CGPRFSED, and CGPR, respectively.

Experimental results show that our MGPRFS method yields meshes of higher quality than the CGPRFSED and GPRFSED methods by up to 7.05 dB and 2.88 dB respectively, with nearly the same computational cost. Moreover, the MGPRFS method outperforms the CGPR and GPR methods in mesh quality by up to 7.08 dB and 0.42 dB respectively, with about 5 to 40 times less computational cost. Lastly, our MED method yields meshes of higher quality than the CED and ED methods by up to 7.08 and 4.72 dB respectively, where all three of these methods have a similar computational cost.

Contents

List of Tables

List of Figures

ACKNOWLEDGMENTS

This thesis would never have been written without the help and support from numerous people. I would like to express my thanks to certain individuals in particular:

To my supervisor Dr. Michael Adams. Thank you for your mentorship, help, and guidance throughout my graduate studies. Without your mentoring in my research project coding, result analysis, and academic writing, this thesis would not have been written. Thank you for your meticulous and earnest teaching in C++, which really is an asset for my career. Your hard working, your dedication to research and teaching have motivated me in my studies, and would continue influencing me in the future. It has been such a pleasure and honor working with you.

To my course instructors. I would like to express my sincere gratitude to the course instructors during my graduate studies, Dr. Wu-Sheng Lu, Dr. Sue Whitesides, Dr. Alexandra Branzan Albu, Dr. Wendy Myrvold, and Dr. Mihai Sima. Thank you for offering all such interesting lectures. I have learnt a lot related to image processing, signal processing, data structure and algorithms, and programming skills, which developed my skills and expand my knowledge.

To my friends and group members Jiacheng Guo, Yue Fang, and some other members. Thank you for all the help during my studies. I have learnt a lot from all of you and I am grateful for being in the same research group with you.

To my dearest parents. I would like to thank my dearest parents, Guifeng Shen and Fangneng Luo. Thank you for your unconditional love, support and trust. Your encouragement and support have given me the strength to overcome the difficulties.

DEDICATION

To my family.

Chapter 1

Introduction

1.1 Mesh Representation of Images

Digital images can be represented in various ways. Despite being the most straightforward and commonly used method for representing images, uniform (i.e., lattice-based) sampling is far from optimal. Due to the fact that most images are nonstationary, such sampling inevitably leads to oversampling in some regions and undersampling in others. This motivates an interest in nonuniform (i.e., content-adaptive) sampling for image representation. In nonuniform sampling, by intelligently choosing the sample points based on image content, the number of sample points used for representing the image can be greatly reduced while still maintaining good fidelity. Moreover, image representations based on nonuniform sampling can also have the ability to better capture geometric image structure, such as edges. Nonuniform sampling has shown to be useful in many applications such as: pattern recognition [1], feature detection [2], computer vision [3], tomographic reconstruction [4, 5], restoration [6], filtering [7], interpolation [8], and image/video coding [9–15].

Many approaches to nonuniform sampling have been proposed over the years, and those based on triangle meshes [16–31] have shown to be particularly efficient, and are the focus of our work herein. Triangle meshes are well suited to capturing geometric structure in images (i.e., edges and corners). In a triangle mesh model, every single triangle with only three sample points can cover a large region of an image, thus, significantly reducing the number of samples required for representing the image. Given a triangle mesh model, one can easily obtain the image approximation by performing linear interpolation over each triangle face.

In order to use a triangle mesh model to represent an image, a scheme to select a good subset of sample points from the original image to form the basis for a mesh model (of the image) is of particular importance. This is the so called mesh-generation problem and is of interest in this thesis.

1.2 Historical Perspective

As mentioned before, a great number of mesh-generation methods have been developed to date [16–31]. Based on how sample points are selected, most of the methods can be classified into three categories: one-shot methods, mesh-refinement methods and mesh-simplification methods. The first category of methods determine all sample points in one shot based on image content, and then construct a triangulation using the selected sample points [4–6, 17, 30, 31]. Among them, one of the most popular and effective methods is the error diffusion (ED) scheme proposed by Yang et al. [17], which employs Floyd-Steinberg error diffusion (FSED) [32] to select sample points such that their density is approximately proportional to the maximum-magnitude second-order directional derivative (MMSODD) of the image, and then triangulates the selected sample points using a Delaunay triangulation. The mesh-refinement and mesh-simplification methods can normally produce meshes with higher quality, but require more computational cost. A mesh-refinement scheme normally begins with a very coarse triangulation (typically with only the extreme convex hull points of the image domain as the vertices), and then repeatedly adds vertices, until the desired number of sample points is achieved. Some examples can be found in [16, 27, 29]. In contrast, a mesh-simplification scheme starts with a refined triangulation (e.g., with a large number of points on the sampling grid as the vertices), and then repeatedly removes vertices, until the desired mesh size is reached [21, 33]. One typical example is the greedy point-removal (GPR) method of Demaret and Iske [21], which is capable of generating meshes with very high quality, but often requires extremely high computational and memory cost.

In addition to how the sample points are selected, the various mesh-generation methods can also be classified by how the triangulation connectivity (i.e., how the vertices in the triangulation are connected by edges) is selected. The two most popular types of triangulation employed in the various mesh-generation methods are Delaunay triangulations and data-dependent triangulations (DDTs). In the case of Delaunay triangulations, the vertices are connected such that the interior of each constructed triangle's circumcircle contains no other vertices. This behavior maximizes the minimum interior angle of each triangle in the triangulation, reducing the possibility of sliver (i.e., long and thin) triangles appearing in the triangulation (which can led to high approximation error). Consequently, Delaunay triangulations are particularly favored in approximation applications [34]. The Delaunay triangulation of a set of points is not guaranteed to be unique if four or more points in the set are cocircular. In order to ensure the Delaunay triangulation of a set of points is uniquely determined, the technique of preferred directions [35] can be applied. More examples of mesh-generation methods using Delaunay triangulations can be found in [16–19,21–23]. In contrast, with a DDT, the vertices can be connected in an arbitrary manner. Examples of mesh-generation methods based on DDT include [16,24–26,28,29]. Due to the flexibility of connectivity offered by DDT, however, achieving good results comes at the expense of a significantly higher computational cost. Consequently, (preferred-direction) Delaunay triangulations are often preferred in practice over DDTs.

As mentioned before, many mesh-generation methods have been proposed over the years, however, most of them are proposed only for single-component (i.e., grayscale) images, with few (if any) methods for multi-component (e.g., RGB) images. This brings our interest in developing a mesh-generation framework that is capable of generating mesh models for multi-component images. In order to develop an efficient mesh-generation method for multi-component images, it is reasonable to use ideas from a good method for grayscale images as our starting point. Among the various mesh-generation methods for grayscale images, the GPRFSED method of Adams [18], which combines the ideas of the ED method of Yang et al. [17] and the GPR method of Demaret and Iske [21], has proven to be particularly effective. Experimental results have shown that the GPRFSED method is capable of generating meshes with quality comparable to, and in many cases better than, the GPR method, while requiring substantially less computational and memory cost. Consequently, the GPRFSED method serves as the foundation of our work in this thesis.

1.3 Overview and Contribution of the Thesis

In this thesis, we modify the grayscale-image mesh-generation method GPRFSED of Adams and propose a more general computational framework that is capable of generating mesh models for single- and multi-component images. Using our framework, we propose two specific mesh-generation methods, known as MED and MGPRFS, which can be seen as improved and extended versions of the ED and GPRFSED methods, respectively. Our proposed two methods have different tradeoff between mesh quality and computational cost. Both of them are applicable to images with an arbitrary number of components. We study and evaluate our proposed mesh-generation methods by focusing mainly on the two most common cases of single- and multi-component images, namely, grayscale and RGB-color images. As we will show later, for both RGB-color and grayscale images, our proposed methods are capable of generating meshes with higher quality than various other methods, while requiring similar or lower computational and memory costs. In addition, one optimality algorithm to the startup policy of the classical FSED, called mirroring method, is proposed and employed in our work, and is shown to improve the algorithm by addressing the startup-effect problem in the algorithm.

The remainder of this thesis is organized as three chapters and two appendixes. In what follows, we provide an overview of these chapters and appendixes.

In Chapter 2, we provide some background information to facilitate a better understanding of the work presented in this thesis. First, the notation and terminology used are introduced. Next, some basic concepts regarding image processing (e.g., smoothing filter and gradient operator) and computational geometry (e.g., convex hull and Delaunay triangulation) are presented. Subsequently, the triangle mesh model and the mesh-generation problem addressed in this thesis are formally introduced. In addition, the classical FSED algorithm is introduced. Lastly, we introduce three well-known grayscale image mesh-generation methods, namely ED [17], GPR [21], and GPRFSED [18]. Of these, the GPRFSED method of Adams [18], which derives from the ED [17] and GPR [21] methods, serves as the foundation of our work herein.

In Chapter 3, we present our proposed mesh-generation method for single- and multi-component images and the process by which the method was developed. First, we introduce a new startup policy for FSED that will be employed in our work. Then, we define our general mesh-generation framework for single- and multi-component images, which has several free parameters and many possible choices are considered for these parameters. Then, we introduce three modes that our framework can operate in. We study the effect of parameter choices in each of the three modes in detail, by exploring in depth the two most common cases of images, namely, RGB-color and grayscale images. Finally, based on experimental results, we advocate two specific mesh-generation methods, called MED and MGPRFS respectively, which correspond to the best set of parameter choices in two of our operation modes. These two methods make different tradeoffs between mesh quality and computational cost, and they are both applicable to images with an arbitrary number of components.

In Chapter 4, we evaluate the performance of our proposed methods on the mesh quality and computational cost by comparing with three highly-effective previously-proposed single-component mesh generators for both grayscale and RGB-color images, namely ED, GPRFSED, and GPR. Since these three methods cannot directly handle RGB-color images, RGB-color images are handled through conversion to grayscale as a preprocessing step, and then as a postprocessing step after mesh generation the grayscale sample values in the generated mesh are replaced by their corresponding RGB values. These color-capable versions of ED, GPRFSED, and GPR are henceforth referred to as CED, CGPRFSED, and CGPR, respectively. Experimental results show that our MGPRFS method can produce meshes with higher quality than the CGPRFSED and GPRFSED methods in mesh quality by up to 7.05 dB and 2.88 dB, where all three of these methods have a similar computational cost. Moreover, our MGPRFS method is shown to yield meshes of higher quality than the CGPR and GPR methods by up to 7.05 dB and 0.42 dB, while requiring about 5 to 40 times less computational cost. Lastly, our MED method is shown to outperform the CED and ED methods by up to 7.08 dB and 4.72 dB, with nearly the same computational cost.

In Chapter 5, we conclude the thesis by summarizing the key results of our work. In addition, some recommendations for future research are made.

In Appendix A, we present the test images employed in our work in more detail. Different images from various standard image test sets are presented.

In Appendix B, we provide an introduction to the software which implements the mesh-generation framework proposed in this thesis and is used to collect experimental results. The software is developed by the author of this thesis using C++, consists of more than 6000 lines of code, and contains many complex data structures and algorithms. The installation of the software is first presented, followed by introducing the file formats used. Then, each of the programs are introduced in detail with their available options. Lastly, some examples of how to use the software are provided.

Chapter 2

Preliminaries

2.1 Overview

In this section, we provide some background information that is essential for understanding the work presented in this thesis. To begin, we introduce the notation and terminology used in the remainder of this thesis. Then, some basic concepts from image processing and computational geometry are introduced. After that, the triangle-mesh models for single- and multi-component images are discussed, followed by introducing the mesh-generation problem addressed in this thesis in more detail. The chapter concludes with a description of the Floyd-Steinberg error diffusion (FSED) algorithm and three well-known grayscale-image mesh-generation methods, namely, ED, GPR, and GPRFSED. Of these, the GPRFSED method, which is derived from the ED and GPR methods, serves as the foundation of our work in this thesis.

2.2 Notation and Terminology

Before proceeding further, some basic notation and terminology used throughout the thesis is introduced. The sets of integers and real numbers are denoted as $\mathbb{Z}$ and $\mathbb{R}$, respectively. For a set P, the cardinality of P is denoted as $|P|$. When presenting equations and algorithms, the symbol ":=" is used to denote variable assignment. For two integers a and b, $[a..b] = \{x \in \mathbb{Z} : a \leq x \leq b\}$, $[a..b) = \{x \in \mathbb{Z} : a \leq x < b\}$, and $(a..b) = \{x \in \mathbb{Z} : a < x < b\}$.

2.3 Image Processing

In this section, some basic knowledge relating to smoothing filters and gradient operators employed in our work is introduced.

Binomial Filters. Binomial filters [36] are lowpass filters that approximate Gaussian filtering [37]. They do not require multipliers and can therefore be implemented efficiently in programmable hardware. Due to their simplicity and efficiency, binomial filters are particularly useful for smoothing in image processing applications [38]. The transfer function H_n of a one-dimensional (1-D) n-th order binomial filter (with zero phase and unity DC gain) is given by

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

where n is an odd integer. A 2-D binomial filter can be defined as the tensor product of two 1-D binomial filters. For example, the nonzero coefficients of the impulse response of a third-order 2-D binomial filter are

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix},$$

which is the tensor product of two 1-D third-order binomial filters with the coefficients $\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$. In order to reduce the influence of noise in the input image, a third-order binomial smoothing filter is found to be particularly effective in our work, and therefore, is always employed to smooth each image component when computing any first- or second-order derivatives of the image.

Laplacian. The Laplacian operator is a second-order differential operator for single-component images. For a function f defined on $\mathbb{R}^2$, the Laplacian [39] $L(x, y)$ of f at (x, y) is given by

$$L(x, y) = \frac{\partial^2}{\partial x^2} f(x, y) + \frac{\partial^2}{\partial y^2} f(x, y). \tag{2.1}$$

The second-order derivative operators are computed using the filter with the transfer function $z - 2 + z^{-1}$. Edges of the image can be obtained by thresholding $|L|$ (i.e., the magnitude of L at each point). The magnitude of Laplacian has the property that it yields a double response to image edges, with the maxima being attained along the two sides of an image edge. An example of a grayscale image and its magnitude of Laplacian are provided in Figures 2.1(a) and (b), respectively.

MMSODD. The maximum-magnitude second-order directional derivative (MMSODD) [17] is a quantity of interest for single-component (i.e., grayscale) images. For a function f defined on $\mathbb{R}^2$, the MMSODD $\tilde{d}(x, y)$ of f at (x, y) is given by

$$\tilde{d}(x,y) = \max\Big\{|\alpha(x,y) + \beta(x,y)|, |\alpha(x,y) - \beta(x,y)|\Big\}, \tag{2.2}$$

where

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

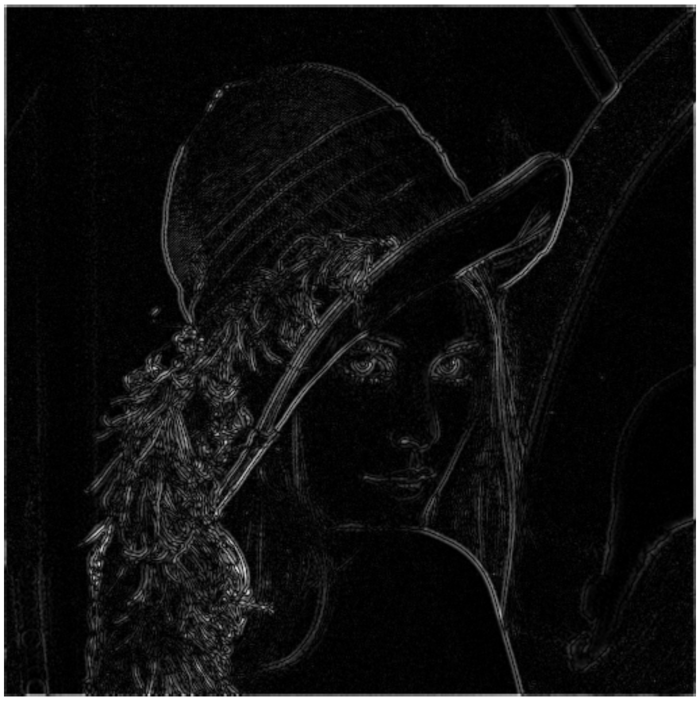
and

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

The 1-D second-order derivative operators $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$ are both computed using the filter with the transfer function $z - 2 + z^{-1}$. The derivative operator $\frac{\partial^2}{\partial x \partial y}$ is formed by computing the tensor product of two 1-D first-order derivative operators with the transfer function $\frac{1}{2}z - \frac{1}{2}z^{-1}$. The MMSODD also has the property that it yields a double response to image edges, and its response to image edges is shown to be stronger than the magnitude of Laplacian. This property turns out to be particularly important in our work. An example of a grayscale image and its MMSODD are shown in Figures 2.2(a) and (b), respectively.

Vector Gradient Operator. The vector gradient operator proposed by Di Zenzo [40] is a derivative operator to compute a single gradient for a RGB-color image, which has been widely used in color-image edge-detection applications [41]. Different from the MMSODD and Laplacian (which are defined for single-component images), the vector gradient operator is defined for 2-D three-component vector images. The vector gradient operator first obtains vector-space directional derivatives by computing the tensor product of two tangent vectors with respect to the X and Y spaces (i.e., the horizontal and vertical coordinate spaces), and then computes the gradient of the color image based on these derivatives. Let the RGB-color image be a vector function f with $f(x, y) = (R(x, y), G(x, y), B(x, y))$ at (x, y), and let $\mathbf{r}, \mathbf{g}, \mathbf{b}$ be the unit vectors

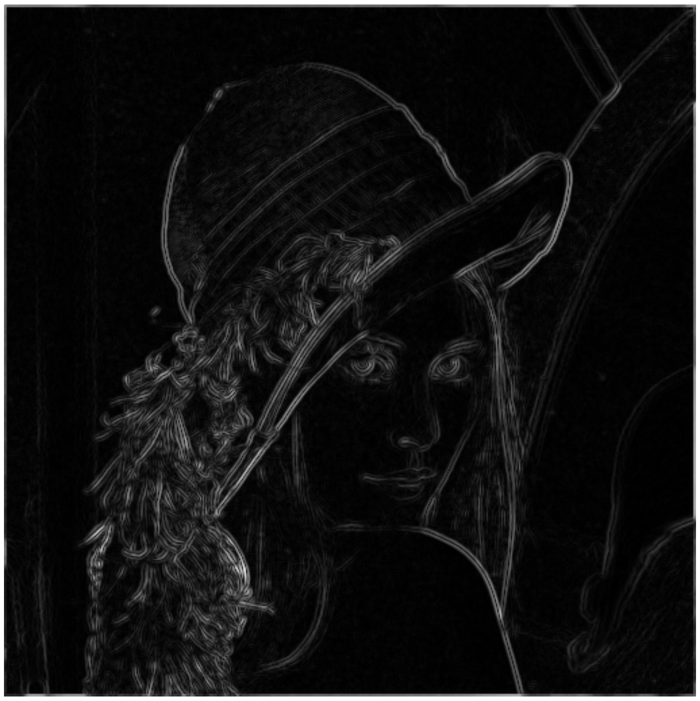
(a)



(b)

Figure 2.1: (a) A grayscale image and its (b) magnitude of Laplacian.

(a)



(b)

Figure 2.2: (a) A grayscale image and its (b) MMSODD.

along the R, G, B axes, respectively. The two tangent vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ with respect to the X and Y spaces are given by

$$\begin{aligned} \boldsymbol{u} &= \frac{\partial R}{\partial x}\boldsymbol{r} + \frac{\partial G}{\partial x}\boldsymbol{g} + \frac{\partial B}{\partial x}\boldsymbol{b} \quad \text{and} \\ \boldsymbol{v} &= \frac{\partial R}{\partial y}\boldsymbol{r} + \frac{\partial G}{\partial y}\boldsymbol{g} + \frac{\partial B}{\partial y}\boldsymbol{b}, \end{aligned}$$

based on which, the vector-space directional derivatives are computed as

$$\begin{aligned} g_{xx} &= \boldsymbol{u}\cdot\boldsymbol{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2, \\ g_{yy} &= \boldsymbol{v}\cdot\boldsymbol{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2, \text{ and} \\ g_{xy} &= \boldsymbol{u}\cdot\boldsymbol{v} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y}. \end{aligned}$$

Finally, the direction of the maximum contrast and the maximum rate of change of f can be computed respectively, as

$$\begin{aligned} \theta &= \frac{1}{2}\arctan\frac{2g_{xy}}{g_{xx} - g_{yy}} \quad \text{and} \\ F(\theta) &= \sqrt{\frac{1}{2}\big[(g_{xx} + g_{yy}) + cos2\theta(g_{xx} - g_{yy}) + 2g_{xy}sin2\theta\big]}. \end{aligned} \tag{2.3}$$

All first-order derivative operators are computed using the filter with transfer function $\frac{1}{2}z - \frac{1}{2}z^{-1}$. The color image edges can be obtained by thresholding F. An example of a RGB-color image and its magnitude of gradient are shown in Figures 2.3(a) and (b), respectively.

RGB to Grayscale Image Conversion. Given a RGB-image function f with $f(x, y) = (R(x, y), G(x, y), B(x, y))$ at (x, y), we can convert f to a grayscale-image function $\tilde{f}$ by forming a weighted sum of the R, G, B components. More specifically, the grayscale image function $\tilde{f}(x, y)$ at (x, y) is given by

$$\tilde{f}(x, y) = w_r R(x, y) + w_g G(x, y) + w_b B(x, y), \tag{2.4}$$

where w_r, w_g, and w_b are real constants. In our work, the following standard choice

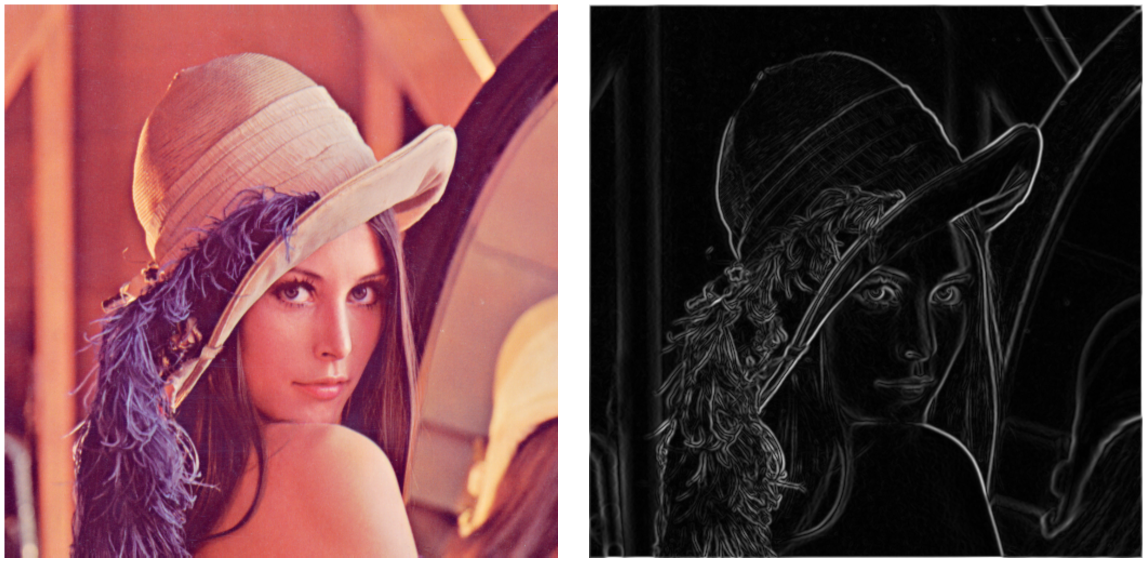

(a) (b)

Figure 2.3: (a) A color image and its (b) magnitude of gradient.

of weights based on Rec.ITU-R BT.601-7 [42], is used:

$$w_r = 0.299,\ w_g = 0.587, \text{ and } w_b = 0.144.$$

2.4 Geometry Processing

Since our mesh models are based on Delaunay triangulations, some fundamental geometry concepts such as the notions of a triangulation and a Delaunay triangulation need to be introduced. Before presenting the definition of a triangulation, we need to first introduce the concepts of convex set and convex hull.

Definition 1. *(Convex set). A set P of points in $\mathbb{R}^2$ is convex if for every pair of points $a, b \in P$, every point on the line segment $\overline{ab}$ is also in P.*

Two different sets are provided in Figure 2.4 to better illustrate the notion of a convex set. The set P, in Figure 2.4(a), is convex since for every pair of points in P the line segment that joins them is contained in P, such as the line segment $\overline{ab}$. The set P, in Figure 2.4(b), is not convex since not every line segment that joints two points in P is in P, such as the line segment $\overline{ab}$. With the concept of convex set in place, we can now introduce the definition of the convex hull.

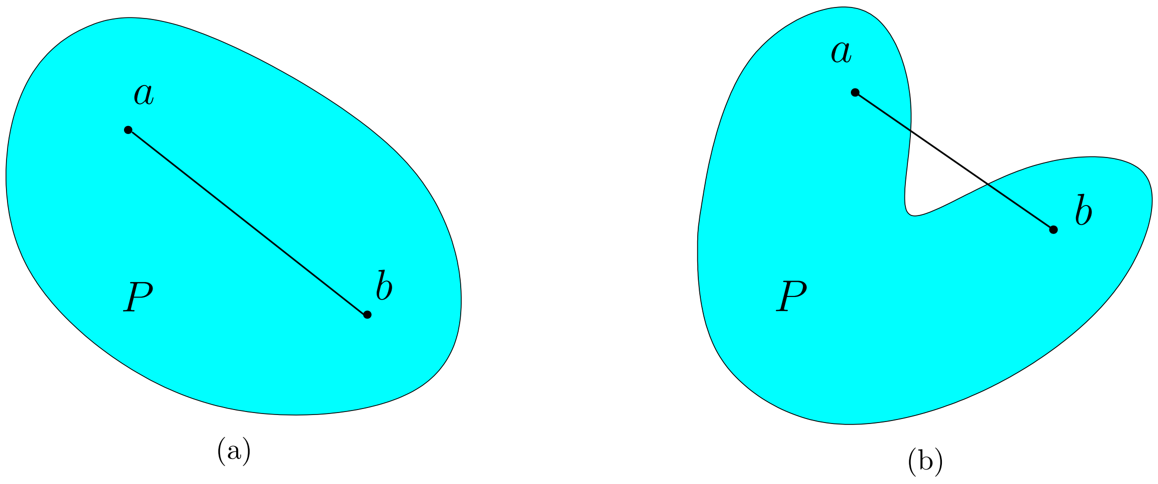


Figure 2.4: Examples of (a) convex and (b) nonconvex sets.

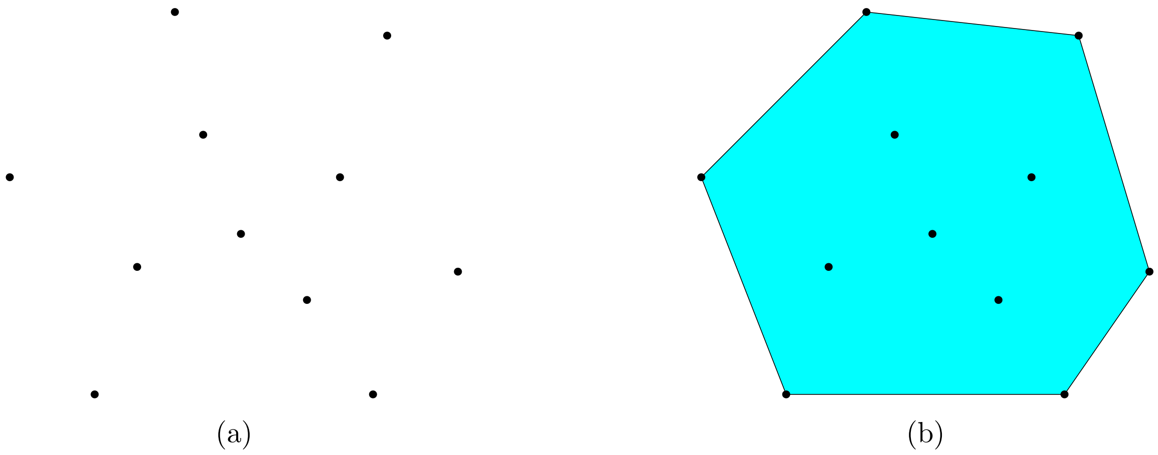


Figure 2.5: Convex hull example. (a) A set P of points and (b) the convex hull of P.

