An Improved Incremental/Decremental Delaunay Mesh-Generation Strategy for Image Representation

by

Badr Eddine El Marzouki
M.Eng., University of Sheffield, 2012

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

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

An Improved Incremental/Decremental Delaunay Mesh-Generation Strategy for Image Representation

by

Badr Eddine El Marzouki
M.Eng., University of Sheffield, 2012

ABSTRACT

Two highly effective content-adaptive methods for generating Delaunay mesh models of images, known as IID1 and IID2, are proposed. The methods repeatedly alternate between mesh simplification and refinement, based on the incremental/decremental mesh-generation framework of Adams, which has several free parameters. The effect of different choices of the framework's free parameters is studied, and the results are used to derive two mesh-generation methods that differ in computational complexity. The higher complexity IID2 method generates mesh models of superior reconstruction quality, while the lower complexity IID1 method trades mesh quality in return for a decrease in computational cost. Some of the contributions of our work include the recommendation of a better choice for the growth-schedule parameter of the framework, as well as the use of Floyd-Steinberg error diffusion for the initial-mesh selection.

As part of our work, we evaluated the performance of the proposed methods using a data set of 50 images varying in type (e.g., photographic, computer generated, and medical), size and bit depth with multiple target mesh densities ranging from 0.125% to 4%. The experimental results show that our proposed methods perform extremely well, yielding high-quality image approximations in terms of peak-signal-to-noise ratio (PSNR) and subjective visual quality, at an equivalent or lower computational cost compared to other well known approaches such as the ID1, ID2, and IDDT methods of Adams, and the greedy point removal (GPR) scheme of Demaret and Iske. More specifically, the IID2 method outperforms the GPR scheme in terms of mesh quality by 0.2–1.0 dB with a 62–93% decrease in computational cost. Furthermore, the IID2 method yields meshes of similar quality to the ID2 method at a computational cost that is lower by

9–41%. The IID1 method provides improvements in mesh quality in 93% of the test cases by margins of 0.04–1.31 dB compared to the IDDT scheme, while having a similar complexity. Moreover, reductions in execution time of 4–59% are achieved compared to the ID1 method in 86% of the test cases.

Contents

List of Tables

List of Figures

List of Acronyms

DT	Delaunay triangulation
PSNR	Peak signal-to-noise ratio
MSE	Mean-squared error
SE	Squared error
SSE	Sum of squared error
SAE	Sum of absolute error
PAE	Peak absolute error
PWAE	Peak weighted absolute error
MMSODD	Maximum-magnitude second-order directional derivative
ED	Error diffusion
PDS	Poisson disk sampling
GPR	Greedy point removal
ALSEM	Approximate local squared-error minimizer
CGAL	Computational Geometry Algorithms Library
SPL	Signal Processing Library
SPLEL	Signal Processing Library Extensions Library

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and deep appreciation to my supervisor Dr. Michael Adams. Thank you for all the assistance and knowledge you have given me throughout my graduate studies. The work of this thesis would not have been possible without your guidance. And a special thank you for your support with the writing of this thesis, I am eternally grateful.

I must also acknowledge the graduate students in our research group, past and present. Our meetings have helped me develop my skills and expand my knowledge.

Finally, I am thankful to the faculty and staff who have assisted me during my time at the University of Victoria.

DEDICATION

To my family, for their constant support and encouragement, and all their sacrifices.

Chapter 1

Introduction

1.1 Mesh Representation of Images

Despite being the most common form of image representation, lattice-based sampling is far from optimal. Typically, images are nonstationary, which inevitably leads to oversampling in some regions of the sampled domain and undersampling in others, due to the uniform sampling used. Nonuniform sampling addresses this issue by adapting the sampling density to the content of the image. A more intelligent placement of the sample points leads to greater efficiency, thereby significantly reducing the number of samples needed while keeping the visual quality at acceptable levels. Nonuniform sampling has proven useful in many applications such as: feature detection [1], pattern recognition [2], image/video coding [3–9], tomographic reconstruction [10, 11], restoration [12] and filtering [13].

Several approaches to nonuniform sampling have been proposed over the years, such as radial basis function methods [14, 15], Voronoi and natural neighbor methods [14], inverse distance-weighted methods [14, 15] and finite-element methods [14, 15]. One particular approach to image representation based on content-adaptive sampling that has garnered interest from researchers in the past few years is based on meshes [16–36]. Large regions of the image can be represented using a few polygons instead of a much larger number of pixels. This takes advantage of the inherent geometric structure in images and the correlation between adjacent image pixels. Triangle meshes are particularly well suited for this application, due to their simplicity and their ability to model a wide range of image content as well as some features inherent in images such as sharp edges. In a triangle-mesh model, the image domain is partitioned into triangle faces whose vertices are the sample points, and an interpolant is constructed over each face. The sample points, their connectivity, and the function value at the sample points uniquely characterize the model.