Definition 2. *(Convex hull). The convex hull of a set P of points in $\mathbb{R}^2$ is the intersection of all convex sets that contain P.*

An example of a convex hull is provided in Figure 2.5. Given a set P of points as shown in Figure 2.5(a), the convex hull of P is shown as the shaded area in Figure 2.5(b), which can be visualized as the area enclosed by a rubber band stretched around all of the points in P. The resultant polygon formed by the rubber band is the boundary of the convex hull of P. Having introduced the concept of convex hull, we can now present the notion of a triangulation.

Definition 3. *(Triangulation). A triangulation of a finite set P of points in $\mathbb{R}^2$ is a set T of (non-degenerate) triangles satisfying the following conditions:*

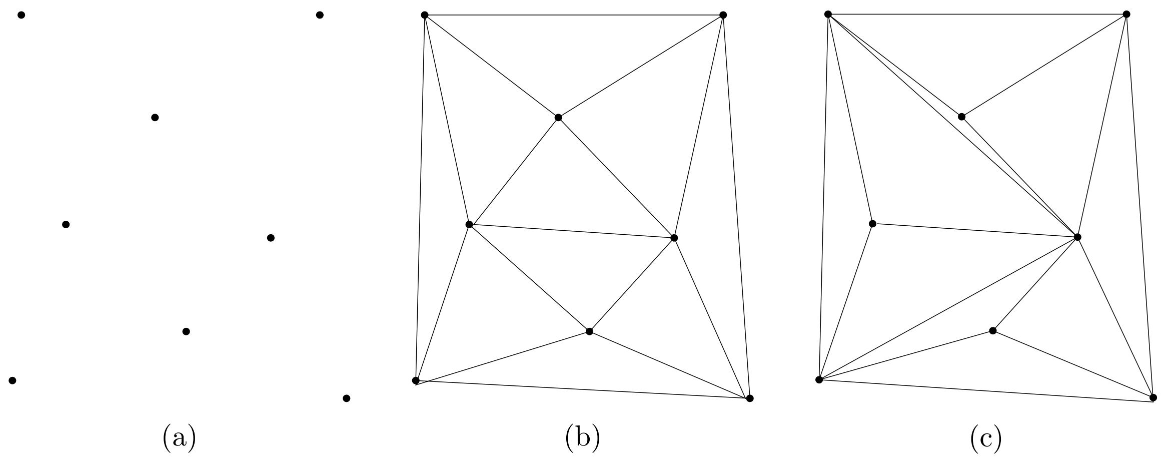


Figure 2.6: Triangulation example. (a) A set P of points, (b) a triangulation of P, and (c) another triangulation of P.

1. *the set of all vertices of triangles in T is P;*
2. *the union of all triangles in T is the convex hull of P; and*
3. *the interiors of any two triangles in T are disjoint.*

A set P of points can have many possible triangulations. An example to illustrate this is given in Figure 2.6. Given a set P of points as shown in Figure 2.6(a), by varying how the vertices are connected by edges, we can obtain the triangulations shown in Figures 2.6(b) and (c) amongst others.

Many types of triangulation have been proposed over the years. One widely used type is the Delaunay triangulation [43], which is of particular interest in our work. Before introducing the Delaunay triangulation, we need to first introduce the concept of a circumcircle of a triangle.

Definition 4. *(Circumcircle of a triangle). The unique circle passing through all three vertices of a triangle is called the circumcircle of the triangle.*

Having introduced the concept of circumcircle, we can now define the concept of Delaunay triangulation as below.

Definition 5. *(Delaunay triangulation). A triangulation T of a set P of points in $\mathbb{R}^2$ is said to be Delaunay if each triangle in T is such that the interior of its circumcircle contains no vertices of T.*

An example of the Delaunay triangulation is shown in Figure 2.7. A set P of points and its Delaunay triangulation are shown in Figures 2.7(a) and (b), respectively. In

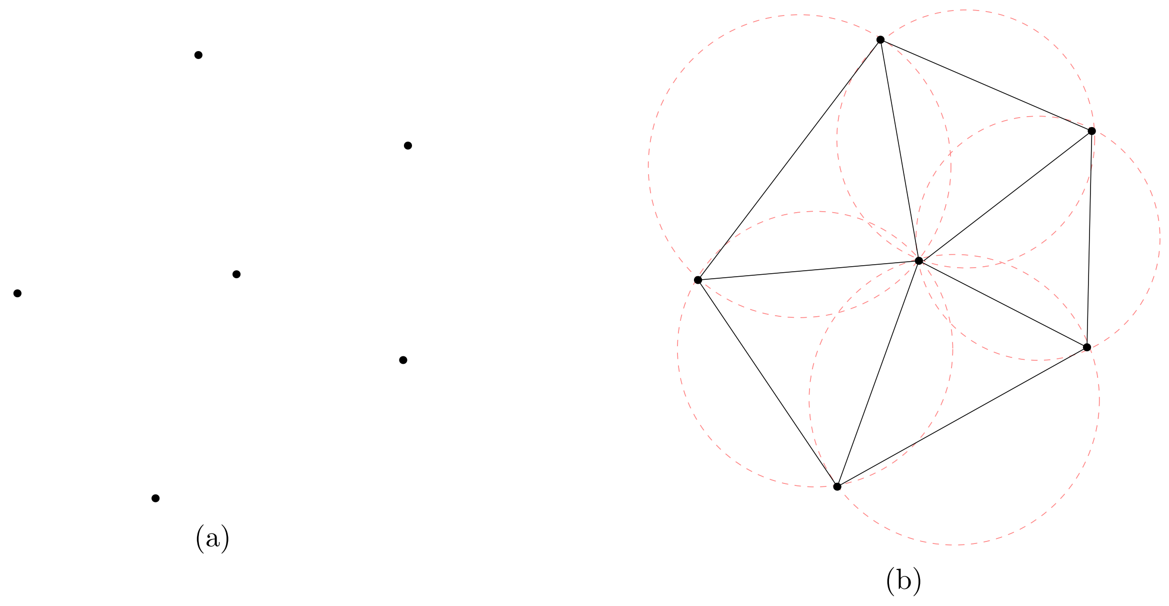


Figure 2.7: Delaunay triangulation example. (a) A set P of points. (b) The Delaunay triangulation of P.

the figure, the circumcircle of each triangle is displayed using dashed lines. As we can see from figure, no vertex of the triangulation falls strictly inside any of the circumcircles.

The Delaunay triangulation has the property that it maximizes the minimum interior angle of all triangles in the triangulation, thus minimizing the incidence of sliver (i.e., long and thin) triangles. This makes the Delaunay triangulation desirable in many approximation applications [34]. Another advantageous property of the Delaunay triangulation is that, the effect of point deletion is local [44]. That is, when a vertex p is removed from a triangulation, only the faces incident to p are changed. An example is shown in Figure 2.8, where Figure 2.8(a) shows the original Delaunay triangulation with the vertex p marked for deletion, and Figure 2.8(b) shows the updated Delaunay triangulation after removing the vertex p. The shaded areas in the figure are the faces which are affected by the removal of p from the triangulation. We can see that the deletion of p from the triangulation only affects the faces that are incident to p (i.e., the faces in the shaded region). This property can sometimes be exploited to obtain more efficient algorithms involving Delaunay triangulations.

Due to the fact that multiple (i.e., more than three) points may be cocircular, the Delaunay triangulation of a set P of points is not guaranteed to be unique. This behavior is illustrated by the example in Figure 2.9, where two different Delaunay

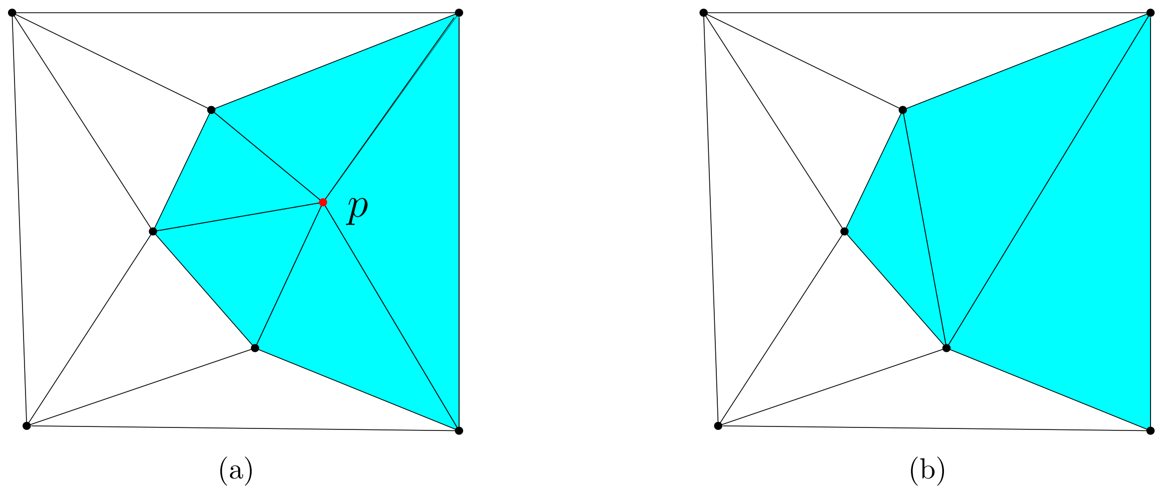
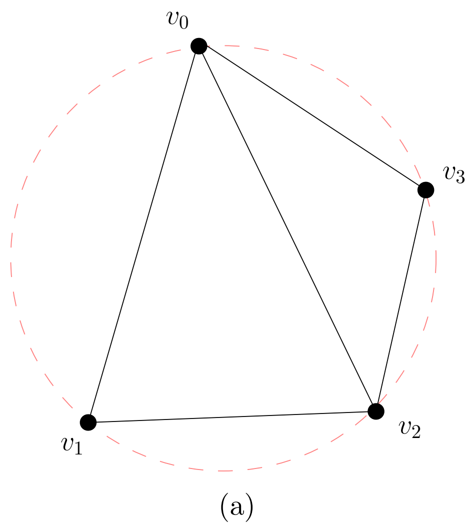


Figure 2.8: Example of the local effect of removing a vertex from the Delaunay triangulation. (a) A Delaunay triangulation of a set of points, and the vertex p to be removed. (b) The updated Delaunay triangulation after removing the vertex p.

triangulations of the same set of points can be constructed, as shown in Figures 2.9(a) and (b). In many applications, it is desirable that the Delaunay triangulation of a set P of points is uniquely determined by P alone, since this avoids the need for additional information during the triangulation process. To achieve the above goal, the preferred-directions technique of Dyken and Floater [35] can be employed while constructing the Delaunay triangulation of P, which forms the preferred-directions Delaunay triangulation (PDDT) of P. In the PDDT of P, the triangulation connectivity is determined solely by P and we are always guaranteed to get a unique triangulation from P.



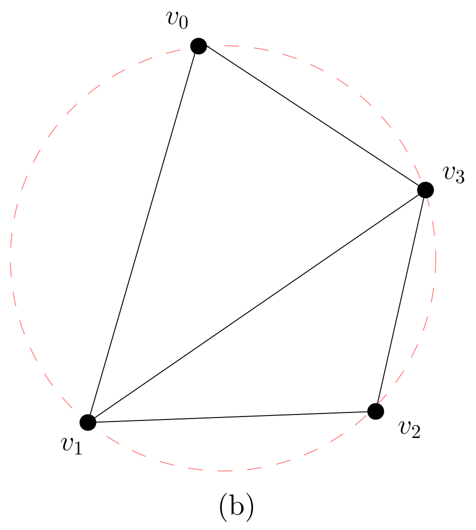


Figure 2.9: Example of a Delaunay triangulation with at least four cocircular points. (a) A Delaunay triangulation of a set of points, and (b) another Delaunay triangulation of the same set of points.

2.5 Mesh Models of Images

Having introduced the definition of the preferred-directions Delaunay triangulation, we now introduce the concept of a triangle mesh model. In the context of this thesis, an M-component image of width W and height H is an integer-vector-valued function ϕ defined on $D = [0, W-1] \times [0, H-1]$ and sampled on the 2-D integer lattice $\Lambda = [0..W-1] \times [0..H-1]$. A triangle mesh model consists of:

1. a set $P = \{p_i\}_{i=0}^{|P|-1}$ of sample points, where $P \subset \Lambda$;
2. a PDDT T of P; and
3. a set $Z = \{z_i\}_{i=0}^{|P|-1}$ of vector-integer-valued function values, where $z_i = \phi(p_i)$.

In order to ensure that all points of Λ are covered by the triangulation T, the set P must always include all of the extreme convex hull points of the image domain D, namely, the four points $(0,0)$, $(W-1,0)$, $(0,H-1)$ and $(W-1,H-1)$. As a matter of terminology, we use $|P|$ and $|P|/|\Lambda|$ to denote the size and sampling density of the mesh model, respectively.

The above mesh model is associated with a vector-integer-valued function $\hat{\phi}$ that approximates ϕ. For each image component $i \in [0..M)$, we first construct a function $\tilde{\phi}_i$ by doing linear interpolation over each face f in the triangulation T, where the linear interpolant over each face f is constructed based on the three vertices of f and their corresponding function values from Z. Since ϕ_i is integer-valued, we set $\hat{\phi}_i$ as integer-valued as well. Thus, we define the approximation function $\hat{\phi}$ as $\hat{\phi}_i = \text{round}\left(\tilde{\phi}_i\right)$, where round denotes an operator that rounds to the nearest integer value.

To illustrate the triangle-mesh modeling process, an example is provided in Figure 2.10. In an M-component image, each component $i \in [0..M)$ can be seen as a single-component (i.e., grayscale) image ϕ_i as shown in Figure 2.10(a). We can view ϕ_i as the surface shown in Figure 2.10(b), where the function value at each sample point in the image domain corresponds to its height above the plane. By selecting a subset P of the sample points in the image and triangulating them, we form the unique triangulation T shown in Figure 2.10(c). Then, the triangle mesh model can be constructed by combining the function values of each point in P with the triangulation T. The mesh model of ϕ_i is presented in Figure 2.10(d). Lastly, the approximation function $\hat{\phi}_i$ of ϕ_i can be obtained by rasterizing [45] the mesh model, as shown in Figure 2.10(e).

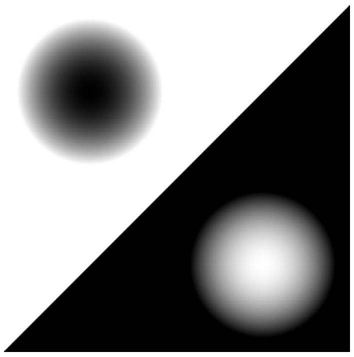
In our work, we seek to minimize the error between the original image ϕ and the reconstructed image $\hat{\phi}$. The error between ϕ and $\hat{\phi}$ is often measured by the mean squared error (MSE), which is defined as

$$\mathsf{MSE} = \frac{1}{M|\Lambda|}\sum_{i=0}^{M-1}\sum_{p\in\Lambda}[\hat{\phi}_i(p) - \phi_i(p)]^2,$$

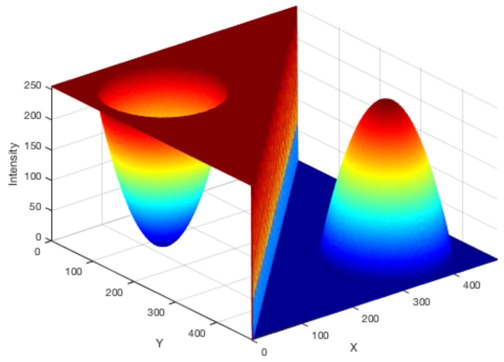
where M is the number of components in the image. For convenience, we express the MSE in terms of peak signal to noise ratio (PSNR), which is defined as

$$\mathsf{PSNR} = 20\log_{10}\left(\frac{2^\rho - 1}{\sqrt{\mathsf{MSE}}}\right), \tag{2.5}$$

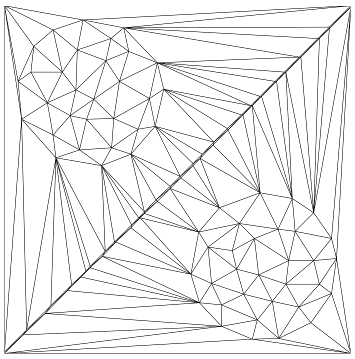
where ρ is the number of bits/sample in each component of ϕ. The PSNR is a logarithmic representation of the MSE relative to the dynamic range of the signal, with a higher PSNR value corresponding to better quality.



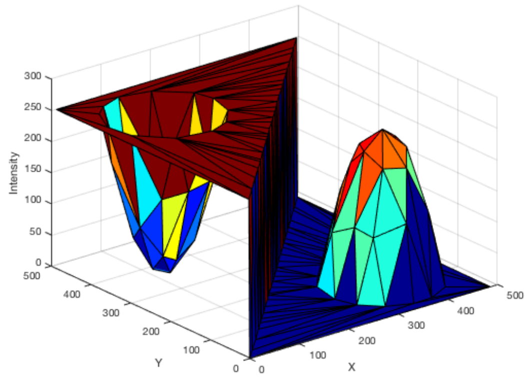
(a)



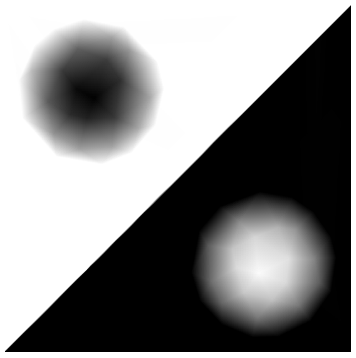
(b)



(c)



(d)



(e)

Figure 2.10: Mesh model of a single-component (i.e., grayscale) image. (a) A single-component image, (b) image viewed as a surface, (c) triangulation of the image domain, (d) resulting triangle mesh model, and (e) reconstructed image obtained by rasterizing the mesh model.

2.6 Grid-Point to Face Mapping

Given an image ϕ defined on a rectangular grid and a triangulation T of the image domain, let $\Gamma(T)$ denote the set of all integer grid points falling inside or on the boundary of T. For reasons that will become clear later, a mapping from points in $\Gamma(T)$ to faces in T is needed. A slightly modified version of the grid-points to face mapping scheme proposed in [45] is applied in our work. More specifically, a grid point $p \in \Gamma(T)$ is uniquely mapped to a face f of the triangulation T as follows:

1. If p is strictly inside a face f, map p to the face f.
2. If p is on an edge e, excluding the endpoints of e:
   (a) If e is horizontal, map p to the face below e unless no such face exists, in which case p is mapped to the face above e.
   (b) If e is not horizontal, map p to the face to the left of e. If no such face exists, map p to the face to the right of e.
3. If p is a vertex:
   (a) If p is the right endpoint of a horizontal edge e, map p to the face below e, unless no such face exists, in which case map p to the face above e.
   (b) If p is not the right endpoint of any horizontal edge, map p to the face to the left of p, unless no such face exists, in which case map p to the face to the right of p.

An example is provided in Figure 2.11 to illustrate the mapping rules described above. In Figures 2.11(a) and (b), an image ϕ is defined on the rectangular grid $[0..9] \times [0..6]$, and a triangulation T of the points $\{v_i\}_{i=0}^{7}$ is superimposed on the grid. In order to better illustrate the mapping rules, the grid points in Figure 2.11(b) are marked with different symbols, with the ones that are mapped to the same face sharing the same symbol. For example, based on rule 1, the grid point p_0 is mapped to the face f_0 since p_0 is strictly inside the face f_0. According to rule 2a, since the grid point p_1 is on the horizontal edge $\overline{v_0v_6}$ but is not either of its endpoints, p_1 is mapped to the face that below edge $\overline{v_0v_6}$, which is the face f_0. The grid point p_2 is on the horizontal edge $\overline{v_1v_2}$ but no face exists below $\overline{v_1v_2}$, so p_2 is mapped to the face above the edge $\overline{v_1v_2}$, which is the face f_2. The points p_3 and p_4 are on the non-horizontal edges $\overline{v_2v_7}$ and $\overline{v_0v_1}$, respectively, and are mapped to the faces f_2 and f_1, respectively, according

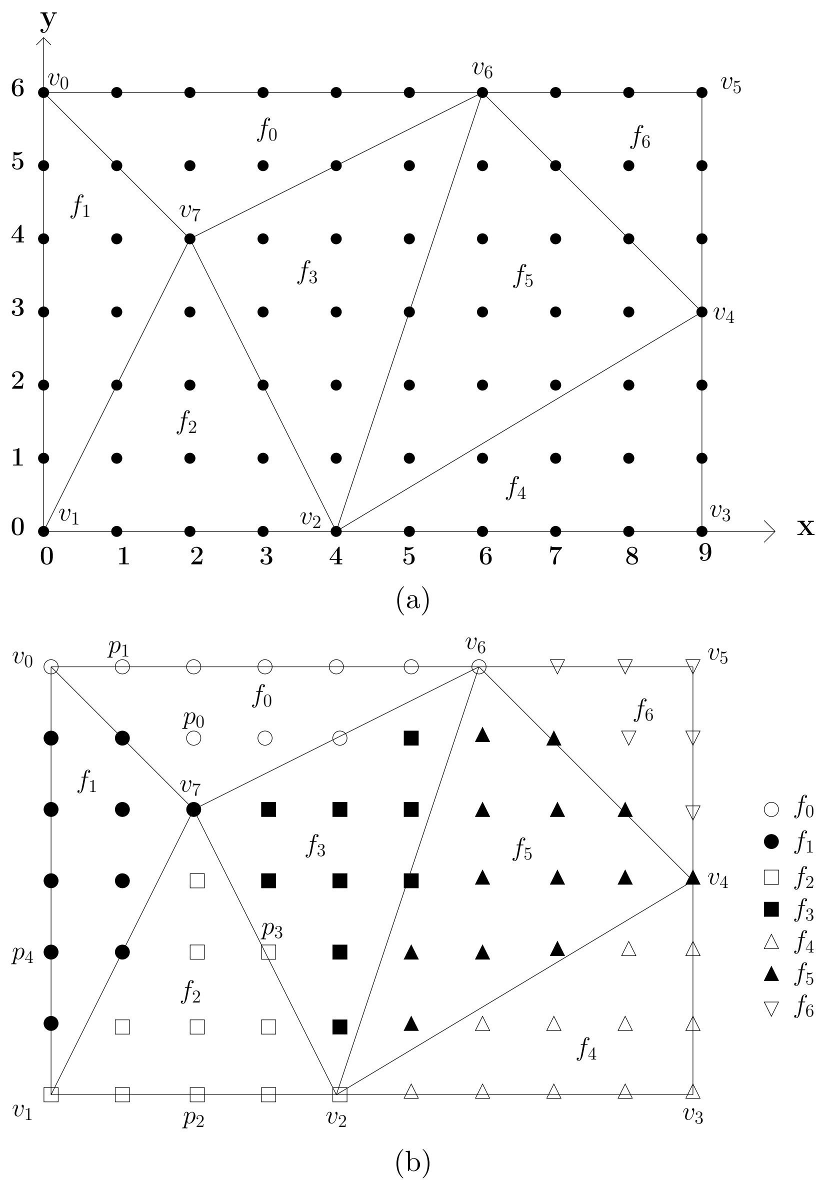


Figure 2.11: Example of the grid-point to face mapping. (a) A triangulation of a set of points on the rectangular grid, and (b) the mapping of the grid points to faces of the triangulation.

to rule 2b. Rules 3a and 3b apply to the grid points v_2 and v_4, respectively. Thus, v_2 and v_4 are mapped to the faces f_2 and f_5, respectively.

2.7 Floyd-Steinberg Error-Diffusion Algorithm

Before preceding further, we need to introduce the Floyd-Steinberg error-diffusion (FSED) algorithm [32], which is key to some of the mesh-generation methods discussed later. The FSED algorithm is a classic error-diffusion method used primarily for digital halftoning [46]. The algorithm is aimed at distributing points in a region so that their density is proportional to some prescribed density function. Given a density function d of an image ϕ of width W and height H and a threshold τ, the algorithm generates a binary-valued function b indicating the locations of selected points. The algorithm has the following additional free parameters, namely, initial diffused-in errors $\tilde{e}$ (of length W), a boundary weights mode boundMode (e.g., leaky or nonleaky [17]), and a prescribed scan order scanOrder (e.g., raster or serpentine [17]). More specifically, the algorithm generates a binary-valued function b as follows:

1. Initialize a function b as $b = d$.
2. Diffuse $\tilde{e}$ to the startup row of b. That is, $b(x,0) := b(x,0) + \tilde{e}(x)$, where $x \in [0..W)$.
3. For each point (x,y), using left bottom to right top order, perform the following:
   (a) Record the value of $b(x,y)$ as $b_{\text{old}}(x,y) = b(x,y)$.
   (b) Update b by comparing $b_{\text{old}}(x,y)$ against the threshold τ as follows:
   $$b(x,y) = \begin{cases} 1, & \text{if } b_{\text{old}}(x,y) \geq \tau \\ 0, & \text{otherwise,} \end{cases}$$
   where $b(x,y) = 1$ indicates the point (x,y) is a selected point.
   (c) Compute the quantization error $e(x,y)$ at position (x,y) as $e(x,y) = b_{\text{old}}(x,y) - (2\tau)b(x,y)$.
   (d) Diffuse e at (x,y) to its four immediate neighbors in b. For example, if serpentine scan order is selected, we diffuse the error e at (x,y) in the

following manner:

$$\begin{aligned} b(x+1,y) &:= b(x+1,y) + \frac{w_1}{w_{\text{sum}}} e(x,y), \\ b(x-s,y+1) &:= b(x-s,y+1) + \frac{w_2}{w_{\text{sum}}} e(x,y), \\ b(x,y+1) &:= b(x,y+1) + \frac{w_3}{w_{\text{sum}}} e(x,y), \text{ and} \\ b(x+s,y+1) &:= b(x+s,y+1) + \frac{w_4}{w_{\text{sum}}} e(x,y). \end{aligned}$$

where $s = 1$ if y is odd and $s = -1$ if y is even, w_1, w_2, w_3, w_4 are real constants, and $w_{\text{sum}} = \sum_{i=1}^{4} w_i$. The weights w_1, w_2, w_3, and w_4 are nominally chosen as $\frac{7}{16}, \frac{3}{16}, \frac{5}{16}$, and $\frac{1}{16}$, respectively. For leaky mode, the weights are fixed. For nonleaky mode, if any of the neighbors is out of the boundary (i.e., weight W and height H) of b, the corresponding weight w will be set as 0.

4. Output the function b.

In classic FSED, the initial diffused-in error $\tilde{e}$ is zero. Due to the feature of diffusing errors to neighboring points, the FSED algorithm can be used to quickly select a set of sample points whose density is proportional to a prescribed function, while at the same time not leaving large areas with no points having been selected. Experimental results have shown that the serpentine scan order and nonleaky error diffusion weights (described in [17]) are particularly effective. For this reason, these choices are always employed for the parameters boundMode and scanOrder in our work herein, unless indicated otherwise.

The obtained binary-valued function b indicates which points are selected. That is, points are selected at positions (x, y) where $b(x, y) = 1$. In our specific application, for reasons mentioned earlier in Section 2.4, we force the (four) extreme convex hull points H of ϕ to always be selected. That is, we always set $b(0, 0) = 1$, $b(W-1, 0) = 1$, $b(0, H-1) = 1$, and $b(W-1, H-1) = 1$. In addition, for some applications, we want to use the FSED algorithm to select a set P of N points. This can be achieved by using binary search [47] to find an optimal threshold τ for error diffusion. The mapping of the threshold τ to the selected number of sample points $|P|$ is approximately monotonic. Thus, in order to find an optimal τ, binary search can be exploited. Since it may not be possible to find a value of τ that exactly satisfies $|P| = N$, a size tolerance T is normally provided. In our application, we choose $T = 10$. More specifically, the binary search method finds an optimal error diffusion threshold τ as follows. Find a τ_1 which results in $|P| > N$ and a τ_2 which results in $|P| < N$. Then perform a binary search in $[\tau_1, \tau_2]$ to find an optimal τ which results in $\big||P| - N\big| \leq T$.

2.8 Methods for Generating Mesh Models of Grayscale Images

We now introduce three highly-effective previously-proposed mesh-generation methods for grayscale images, namely, ED, GPR, and GPRFSED. Of these, the GPRFSED method, which is derived from the ED and GPR method, serves as the foundation of our work.

2.8.1 ED Method

One highly effective method for generating mesh models of grayscale images is the error diffusion (ED) method proposed by Yang et al. [17]. Given a grayscale image ϕ which is sampled on a truncated 2-D integer lattice Λ of width W and height H and a desired number N of sample points, the ED method employs the FSED algorithm to generate a set P of sample points, where $|P| = N$, and constructs a mesh model of ϕ with P. The sample points in P are distributed with a density approximately proportional to the MMSODD of ϕ. Some parameter choices (i.e., boundary handling strategy, smoothing filter order, and scan order) of the ED method are discussed in detail by Adams in [18]. The ED method consists of the following steps:

1. Compute the significance function σ_0, which is the MMSODD of the image ϕ.

2. Compute the density function d by normalizing σ_0. That is, $d(x, y) = \sigma_0(x, y)/(\sigma_{\max} + \varepsilon)$, where $\sigma_{\max} = \max_{(x,y)\in\Lambda} \sigma_0(x, y)$ and ε is a small positive constant (e.g., 10^{-12}).

3. Set the threshold τ to use for FSED to be $\tau_0 = \frac{1}{2N} \sum_{(x,y)\in\Lambda} d(x, y)$.

4. Convert d to a binary-valued function b using serpentine, nonleaky FSED with the threshold τ.

5. Initialize P to the set of all points (x, y) for which $b(x, y) \neq 0$. Then, let $P := P \cup H$, where H is the set of the (four) extreme convex hull points of Λ.

6. If $\big||P| - N\big| \leq T$ (where T is a tolerance), stop. Otherwise, adjust τ appropriately (i.e., if $|P| > N$, increase τ; if $|P| < N$, decrease τ) and go to step 4. (Note that, in order to find an optimal τ, a binary search approach as described previously in Section 2.7 can be applied).

7. Triangulate P using a preferred-directions Delaunay triangulation.

In step 1 above, while computing the MMSODD of the image ϕ, a third-order binomial filter is used to smooth the image, where zero extension is used to handle boundaries.

To demonstrate the results obtained by using the ED method, an example is provided in Figure 2.12. For a grayscale image ϕ in Figure 2.12(a), whose MMSODD is shown in Figure 2.12(b), we generate a mesh at the sampling density of 2% using the ED method. Figures 2.12(c) to (e) show the selected sample points, image-domain triangulation, and reconstructed image, respectively. From the result of Figure 2.12(c), we can see that the selected sample points have a higher density around image edges, and lower density in other areas. The ED method has the advantage of having extremely low computational and memory costs. The quality of the mesh, however, is not very good compared to some other methods with higher complexities, like the GPR and GPRFSED methods.

2.8.2 GPR Method

The greedy point-removal (GPR) method of Demaret and Iske [21] (called "adaptive thinning" in [21]), which is closely related to the adaptive thinning technique of Dyn et al. [33], is the second mesh-generation method of interest herein. Before we can proceed further, some additional notation and terminology need to be introduced. The set of all points that are mapped to a face f is denoted points(f), based on the rules described previously in Section 2.6. The set of faces in a triangulation T that is affected by the deletion of a point p from T is denoted as affFaces(p), and the set of points in T that is affected by the deletion of p from T is denoted as affPoints(p), given by $\bigcup_{f \in \mathrm{affFaces}(p)} \mathrm{points}(f)$. Let $r_S(p)$ denotes the approximation error at the point p for the mesh with sample points S. That is, $r_S(p) = \hat{\phi}_S(p) - \phi(p)$, where ϕ is the original image and $\hat{\phi}_S$ is the image approximation obtained from mesh with the set S of sample points.

With the necessary background in place, we can now introduce the GPR method. The GPR method employs the idea of first constructing a mesh that contains all

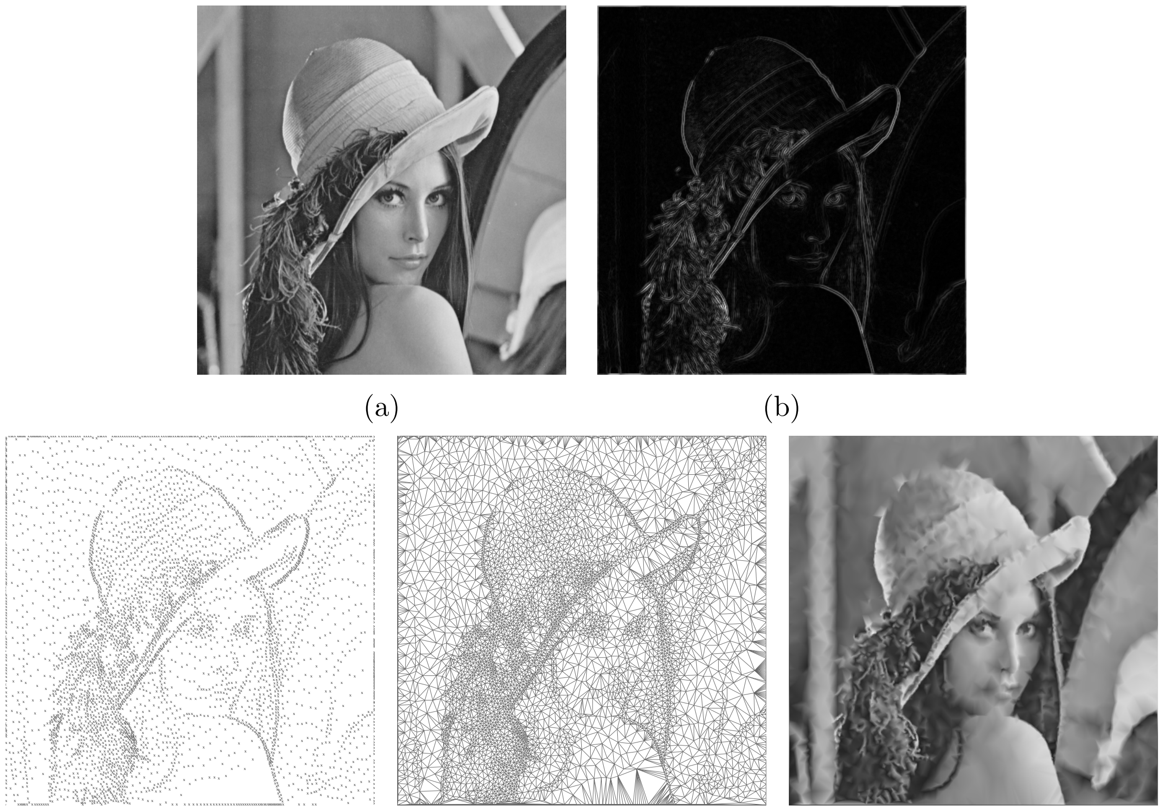


Figure 2.12: Example of the sample points selected by using the ED method. (a) A grayscale image ϕ and its (b) MMSODD. The resulting (c) selected sample points, (d) image-domain triangulation, and (e) reconstructed image (25.83 dB) obtained for ϕ at a sampling density of 2%.

the sample points in the image, and then repeatedly removing the sample point that yields the smallest increase in the squared error of the mesh approximation, until the desired mesh size is reached. More specifically, given a grayscale image ϕ sampled on a truncated 2-D integer lattice Λ of width W and height H and a desired number N of sample points, the GPR method selects a set P of sample points, where $|P| = N$ as follows:

1. Initially, let $P := \Lambda$ (hence, $|P| = WH$).
2. Construct the initial mesh, which is the preferred-directions Delaunay triangulation T of P.
3. If $|P| \leq N$, output P and stop.

4. For each point $p \in P$, compute the increase $\Delta e(p)$ in squared error of the mesh approximation that is incurred if p is removed from the triangulation, given by

$$\Delta e(p) = \sum_{q \in \Lambda \cap \text{affPoints}(p)} [r^2_{P \setminus \{p\}}(q) - r^2_P(q)].$$

5. Choose the point p^* to delete from the mesh, given by

$$p^* = \arg\min_{p \in P \setminus H} \Delta e(p),$$

   where H is the set of the (four) extreme convex hull points of Λ. Let $P := P \setminus \{p^*\}$ (i.e., remove p from the triangulation T).

6. Go to step 3 (i.e., the beginning of the loop).

Note that, since removing a vertex p from a Delaunay triangulation T can only influence the faces incident to p (as described earlier in Section 2.4), in each iteration with the exception of the first, step 4 of the above algorithm only needs to compute the error increase Δe_p for the newly constructed faces resulting from p being removed from the triangulation T. This fact is important from the viewpoint of computational efficiency. To efficiently find the next vertex p to be removed from the triangulation in step 5, the vertices can be placed in a heap-based priority queue sorted by Δe_p. For further details regarding efficient implementation, the reader is referred to [21, 33].

The GPR method can typically generate meshes with very high quality. This is mainly attributable to the fact that, in the mesh-simplification process, the selected point for removal from the mesh in each iteration is the one whose removal yields the least increase in error. Thus, the selected point is always chosen optimally at each iteration. To illustrate the results obtained by using the GPR method, an example is provided in Figure 2.13. For a grayscale image in Figure 2.13(a), we generate a mesh model at a sampling density of 2%, and present the resulting image-domain triangulation and reconstructed image in Figures 2.13(b) and (c), respectively. We can see that the reconstructed image in Figure 2.13(c) looks very close to the original image in Figure 2.13(a). Although the GPR method has been shown to yield meshes with outstanding quality, it has one major weakness, namely its extremely high computational and memory costs, which is mainly due to the initial mesh size being extremely large (i.e., containing all of the sample points of the original image).

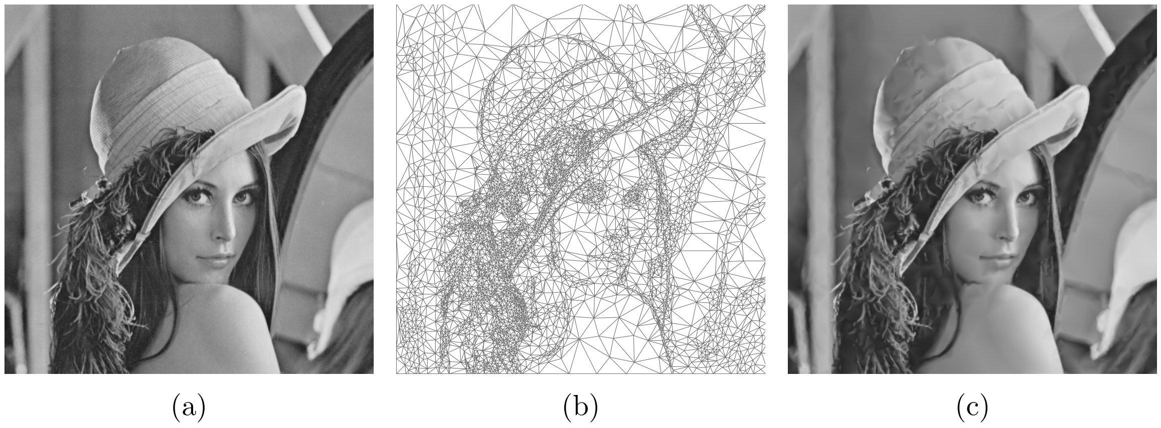


Figure 2.13: Example of the results obtained by using the GPR method. (a) A grayscale image ϕ. The resulting (b) image-domain triangulation and (c) reconstructed image (31.80 dB) obtained for ϕ at a sampling density of 2%.

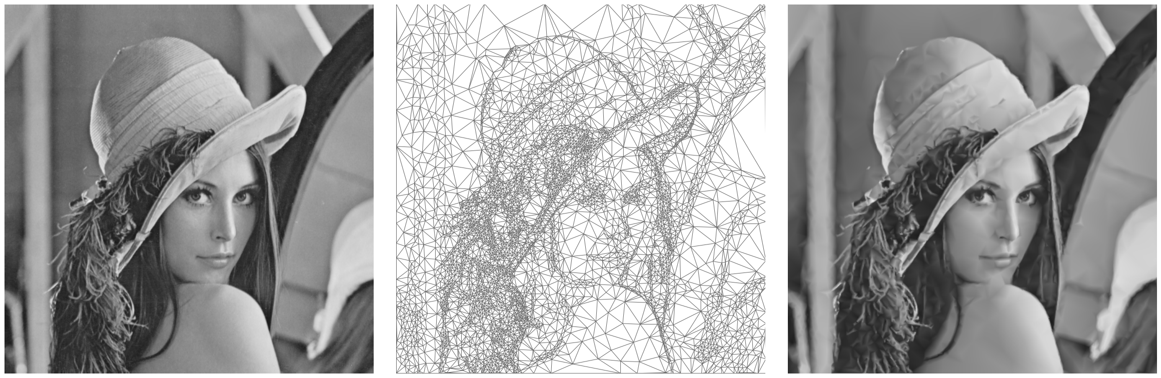
Lastly, one should note that, due to the greedy nature of the GPR method, the algorithm is extremely unlikely to yield a globally optimal solution. This is because, in each iteration, when determining the point to be removed from the mesh, the algorithm does not consider how its removal can affect the evolution of the algorithm in all subsequent iterations. That is, trying to minimize the increase of the squared error in the current iteration may cause the error increment in later iterations to become much larger.

2.8.3 GPRFSED Method

Having introduced the ED and GPR methods, we now introduce the GPRFSED method of Adams [18]. The GPRFSED method is formed by combining the ideas of the ED and GPR methods. That is, the GPRFSED method employs the ED scheme to select initial mesh points, and then performs the mesh-simplification process as proceeded in the GPR scheme. More specifically, given a grayscale image ϕ sampled on a truncated 2-D lattice Λ of width W and height H and a desired number N of sample points, the GPRFSED method selects a set P of N sample points as follows:

1. Initially, use the ED mesh-generation scheme to select a set P of points with size $N_0 = \min\{\gamma N, WH\}$, where γ is a constant satisfying $\gamma \geq 1$ and is recommended to be chosen nominally as 4.

2. Construct the initial mesh, which is the preferred-directions Delaunay triangulation T of P.



(a) (b) (c)

Figure 2.14: Example of the results obtained by using the GPRFSED method. (a) A grayscale image ϕ. The resulting (b) image-domain triangulation and (c) reconstructed image (31.86 dB) obtained for ϕ at a sampling density of 2%.

3. Perform the mesh-simplification process as described in step 3 to 6 of the GPR method.

Note that, the ED and GPR methods can be seen as special cases of the GPRFSED method, with $N_0 = N$ and $N_0 = WH$, respectively. In step 1 of the GPRFSED method, when deciding the value of N_0, a choice of $\gamma \in [4, 5.5]$ can normally result in optimal mesh quality. To achieve results comparable to the GPR method, it is usually sufficient to choose $\gamma = 4$. For the remaining cases, choosing a larger value of γ (e.g., 5.5) can normally lead to slightly better mesh quality, but requires greater computational and memory costs. To illustrate the results obtained by using the GPRFSED method, an example is provided in Figure 2.14. For the grayscale image in Figure 2.14(a), we generated a mesh at the sampling density of 2% using the the GPRFSED method, and present the resulting image-domain triangulation and reconstructed image in Figures 2.14(b) and (c), respectively. We can see that the subjective quality of the reconstructed image in Figure 2.14(c) is very high.

As noted in [18], for grayscale images, the GPRFSED method typically generates meshes with quality comparable to, and in many cases better than, the GPR scheme. Furthermore, this is achieved while requiring substantially less computational and memory costs. In addition, by making a different choice for the parameter γ in the GPRFSED method from the recommended nominal value of 4, one can easily tradeoff between computational/memory cost and mesh quality. Due to its desirable properties, we choose the GPRFSED method as the foundation of our work herein.

Chapter 3

Proposed Mesh-Generation Methods and Their Development

3.1 Overview

In this chapter, we propose a more general computational framework for generating mesh models for single- and multi-component images, based on the GPRFSED method. We begin by proposing a new startup policy for FSED that will be employed in our framework. Then, we introduce our general mesh-generation framework for single- and multi-component images, which has several free parameters and many possible choices are considered for these parameters. Next, we describe three different modes in which our mesh-generation framework can operate, as well as the relationship between theses modes and some previously-proposed mesh-generation methods. After that, the test images used for evaluation and analysis purposes are briefly introduced. Then, for each of the modes for our framework, we study the effect of different choices of free parameters on mesh quality. Based on experimentation, we advocate two specific sets of parameter choices, leading to the proposal of two mesh-generation methods for both single- and multi-component images. Lastly, we evaluate the performance of our proposed mesh-generation methods in terms of mesh quality and computational cost.

3.2 FSED Startup Policy

As introduced earlier in Section 2.7, FSED is a well-known algorithm originally proposed for digital halftoning. This algorithm can be used to select a certain number of points on a rectangular grid, distributed in proportion to a given density function. The error diffusion threshold τ plays an important role in the algorithm, which controls the number of selected points. In the application of halftoning, τ is normally quite small. In our application, we employ FSED for selecting sample points for a mesh model, in which case, the threshold τ used for error diffusion can be quite large, in order to achieve a relatively low sampling density. In the case of larger threshold values, an undesirable startup effect can occur. In particular, at low sampling densities, error diffusion can often result in an abnormally low number of sample points being selected in the startup region for error diffusion (i.e., the bottom of the image, since we scan from bottom to top). This abnormally low number of sample points can lead to very bad distortion in the part of the image that corresponds to the startup region for error diffusion, degrading overall performance of the algorithm.

To illustrate the startup effect, an example is provided in Figure 3.1. Figure 3.1(a) shows an example of a density function typical of those used in our application, and Figure 3.1(b) shows the resulting selected sample points at the sampling density of 1% using classic FSED. Observe that, very few sample points are selected in the startup region for error diffusion (i.e., bottom region). In particular, very few sample points have been selected in the bottom scan line of the image. This often leads to high approximation error in our mesh-generation application (to be considered shortly).

The above behavior of FSED can be explained as follows. In the error diffusion process, if the density function d in the startup region is quite small relative to the error-diffusion threshold τ, diffused error will accumulate very slowly, resulting in very few (or no) sample points being selected in this region. At low sampling densities, the density function d is often quite small in the startup region (relative to τ), resulting in the above startup effect.

To reduce the above startup effect, we propose a method which adds diffused-in errors to the startup row (i.e., first row) of error diffusion when running FSED. Given a density function d of an image ϕ which is sampled on a truncated 2-D lattice Λ of width W and height H, this method generates an array $\tilde{e}$ of length W which stores the initial diffusion errors. This method can be seen as a preprocessing step to FSED that selects the initial diffused-in errors. In this method, we need to first

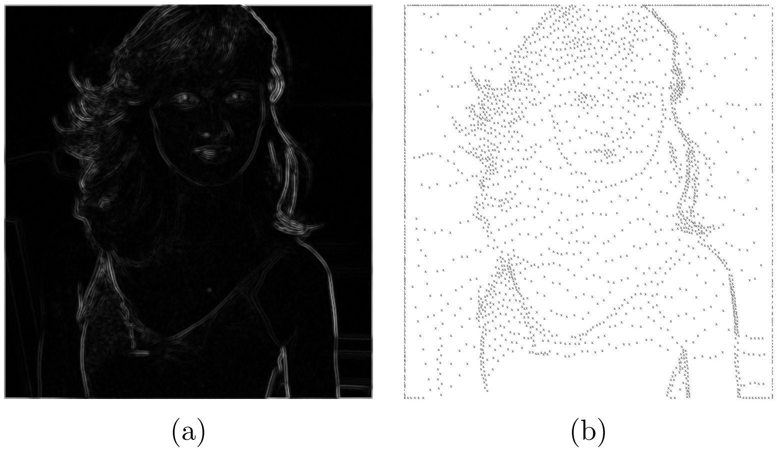


Figure 3.1: Example of the startup effect in error diffusion. (a) A density function of an image and (b) the resulting selected sample points at a sampling density of 1% using the classic FSED algorithm.

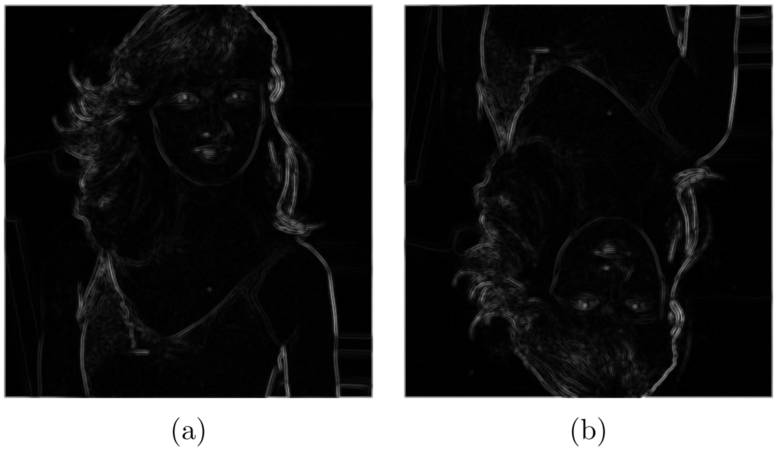


Figure 3.2: Example of mirroring a density function. (a) A density function and its (b) mirrored version.

construct a mirrored version d_m of the density function d, where d_m is given by $d_m(x, y) = d(x, H - 1 - y)$, and $(x, y) \in \Lambda$. To illustrate the results of mirroring a density function, an example is provided in Figure 3.2. Figures 3.2(a) and (b) show a density function and its mirrored version, respectively. Note that, the last row (i.e., top row) of obtained mirrored version actually corresponds to the first row of the original density function. More specifically, the mirroring method generates the initially diffused-in errors as follows:

1. Construct a mirrored version d_m of the density function d, given by $d_m(x, y) = d(x, H - 1 - y)$, where $(x, y) \in \Lambda$.

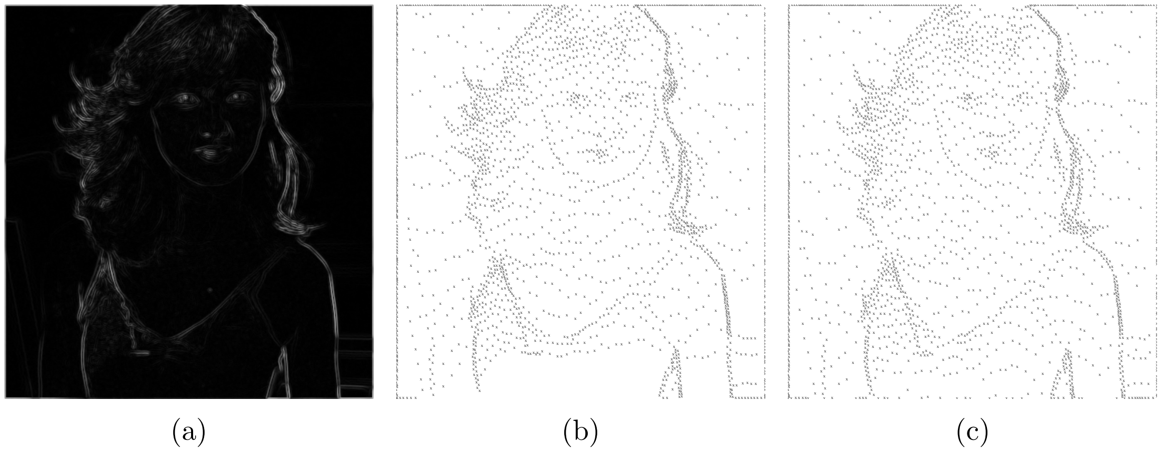


Figure 3.3: Comparison of using the classic and mirroring methods in error diffusion. (a) A density function of an image. The resulting selected sample points for the image at a sampling density of 1% with the (b) classic and (c) mirroring methods.

2. Take d_m as the input density function, scan from row 0 to $H-2$ (i.e., the row before the last row), run the nonleaky, serpentine FSED as described in Section 2.7.

3. Record and output the errors $\tilde{e}[i]$ (where $i \in [0..W)$) that have been diffused from row $H-2$ to row $H-1$ (i.e., the last row).

In our work, experiments are conducted to compare the mirroring method with the classic method (i.e., the initially diffused-in errors are all zero). To demonstrate the benefit of applying the mirroring method, we provide an example in Figure 3.3. Figure 3.3(a) shows a density function of an image, and Figures 3.3(b) and (c) show the selected sample points at a sampling density of 1% using FSED with the classic and mirroring methods, respectively. We can see that, in the case of using the classic method, very few sample points are selected in the startup (i.e., bottom) region. In contrast, in the case of using the mirroring method, the sample point distribution does not suffer from this problem.

3.3 Computational Framework for Mesh Generation

Having introduced the necessary background for several mesh-generation methods (i.e., ED, GPR, and GPRFSED) for grayscale (i.e., single-component) images in

Chapter 2, we now turn our attention to introducing the mesh-generation framework for single- and multi-component images proposed in this thesis. Our framework has several free parameters and numerous choices are considered for these parameters. We study the proposed mesh-generation framework by exploring in depth the two most common cases, namely grayscale and RGB-color images.

Before we can proceed further, some additional notation and terminology need to be introduced. For each component $i \in [0..M)$, let $r_{i,S}(p)$ denote the approximation error on component i at the point p for the mesh with set S of sample points. That is, $r_{i,S}(p) = \hat{\phi}_{i,S}(p) - \phi_i(p)$, where i is the component number, ϕ is the original image, and $\hat{\phi}_{i,S}$ is the image approximation obtained from mesh with the set S of sample points. Let $\operatorname{points}(f)$ denote the set of all points that are mapped to a face f, based on the rules described in Section 2.6. The set of faces in a triangulation T that is affected by the deletion of a point p from T is denoted as $\operatorname{affFaces}(p)$. The set of points in T that is affected by the deletion of p from T is denoted as $\operatorname{affPoints}(p)$, given by $\bigcup_{f \in \operatorname{affFaces}(p)} \operatorname{points}(f)$.

With the necessary background in place, we can now describe our framework. Our general mesh-generation framework employs the idea of the GPRFSED method, which first generates an initial mesh by using error diffusion and then applies a mesh-simplification process to obtain the final mesh. Given an M-component image ϕ (where $M \geq 1$) sampled on a truncated 2-D integer lattice Λ of width W and height H and a desired mesh size N, our general framework produces a mesh model of ϕ having a set P of sample points, with $|P| = N$, and the associated triangulation T. More specifically, our framework consists of the following steps (in order):

1. Initial sample-point selection. Select a set P of N_0 sample points, by applying the initial sample-point selection policy initSampSelPolicy, which employs FSED with the startup policy startupPolicy, where N_0, initSampSelPolicy, and startupPolicy, are three free parameters of our framework, and $N_0 \in [N..WH]$.

2. Initial mesh construction. Construct the preferred-directions Delaunay triangulation T of P.

3. Top of mesh-simplification loop. If $|P| \leq N$, output P and the associated triangulation T, and stop; otherwise, proceed to step 4.

4. For each point $p \in P$, compute the increase $\Delta e(p)$ in squared error of the mesh

approximation that is incurred if p is removed from T, given by

$$\Delta e(p) = \sum_{i=0}^{M-1} \sum_{q \in \Lambda \cap \text{affPoints(p)}} [r_{i,P\setminus\{p\}}^2(q) - r_{i,P}^2(q)],$$

5. Choose point p^* to delete from the mesh, given by

$$p^* = \operatorname*{arg\,min}_{p \in P \setminus H} \Delta e(p),$$

   where H is the set of the (four) extreme convex hull points of Λ. Let $P := P \setminus \{p^*\}$ (i.e., remove p from the triangulation T).

6. Go to step 3 (i.e., the beginning of the mesh-simplification loop).

As shown above, our framework has three free parameters, namely, N_0, initSampSelPolicy, and startupPolicy. The parameter N_0, which is size of the set of initial sample points, plays an important role in the framework. The quantity N_0 can be chosen as any value in $[N..WH]$, with larger values of N_0 leading to higher computational cost but often better mesh quality. For the parameter startupPolicy, the choices considered in our work are the mirroring and classic methods discussed in Section 3.2.

Significance Function Options. Before proceeding to introduce the choices considered for the initSampSelPolicy parameter in detail, we first introduce different choices of the significance function, which describes the importance of each point in each image component. In step 1 of the ED method (introduced in Section 2.8.1), the choice of the significance function σ_0 plays an important role, as it determines the density function d used for error diffusion (i.e., d is obtained by normalizing σ_0). As will be shown later, a density function which has a double response to image edges is particularly desirable in our work. Given an M-component image ϕ, where $M \geq 1$, we consider the following choices of significance function σ_i for each image component ϕ_i, where $i \in [0..M)$:

- MMSODD. The significance function σ_i is computed as the MMSODD of ϕ_i, by using (2.2).

- MoL (Magnitude of Laplacian). The significance function σ_i is computed as the magnitude of the Laplacian of ϕ_i, by using (2.1).

Initial Sample-Point Selection Policy. Recall that, in step 1 of our framework, the parameter initSampSelPolicy selects the initial sample-point selection policy. The choice of this policy is critical in our work, as it not only determines the selection of the initial sample points, but also to a high degree determines the resulting mesh quality. In what follows, we introduce all the choices considered for the parameter initSampSelPolicy in our framework. As a matter of terminology, FSED with a binary search method applied (as described in Section 2.7) is called the FSED point-selection algorithm. The startup policy employed in the FSED point-selection algorithm is selected based on the choice of the parameter startupPolicy in our proposed framework.

In our work, we consider five different initial sample-point selection policies, namely, PSA, PSB, PSC, PSD, and PSE. Among them, some of the policies are applicable to only grayscale images or only RGB-color images. Given an M-component image ϕ which is sampled on a truncated 2-D integer lattice Λ of width W and height H and a desired number N of sample points, each of the initial sample-point selection policies generates a set P of sample points, where $\big||P|-N\big| \leq T$, and T is a tolerance with a default value of 10. Some additional free parameters also exist in the various initial sample-point selection policies, which have to be specified as well. In what follows, we introduce each of our five initial sample-point selection policies.

PSA. The first initial sample-point selection policy to be considered is PSA. The PSA policy is only applicable to grayscale images (i.e., $M = 1$). This policy employs an idea similar to the sample-point selection scheme in the ED method to select the initial sample points. The PSA policy has an additional free parameter, namely, the significance function σ_0. More specifically, the PSA policy selects a set P of sample points as follows:

1. Compute the specified significance function σ_0 of the image ϕ.
2. Compute the density function d, given by $d(x, y) = \sigma_0(x, y)/(\sigma_{\max} + \varepsilon)$, where $\sigma_{\max} = \max_{(x,y)\in\Lambda}\sigma_0(x, y)$ and ε is a small positive constant (e.g., 10^{-12}).
3. Invoke the FSED point-selection algorithm to select a set P of sample points, by taking d as the input density function and N as the desired number of sample points.

In the remainder of this thesis, PSA(D) and PSA(L) denote the PSA policy with the parameter σ_0 chosen as MMSODD and MoL, respectively.

PSB. The second initial sample-point selection policy to be considered is PSB. The PSB policy is only applicable to RGB-color images (i.e., $M = 3$). This policy selects P by first converting the color image ϕ into a grayscale image $\tilde{\phi}$. Then, it applies the grayscale-image initial sample-point selection policy to $\tilde{\phi}$. The PSB policy also has an additional free parameter, namely, the significance function σ_0. More specifically, the PSB policy selects a set P of sample points as follows:

1. Convert the color image ϕ into a grayscale image $\tilde{\phi}$, by using (2.4).

2. Compute the specified significance function σ_0 of $\tilde{\phi}$.

3. Compute the density function d, given by $d(x, y) = \sigma_0(x, y)/(\sigma_{\max} + \varepsilon)$, where $\sigma_{\max} = \max_{(x,y)\in\Lambda}\sigma_0(x, y)$ and ε is a small positive constant (e.g., 10^{-12}).

4. Invoke the FSED point-selection algorithm to select a set P of sample points, by taking d as the input density function and N as the desired number of sample points.

In the remainder of this thesis, PSB(D) and PSB(L) respectively denote the PSB policy with the parameter σ_0 chosen as MMSODD and MoL.

PSC. The third initial sample-point selection policy to be considered is PSC. The PSC policy is only applicable to RGB-color images (i.e., $M = 3$). This policy first obtains a density function d of the image by normalizing a function g of the image computed by using (2.3). Then, it invokes the FSED point-selection algorithm to select the sample points. More specifically, the PSC policy selects a set P of sample points as follows:

1. Compute the function g of the RGB-color image ϕ by using (2.3).

2. Compute the density function d, given by $d(x, y) = g(x, y)/(g_{\max} + \varepsilon)$, where $g_{\max} = \max_{(x,y)\in\Lambda} g(x, y)$ and ε is a small positive constant (e.g., 10^{-12}).

3. Invoke the FSED point-selection algorithm to select a set P of sample points, by taking d as the input density function and N as the desired number of sample points.

PSD. The fourth initial sample-point selection policy to be considered is PSD. The PSD policy is applicable to images with an arbitrary number M of components. This policy first computes a significance function for each image component, and then aggregates the significance functions to produce a density function for FSED. The PSD policy has these additional free parameters: 1) a significance function σ_i for each image component, where $i \in [0..M)$; 2) an aggregation function F, where $F \in \{F_{\max}, F_{\text{avg}}\}$. The aggregation functions $F_{\max}$ and F_{avg} are defined as

$$F_{\max}(x_1, x_2, ..., x_n) = \max\{x_1, x_2, ..., x_n\} \text{ and}$$
$$F_{\text{avg}}(x_1, x_2, ..., x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

More specifically, the PSD policy selects a set P of sample points as follows:

1. Compute the specified significance function σ_i for each image component ϕ_i, where $i \in [0..M)$.
2. Compute the aggregated significance function a of the image, given by $a(x, y) = F\Big(\sigma_0(x, y), \sigma_1(x, y), ..., \sigma_{M-1}(x, y)\Big)$.
3. Compute the density function d, given by $d(x, y) = a(x, y)/(a_{\max} + \varepsilon)$, where $a_{\max} = \max_{(x,y)\in\Lambda} a(x, y)$ and ε is a small positive constant (e.g., 10^{-12}).
4. Invoke the FSED point-selection algorithm to select a set P of sample points, by taking d as the input density function and N as the desired number of sample points.

In the remainder of this thesis, PSD(Max, D) denotes the PSD policy with the aggregation function F chosen as $F_{\max}$ and the significance function σ_i (where $i \in [0..M)$) chosen as MMSODD. Similarly, PSD(Max, L) denotes the PSD policy with F chosen as $F_{\max}$ and σ_i chosen as MoL; PSD(Avg, D) denotes the PSD policy with F chosen as F_{avg} and σ_i chosen as MMSODD; and PSD(Avg, L) denotes the PSD policy with F chosen as F_{avg} and σ_i chosen as MoL.

PSE. The last initial sample-point selection policy to be considered is PSE. The PSE policy is also applicable to images with an arbitrary number M of components. This policy selects P by first performing FSED on each image component separately to generate M sets of sample points, and then taking the union of these sets to produce one final point set. The PSE policy has the following additional free parameter, namely, the significance function σ_i, where $i \in [0..M)$. More specifically, the PSE policy selects a set P of sample points as follows:

1. Compute the specified significance function σ_i for each image component ϕ_i, where $i \in [0..M)$.

2. For each $i \in [0..M)$, compute the density function d_i, given by $d_i(x,y) = \sigma_i(x,y)/(\sigma_{\max} + \varepsilon)$, where $\sigma_{\max} = \max_{(x,y),i \in \Lambda \times [0..M)} \sigma_i(x,y)$ and ε is a small positive constant (e.g., 10^{-12}).

3. Set a single threshold τ to use for FSED to be $\tau_0 = \frac{M}{2N} \sum_{(x,y) \in \Lambda} d_0(x,y)$.

4. For each $i \in [0..M)$, convert d_i to a binary-valued function b_i using serpentine, nonleaky FSED with the threshold τ. The employed FSED startup policy is selected based on the choice of the parameter startupPolicy.

5. Initialize P to the set of all points (x,y) for which $b_i(x,y) \neq 0$ for at least one of $i \in [0..M)$. Then, let $P := P \cup H$, where H is the set of the (four) extreme convex hull points of Λ.

6. If $\big||P| - N\big| \leq T$, where T is a size tolerance (with a nominal value 10), stop; otherwise, if $|P| > N$, increase τ, or if $|P| < N$, decrease τ, and go to step 4.

In step 6, the mapping of the threshold τ to the selected number of sample points $|P|$ is approximately monotonic. Thus, in order to find an optimal τ, binary search can be exploited in our work. That is, find a τ_1 which results in $|P| > N$ and a τ_2 which results in $|P| < N$, and then perform a binary search in $[\tau_1, \tau_2]$ to find an optimal τ. In the remainder of this thesis, PSE(D) and PSE(L) denote the PSE policy with significance function σ_i (where $i \in [0..M)$) chosen as MMSODD and MoL, respectively.

Some relationships exist between some of the policies introduced above. As mentioned earlier, the PSD and PSE policies are applicable to images with an arbitrary number of components. For an image with only one component, averaging or taking the maximum of the significance function across components is the same as taking the original single significance function, and taking the union of all component-point sets is the same as taking the original single point set. Thus, if $M = 1$, the PSA, PSD, and PSE policies are equivalent if they use the same significance function. More specifically, if $M = 1$, the policies PSA(D), PSD(Max, D), PSD(Avg, D), and PSE(D) are equivalent; and the policies PSA(L), PSD(Max, L), PSD(Avg, L), and PSE(L) are equivalent.

3.4 Different Modes in the Proposed Framework and Their Relationship with Some Previous Methods

Our proposed framework can operate in three distinct modes which corresponds to N_0 been chosen in N, $(N..WH)$, and WH. These different modes result in fundamentally different behavior in our framework. For example, the modes differ in whether mesh-simplification is applied (or the extent to which the mesh-simplification is applied).

The first mode corresponds to $N_0 = N$, in which case, no mesh-simplification is performed. This mode selects all mesh sample points in one shot. Thus, the computational cost of this mode is extremely low, but the mesh quality is often not as good as with other modes. For certain applications which require extremely low computational cost, the methods constructed from this mode can be particularly useful. In this mode, if $M = 1$ and we select the free parameters such that initSampSelPolicy is PSA(D) and startupPolicy is classic, we obtain the ED method proposed in [17]. In addition, if we vary the choices of initSampSelPolicy, and startupPolicy, we can obtain a family of methods that includes the ED method as a special case. Thus, we refer this mode as “ED-like”.

The second mode corresponds to $N_0 \in (N..WH)$, in which case, some mesh-simplification is performed. In this mode, however, the initial mesh size N_0 can be chosen flexibly as any value in $(N..WH)$. As will be shown later, if N_0 is chosen appropriately, we can achieve meshes with very good quality, but at a relatively low computational cost. In this mode, if $M = 1$ and we select the free parameters such that $N_0 = 4N$, initSampSelPolicy is PSA(D), and startupPolicy is classic, we obtain the GPRFSED method proposed in [18]. In addition, by varying the choices of N_0, initSampSelPolicy, and startupPolicy in the framework, we can obtain a family of methods that includes the GPRFSED method as a special case. Thus, we refer to this mode as “GPRFSED-like”.

The third mode corresponds to $N_0 = WH$, in which case, a large amount of mesh-simplification is performed (since the initial mesh contains all points in the sampling grid). Methods constructed from this mode can normally obtain a mesh with very good quality, but the computational cost is typically extremely high. In this mode, if $M = 1$, we obtain the GPR method proposed in [21]. Thus, we refer to this mode as “GPR-like”.

Table 3.1: Images in the representative data set

Image	Width and Height	Description
lena (4.2.04)	512×512	photographic, woman
pens	512×480	photographic, pens
bluegirl	480×512	photographic, girl with blue skirt
kodim23	768×512	photographic, birds
cartoon_bull	1024×768	computer-generated, bull

3.5 Test Data and Experimental Comments

Before proceeding further, a brief digression is in order regarding the test images that we used for analysis and evaluation purposes. In our work, we employed 45 RGB-color images, taken mostly from standard data sets, such as the JPEG-2000 [48], USC-SIPI [49], CIPR-Canon [50], and Kodak [51] test sets. A detailed description of each image is presented in Appendix A. For reasons of consistency, our grayscale image test set is simply formed by converting each of the above 45 color images into grayscale versions using the standard RGB to luminance mapping given by (2.4). We often focus on results for a representative subset of these images, namely, those listed in Table 3.1. This subset was deliberately chosen to cover a variety of image types (i.e., photographic and computer-generated imagery), image sizes, and subject matter.

Only RGB-color (i.e., $M = 3$) and grayscale (i.e., $M = 1$) images are available from the standard test sets. Thus, we study our proposed mesh-generation framework by focusing on exploring these two cases. Between them, our main contributions focus on mesh generation for RGB-color images. For the case of grayscale images, although our contribution is relatively smaller, the work is still of practical benefit as it offers some improvements compared to some other previously-proposed mesh-generation methods for grayscale images.

3.6 Impact on Different Parameter Choices for the ED-like Mode

As introduced earlier in Sections 3.3 and 3.4, our mesh-generation framework has several free parameters and can operate in three different modes. In what follows, we study how different choices of parameters affect the mesh quality in these three modes. We start by discussing the impact of parameter choices in the ED-like mode of the framework.

3.6.1 RGB-Color Case

First, we discuss the impact of the parameter choices on the mesh quality for RGB-color images (i.e., $M = 3$). The selection of free parameters in the proposed framework, including initSampSelPolicy and startupPolicy, will be discussed.

3.6.1.1 Initial Sample-Point Selection Policy

To begin, we consider how the choice of the initial sample-point selection policy affects mesh quality. To do this, we select the free parameter startupPolicy as the classic method. As mentioned earlier, we consider four initial sample-point selection policies that are applicable to RGB-color images, namely, PSB, PSC, PSD, and PSE. By varying the choice of additional parameters (i.e., significance function and aggregation function) in the four initial sample-point selection policies, we have in total nine combinations of possible choices for the parameter initSampSelPolicy, namely, PSB(D), PSB(L), PSC, PSD(Max, D), PSD(Max, L), PSD(Avg, D), PSD(Avg, L), PSE(D), and PSE(L). For each of the 45 color images and five sampling densities per image (for a total of 225 test cases), we generated a mesh using each possible choice of initSampSelPolicy, and measured the approximation error of the reconstructed image in terms of PSNR. In each test case, the PSNR results obtained from different policies were ranked from 1 (best) to 9 (worst). Then, the average and standard deviation of the ranks were computed across each sampling density as well as overall. The overall ranking results are given in Table 3.2(a). Some results for individual test cases are shown in Table 3.2(b). For each of the above tables, the best result in each test case is typeset in bold. We also conducted similar experiments by using other choices of fixed parameters (i.e., using startup policy of mirroring), and obtained similar results to those in Table 3.2.

Table 3.2: Comparison of using the various initial sample-point selection policies in the ED-like mode for RGB-color images. (a) Average ranking across the 45 images in the data set. (b) PSNR results of some representative images.

(a)

Samp. Density (%)	PSNR Average Rank (Standard Deviation)								
	PSB (D)	PSB (L)	PSC	PSD (Max, D)	PSD (Max, L)	PSD (Avg, D)	PSD (Avg, L)	PSE (D)	PSE (L)
0.5	4.00 (1.80)	5.82 (1.45)	**2.16** (2.01)	2.69 (1.28)	4.80 (1.63)	3.69 (1.60)	5.49 (1.90)	7.80 (1.17)	8.56 (1.09)
1.0	3.51 (1.67)	6.02 (1.36)	2.98 (2.29)	**2.60** (1.61)	4.93 (1.55)	2.96 (1.23)	5.42 (1.48)	7.93 (1.14)	8.64 (0.85)
2.0	3.49 (1.73)	5.87 (1.54)	4.24 (2.44)	**2.33** (1.48)	4.58 (1.67)	2.64 (1.23)	5.04 (1.28)	8.07 (0.71)	8.73 (0.49)
3.0	3.16 (1.40)	5.80 (1.45)	5.02 (2.47)	**2.20** (1.24)	4.44 (1.63)	2.42 (1.06)	5.16 (1.23)	7.93 (0.68)	8.87 (0.34)
4.0	3.24 (1.69)	5.98 (1.29)	5.84 (2.04)	**1.80** (0.91)	4.53 (1.42)	2.16 (0.76)	4.73 (0.90)	7.91 (0.78)	8.80 (0.40)
Overall	3.48 (1.69)	5.90 (1.42)	4.05 (2.63)	**2.32** (1.36)	4.66 (1.59)	2.77 (1.32)	5.17 (1.43)	7.93 (0.93)	8.72 (0.70)

(b)

Image	Samp. Density (%)	PSNR Average Rank (Standard Deviation)								
		PSB (D)	PSB (L)	PSC	PSD (Max, D)	PSD (Max, L)	PSD (Avg, D)	PSD (Avg, L)	PSE (D)	PSE (L)
pens	0.5	13.96	13.87	14.99	**15.02**	14.95	14.32	14.05	11.38	10.90
	1.0	17.24	17.07	17.89	**18.17**	17.99	17.83	17.76	15.09	14.89
	2.0	21.98	21.84	21.82	**22.68**	22.28	22.18	22.07	18.11	18.22
	3.0	25.05	24.92	24.28	**25.80**	25.40	25.29	25.02	20.80	20.72
	4.0	27.05	26.80	26.08	27.20	26.94	**27.21**	27.11	23.35	22.59
kodim23	0.5	20.35	20.41	**21.02**	20.61	20.52	20.49	20.20	16.86	16.75
	1.0	24.69	24.73	25.11	**25.52**	25.17	25.00	24.75	21.06	20.95
	2.0	29.19	28.98	28.34	**29.74**	29.56	29.55	29.21	26.00	25.79
	3.0	31.27	30.98	29.98	**31.68**	31.44	31.50	31.29	29.05	28.98
	4.0	32.65	32.44	31.12	**32.86**	32.75	32.77	32.56	31.26	31.01
cartoon _bull	0.5	25.23	25.22	25.00	25.26	25.52	25.45	**25.88**	20.28	20.21
	1.0	32.64	32.52	31.79	32.89	32.83	32.87	**33.16**	25.98	26.50
	2.0	37.00	36.98	37.36	**38.00**	37.60	37.56	37.50	33.80	33.85
	3.0	39.51	39.30	39.26	**40.01**	39.95	39.99	39.74	36.88	37.46
	4.0	40.69	40.52	41.06	**41.43**	41.15	41.29	41.09	38.57	39.03

Examining the results of Table 3.2(a), we can make several observations. First, the PSD(Max, D) policy performs the best among all policies with an overall rank of 2.32, followed by the PSD(Avg, D) policy with an overall rank of 2.77. As the sampling density is increased, both PSD(Max, D) and PSD(Avg, D) become more favored compared to other policies. That is, their average rank increases with sampling density. Second, the worst performers are the PSE(D) and PSE(L) policies, with overall ranks of 7.93 and 8.72, respectively. Lastly, within each of the PSB, PSD, and PSE policies, in terms of the choice of significance function on mesh quality, the MMSODD function yields better results than the MoL function. For example, in terms of overall rank, the PSB(D) policy outperforms the PSB(L) policy, and the PSD(Max, D) policy outperforms the PSD(Max, L) policy. The results for individual test cases in Table 3.2(b) are consistent with the above observations. That is, the PSD(Max, D) policy performs the best and the PSE(L) performs the worst among all policies, and the MMSODD function outperforms the MoL function within each of the PSB, PSD, PSE policies.

Though the PSD(Max, D) policy performs the best among all policies, it is helpful to the reader to understand why this is so. Thus, in what follows, we present a deeper analysis of how each initial sample-point selection policy affects the mesh quality by comparing with PSD(Max, D).

MMSODD versus MoL. Before studying the effect of different initial sample-point selection policies on mesh quality in detail, we first analyze a common free parameter among most of the policies, namely, the significance function. Our experimental results show that, in each of the PSB, PSD, and PSE policies, the MMSODD function outperforms the MoL function in about 90% of all test cases, by 0.01 to 2.22 dB. To explain this behavior, for each of the PSB, PSD and PSE policies, we examined the density functions obtained from different significance functions into more detail. We found that, for each of these policies, the density function obtained from the MMSODD usually has a stronger response to image edges than that obtained from the MoL. This behavior is mainly attributable to the fact that, in the Laplacian, the two second-order derivative operators $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$ can have opposite signs, which can result in cancellation. A density function having a stronger response to image edges tends to result in a better point distribution along image edges, leading to a higher mesh quality. For the above reason, in the remainder of the discussion in this section, we consider the effect of using each of the PSB, PSD, and PSE policies only based on the case of the using the MMSODD significance function.

PSB(D) versus PSD(Max, D). First, we analyze the effect of using the PSB(D) policy on mesh quality by comparing with the PSD(Max, D) policy. Based on our experimentation, we find that the PSD(Max, D) policy outperforms the PSB(D) in 157/225 (70%) of all test cases, by a margin of 0.01 to 7.07 dB. Based on further analysis, we find the above behavior is mainly attributable to a major shortcoming of the PSB(D) policy, namely, the loss of contrast during color conversion. When converting a RGB-color image to a grayscale image, the conversion process can sometimes lead to the case that two pixels with different RGB values are converted to similar or even the same gray value. This will cause the grayscale image lose contrast in some places or even drop some important image details (e.g., lines or edges) that exists in the original RGB image, which is undesirable. For example, two pixels with the RGB values (255, 191, 0) and (128, 255, 0) will result in same grayscale value 188 by using (2.4).

PSC and PSD(Avg, D) versus PSD(Max, D). Next, we analyze the effect of using each of the PSC and PSD(Avg, D) policies on mesh quality, by comparing with the PSD(Max, D) policy. The key difference between these policies is the use of different density functions before invoking the FSED algorithm. Our experimental results show that the PSD(Max, D) policy beats the PSC policy in 138/225 (61.3%) of test cases by 0.01 to 3.65 dB, and beats the PSD(Avg, D) in 144/225 (64.0%) of test cases by 0.01 to 3.43 dB. Based on further experiments, we find that, the density function obtained from the PSC policy has a single response to image edges, while those obtained from the PSD(Max, D) and PSD(Avg, D) policies have a double response to image edges. In addition, the response to image edges is stronger in the density function obtained from the PSD(Max, D) policy. The density function having a strong double response to image edges is shown to be particularly helpful in our work.

To help illustrate the effect of using each of these policies on the mesh quality, an example is provided in Figure 3.4. For the RGB-color image shown in Figure 3.4(a), we generate a mesh using each of the PSC, PSD(Max, D) and PSD(Avg, D) policies. In each case, for the small region of interest highlighted in Figure 3.4(a), shown enlarged in Figure 3.4(b), we present the resulting sample-point density function σ, image-domain triangulation, and reconstructed image. The density functions shown in Figures 3.4(c) to (e) are displayed with the same brightness scale.

Examining the results of Figure 3.4, we can see that the density function in Figure 3.4(c) has a single response to an image edge, which results in many sample points

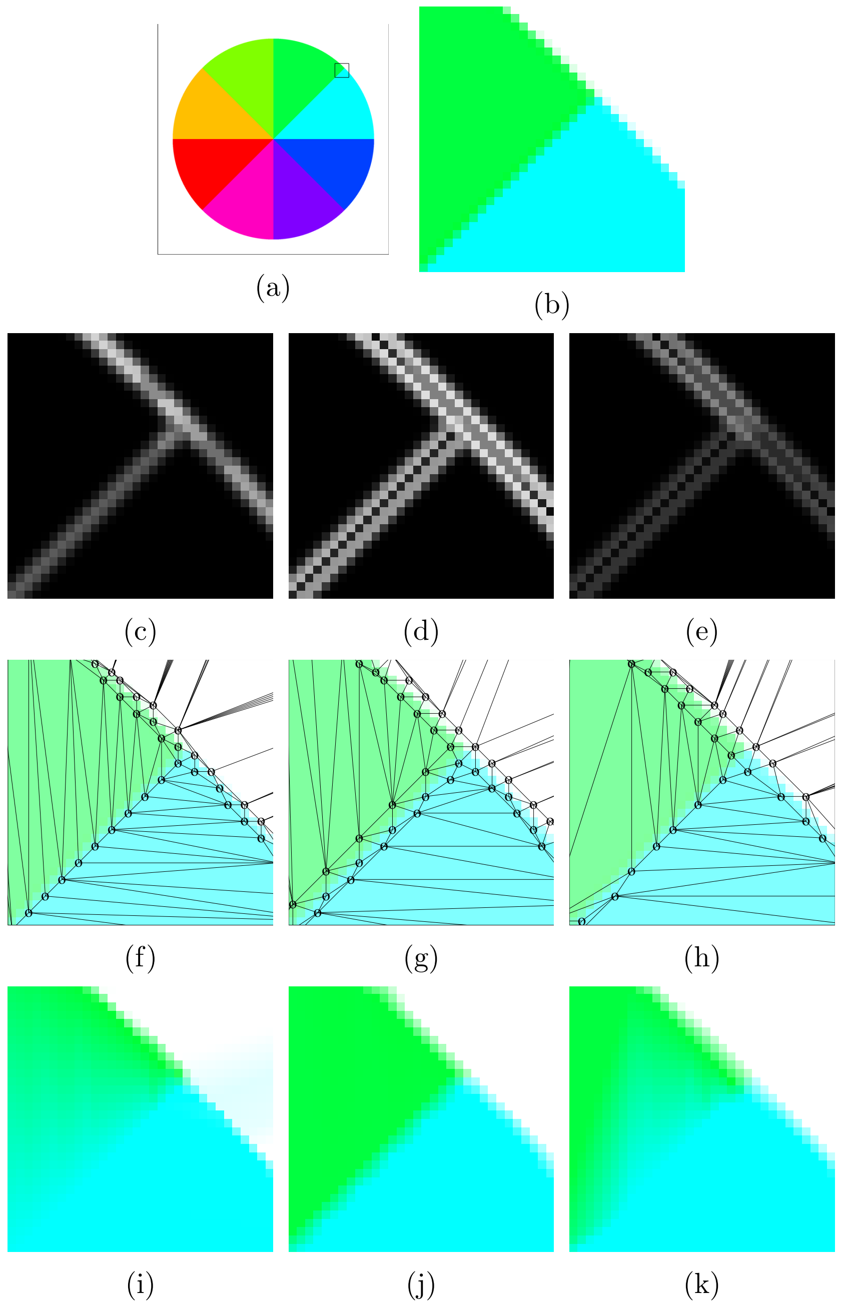
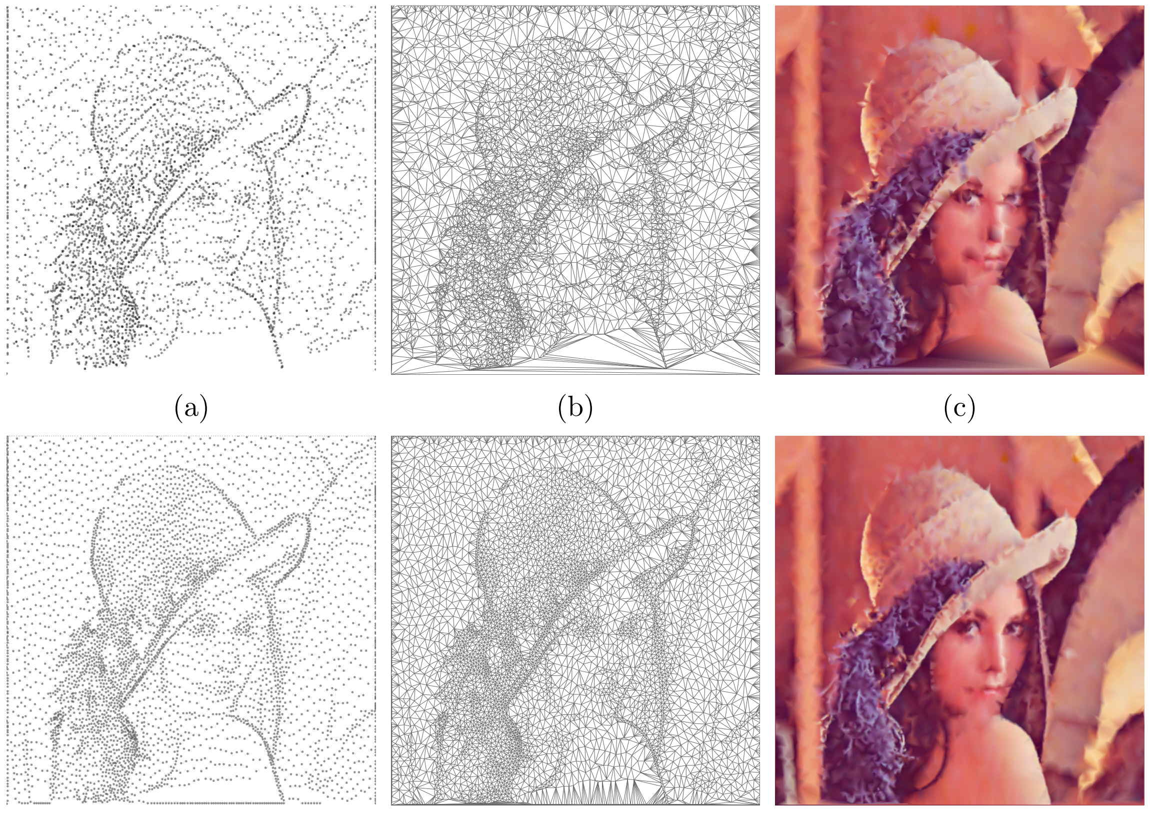


Figure 3.4: Comparison of the PSC and PSD policies in the ED-like mode for RGB-color images. (a) A color wheel image, showing a rectangular region of interest. (b) Region of interest under magnification. The density function used for error diffusion with the (c) PSC, (d) PSD(Max, D), and (e) PSD(Avg, D) policies; The image-domain triangulations obtained at a sampling density of 1% by using the (f) PSC (g) PSD(Max, D) and (h) PSD(Avg, D) policies, and the corresponding reconstructed images (i), (j), and (k).

being selected exactly on the image edges in Figure 3.4(f). This leads to a bad distortion in the reconstructed image in Figure 3.4(i). In contrast, the density functions in Figures 3.4(d) and (e) both have a double response to an image edge, and the response to image edges is weaker in Figure 3.4(e). Due to the density function having a weak response to image edges, not enough sample points are selected on those image edges in the triangulation shown in Figure 3.4(h). This leads to a bad distortion in the reconstructed image in Figure 3.4(k). The above behavior of density function having weaker response to image edges is mainly attributed to the averaging effect in the PSD(Avg, D) policy. By using the PSD(Avg, D) policy, its aggregated significance function a can be influenced by the component with the minimum significance value. In contrast, the density function in Figure 3.4(d) has a stronger double response to all image edges and the reconstructed image in Figure 3.4(j) is of higher quality.

PSE(D) versus PSD(Max, D). Lastly, we analyze the effect of using the PSE(D) policy on mesh quality by comparing with the PSD(Max, D) policy. Our experimental results show that the PSD(Max, D) policy outperforms the PSE(D) policy in 223/225 (99.1%) of all test cases by a margin of 0.14 to 8.18 dB. To explain this behavior, we provide an example. In Figure 3.5, for a RGB-color image, we generate a mesh using each of the PSE(D) and PSD(Max, D) policies, and present the resulting selected sample points, image-domain triangulation, and reconstructed image.

Examining the results in Figure 3.5(a), we can make a few observations. First, many sample points are selected extremely close to each other. Second, many areas have relatively low sampling density. This manner of selecting sample points is shown to be ineffective and normally leads to a higher distortion in the reconstructed images. The above behavior is mainly due to the fact that, during the FSED process, the threshold τ is very high, which results in the selected sample points in each component often corresponding to only some key image elements (i.e., edges). As shown in the triangulation in Figure 3.5(b), many big sliver triangles exist, which results in bad distortion in the reconstructed image in Figure 3.5(c). In contrast, the sample points are distributed reasonably well in Figure 3.5(d) and the quality of the resulting reconstructed image in Figure 3.5(f) is substantially better.



(a) (b) (c)

(d) (e) (f)

Figure 3.5: Comparison of the PSE(D) and PSD(Max, D) policies in the ED-like mode for RGB-color images. The selected sample points at a sampling density of 2% by using the (a) PSE(D) and (d) PSD(Max, D) policy, the corresponding obtained image-domain triangulations (b) and (e), and the corresponding reconstructed images (c) (21.99 dB) and (f) (25.56 dB).

3.6.1.2 FSED Startup Policy

Next, we study the effect of the choice of the FSED startup policy on mesh quality in the ED-like mode for RGB-color images. To do this, we select the free parameter initSampSelPolicy as PSD(Max, D). For each of the 45 RGB-color images with five sampling densities per image, we generated a mesh using each of the startup policies (i.e., mirroring and classic), and measured the approximation error of the reconstructed image in terms of PSNR. For each sampling density as well as overall, we calculated the fraction of cases where the mirroring method outperforms classic method, in addition to the minimum, median, and maximum differences in PSNR by which mirroring beats classic. The overall results are shown in Table 3.3(a). Results for some individual test cases are shown in Table 3.3(b). We also conducted similar

Table 3.3: Comparison of using different startup policies in the ED-like mode for RGB-color images. (a) Overall results for the change in PSNR obtained by using the mirroring method compared to the classic method, where win ratio is the fraction of cases the mirroring method beats the classic method. (b) PSNR results for some representative images, where the column diff shows the PSNR differences in the margin by mirroring beats classic.

(a)

Samp. Density (%)	PSNR Change (dB)			Win Ratio (%)
	Minimum	Median	Maximum	
0.5	-2.29	0.76	3.87	77.8
1.0	-0.57	0.38	2.65	80.0
2.0	-0.53	0.19	3.68	68.9
3.0	-3.45	0.09	3.32	64.4
4.0	-0.90	0.05	1.58	66.7
Overall	-3.45	0.17	3.87	71.6

(b)

Image	Samp. Density (%)	PSNR (dB)		
		mirroring	classic	diff.
lena	0.5	**19.18**	18.00	1.18
	1.0	**22.12**	21.66	0.46
	2.0	**25.77**	25.56	0.21
	3.0	**27.73**	27.39	0.34
	4.0	**28.83**	28.57	0.26
pens	0.5	**15.78**	15.02	0.76
	1.0	**19.27**	18.17	1.10
	2.0	**23.48**	22.68	0.80
	3.0	**25.91**	25.80	0.11
	4.0	**27.92**	27.20	0.72
bluegirl	0.5	**21.17**	19.49	1.68
	1.0	**25.30**	22.65	2.65
	2.0	**29.40**	25.72	3.68
	3.0	**31.90**	29.08	2.82
	4.0	**33.40**	31.82	1.58

experiments by using other choices of fixed parameters (i.e., using other choices of initSampSelPolicy), and the results obtained were found to be similar to those in Table 3.3.

Examining the results of Table 3.3(a), we can make several observations. First, the mirroring method outperforms the classic method at all sampling densities. For example, the mirroring method beats the classic method in more than 29/45 (64.6%) of the test cases at each sampling density. Looking at the full results in more detail, we find that the mirroring method outperforms the classic method in 161/225 (71.6%) of the test cases by a margin of 0.01 to 3.87 dB. Second, at lower sampling densities, the fraction of cases in which the mirroring method outperforms the classic method is higher, and the differences in PSNR by which mirroring beats classic are larger. For example, at the sampling density of 0.5% and 1.0%, the mirroring method outperforms the classic method in 35/45 (77.8%) and 36/45 (80.0%) of test cases, and the PSNR differences in the margin by which mirroring beats classic have a median value of 0.76 dB and 0.38 dB, respectively. Lastly, at higher sampling densities, the mirroring method still outperforms the classic method, but the differences become smaller. For example, at each of the sampling densities of 2.0%, 3.0%, and 4.0%, the mirroring method beats the classic method in about 65% of test cases, and the median differences in PSNR by which mirroring beats classic is less than 0.19 dB. The results for some individual test cases shown in Table 3.3(b) are consistent with the above observations, where the mirroring method outperforms the classic method in all test cases, by a margin of 0.11 to 3.68 dB.

The reason that the mirroring method beats the classic method can be largely attributed to the startup effect that exists in the classic method, as described earlier in Section 3.2. In particular, at lower sampling densities, by using the classic method, an abnormally small number of sample points are selected at the bottom region of the image domain. The mirroring method, however, does not suffer from such a problem. To illustrate the effect of the above behaviors on mesh quality, an example is provided in Figure 3.6. For a RGB-color image, we generate a mesh using each of the classic and mirroring methods and present the corresponding image-domain triangulations in Figures 3.6(a) and (b), respectively. We can see clearly, in Figure 3.6(a), that the number of sample points selected in the bottom region of the image domain is quite small, resulting in many large sliver triangles. This leads to a high approximation error when reconstructing the image using the mesh. In contrast, in the case of using the mirroring method, a better point distribution is obtained at the bottom of the image domain, as shown in Figure 3.6(b).

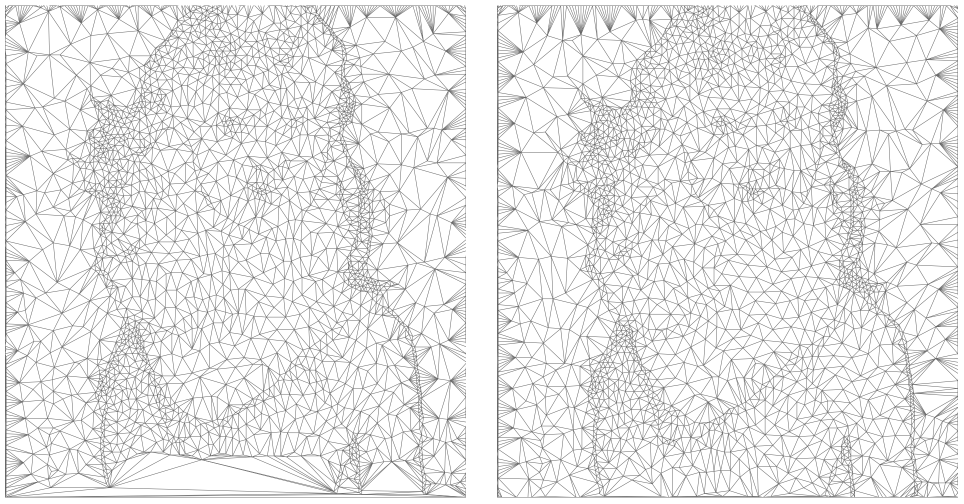


Figure 3.6: Triangulations obtained for the color bluegirl image at a sampling density of 1% in the ED-like mode with error diffusion employing the (a) classic and (b) mirroring methods.

3.6.2 Grayscale Case

Having discussed the effect of parameter choices on the mesh quality in the ED-like mode for RGB-color images, we now consider the case of grayscale images (i.e., $M = 1$). As mentioned earlier, if $M = 1$, the PSD and PSE policies are equivalent to the PSA policy. Thus, in our work, though the PSA, PSD, and PSE policies are all applicable to grayscale images, we only study the case of applying the PSA policy in the framework for grayscale images. The impact of the choices of the other free parameters on mesh quality, including the significance function within the PSA policy (or equivalently, the significance function within the PSD or PSE policy), and startupPolicy, will be discussed.

3.6.2.1 Significance Function

To begin, we study how the choice of the significance function in the PSA policy (or equivalently, the PSD or PSE policy) affects mesh quality. To do this, we select the parameter startupPolicy as classic. For each of the 45 grayscale images in the test set and five sampling densities per image (for a total of $45 \cdot 5 = 225$ test cases), we generated a mesh using each of the significance functions (i.e., MMSODD and MoL), and measured the approximation error of the reconstructed image in terms of PSNR. For each sampling density as well as overall, we calculated the fraction of cases where the MMSODD function outperforms the MoL function, in addition to the minimum, median, and maximum difference in PSNR by which MMSODD beats MoL. The overall results are shown in Table 3.4(a). Results for some individual test cases are shown in Table 3.4(b). In each of these tables, the best result for each test case is shown in bold font. We also conducted similar experiments by using other choices of fixed parameters (i.e., using a startup policy of mirroring), and the results obtained were found to be similar to those in Table 3.4.

Examining the results of Table 3.4(a), we can see that the significance function MMSODD outperforms MoL at all sampling densities. Looking at the full results in more detail, we find that the MMSODD function outperforms the MoL function in 208/225 (92.4%) of test cases by a margin of 0.01 to 5.37 dB. The results for individual test cases shown in Table 3.4(b) are consistent with the overall results. For example, the MMSODD function outperforms the MoL function in 14/15 (93.3%) of the test cases. The reason for this behavior is similar to what we described earlier in Section 3.6.1 for RGB-color images. That is, the density function obtained from the MMSODD function has a stronger response to images edges than the density function obtained from the MoL function.

3.6.2.2 FSED Startup Policy

Lastly, we study how the choice of the startup policy in the FSED algorithm affects mesh quality. To do this, we select the parameter initSampSelPolicy as PSA(D). For each of the 45 grayscale images and five sampling densities, we generated a mesh using each of the startup policies (i.e., mirroring and classic), and measured the approximation error of the reconstructed image in terms of PSNR. For each sampling density as well as overall, we calculated the fraction of cases where the mirroring method outperforms the classic method, in addition to the minimum, median, and maximum difference in PSNR by which mirroring beats classic. The overall results are shown in Table 3.5(a). Results for some individual test cases are shown in Table 3.5(b). We also conducted similar experiments by using other choices of fixed parameters (i.e., using PSA(L) as initSampSelPolicy), and the results obtained were found to be similar to those in Table 3.5.

Examining the results of Table 3.5(a), we can make similar observations as what we saw previously for RGB-color images in Section 3.6.1. That is, the mirroring method beats the classic method at all sampling densities. Looking at the full results

Table 3.4: Comparison of using different significance functions within the PSA policy in the ED-like mode. (a) Overall results for the change in PSNR obtained by using the MMSODD function compared to the MoL function, where win ratio is the fraction of cases the MMSODD function beats the MoL function. (b) PSNR results for some representative images, where the column diff shows the PSNR differences in the margin by MMSODD beats MoL.

(a)

Samp. Density (%)	PSNR Change (dB)			Win Ratio (%)
	Minimum	Median	Maximum	
0.5	-0.53	0.47	2.13	82.2
1.0	-1.05	0.49	5.37	93.3
2.0	-0.20	0.36	1.27	91.1
3.0	0.02	0.38	1.94	100.0
4.0	-0.53	0.38	1.49	95.6
Overall	-1.05	0.42	5.37	92.4

(b)

Image	Samp. Density (%)	PSNR (dB)		
		MMSODD	MoL	Diff.
lena	0.5	**17.17**	17.12	0.05
	1.0	**21.13**	20.96	0.17
	2.0	**25.83**	25.40	0.43
	3.0	**28.06**	27.78	0.28
	4.0	**29.59**	29.40	0.19
pens	0.5	**14.37**	14.04	0.33
	1.0	**17.83**	17.46	0.37
	2.0	**22.87**	22.58	0.29
	3.0	**26.19**	25.87	0.32
	4.0	**28.10**	27.89	0.21
bluegirl	0.5	**19..99**	19.15	0.84
	1.0	**22.72**	21.46	1.26
	2.0	25.42	**25.54**	-0.12
	3.0	**30.10**	28.76	1.34
	4.0	**28.76**	32.09	1.49

in more detail, we find that the mirroring method outperforms the classic method in 155/225 (68.9%) of test cases by a margin of 0.01 to 4.72 dB. In particular, at lower sampling densities, the fraction of cases in which mirroring outperforms classic is higher, and the differences in PSNR by which mirroring beats classic is larger. For example, at sampling densities of 0.5% and 1.0% respectively, the mirroring method

Table 3.5: Comparison of using different startup policies in the ED-like mode for grayscale images. (a) Overall results for the change in PSNR obtained by using the mirroring method compared to the classic method, where win ratio is the fraction of cases the mirroring method beats the classic method. (b) PSNR results for some representative images, where the column diff shows the PSNR differences in the margin by mirroring beats classic.

(a)

Samp. Density (%)	PSNR Change (dB)			Win Ratio (%)
	Minimum	Median	Maximum	
0.5	-1.63	0.61	2.40	84.4
1.0	-3.33	0.35	3.51	75.6
2.0	-2.72	0.10	4.72	55.6
3.0	-0.92	0.11	2.46	66.7
4.0	-1.81	0.03	3.08	62.2
Overall	-3.33	0.16	4.72	68.9

(b)

Image	Samp. Density (%)	PSNR (dB)		
		mirroring	classic	diff.
lena	0.5	**17.96**	17.17	0.79
	1.0	**21.83**	21.13	0.70
	2.0	**26.14**	25.83	0.31
	3.0	**28.43**	28.06	0.37
	4.0	**29.76**	29.59	0.17
pens	0.5	**15.65**	14.37	1.28
	1.0	**19.24**	17.83	1.41
	2.0	**23.96**	22.87	1.09
	3.0	**26.93**	26.19	0.74
	4.0	**28.79**	28.10	0.69
cartoon_bull	0.5	**26.57**	26.13	0.44
	1.0	**33.62**	33.38	0.24
	2.0	37.57	**37.63**	-0.06
	3.0	**40.35**	40.33	0.02
	4.0	**41.57**	41.55	0.02

beats the classic method in 38/45 (84.4%) and 34/45 (75.6%) of the test cases, and the median differences in PSNR by which mirroring beats classic are 0.61 dB and 0.35 dB. The individual results in Table 3.5(b) shows that the mirroring method outperforms the classic method in 14/15 of test cases by a margin of 0.02 to 1.41 dB. The above behavior can be attributed to the startup effect in the classic method, similar to what we described earlier for the RGB-color image case in Section 3.6.1.

3.7 Impact on Different Parameter Choices for the GPRFSED-like Mode

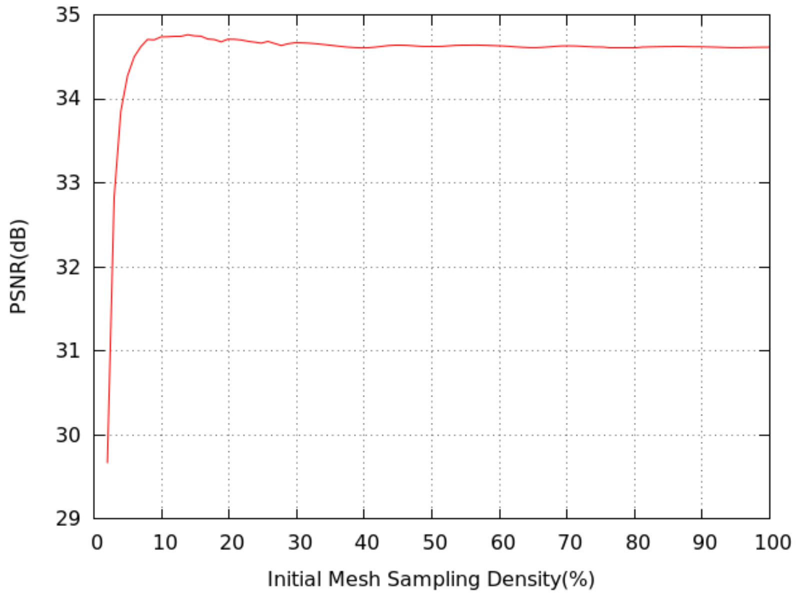
Having discussed the effect of the choice of the free parameters on mesh quality for the ED-like mode in our framework, we now consider the impact of parameter choices on mesh quality for the GPRFSED-like mode. First, we consider the effect of the choice of free parameters on mesh quality for RGB-color images. Then, we consider the case of grayscale images.

3.7.1 RGB-Color Case

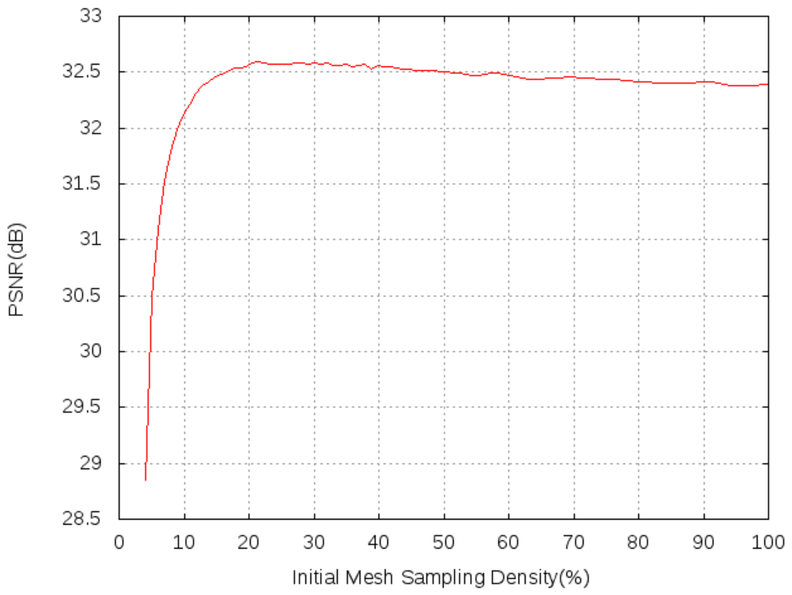
We start by analyzing the impact of the parameter choices on mesh quality in the GPRFSED-like mode in the case of RGB-color images. The impact of choices of free parameters on mesh quality, including N_0, initSampSelPolicy, and startupPolicy, will be discussed.

3.7.1.1 Initial Mesh Size N_0

In our work, we aim to find a good strategy to choose the initial mesh size N_0, or equivalently, the initial sampling density $D_0 = \frac{N_0}{WH}$. To do this, we selected the parameters such that initSampSelPolicy is PSD(Max, D) and startupPolicy is mirroring. For each of the 45 RGB-color images and five sampling densities, the following experiment was conducted. For the given image and desired sampling density D, we measured the mesh quality (in terms of PSNR) as a function of D_0 while keeping D fixed. Figure 3.7 shows the results obtained for two representative test cases. We also conducted similar experiments by using other choices of fixed parameters (i.e., using different choices of initSampSelPolicy and startupPolicy), and found that most results obtained were similar to the graphs shown in Figure 3.7. There were a few exceptions, however. For some combinations of parameter choices the results obtained were quantitatively different, but from a practical point of view, they are not of our interest due to their poor performance. As we will show later, the choice of PSD(Max, D) as initSampSelPolicy and mirroring as startupPolicy is still superior to the other alternatives.



(a)



(b)

Figure 3.7: Effect of varying the initial sampling density on mesh quality the GPRFSED-like mode for RGB-color images.(a) The kodim23 image with a desired sampling density of 2.0%; and (b) The lena image with a desired sampling density of 4.0%.

From the results of Figure 3.7, we can see that, in both graphs, the PSNR increases rapidly with D_0 initially, and then gradually decreases. This shows that the best mesh quality is not obtained at the point where the initial mesh sampling density is highest (i.e., $D_0 = 100\%$). Instead, the best result is obtained at D_0 chosen much smaller than 100%, where less computational cost is required. Based on further experimentation, we found that, the GPRFSED-like mode can usually achieve a PSNR very close to, and in many cases better than, the case that $D_0 = 100\%$, by choosing D_0 in the approximate range $[4D, 5.5D]$. Consequently, we propose that D_0 be chosen using a simple formula γD, or equivalently, that N_0 be chosen as $N_0 = \gamma N$, where γ is a real constant satisfying $\gamma \in [4, 5.5]$. Typically, for photographic imagery (except at very low sampling densities), in order to achieve results comparable to the case $D_0 = 100\%$, a choice of $\gamma = 4$ is sufficient, in the remaining cases, a choice of $\gamma = 5.5$ may be more appropriate. For the interest of minimizing computational and memory costs, we propose γ nominally be chosen as 4.

3.7.1.2 Initial Sample-Point Selection Policy

Now, we study the effect of the choice of initial sample-point selection policy on mesh quality in the GPRFSED-like mode in the case of RGB-color images. To do this, we select the free parameters such that $N_0 = 4N$ and startupPolicy is mirroring. For each of the 45 RGB-color images and five sampling densities, we generated a mesh using each of the initial sample-point selection policies. In each test case, the PSNR results obtained by different policies were ranked from 1 (best) to 9 (worst). Then, the average and standard deviation of the ranks were computed across each sampling density as well as overall. The overall ranking results are given in Table 3.6(a), with the best result in each row typeset in bold. Results for some individual test cases are shown in Table 3.6(b). We also conducted experiments using other choices of fixed parameters (e.g., using different startup policy), and the results obtained were found to be similar to those in Table 3.6.

Examining the results of Table 3.6, we can clearly see that the PSD(Max, D) and PSD(Avg, D) policies perform best. More specifically, the PSD(Max, D) policy performs slightly better at lower sampling densities, while the PSD(Avg, D) policy performs slightly better at higher sampling densities. The above behavior can be attributed to the fact that, the density function obtained from the PSD(Max, D) policy results in a stronger response to image edges than the PSD(Avg, D) policy. Thus, at higher sampling densities (e.g., 2.0% and higher), due to the larger initial mesh size, having sample points been selected extremely close to each other along image edges is

Table 3.6: Comparison of using the various initial sample-point selection policies in GPRFSED-like mode for RGB-color images. (a) Average ranking across the 45 images in the data set. (b) Some representative images results.

(a)

Samp. Density (%)	PSNR Average Rank (Standard Deviation)								
	PSB (D)	PSB (L)	PSC	PSD (Max, D)	PSD (Max, L)	PSD (Avg, D)	PSD (Avg, L)	PSE (D)	PSE (L)
0.5	4.87 (1.87)	6.67 (1.26)	4.73 (2.32)	**1.96** (1.05)	3.87 (1.39)	2.27 (1.34)	4.40 (1.40)	7.71 (0.96)	8.53 (0.83)
1.0	5.53 (2.00)	7.36 (1.55)	5.84 (2.29)	1.80 (0.91)	4.11 (1.48)	**1.78** (0.89)	4.18 (1.35)	6.58 (1.47)	7.82 (1.22)
2.0	6.44 (1.98)	7.87 (1.48)	7.29 (2.03)	1.98 (1.04)	4.29 (1.26)	**1.73** (1.00)	4.38 (1.18)	4.29 (1.72)	6.73 (1.40)
3.0	6.60 (1.89)	8.13 (0.93)	7.60 (1.84)	2.09 (1.11)	4.44 (1.34)	**1.82** (0.93)	4.47 (1.26)	3.53 (1.67)	6.31 (1.31)
4.0	6.69 (1.82)	8.20 (0.88)	7.80 (1.56)	2.33 (0.94)	4.87 (1.13)	**1.78** (1.05)	4.58 (1.04)	2.98 (1.76)	5.78 (1.65)
Overall	6.03 (2.04)	7.64 (1.38)	6.65 (2.35)	2.03 (1.03)	4.32 (1.37)	**1.88** (1.07)	4.40 (1.26)	5.02 (2.39)	7.04 (1.65)

(b)

Image	Samp. Density (%)	PSNR Average Rank (Standard Deviation)								
		PSB (D)	PSB (L)	PSC	PSD (Max, D)	PSD (Max, L)	PSD (Avg, D)	PSD (Avg, L)	PSE (D)	PSE (L)
pens	0.5	23.43	23.23	23.30	**24.05**	23.93	23.96	23.90	22.22	22.22
	1.0	26.30	26.25	26.26	26.77	26.52	**26.83**	26.56	25.68	26.11
	2.0	28.98	28.90	29.03	29.43	39.35	**29.49**	29.34	29.05	28.98
	3.0	30.69	30.61	30.58	31.16	31.08	**31.18**	31.11	31.02	30.96
	4.0	32.07	31.88	31.82	32.45	32.44	32.49	32.44	**32.51**	32.28
kodim23	0.5	28.21	27.78	28.40	**29.11**	28.97	29.00	28.91	28.14	28.30
	1.0	30.86	30.95	30.96	**31.89**	31.85	31.88	31.71	31.66	31.47
	2.0	33.71	33.70	33.76	**34.71**	34.60	34.65	34.54	34.60	34.50
	3.0	35.45	35.41	35.65	**36.43**	36.35	36.41	36.32	36.35	36.20
	4.0	36.66	36.61	37.01	**37.69**	37.62	37.68	37.60	37.66	37.54
cartoon _bull	0.5	36.30	36.45	38.80	**39.61**	**39.61**	38.96	39.08	36.87	37.51
	1.0	39.55	39.39	42.08	**42.60**	42.28	42.59	42.47	40.49	40.38
	2.0	42.40	42.24	44.66	**44.92**	44.66	44.82	44.58	44.55	44.37
	3.0	44.37	44.14	46.05	**46.24**	45.95	46.23	45.92	46.03	45.85
	4.0	45.89	45.50	47.18	**47.37**	47.07	47.34	47.05	47.23	47.00

more likely to happen, which leads to slightly poorer performance. Nevertheless, we found that the PSNR difference between the PSD(Max, D) and PSD(Avg, D) policies is usually quite small at sampling densities 2.0% and higher, and relatively large at sampling densities 1.0% and lower. For example, for the individual results shown in Table 3.6(b), the PSNR difference between PSD(Max, D) and PSD(Avg, D) is less than 0.10 dB at higher sampling densities. While at lower sampling densities, the PSD(Max, D) policy beats the PSD(Avg, D) policy by a margin of 0.01 to 0.57 dB. Given the fact the PSD(Max, D) policy performs very well in the GPRFSED-like mode and performs better than the PSD(Avg, D) policy in the ED-like mode, we choose PSD(Max, D) nominally in the framework, though incurring a small mesh-quality penalty relative to the PSD(Avg, D) policy at higher sampling densities.

3.7.1.3 FSED Startup Policy

Lastly, we study the effect of the choice of the FSED startup policy on mesh quality in the GPRFSED-like mode. To do this, we select the free parameters such that $N_0 = 4N$ and initSampSelPolicy is PSD(Max, D). For each of the 45 RGB-color images with five sampling densities per image, we generated a mesh using each of the startup policies (i.e., mirroring and classic), and measured the approximation error of the reconstructed image in terms of PSNR. For each sampling density as well as overall, we calculated the fraction of cases where the mirroring method outperforms classic method, in addition to the minimum, median, and maximum differences in PSNR by which mirroring beats classic. The overall results are shown in Table 3.7(a). Results for some individual test cases are shown in Table 3.7(b). We also conducted similar experiments by using other choices of fixed parameters (e.g., using different initial sample-point selection policy), and the results obtained were found to be similar to those in Table 3.7.

Examining the results of Table 3.7(a), we can see that the mirroring method still outperforms the classic method in terms of overall performance. For example, the overall results show that the mirroring method beats the classic method in 134/225 (59.6%) of test cases. More specifically, the mirroring method beats the classic method at sampling densities of 1.0% and lower, and behaves similarly to the classic method at sampling densities of 2.0% and higher. More detailed analysis shows that at sampling densities of 1.0% and lower, the mirroring method beats the classic method in 68/90 (75.6%) of the test cases by a margin of 0.02 to 2.29 dB. At sampling densities of 2.0%

Table 3.7: Comparison of using different startup policies in the GPRFSED-like mode for RGB-color images. (a) Overall results for the change in PSNR obtained by using the mirroring method compared to the classic method, where win ratio is the fraction of cases the mirroring method beats the classic method. (b) PSNR results for some representative images, where the column diff shows the PSNR differences in the margin by mirroring beats classic.

(a)

Samp. Density (%)	PSNR Change (dB)			Win Ratio (%)
	Minimum	Median	Maximum	
0.5	-0.28	0.21	2.29	82.2
1.0	-0.15	0.05	1.30	68.9
2.0	-0.05	0.01	0.20	53.3
3.0	-0.42	-0.01	0.14	48.9
4.0	-0.34	-0.01	0.06	44.4
Overall	-0.42	0.01	2.29	59.6

(b)

Image	Samp. Density (%)	PSNR (dB)		
		mirroring	classic	diff.
lena	0.5	**26.04**	25.83	0.21
	1.0	**28.38**	28.23	0.15
	2.0	30.48	**30.50**	-0.02
	3.0	31.68	**31.71**	-0.03
	4.0	32.49	**32.49**	0.00
pens	0.5	**24.05**	23.37	0.68
	1.0	**26.77**	26.30	0.47
	2.0	29.43	**29.45**	-0.02
	3.0	**31.16**	31.15	0.01
	4.0	**32.45**	32.44	0.01
bluegirl	0.5	**29.37**	27.08	2.29
	1.0	**32.68**	31.38	1.30
	2.0	**35.38**	35.28	0.10
	3.0	**36.85**	36.84	0.01
	4.0	**37.86**	37.85	0.02

and higher, the fraction of cases in which mirroring beats classic is close to 50%, and the PSNR difference between these two startup policies is quite small (i.e., the median difference in PSNR between them is less than 0.01 dB). The above behavior is mainly due to the fact that, at higher sampling densities, the error diffusion threshold τ employed is quite small in the GPRFSED-like mode. Thus the startup effect becomes much less of an issue. At lower sampling densities, however, τ is still high and the performance of the classic method is still influenced by the startup effect. The individual results shown in Table 3.7(b) are consistent with the above observations. That is, the mirroring method beats the classic method at lower sampling densities in all test cases by 0.15 to 2.29 dB, and behaves similarly to the classic method at higher sampling densities.

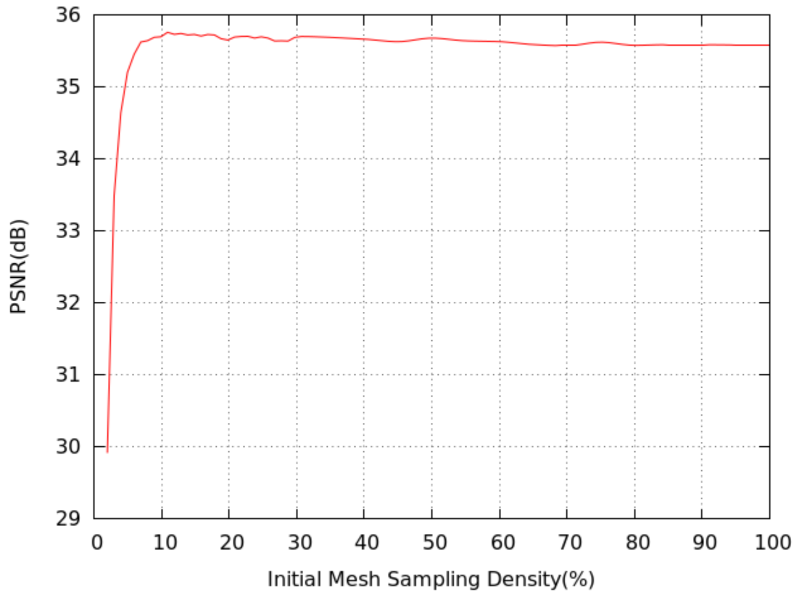
3.7.2 Grayscale Case

For the GPRFSED-like mode, we have discussed the impact of parameter choices on mesh quality for RGB-color images. Now we consider the case of grayscale images. For the initial sample-point selection policy, we still only consider the case of using the PSA policy (which is equivalent to the PSD or PSE policy in this context). The impact of the choice of free parameters on mesh quality, including N_0, the significance function within PSA, and startupPolicy, will be discussed.

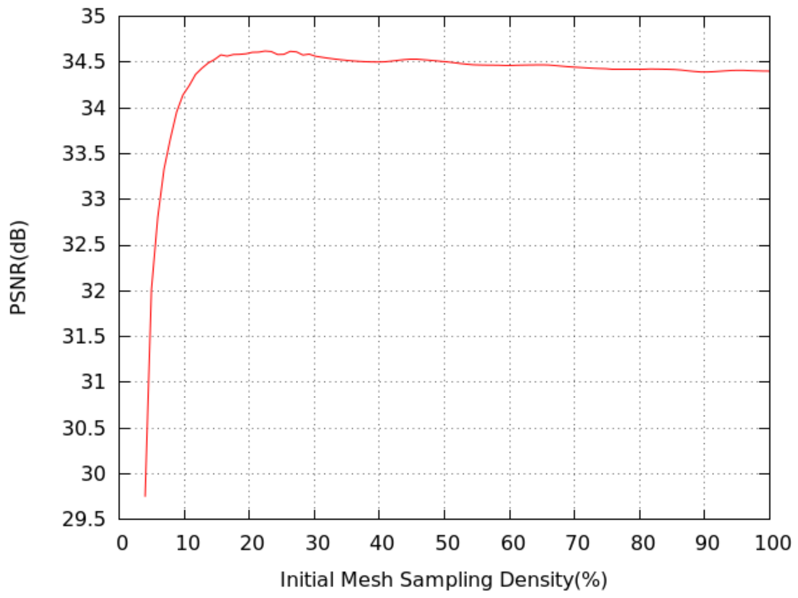
3.7.2.1 Initial Mesh Size N_0

We start by studying the effect of the choice of N_0 on mesh quality in the GPRFSED-like mode. Like what we did for the RGB-color image case in Section 3.7.1, we still aim to find a good strategy to select N_0, or equivalently, the initial sampling density $D_0 = \frac{N_0}{WH}$. To do this, we selected the parameters such that initSampSelPolicy is PSA(D) and startupPolicy is mirroring. For each of the 45 grayscale images and five sampling densities, we conducted experiments as follows. For the given image and desired sampling density D, we measured the mesh quality (in terms of PSNR) as a function of D_0 while keeping D fixed. Figure 3.8 shows the results obtained for two representative test cases. We also conducted similar experiments using other choices of fixed parameters (i.e., using different the choices of significance function within the PSA policy and FSED startup policy), and obtained results similar to the ones shown in Figure 3.8.

Examining the results of Figure 3.8, we can see a similar behavior as earlier for the RGB-color case in Section 3.7.1. That is, in both Figures 3.7(a) and (b), the curve increases rapidly to a maximum and then slowly decreases. Furthermore, based on our experiments, we notice that in most cases the maximum value is located at $D_0 \in [4D, 5.5D]$. Therefore, we still propose that D_0 be chosen as $D_0 = \gamma D$, or equivalently, that N_0 be chosen as $N_0 = \gamma N$, where γ is a real constant satisfying



(a)



(b)

Figure 3.8: Effect of varying the initial sampling density on mesh quality the GPRFSED-like mode for grayscale images.(a) The kodim23 image with a desired sampling density of 2.0%; and (b) The lena image with a desired sampling density of 4.0%.

$\gamma \in [4, 5.5]$. In the interest of minimizing computational cost, we nominally chose γ as 4.

3.7.2.2 Significance Function

Next, we study the effect of the choice of the significance function within the PSA policy (or equivalently, the PSD or PSE policy) on mesh quality in the GPRFSED-like mode for grayscale images. To do this, we select the free parameters such that $N_0 = 4N$ and startupPolicy is mirroring. For each of the 45 grayscale images in the test set and five sampling densities per image, we generated a mesh using each of the significance functions (i.e., MMSODD and MoL), and measured the approximation error of the reconstructed image in terms of PSNR. For each sampling density as well as overall, we calculated the fraction of cases where the MMSODD function outperforms the MoL function, in addition to the minimum, median, and maximum difference in PSNR by which MMSODD beats MoL. The overall results are shown in Table 3.8(a). Results for some individual test cases are shown in Table 3.8(b). We also conducted similar experiments by using other choices of fixed parameters (e.g., using the FSED startup policy of classic), and the results obtained were found to be similar to those in Table 3.8.

Examining the results of Table 3.8(a), we can clearly see that the MMSODD function outperforms the MoL function at all sampling densities. More detailed analysis shows that our the MMSODD function outperforms the MoL function in 211/225 (93.8%) of test cases by a margin of 0.01 to 2.03 dB. The above behavior is still due to the fact that the density function obtained from the MMSODD function has a stronger response to image edges than the MoL function. The individual results shown in Table 3.8(b) are consistent with the overall results, where the MMSODD function beats the MoL function at all sampling densities.

3.7.2.3 FSED Startup Policy

Lastly, we study the effect of the choice of the startup policy in the FSED algorithm on mesh quality. To do this, we select the free parameters such that $N_0 = 4N$ and initSampSelPolicy is PSA(D). For each of the 45 grayscale images and five sampling densities, we generated a mesh using each of the startup policies (i.e., mirroring and classic), and measured the approximation error of the reconstructed image in terms of PSNR. For each sampling density as well as overall, we calculated the fraction of cases where the mirroring method outperforms the classic method, in addition to the minimum, median, and maximum difference in PSNR by which mirroring beats classic. The overall results are shown in Table 3.9(a). Results for some individual

Table 3.8: Comparison of using different significance functions within the PSA policy in the GPRFSED-like mode. (a) Overall results for the change in PSNR obtained by using the MMSODD function compared to the MoL function, where win ratio is the fraction of cases the MMSODD function beats the MoL function. (b) PSNR results for some representative images, where the column diff shows the PSNR differences in the margin by MMSODD beats MoL.

(a)

Samp. Density (%)	PSNR Change (dB)			Win Ratio (%)
	Minimum	Median	Maximum	
0.5	-0.36	0.19	2.03	84.4
1.0	-0.18	0.16	0.56	91.1
2.0	-0.53	0.19	0.33	93.3
3.0	0.02	0.18	0.37	100.0
4.0	-0.07	0.17	0.36	100.0
Overall	-0.53	0.17	2.03	93.8

(b)

Image	Samp. Density (%)	PSNR (dB)		
		MMSODD	MoL	Diff.
lena	0.5	**26.49**	26.45	0.04
	1.0	**29.11**	29.01	0.10
	2.0	**31.92**	31.84	0.08
	3.0	**33.51**	33.34	0.17
	4.0	**34.59**	34.45	0.14
pens	0.5	**24.79**	24.71	0.08
	1.0	**27.79**	27.60	0.19
	2.0	**30.73**	30.69	0.04
	3.0	**32.64**	32.56	0.08
	4.0	**34.20**	34.12	0.08
cartoon _bull	0.5	**43.36**	40.20	0.16
	1.0	**43.54**	43.45	0.09
	2.0	**45.67**	45.35	0.32
	3.0	**46.94**	46.58	0.36
	4.0	**48.03**	47.67	0.36

test cases are shown in Table 3.9(b). We also conducted similar experiments by using other choices of fixed parameters (e.g., using PSA(L) for initSampSelPolicy), and the results obtained were found to be similar to those in Table 3.9.

Examining the results of Table 3.9(a), we observe a similar behavior as that seen earlier in Section 3.7.1 for the RGB-color image case. That is, the mirroring method

Table 3.9: Comparison of using different startup policies in the GPRFSED-like mode for grayscale images. (a) Overall results for the change in PSNR obtained by using the mirroring method compared to the classic method, where win ratio is the fraction of cases the mirroring method beats the classic method. (b) PSNR results for some representative images, where the column diff shows the PSNR differences in the margin by mirroring beats classic.

(a)

Samp. Density (%)	PSNR Change (dB)			Win Ratio (%)
	Minimum	Median	Maximum	
0.5	-3.27	0.19	2.88	77.8
1.0	-0.50	0.03	1.20	71.1
2.0	-0.82	-0.01	0.22	46.7
3.0	-0.40	0.01	0.27	68.9
4.0	-0.38	-0.01	0.10	48.9
Overall	-3.27	0.01	2.88	62.7

(b)

Image	Samp. Density (%)	PSNR (dB)		
		mirroring	classic	Diff.
lena	0.5	**26.49**	26.22	0.27
	1.0	**29.11**	29.00	0.11
	2.0	**31.92**	31.86	0.06
	3.0	**33.51**	33.50	0.01
	4.0	**34.59**	34.54	0.05
pens	0.5	**24.79**	24.10	0.69
	1.0	**27.79**	27.43	0.36
	2.0	30.73	**30.77**	-0.04
	3.0	32.64	**32.69**	-0.05
	4.0	34.20	**34.21**	-0.01
cartoon _bull	0.5	**40.36**	39.85	0.51
	1.0	43.54	**43.55**	-0.01
	2.0	45.67	**45.70**	-0.03
	3.0	**46.94**	**46.94**	0
	4.0	**48.03**	**48.03**	0

outperforms the classic method at lower sampling densities, and is comparable to the classic method at higher sampling densities. Looking at the full results in more detail, we find that at lower sampling densities, the mirroring method beats the classic method in 67/90 (74.4%) of the test cases by a margin of 0.01 to 2.88 dB. The individual results shown in Table 3.9(b) are consistent with the overall results.

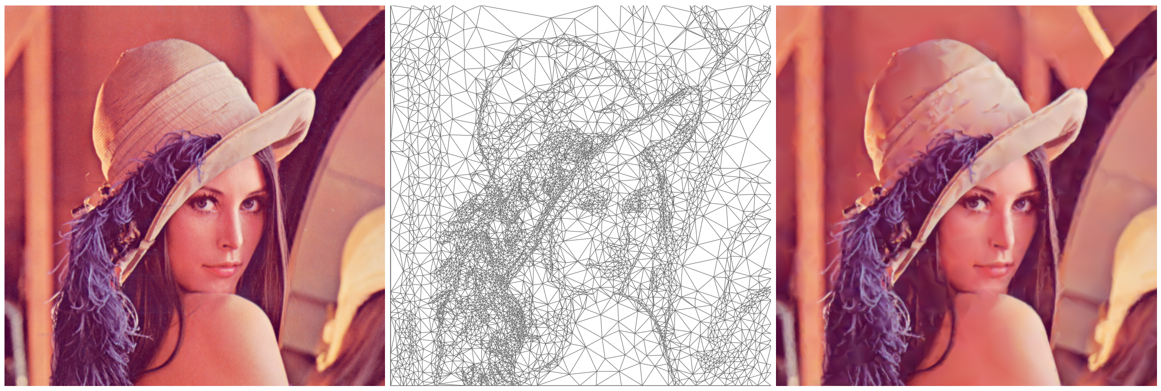


Figure 3.9: (a) The color lena image. (b) Triangulation obtained at the sampling density of 2.0% in the GPR-like mode, and (c) the reconstructed image.

That is, the mirroring method beats the classic method in 5/6 (83.0%) of test cases at lower sampling densities, and performs similarly to the classic method at higher sampling densities.

3.8 Impact on Different Parameter Choices for the GPR-like Mode

Lastly, we consider the the GPR-like mode in our framework. In this mode, since the initial mesh always contains all sample points from the sampling grid, the parameters initSampSelPolicy and startupPolicy are not used. Thus, in this mode, no free parameters can be chosen to affect the mesh quality. Experiments has shown that the GPR-like mode in our framework can produce meshes with very high quality. An example is provided in Figure 3.9. For the RGB-color image in Figure 3.9(a), we generated a mesh at the sampling density of 2.0%. Figures 3.9(b) and (c) show the resulting image-domain triangulation and reconstructed image, respectively. We can see that quality of the reconstructed image is very high compared to the original image.

Although the method constructed from the GPR-like mode has shown to yield meshes with excellent quality, it has one major weakness, namely its very high computational and memory costs. This is mainly due to fact that the initial mesh size is very large, i.e., contains WH points. Thus the mesh-simplification process becomes extremely time consuming. In addition, as shown earlier in Sections 3.7.1.1

and 3.7.2.1, if N_0 is chosen appropriately in the GPRFSED-like mode, the obtained mesh quality (in terms of PSNR) can beat the case that $D_0 = 100\%$, the point at which the result obtained is same as the GPR-like mode. This indicates that our GPRFSED-like mode can yield meshes with higher quality than the the GPR-like mode, while requiring substantially lower computational and memory costs. Thus, the GPRFSED-like mode is more favored compared to the GPR-like mode.

3.9 Proposed Methods

In the preceding sections, we studied how various choices of free parameters affect the performance of our mesh-generation framework in different modes. This leads us to conclude that, in both the ED-like and GPRFSED-like modes, initSampSelPolicy and startupPolicy are best chosen as PSD(Max, D) and mirroring, respectively. In addition, in the GPRFSED-like mode, a choice of $N_0 = \gamma N$, where γ is a real constant satisfying $\gamma \in [4, 5.5]$ and nominally chosen as 4, was also found to be quite effective compared to alternatives. As a result, we chose to propose two methods, known as MED and MGPRFS, which corresponds to the methods with best parameter choices in the ED-like and GPRFSED-like modes, respectively. More specifically, the MED method is derived from the ED-like mode, and it selects the free parameters such that $N_0 = N$, initSampSelPolicy is PSD(Max, D), and startupPolicy is mirroring. The MGPRFS method is derived from the GPRFSED-like mode, and it selects the free parameters such that $N_0 = \gamma N$ with $\gamma \in [4, 5.5]$ and nominally chosen as 4, initSampSelPolicy is PSD(Max, D), and startupPolicy is mirroring. Both of the MED and MGPRFS methods are applicable to images with an arbitrary number of components M. The MED method simply selects the mesh sample points in one shot based on FSED, which has extremely low computational cost, but the mesh quality is not as good compared to the MGPRFS method. The MGPRFS method selects an initial mesh with size larger than desired and then applies mesh-simplification to obtain the final mesh, which can generate meshes with very good quality but requires relatively more computational cost than the MED method.

Chapter 4

Evaluation of the Proposed Methods

Having introduced our two proposed mesh-generation methods for single- and multi-component images, we now evaluate their performance in terms of mesh quality and computational cost. We evaluate these methods based on the two most representative cases of single- and multi-component images, namely, grayscale and RGB-color images. We compare our methods to three highly-effective single-component mesh-generation schemes, namely, ED, GPRFSED, and GPR. As mentioned earlier, in terms of generating mesh models for images, the vast majority of the methods proposed over the years are for single-component (i.e., grayscale) images, only few methods for multi-component (e.g., RGB) images. In addition, authors normally do not want to make their software/code available due to the large effort required for implementation. Thus, approaches to evaluate the performance of our proposed methods for color images is very limited. Ideally, we want to compare with mesh-generation methods which are PDDT-based and having the function value of the mesh model being continuous. To the knowledge of the author of this thesis, however, no such methods for color/multi-component images have been proposed. Consequently, for color case, in the absence of other mesh-generation methods for comparison, the most obvious approach to compare is using single-component mesh generators adapted in a very straightforward way to handle color images. We adapt the three effective single-component mesh-generation methods, namely, ED, GPRFSED, and GPR, in a straightforward way as follows. RGB-color images are handled through conversion to grayscale as a preprocessing step, and then as a postprocessing step after mesh generation, the grayscale sample values in the generated mesh were replaced by their corresponding RGB values. The resulting color-capable versions of ED, GPRFSED, and GPR are henceforth referred to as CED, CGPRFSED, and CGPR, respectively. In addition, to better show the effectiveness of our MGPRFS method, we also compare to the method that results from the GPR-like mode (i.e., $N_0 = WH$) of our framework, which we refer to as MGPR. For the case of grayscale images, the MGPR method is equivalent to the GPR method. The software implementations of the various methods used in this evaluation were developed by the author of this thesis and are written in C++. In what follows, we evaluate the performance of our proposed MED and MGPRFS methods in terms of mesh quality, computational cost, and memory cost by comparing with the ED, CED, GPRFSED, CGPRFSED, GPR, CGPR, and MGPR methods.

4.1 Mesh Quality

We first compare the mesh quality obtained from the various mesh-generation methods. For each of the 45 color images and 45 grayscale images in our test set (where the grayscale images were generated from those color images) and five sampling densities per image, we generated a mesh using each of the methods under consideration. In each test case, the difference between the reconstructed image obtained from the mesh and the original image was measured in terms of PSNR using (2.5). Note that, competing methods to which we compare impose limitations on the type of data that they can handle (i.e., restrictions on the number of components). For example, the methods CED, CGPRFSED, and CGPR can only handle RGB-color images, and the methods ED, GPRFSED, and GPR can only handle grayscale images. Thus, in order to analyze the experimental results, we partition them into two subsets: 1) results for RGB-color images for the methods CED, MED, CGPRFSED, MGPRFS, CGPR, and MGPR; and 2) results for grayscale images for the methods ED, MED, GPRFSED, MGPRFS, and GPR (or equivalently, MGPR). The results for each of the subsets are analyzed separately. For each of the RGB-color and grayscale subsets, the methods were ranked for each test case from best to worst, where 1 is best. The average and standard deviation of the ranks were computed across each sampling density as well as overall with the results presented in Tables 4.1(a) and (b). Results for some individual test cases are shown in Tables 4.2(a) and (b). In what follows, we make various comparisons based on the data in these tables.

Table 4.1: Ranking of mesh quality obtained with the various mesh-generation methods for (a) RGB-color and (b) grayscale images

(a)

Samp. Density (%)	PSNR Average Rank (Standard Deviation)					
	CED	MED	CGPRFSED $\gamma = 4$	MGPRFS $\gamma = 4$	CGPR	MGPR
0.5	5.88 (0.32)	5.12 (0.32)	3.81 (0.45)	2.24 (0.61)	2.67 (0.92)	**1.29** (0.55)
1.0	5.90 (0.29)	5.10 (0.29)	3.40 (0.66)	**1.57** (0.73)	3.36 (0.65)	1.67 (0.71)
2.0	5.90 (0.29)	5.10 (0.29)	3.12 (0.66)	**1.36** (0.53)	3.69 (0.51)	1.83 (0.69)
3.0	5.81 (0.39)	5.17 (0.43)	3.07 (0.63)	**1.33** (0.52)	3.76 (0.53)	1.86 (0.68)
4.0	5.81 (0.39)	5.19 (0.39)	3.02 (0.60)	**1.33** (0.56)	3.79 (0.51)	1.86 (0.60)
Overall	5.86 (0.34)	5.13 (0.35)	3.29 (0.67)	**1.57** (0.69)	3.45 (0.77)	1.70 (0.68)

(b)

Samp. Density (%)	PSNR Average Rank (Standard Deviation)				
	ED	MED	GPRFSED $\gamma = 4$	MGPRFS $\gamma = 4$	GPR/MGPR
0.5	4.83 (0.37)	4.17 (0.37)	2.79 (0.41)	2.12 (0.50)	**1.10** (0.37)
1.0	4.76 (0.43)	4.24 (0.43)	2.38 (0.72)	**1.71** (0.76)	1.90 (0.81)
2.0	4.57 (0.49)	4.43 (0.49)	**1.76** (0.68)	1.90 (0.78)	2.33 (0.86)
3.0	4.64 (0.48)	4.36 (0.48)	2.10 (0.72)	**1.52** (0.66)	2.38 (0.82)
4.0	4.62 (0.49)	4.38 (0.49)	**1.71** (0.59)	1.79 (0.74)	2.50 (0.85)
Overall	4.69 (0.46)	4.31 (0.46)	2.15 (0.75)	**1.81** (0.72)	2.04 (0.92)

Table 4.2: Comparison of mesh quality obtained with the various mesh-generation methods for (a) RGB-color and (b) grayscale images

(a)

Image	Samp. Density (%)	PSNR (dB)					
		CED	MED	CGPRFSED $\gamma = 4$	MGPRFS $\gamma = 4$	CGPR	MGPR
lena (color)	0.5	17.48	19.18	25.63	26.04	26.09	**26.15**
	1.0	21.31	22.12	28.02	**28.38**	28.09	28.28
	2.0	25.54	25.77	30.33	**30.48**	30.13	30.44
	3.0	27.42	27.73	31.44	**31.68**	31.29	31.58
	4.0	28.82	28.83	32.13	**32.49**	32.04	32.39
pens (color)	0.5	13.96	15.78	22.49	24.05	23.60	**24.31**
	1.0	17.24	19.27	25.95	**26.77**	26.40	26.76
	2.0	21.98	23.48	29.08	**29.43**	29.12	29.42
	3.0	25.05	25.91	30.59	**31.16**	30.77	31.15
	4.0	27.05	27.92	31.97	**32.45**	32.01	32.44
bluegirl (color)	0.5	19.73	21.17	27.10	29.37	**29.68**	29.67
	1.0	22.49	25.30	31.99	32.67	32.54	**32.74**
	2.0	25.29	29.40	34.97	35.38	34.98	**35.42**
	3.0	29.49	31.90	36.39	**36.85**	36.33	36.82
	4.0	32.67	33.40	37.29	**37.86**	37.23	37.75

(b)

Image	Samp. Density (%)	PSNR (dB)				
		ED	MED	GPRFSED $\gamma = 4$	MGPRFS $\gamma = 4$	GPR/MGPR
lena (grayscale)	0.5	17.17	17.96	26.22	26.49	**26.55**
	1.0	21.13	21.83	29.00	**29.11**	29.10
	2.0	25.83	26.14	31.86	**31.92**	31.80
	3.0	28.06	28.43	33.50	**33.51**	33.33
	4.0	29.59	29.76	34.54	**34.59**	34.40
pens (grayscale)	0.5	14.37	15.65	24.10	24.79	**25.36**
	10.	17.83	19.24	27.43	**27.79**	27.77
	2.0	22.87	23.96	**30.77**	30.73	30.68
	3.0	26.19	26.93	**32.69**	32.64	32.62
	4.0	28.10	28.79	**34.21**	34.20	34.16
bluegirl (grayscale)	0.5	19.99	20.76	27.72	30.59	**30.97**
	1.0	22.72	26.24	33.43	34.18	**34.31**
	2.0	25.42	30.14	37.40	**37.51**	37.49
	3.0	30.10	32.55	39.21	**39.31**	39.21
	4.0	33.58	34.34	40.52	**40.55**	40.47

MED versus CED and ED. As mentioned earlier, our MED method is an improved and extended version of the ED method. Because of this, we compare our MED method to the ED method and its color-capable version CED here. For RGB-color images, examining the results of Table 4.1(a), we can see that our MED method ranks clearly better than the CED method at all of the sampling densities. For grayscale images, from Table 4.1(b), we can see that our MED method also outperforms the ED method at all sampling densities. More detailed analysis shows that, for RGB-color images, our MED method beats the CED method in 190/225 (84.4%) of the test cases by up to 7.08 dB. For grayscale images, our MED method beats the ED method in 155/225 (68.9%) of the test cases by up to 4.72 dB. Consequently, in terms of mesh quality, our MED method is clearly superior to both the CED and ED methods.

MGPRFS versus CGPRFSED and GPRFSED. As mentioned earlier, our MGPRFS method is an improved and extended version of the GPRFSED method. For this reason, we compare our MGPRFS method to the GPRFSED method and its color-capable version CGPRFSED here. For RGB-color images, examining the overall results of Table 4.1(a), we can see that our MGPRFS method ranks clearly better than the CGPRFSED method at all sampling densities. For grayscale images, examining the results of Table 4.1(b), we can see that, the ranking between our MGPRFS method and the GPRFSED method are mixed at some sampling densities, but overall our MGPRFS method ranks better than the GPRFSED method. Looking at the full results in more detail, we find that, for RGB-color images, our MGPRFS method beats the CGPRFSED method vastly in 220/225 (97.8%) of the test cases by up to 7.05 dB. For grayscale images, our MGPRFS method beats the GPRFSED method at sampling densities of 1.0% and lower in 67/90 (74.4%) of the test cases by up to 2.88 dB, and performs similarly to the GPRFSED method at higher sampling densities. The individual results shown in Tables 4.2 (a) and (b) are consistent with the overall ranking results. For RGB-color images, our MGPRFS method beats the CGPRFSED method in 15/15 of the test cases by 0.15 to 2.88 dB. For grayscale images, our MGPRFS method beats the GPRFSED method in 12/15 of the test cases by 0.03 to 2.86 dB. Consequently, in terms of overall performance on mesh quality, our MGPRFS method is superior to both the CGPRFSED and GPRFSED methods.

MGPRFS versus CGPR, MGPR, and GPR. We now compare our MGPRFS method to the CGPR, MGPR and GPR methods. For reasons we shall see shortly, such a comparison is quite interesting as the method with lower complexity also yields the best result. For RGB-color images, examining the results of Table 4.1(a), we can see that, our MGPRFS method has a better overall rank than both the CGPR and MGPR methods. More specifically, our MGPRFS method ranks clearly better than the CGPR method at all sampling densities. Our MGPRFS method outperforms the MGPR method at most of the sampling densities (except 0.5%). For grayscale images, examining the results of Table 4.1(b), we can see that, our MGPRFS method also performs better than the GPR/MGPR method at most sampling densities (except 0.5%). Looking at the full results in more detail, we find that, for RGB-color images, our MGPRFS method vastly outperforms the CGPR method in 200/225 (89.0%) of the test cases by up to 5.15 dB. In addition, for RGB-color images, our MGPRFS method yields meshes with quality comparable to, and in many cases better than, the MGPR method. In particular, at the sampling densities of 1.0% and higher, our MGPRFS method beats the MGPR method in 123/180 (68.3%) of the test cases by 0.01 to 0.29 dB. At the sampling density of 0.5%, our MGPRFS method performs slightly worse than the MGPR method, typically, by less than 0.6 dB in most cases (which is clearly still comparable). Nevertheless, at the sampling density of 0.5%, as will be seen shortly, our MGPRFS method requires from about 16 to 40 times less computational cost and about 50 times less memory cost than the MGPR method. Thus, any small difference in mesh quality at the sampling density of 0.5% is arguably a small price to pay, considering the reduction in computational and memory costs. For grayscale images, in terms of the comparison between our MGPRFS method and the GPR/MGPR method, we obtained similar results to our comparison between the MGPRFS and MGPR methods for RGB-color images. That is, our MGPRFS method outperforms the GPR/MGPR method at sampling densities of 1.0% and higher in 125/180 (69.4%) of the test cases by up to 0.42 dB. At the sampling density of 0.5%, our MGPRFS method performs slightly worse than the GPR/MGPR method, but is still competitive, especially when one considers the reduction in computational/memory cost of our MGPRFS method. Consequently, in terms of mesh quality, our MGPRFS method is superior to the CGPR method and is comparable to, or better than, the MGPR and GPR methods.

MGPRFS versus MED. Lastly, to show the tradeoff between complexity and mesh quality for our proposed MGPRFS and MED methods, we make a comparison between them. As mentioned earlier, compared to the MED method, our MGPRFS method trades off complexity for mesh quality. Detailed analysis shows that, for both

RGB-color and grayscale cases, our MGPRFS method beats the MED method in all test cases, by a margin of 3.66 to 13.80 dB. Consequently, our MGPRFS method is particularly beneficial for applications which require relatively higher mesh quality.

Subjective Quality. In the above evaluation, PSNR was found to correlate reasonably well with subjective quality. For the benefit of the reader, however, to illustrate the subjective quality achieved by the various methods, we provide an example for each of the RGB-color and grayscale cases. All of the following examples are selected from our $225 \cdot 2 = 450$ test cases for Table 4.1. We begin by examining one of the test cases for RGB-color images, specifically for the cartoon_bull image at a sampling density of 0.5%. For the region of interest shown enlarged in Figure 4.1(a), we present the resulting image-domain triangulation and reconstructed image obtained from each of the various methods considered. Examining the results of Figures 4.1 and 4.2, we can clearly see that, the quality of the reconstructed images obtained from our MGPRFS method is comparable to (if not better than) the MGPR method, and is clearly superior to the CGPRFSED and CGPR methods. Also, the quality of the reconstructed image produced by our MED method is superior to the CED method. For grayscale images, we present another set of examples in Figures 4.3 and 4.4, specifically for the bluegirl image. Examining the results, we can see that, the reconstructed image produced by our MGPRFS method is superior to the GPRFSED method (in the bottom region of the image), and comparable to the GPR/MGPR method. Moreover, the reconstructed image produced by our MED method is better than the ED method (in the bottom region of the image).

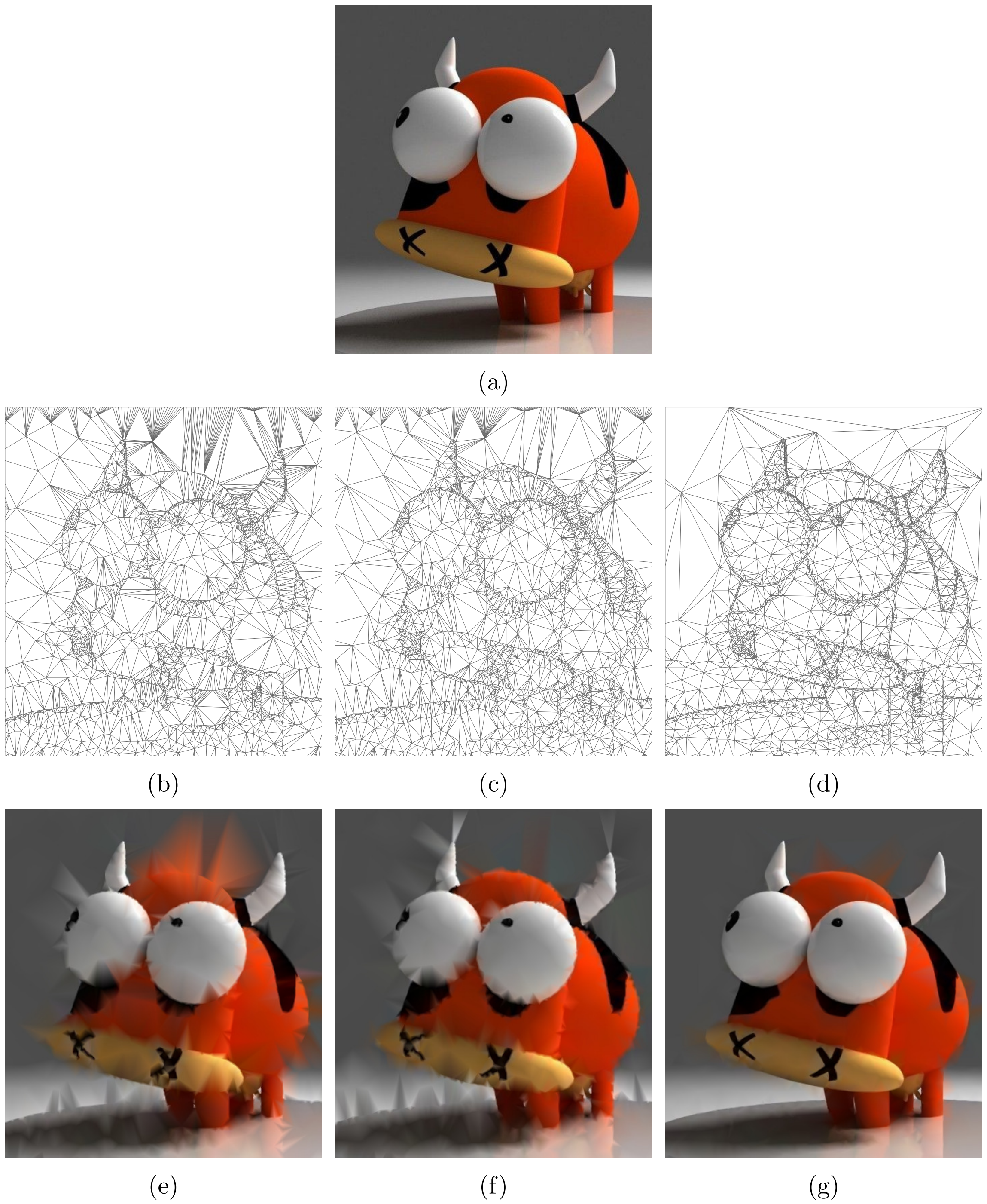
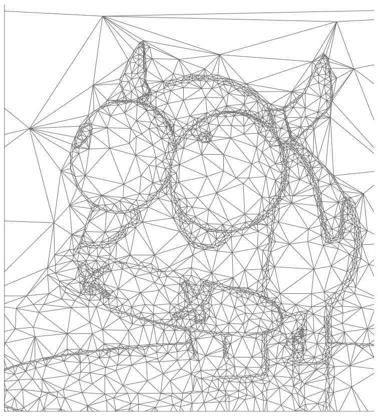
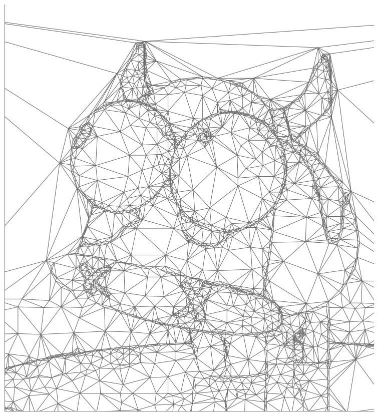
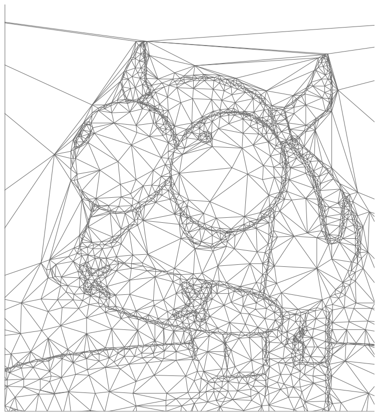
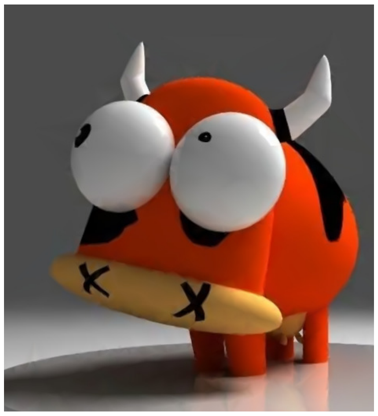


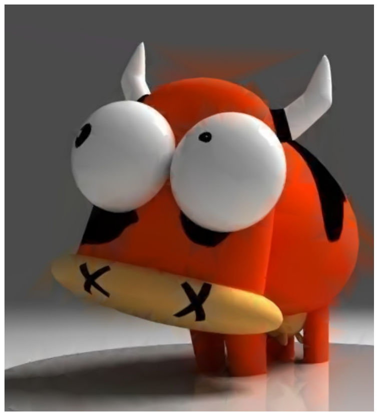
Figure 4.1: (a) Part of the RGB cartoon_bull image. Triangulations obtained at a sampling density of 0.5% using the (b) CED, (c) MED, and (c) CGPRFSED methods, and the reconstructed images obtained using the (e) CED (25.23 dB), (f) MED (26.17 dB), and (g) CGPRFSED (36.98 dB) methods.


(a)


(b)


(c)


(d)


(e)

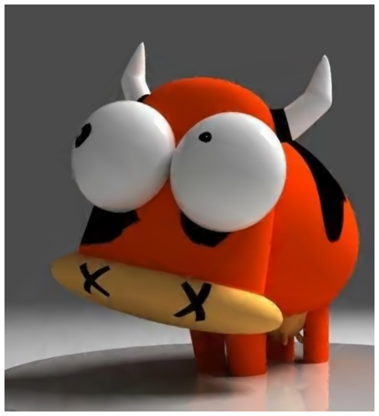
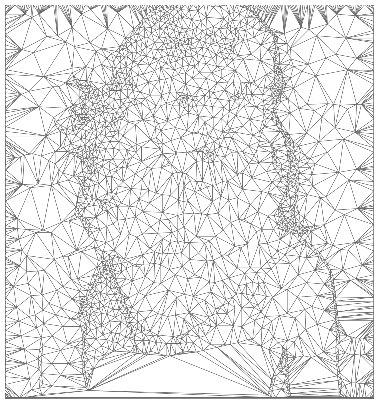
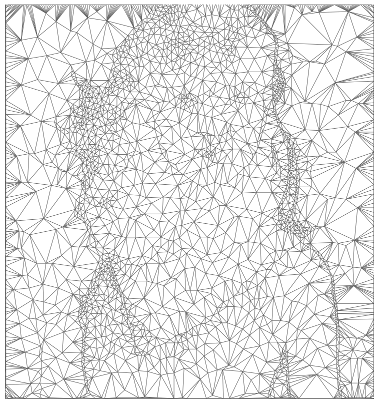

(f)

Figure 4.2: Part of triangulations obtained for the RGB cartoon_bull image at a sampling density of 0.5% using the (a) MGPRFS, (b) CGPR, and (c) MGPR methods, and the reconstructed images obtained using the (d) MGPRFS (39.61 dB), (e) CGPR (36.53 dB), and (f) MGPR (40.57 dB) methods.

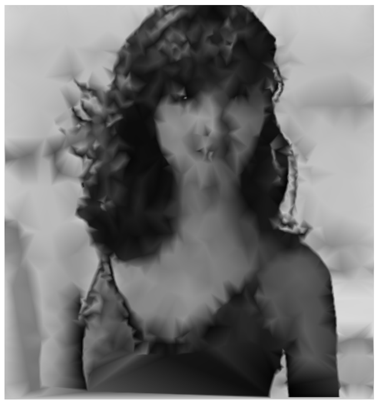
(a)



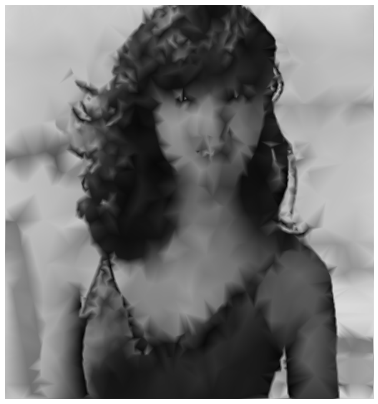
(b)



(c)

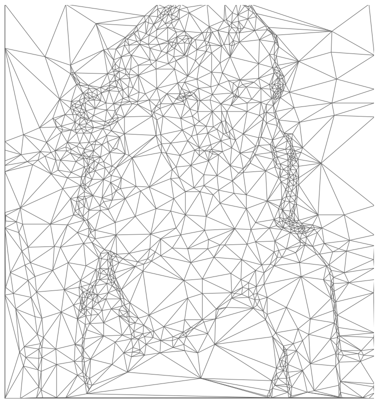


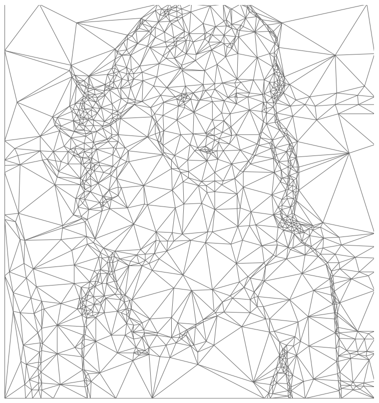
(d)

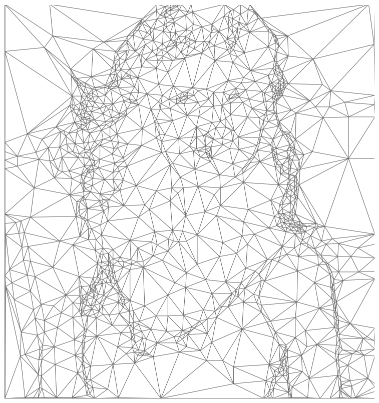


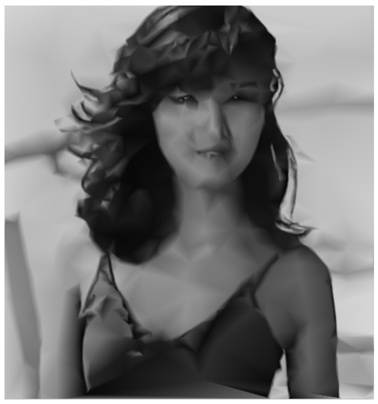
(e)

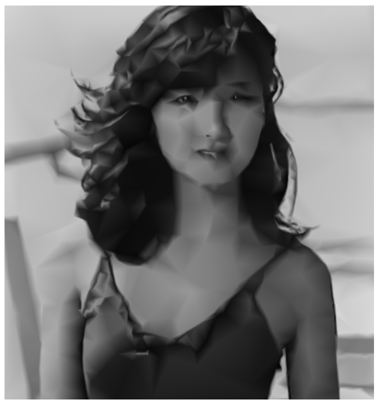
Figure 4.3: (a) The grayscale bluegirl image. Triangulations obtained at a sampling density of 1.0% using the (b) ED and (c) MED methods, and the reconstructed images obtained using the (d) ED (22.72 dB) and (e) MED (26.24 dB) methods.


(a)


(b)


(c)


(d)


(e)

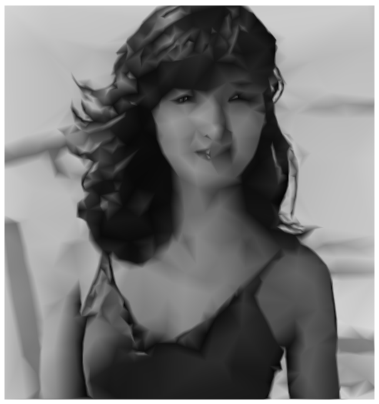

(f)

Figure 4.4: Triangulations obtained for grayscale bluegirl image at a sampling density of 0.5% using the (a) GPRFSED, (b) MGPRFS, and (c) GPR methods, and the corresponding reconstructed images obtained using the (d) GPRFSED (27.72 dB), (e) MGPRFS (30.59 dB), and (f) GPR/MGPR (30.97 dB) methods.

4.2 Computational Complexity

Next, we evaluate the computational cost (i.e., execution time) of the various methods considered. For this purpose, we provide a representative subset of timing results collected using a 3 year old laptop with a 1.60 GHz Intel Core i5 processor and 2GB of RAM. For three of the representative images in Table 3.1 with both RGB and grayscale versions and five sampling densities per image, we measured the mesh-generation time required for each of the applicable methods considered, and present the results in Tables 4.3(a) and (b).

Examining the results of Tables 4.3(a) and (b), we can see that the execution time required for the CED, ED, and MED methods are normally quite small, and the execution time required for the CGPRFSED, GPRFSED, CGPR, GPR, MGPR, and MGPRFS methods are relatively larger. This behavior is as expected, since the CGPRFSED, GPRFSED, CGPR, GPR, MGPR, and MGPRFS methods all have larger initial mesh size and perform mesh-simplification. Also, we would expect that, at higher sampling densities, except for the GPR, CGPR and MGPR methods, most methods require more execution time, since the initial mesh size is larger. Lastly, for our MGPRFS and MED methods, we would also expect that, as the number of input image components increases from 1 to M (e.g., $M > 1$), the execution time will normally scale approximately by a factor of M (normally less than M) due to the interpolation process operating on M components.

MGPRFS versus CGPR, MGPR, and GPR. The most important comparison to be made here is between our MGPRFS method and each of the CGPR, MGPR, and GPR methods. As all of the CGPR, MGPR, and GPR methods have higher complexity, and can normally yield a mesh quality that is comparable to our MGPRFS method. For this reason, it is particularly meaningful to show how our MGPRFS method outperforms these methods in computational cost. For the results shown in Table 4.3(a) and (b), our MGPRFS method requires from about 5 to 40 times less execution time than the CGPR, MGPR, and GPR methods. The difference is most distinct at the sampling density of 0.5%, where our MGPRFS method requires from about 16 to 40 times less execution time than the CGPR, MGPR, and GPR methods. In other words, although (as seen previously) our MGPRFS method yields meshes with quality comparable, and in most cases better than, the CGPR, MGPR, and GPR methods, this is accomplished with substantially less computational cost.

MGPRFS versus CGPRFSED and GPRFSED. As mentioned earlier, our

Table 4.3: Comparison of computational cost obtained with the various methods for (a) RGB-color and (b) grayscale images

(a)

Image	Samp. Density (%)	Time (s)					
		CED	MED	CGPRFSED $\gamma = 4$	MGPRFS $\gamma = 4$	CGPR	MGPR
lena (color)	0.5	0.16	0.35	0.68	1.72	29.97	41.03
	1.0	0.18	0.46	1.03	2.09	29.62	40.46
	2.0	0.23	0.58	2.31	2.79	29.33	38.66
	3.0	0.30	0.73	3.15	3.49	29.19	37.32
	4.0	0.38	0.80	4.32	5.04	29.00	37.18
bluegirl (color)	0.5	0.14	0.40	0.62	1.66	26.90	36.65
	1.0	0.16	0.45	0.94	2.28	26.78	42.76
	2.0	0.25	0.62	1.79	2.91	26.40	38.46
	3.0	0.26	0.66	2.96	3.89	25.82	35.59
	4.0	0.31	0.72	3.37	5.45	25.29	35.05
cartoon _bull (color)	0.50	0.50	1.38	2.49	4.23	125.86	150.07
	1.0	0.60	1.54	4.20	6.06	114.16	146.56
	2.0	0.80	1.97	6.64	9.12	104.15	140.98
	3.0	1.02	2.40	10.28	11.90	100.82	133.59
	4.0	1.29	2.84	15.64	17.38	97.68	125.38

(b)

Image	Samp. Density (%)	Time (s)				
		ED	MED	GPRFSED $\gamma = 4$	MGPRFS $\gamma = 4$	GPR/MGPR
lena (grayscale)	0.5	0.13	0.22	0.63	0.77	28.19
	1.0	0.15	0.31	1.00	1.19	27.60
	2.0	0.20	0.36	1.77	1.91	27.40
	3.0	0.27	0.44	2.74	2.76	27.00
	4.0	0.33	0.54	3.49	3.53	26.70
bluegirl (grayscale)	0.5	0.10	0.23	0.56	0.71	25.42
	1.0	0.12	0.27	0.92	1.06	25.35
	2.0	0.18	0.32	1.68	1.74	24.79
	3.0	0.24	0.39	2.50	2.48	24.55
	4.0	0.29	0.47	3.22	3.30	24.61
cartoon _bull (grayscale)	0.5	0.43	0.86	2.35	2.52	101.10
	1.0	0.51	0.56	3.70	4.20	97.06
	2.0	0.72	0.74	6.43	6.35	95.89
	3.0	0.92	1.62	9.02	9.12	95.23
	4.0	1.18	1.84	11.87	12.00	93.79

MGPRFS method is an improved and extended version of the GPRFSED method. For this reason, we compare our MGPRFS method to the GPRFSED method and its color-capable version CGPRFSED here. It makes sense that, for RGB-color images, our MGPRFS method requires slightly more execution time than the CGPRFSED method, as the CGPRFSED method performs mesh-simplification on only a single component while our MGPRFS method operates on three components. Also, for grayscale images, our MGPRFS method is expected to take slightly more execution time than the GPRFSED method, as additional round of FSED is performed for the mirroring method. Examining the results of Tables 4.3(a) and (b), we can see that although our MGPRFS method takes about 1.1 to 2.7 times more execution time than the CGPRFSED and GPRFSED methods, the absolute time difference is less than 2 seconds in all test cases. Given the fact that our MGPRFS method yields meshes of vastly superior quality relative to the CGPRFSED and GPRFSED methods (typically, by a margin of up to 7.05 dB and 2.88 dB, respectively), our MGPRFS method is still arguably quite competitive with the these two methods.

MED versus CED and ED. As mentioned earlier, our MED method is an improved and extended version of the ED method. For this reason, we compare our MED method to the ED method and its color-capable version CED here. Examining the results of Tables 4.3(a) and (b), we can see that although our MED method takes about 1.6 to 2.8 times more execution time than the CED and ED methods, the absolute time difference is less than 1 second in most cases. Given the fact that our MED method produces meshes of vastly superior quality relative to the CED and ED methods (typically, by a margin of up to 7.08 dB and 4.72 dB, respectively), our MED method is still arguably quite competitive with the CED and ED methods.

MED versus MGPRFS. Lastly, to show the tradeoff between computational cost and mesh quality for our proposed MED and MGPRFS methods, we make a comparison between them. Examining the results of Tables 4.3, our MED method requires from about 4 to 8 times less execution time than the MGPRFS method in all the test cases. Looking at the results in more detail, we can see that, the execution time required for our MED method is extremely small, typically, within 1 second in most cases. Consequently, our MED method is particularly beneficial for applications which require extremely low computational cost.

4.3 Memory Complexity

Lastly, we compare the memory complexities of the various methods under evaluation. In each the of CGPRFSED, GPRFSED, CGPR, GPR, MGPR, and MGPRFS methods, the memory usage is largely determined by the triangulation data structure and a vertex priority queue data structure (which has one entry per triangulation vertex). Due to the similarity between these methods, we implemented all of the various methods by using similar triangulation and priority queue data structures. For the MED, CED, and ED methods, we still implemented them using the same triangulation and priority queue data structures but leaving the priority queue as unused. In addition, for reason of saving memory, the function value (e.g., RGB or gray value) of each point was not stored directly in the vertex of the triangulation. Instead, we stored the image data in a separate image object and looked up the function value of each vertex by simply using the coordinates of the vertex. Thus, no matter if to process a single- or multi-component image, the memory cost for representing the triangulation and priority queue data structures does not change. For practical cases of M-component images (e.g., grayscale and RGB images) and sampling densities, the memory usage of each of the various methods considered is largely dominated by the triangulation and priority queue data structures. The memory usage of the triangulation and priority queue data structures are approximately proportional to the number of vertices in the mesh. Consequently, the maximum memory usage of each method is approximately proportional to the maximum number of vertices in the mesh. Thus, we use the maximum mesh size as an indicator of maximum memory cost. Given a RGB or grayscale image of width W and height H, and a desired sampling density D, we present the comparison of the maximum mesh size for the various methods under consideration in Table 4.4

Examining the results of Table 4.4, we can see that the memory cost for our MGPRFS method is nearly the same as the CGPRFSED and GPRFSED methods. The most important comparison to be made here is between our MGPRFS method and each of the CGPR, MGPR, and GPR methods. We can see that, at sampling densities from 0.5% to 4.0%, the MGPRFS method requires from $\frac{25}{4} \approx 6.2$ to $\frac{200}{4} = 50$ times less memory than all of the CGPR, MGPR, and GPR methods. The difference is most distinct at lower sampling densities. In particular, at the sampling density of 0.5%, our MGPRFS method requires $\frac{200}{4} = 50$ times less memory than all of the CGPR, MGPR, and GPR methods. Thus, we can see that our MGPRFS method

Table 4.4: Comparison of the maximum mesh size for the various methods

Method	Maximum Mesh Size	Relative Maximum Mesh Size		
		General	D = 0.5%	D = 4%
CED	DWH	1	1	1
ED	DWH	1	1	1
CGPRFSED, $\gamma = 4$	$4DWH$	4	4	4
GPRFSED, $\gamma = 4$	$4DWH$	4	4	4
CGPR	WH	$1/D$	200	25
GPR	WH	$1/D$	200	25
MGPR	WH	$1/D$	200	25
MED	DWH	1	1	1
MGPRFS, $\gamma = 4$	$4DWH$	4	4	4

outperforms the CGPR, MGPR, and GPR methods substantially in terms of memory cost.

Lastly, from Table 4.4, we can see that the memory costs of our MED method and the CED and ED methods are nearly the same. All of these three methods require extremely low memory cost, which is about 4 times less than the MGPRFS, CGPRFSED, and GPRFSED methods, and 200 times less than the MGPR, CGPR, and GPR methods. This also shows the favor of our MED method for the applications which require extremely low memory cost.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

In this thesis, we have studied PDDT-based triangle mesh models for image representation. In particular, we have proposed a general computational framework for generating mesh models for images with an arbitrary number of components, based on the GPRFSED method. As the proposed mesh-generation framework has several free parameters, we studied how various choices of those parameters affect the mesh quality. Based on experimentation and analysis, we recommended two particular sets of parameter choices, leading to two specific mesh-generation methods for single- and multi-component images, known as MED and MGPRFS.

We evaluated our proposed mesh-generation methods based on the two most common cases of single- and multi-component images, namely, grayscale and RGB-color images. Through experimental results, our proposed methods MED and MGPRFS were shown to outperform competing mesh-generation methods of similar or higher complexity, namely the ED, CED, GPRFSED, CGPRFSED, GPR, and CGPR methods. The higher complexity MGPRFS method was shown to achieve meshes with better quality than the GPRFSED, CGPRFSED, GPR and CGPR methods, with similar or lower computational and memory costs. The lower complexity MED method was shown to yield better quality meshes than the ED and CED methods, with similar computational and memory costs. Our two proposed methods allow a different trade-off to be made between mesh quality and computational cost, allowing our approach to be useful in a wider range of applications. As part of our work, we proposed better choices for some of the free parameters of the mesh-generation framework, namely: a good significance function and a new startup policy for FSED, a good initial sample-point selection policy, and a good strategy to choose the initial mesh size. These choices were shown to lead to an increase in mesh quality or reduction in computational and memory costs or both.

5.2 Future Work

Although this thesis has made a significant contribution to mesh-generation schemes for single- and multi-component images, further work in this area could still be done. In what follows, some potential areas for future work are discussed.

In our work, we evaluated the objective mesh quality by the PSNR. Thus, in the mesh-simplification process of our proposed mesh-generation framework, we optimized the mesh by minimizing squared error. Other error metrics, however, also would be worth considering. Some other popular error metrics for comparing images include: structural similarity index measure (SSIM) and mean structural similarity index (MSSIM). By changing the optimization error metric in our mesh-generation framework correspondingly, a mesh model with better subjective quality might possibly be achievable.

Furthermore, although our initial sample-point selection policy based on error diffusion has shown to produce a mesh reasonably well adapted to image content, one might develop better sample-point selection policies for use in our proposed mesh-generation framework to further enhance the mesh quality produced.

Appendix A

Image Datasets

In order to help analyze and evaluate our proposed mesh-generation methods, numerous test images were used. In our work, we employed 45 RGB-color images, taken mostly from the well-known JPEG-2000 [48], USC-SIPI [49], CIPR-Canon [50], and Kodak [51] test sets. The images in each test set are listed in Tables A.1, A.2, A.3, A.4, and A.5. The test images contain both photographic, and computer-generated images. All of these images contain three components (i.e., RGB) and have a precision of 8 bits/sample. For reasons of simplicity and consistency, we formed our grayscale image test set by converting each RGB-color image into its grayscale version using the standard RGB to luminance mapping given by [42].

Table A.1: JPEG 2000 images (photographic images)

Image	Width and Height	Description
bike_rgb	2048×2560	bike and fruits
woman_rgb	2048×2560	woman with gray background

Table A.2: USC SIPI miscellaneous images (photographic images)

Image	Width and Height	Description
4.1.01	256×256	female with flower
4.1.03	256×256	female with gray background
4.1.04	256×256	female with red clothes
4.1.07	256×256	arranged jellybeans
4.1.08	256×256	scattered jellybeans
4.2.01	512×512	splash
4.2.02	512×512	woman with blond hair
4.2.04	512×512	lena
4.2.07	512×512	peppers

Table A.3: RPI CIPR Canon images (photographic images)

Image	Width and Height	Description
annnouncer	512×480	female announcer
beachgirl	480×512	girl on beach
bluegirl	480×512	girl with blue skirt
flower	512×480	flower with a bee
fruits	512×480	oranges and lemons
girl_ldisk	512×480	girl with silver necklace
girl	480×512	girl with black hat
hustler	512×480	hustler playing pool
masuda1	512×480	girl with pink background
masuda2	512×480	girl with green background
model	512×480	female model
pens	512×480	pens
tanaka	512×480	woman
yacht	512×480	yachts

Table A.4: Miscellaneous images (computer-generated images)

Image	Width and Height	Description
3d_cartoon_character	1600×1200	cartoon animal
android_apple_jedi_battle	2560×1600	cartoon jedis battle
carbon_fiber_tux_by_Cope57	1280×1280	cartoon penguin
cartoon_bull	1024×768	cartoon bull

Table A.5: Kodak images (photographic images)

Image	Width and Height	Description
kodim01	768×512	walls
kodim02	768×512	part of a door
kodim03	768×512	hats and blue sky
kodim04	512×768	woman
kodim07	768×512	window and flower
kodim08	768×512	buildings
kodim09	512×768	yachts on ocean
kodim10	512×768	yachts on ocean
kodim12	768×512	couple on beach
kodim15	768×512	kid
kodim17	512×768	statue
kodim18	512×768	female model in front of trees
kodim19	512×768	gazebo
kodim20	768×512	aircraft
kodim23	768×512	birds
kodim24	768×512	house

Appendix B

Software User Manual

B.1 Introduction

As part of the work for this thesis, software that implements the mesh-generation framework and methods proposed herein was developed by the author. The software was written in C++, consists of more than 6000 lines of code, and contains many complex data structures and algorithms.

The software package consists primarily of two executable programs:

1. `make_model`, which reads a RGB or grayscale image in PNM/PPM/PGM format from standard input, creates a mesh model of the image in MODEL format and writes the mesh model to standard output.

2. `rasterize_model`, which reads a mesh model from standard input, creates a raster image reconstruction, and writes the image to standard output.

In sections that follow, the installation and use of the software are described in detail.

B.2 Software Installation

Since our program utilizes many features of C++17, in order to build and run the program, the compiler should support C++17. We recommend the use of GCC with a version 7.2.0 or higher as the C++ compiler. Our software implementation makes heavy use of some C++ libraries, including the Computational Geometry Algorithm Library (CGAL) [52], the Boost Library [53], the Signal Processing Library (SPL) [54] and its extension library SPLEL. These libraries should be installed before building the software. The following versions of these libraries have been verified to work correctly with our software:

- Boost 1.58.0;
- CGAL 3.8.2;
- SPL 2.0.7; and
- SPLEL 2.0.9.

In what follows, `$SOURCE_DIR` denotes the top-level directory of the software distribution (i.e., the directory that contains the `INSTALL` file), `$BUILD_DIR` denotes a directory (which is either empty or to be newly created) to be used for building the software, and `$INSTALL_DIR` denotes the directory under which the software should be installed. To build and install the software, perform the following:

1. Create the installation directory and the build directory. If the directories `$INSTALL_DIR` and `$BUILD_DIR` do not exist, create them by using the command sequence:

   ```
   mkdir -p $INSTALL_DIR
   mkdir -p $BUILD_DIR
   ```

2. Generate build files for the native build tool by using the command:

   ```
   cmake -H$SOURCE_DIR -B$BUILD_DIR -DCMAKE_INSTALL_PREFIX=$INSTALL_DIR
   ```

3. Build the software by using the command:

   ```
   cmake --build $BUILD_DIR --clean-first
   ```

4. Install the software by using the command:

   ```
   cmake --build $BUILD_DIR --target install
   ```

B.3 File Formats

The input of the `make_model` program and the output of the `rasterize_model` program are both images in PGM/PNM/PPM [55] [56] [57] format. A slightly modified version of the Object File Format (OFF) [58], called the MODEL format, is used for representing a mesh model. A mesh in MODEL format is output by the `make_model` program, and used as input by the `rasterize_model` program. Basically, the MODEL format consists of the following contents (in order), with each field separated by whitespace:

1. the width of the image;
2. the height of the image;
3. the number of components in the image;
4. the precision of each component sample (i.e., bits/sample); and
5. the mesh model in OFF format.

B.4 Detailed Program Descriptions

As mentioned before, our software consists of two executable programs, namely `make_model` and `rasterize_model`. In this section, we will give a detailed description of the command line interface for each program:

B.5 The `make_model` Program

Synopsis

```
make_model [OPTIONS] < input_file > output_file
```

Description
This program reads an image in PNM format from standard input, generates and writes a mesh model of the image in MODEL format to standard output. The `make_model` program consists of three main functional blocks: the initial mesh generation block, the mesh simplification block, and a postprocessing block. The initial-mesh-generation block will select the initial mesh points based on the specified initial mesh generator (i.e. `all`, `random`, `ed`). The mesh simplification block will remove sample points from the mesh until the desired mesh size is achieved. The postprocessing block provides the ability to keep removing points if this will reduce the overall approximation error. In order to run the `make_model` program, an image must be

provided from standard input, and the desired sampling density (or size) must be specified by the option `--density` (or `--size`).

Options

`--help`

Print a help message listing all free parameters and the valid arguments for each parameter.

`--density $arg`

Set the desired output mesh density as a percentage of the total number of samples in the original image to `$arg`. Valid values for `$arg` are as follows: $0 <$ `$arg` ≤ 100.

`--size $arg`

Set the desired output mesh size (in number of samples) to `$arg`. If specified, the value is required to be greater than 4 (at least 4 extreme convex hull points).

`--relative-initial-density $arg`

Set the initial density D_0 relative to the target density D to $D_0 = \frac{1}{100 \times D} \times$ `$arg`.

`--initial-density $arg`

Set the initial mesh density (in percent). The default value for `$arg` is `100`.

`--initial-size $arg`

Set the initial mesh size (in number of samples) to `$arg`.

`--initial-generator $arg`

Set the method used for selecting the initial mesh points to `$arg`. The default value of `$arg` is `ed`. Valid values for `$arg` are listed in Table B.1.

`--ed-size-tolerance $arg`

Set the size tolerance of the mesh that generated by the error diffusion method to `$arg`. The default value for `$arg` is `10`.

`--smooth-order $arg`

Set the order of the smoothing filter to `$arg`. The value of `$arg` must be an odd positive integer. The default value of `$arg` is `3`.

`--smooth-operator $arg`

Set the smoothing operator to `$arg`. The default value for `$arg` is `binomial`.

Valid values of `$arg` are:

1. `binomial`: use a binomial filter as the smoothing filter.
2. `mean`: use a mean filter as the smoothing filter.

`--derivative-bound-policy $arg`

Set the boundary handling policy used for computing the partial derivatives to `$arg`. The default value for `$arg` is `zero_ext`.

1. `zero_ext`: zero extension.
2. `const_ext`: const extension.
3. `sym_ext`: symmetric extension.

`--ed-strategy $arg`

Set the error diffusion strategy to `$arg`. The default value set for `$arg` is `max_comp_mmsodd_ed`. Valid values of `$arg` are listed in Table B.2. If the input image is a color image, each of the valid values corresponds to a color image error diffusion strategy in Section 3.3. If the input image is a grayscale image: if `$arg` is set as one of `gray_comp_laplacian_ed`, `max_comp_laplacian_ed`, `mean_comp_laplacian_ed`, the program will use the MoL of the image as the density function for the error diffusion algorithm; otherwise, the program will use the MMSODD.

`--ed-scan-order $arg`

Set the error diffusion scan order to `$arg`. The default value is for `$arg` is `serpentine`. Valid values for `$arg` are:

1. `raster`: raster scan order.
2. `serpentine`: serpentine scan order.

`--ed-leaky-mode $arg`

Set the error diffusion scan order to `$arg`. The default value for `$arg` is `no_leaky`. Valid values for `$arg` are:

1. `leaky`: leaky mode
2. `no_leaky`: no leaky mode

`--ed-initial-error-mode $arg`

Set the error diffusion startup error condition to `$arg`. The default value for `$arg` is `mirror`. Valid values for `$arg` are:

Table B.1: Choices of initial generator

Generator	Description
`all`	select all sample points on the sampling grid as the initial mesh points
`ed`	use the error diffusion method to select the initial mesh points
`random`	randomly select points on the sampling grid as the initial mesh points
`uniform`	uniformly select points on the sampling grid as the initial mesh points

Table B.2: Choices of error diffusion strategy (Detail of each strategy can refer to Section 3.3)

Strategy	Description
`gray_comp_mmsodd_ed`	PSB(D)
`gray_comp_laplacian_ed`	PSB(L)
`vector_space_comb`	PSC
`max_comp_mmsodd_ed`	PSD(Max, D)
`max_comp_laplacian_ed`	PSD(Max, L)
`mean_comp_mmsodd_ed`	PSD(Avg, D)
`mean_comp_laplacian_ed`	PSD(Avg, L)
`three_comps_union_mmsodd`	PSE(D)
`three_comps_union_laplacian`	PSE(L)

1. `zero`: classic method for the startup policy of FSED
2. `mirror`: mirroring method for the startup policy of FSED.

`--bad-point-removal $arg`

Specify if bad point removal is enabled. If `$arg` is 0, then enable bad point removal, otherwise, disable bad point removal. The default value for `$arg` is `1`.

B.6 The `rasterize_model` Program

Synopsis

```
rasterize_model [OPTIONS] < input_file > output_file
```

Description

The `rasterize_model` program reads a mesh model in MODEL format from standard input, reconstructs an image from the mesh, and writes the image to standard output. The reconstructed image is written in PGM/PNM format for grayscale images, and in PPM/PNM format for color images.

B.7 Examples of Software Usage

A few examples are provided in what follows to illustrate the use of our software.

Example A1. Given a color or grayscale image `lena`, one can generate a mesh model with a sampling density of 2% using the MED method, by using command:

```
make_model --density 2 --initial-density 2 --initial-generator ed \
   --ed-strategy max_comp_mmsodd_ed --ed-initial-error-mode mirror \
   --bad-point-removal 0 < lena.pnm > output.model
```

Example A2. Given a color or grayscale image `lena`, one can generate a mesh model with a sampling density of 2% using the MGPRFS method with $\gamma = 4$, by using command:

```
make_model --density 2 --relative-initial-density 400 \
   --initial-generator ed --ed-strategy max_comp_mmsodd_ed \
   --ed-initial-error-mode mirror --bad-point-removal 0 \
    < lena.pnm > output.model
```

Example A3. Given a color or grayscale image `lena`, one can generate a mesh model with a sampling density of 2% using the MGPR method, by using command:

```
make_model --density 2 --initial-density 100 --initial-generator all \
   --bad-point-removal 0 < lena.pnm > output.model
```

Example B. Given a mesh model `input.model`, one can generate the reconstructed image, by using command:

```
rasterize_model < input.model > reconstructed_image.pnm
```

Bibliography

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

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

[3] M. Sarkis and K. Diepold. A fast solution to the approximation of 3-D scattered point data form stereo images using triangular meshes. *Proceedings of IEEE-RAS International Conference on Humaniod Robots, Pittsburg, PA, USA*, pages 235–241, Nov 2007.

[4] J. G. Brankov, Y. Yang, and M. N. Wernick. Tomographic image reconstruction based on a content-adaptive mesh model. *IEEE Transactions on Medical Imaging*, 23(1):202–212, 2004.

[5] Y. Yang, J. G. Brankov, and M. N. Wernick. Content-adaptive mesh modeling for fully-3D tomographic image reconstruction. *Proceedings of International Conference on Image Processing*, 2:621–624, 2002.

[6] J. G. Brankov, Y. Yang, and N. P. Galatsanos. Image restoration using content-adaptive mesh modeling. *Proceedings of IEEE International Conference on image Processing*, 2:997–1000, 2003.

[7] M. A. Garcia and B. X. Vintimilla. Acceleration of filtering and enhancement operations through geometric processing of gray-level images. *Proceedings of IEEE International Conference on image Processing*, 1:97–100, 2000.

[8] D. Su and P. Willis. Image interpolation by pixel-level data-dependent triangulation. *Computer Graphics Forum*, 23(2):189–201, 2004.

[9] M. D. Adams. Progressive lossy-to-lossless coding of arbitrarily-sampled image data using the modified scattered data coding method. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, Taiwan*, pages 1017–1020, 2009.

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

[11] P. Lechat, H. Sanson, and L. Labelle. Image approximation by minimization of a geometric distance applied to a 3D finite elements based model. *Proceedings of IEEE International Conference on image Processing*, 2:724–727, 1997.

[12] 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 Transactions on Circuits and Systems for Video Technology*, 6(6):647–659, Dec 1996.

[13] F. Davoine, M. Antonini, J.-M. Chassery, and M. Barlaud. Fractal image compression based on delaunay triangulation and vector quantization. *IEEE Transactions on Image Processing*, 5:383–346, Feb 1996.

[14] K.L. Hung and C.-C. Chang. New irregular sampling coding method for transmitting images progressively. *IEEE Proceedings - Vision, Image and Signal Processing*, 150(1):44–50, 2003.

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

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

[17] Y. Yang, M. N. Wemick, and J. G. Brankov. A fast approach for accurate content-adaptive mesh generation. *IEEE Transactions on Image Processing*, 12(8):866–881, 2003.

[18] M. D. Adams. A flexible content-adaptive mesh-generation strategy for image representation. *IEEE Transactions on Image Processing*, 20(9):2414–2427, 2011.

[19] M. D. Adams. A highly-effective incremental/decremental delaunay mesh-generation strategy for image representation. *Signal Processing*, 93(4):2414–2427, Apr 2013.

[20] X. Tu and M. D. Adams. Image representation using triangle meshes with explicit discontinuities. *Proceeding of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 97–101, Aug 2011.

[21] L. Demaret and A. Iske. Advances in digital image compression by adaptive thinning. *Annuals of the Marie-Curie Fellowship Association*, 3:105–109, 2004.

[22] L. Demaret and A. Iske. Scattered data coding in digital image compression. *Curve and Surface Fitting: Siant-Malo*, 2003:107–117, 2002.

[23] L. Demaret and A. Iske. Adaptive image approximation by linear splines over locally optimal delaunay triangulations. *IEEE Signal Processing Letters*, 13(5):281–284, 2006.

[24] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolations. *IMA Journal of Numerical Analysis*, 10(1):137–154, 1990.

[25] N. Dyn, D. Levin, and S. Rippa. Boundary correction for piecewise linear interpolation defined over data-dependent triangulations. *Journal of Computational and Applied Mathematics*, 39:179–192, 1992.

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

[27] S. Rippa. Long and thin triangles can be good for linear interpolation. *SIAM Journal on Numberical Analysis*, 29(1):257–270, 1992.

[28] X. Yu, B. S. Morse, and T. W. Sederberg. Image reconstruction using data-dependent triangulation. *IEEE Computer Graphics and Applications*, 21(3):62–68, May 2001.

[29] P. Li and M. D. Adams. A tuned mesh-generation strategy for image representation based on data-dependent triangulation. *IEEE Transactions on Signal Processing*, 22(5):2004–2018, May 2013.

[30] M. A. Garcia and A. D. Sappa. Efficient generation of discontinuity-preserving adaptive triangulations from range images. *IEEE Transactions on Systems, Man, and Cybernetics (Part B): Cybernetics*, 34(5):2003–2014, Oct 2004.

[31] D.-M. Yang and P. Wonka. Gap processing for adaptive maximal poisson-disk sampling. *ACM Transactions on Graphics*, 32:1–15, Oct 2013.

[32] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial greyscale. *Proceedings of the Society for Information Display*, 17(2):75–77, 1976.

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

[34] J. Shewchuk. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures. *In Proceedings of the 11th International Meshing Roundtable*, 73:115–126, 2002.

[35] C. Dyken and M. S. Floater. Preferred directions for resolving the non-uniqueness of delaunay triangulations. *Computational Geometry-Theory and Applications*, 34:96–101, 2006.

[36] M. Aubury and W. Luk. Binomial filters. *Journal of VLSI Signal Processing*, 12(1):35–50, 1996.

[37] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, Oct 1973.

[38] R. A. Haddad and A. N. Akansu. A class of fast guassian binomial filters for speech and image processing. *IEEE Transactions on Signal Processing*, 39(3):723–727, 1991.

[39] R. Gonzalez and R. Woods. *Digital Image Processing.* Addison Wesley, 1992.

[40] S. D. Zenzo. A note on the gradient of a multi-image. *Computer Vision, Graphics, and Image Processing*, 33(1):116–125, 1986.

[41] K. N. Plataniotis and A. N. Venetsanopoulos. Color image processing and applications. 12(2):179–206, 2000.

[42] Recommendation: Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. https://www.itu.int/rec/R-REC-BT.601-7-201103-I/en, Mar 2011.

[43] B. Delaunay. Sur la s sphere vide. *Bulletin of the Academy of Sciences of the USSR, Classe des Sciences Mathematiques et Naturelle*, 7(6):793–800, 1934.

[44] O. Devillers. On deletion in delaunay triangulations. *International Jounal of Computational Geometry and Applications*, 12(3):193–205, 2002.

[45] K. Fleischer and D. Salesin. Accurate polygon scan conversion using half-open intervals. *In Graphics Gems III*, pages 362–365, 1992.

[46] R. Ulichney. *Digital Halftoning.* MIT Press Cambridge, MA, USA, 1987.

[47] D. Knuth. *The art of computer programming sorting and searching*, volume 3. Addison Wesley, 1998.

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

[49] USC-SIPI image database. http://sipi.usc.edu/database, 2016.

[50] CIPR still images, Canon. http://www.cipr.rpi.edu/resource/stills/canon.html, 2002.

[51] Kodak lossless true color image suite. http://r0k.us/graphics/kodak, 2016.

[52] CGAL - Computational Geometry Algorithm Library. http://www.cgal.org, 2018.

[53] Boost library. http://www.boost.org, 2018.

[54] M.D. Adams. Signal processing library. https://github.com/mdadams/SPL, 2018.

[55] PGM - Netpbm grayscale image format. http://netpbm.sourceforge.net/doc/pgm.html, 2016.

[56] PNM - Netpbm superformat. http://netpbm.sourceforge.net/doc/pnm.html, 2013.

[57] PPM - Netpbm color image format. http://netpbm.sourceforge.net/doc/ppm.html, 2016.

[58] OFF - Object file format. http://www.geomview.org/docs/html/OFF.html, 2018.