The so called mesh-generation problem is concerned with creating a mesh approximation of an image, more specifically, a mesh of a fixed size that minimizes some approximation error criteria. As it turns out, finding computationally-efficient solutions to this type of problem is quite challenging [37].

1.2 Historical Perspective

Over the years, a great number of methods for generating mesh models of bivariate functions have been developed [16–36]. Typically, they can be classified based on how they select the sample points. One category of methods determine all the sample points in one shot based on the local image content, then construct a triangulation of the generated set of sample points. The highly effective scheme proposed by Yang et al. [36], for instance, uses the classical Floyd-Steinberg error diffusion algorithm [38] to select the sample points such that their density is proportional to the maximum magnitude second order directional derivative of the image. A Delaunay triangulation of the selected sample points is then constructed. Other examples of similar techniques can be found in [10–12, 39, 40]. Another group of mesh-generation methods update the set of selected sample points over multiple iterations on the mesh, in an attempt to gradually improve the mesh model's approximation quality [17, 18, 23, 41]. They achieve this by either adding or removing points in each iteration, which is why they tend to be more computationally expensive but, generally, produce higher quality image approximations than methods of the first category. In mesh-simplification schemes, sample points are gradually removed from the mesh until the desired number of samples or approximation error has been reached [17, 42]. This can be with one or more sample points removed at a time, starting from a mesh containing many or all of the sample points in the image domain. The adaptive thinning method of Demaret and Iske [17] is able to generate high-quality mesh models using a mesh-simplification approach based on minimizing the increase in approximation error of piecewise-linear interpolants over the Delaunay triangulation. Unfortunately, as with many mesh-simplification methods, it suffers from a high computational cost. Moreover, the memory cost associated with a mesh containing most or all of the sample points quickly becomes prohibitive as the image size becomes large. Conversely, mesh refinement methods start with a low-density mesh consisting of the four extreme convex-hull points of the image domain, and possibly a small subset of the sample points as well. The mesh is then refined by adding one or more points until the termination criteria, such as mesh size or mesh approximation error, is reached [18, 23, 34, 35]. Mesh-generation methods that use refinement, such as the one proposed by Rippa [23], generally have a lower computational and memory cost than simplification methods, but at the expense of mesh quality.

In addition to the sample points selected, mesh models are characterized by the mesh connectivity (i.e., how the vertices are connected by edges). Consequently, mesh-generation methods may also be classified by the type of triangulation connectivity they use. The use of a Delaunay triangulation is quite popular, as Delaunay triangulations have some useful properties for approximation [43]. Delaunay triangulations minimize the maximum interior angle of all the triangles in the triangulation, thereby avoiding poorly chosen sliver (i.e. long and thin) triangles, which can lead to high approximation error. Furthermore, when used in conjunction with a technique such as preferred directions [44], the connectivity of a Delaunay triangulation can be uniquely determined from the position of the points being triangulated. Another class of triangulations used in mesh-generation applications is data-dependent triangulations (DDTs) [18, 23–26, 29, 30, 32, 33, 35]. In a DDT, the mesh connectivity is chosen arbitrarily based on information from the data being triangulated. DDTs are therefore more flexible than Delaunay triangulations, and can outperform them if the sample points and the connectivity are chosen well. Nevertheless, the flexibility afforded by DDTs means that, in mesh-generation, achieving acceptable results comes at a very high computational cost. Consequently, Delaunay triangulations were chosen over DDTs in our work.

In [20], Adams proposed a framework for generating mesh models of images based on the Delaunay triangulation. The framework is iterative and has many degrees of freedom to accommodate a wide variety of mesh-generation methods. The IDDT [20], and later, ID1 and ID2 [21] methods were derived from this framework by fixing the degrees of freedom to specific choices, which were determined based on experimentation. A combination of mesh refinement and mesh simplification is used in these methods. This allows these methods to benefit from mesh quality comparable to methods relying solely on mesh simplification, such as adaptive thinning, but at a much lower computational cost. Despite the improved speed over using pure mesh simplification, the IDDT, ID1, and ID2 methods are not as fast as methods that generate the sample points in one shot such as error diffusion. Nevertheless, one-shot methods perform substantially worse in terms of mesh quality. The work of this thesis is based on the framework by Adams [20, 21], and uses a fast approach for selecting the initial mesh samples in one shot, specifically error diffusion, along with a combination of mesh simplification and mesh refinement such that the advantages of all these different classes of methods in terms of mesh quality and computation cost are achieved.

1.3 Overview and Contribution of the Thesis

In this thesis, we explore the generation of triangle-mesh models of images. Given a grayscale image, our goal is to generate a mesh model of the image that minimizes the mean-squared error. Our work focuses on the generation of triangle-mesh models using the computational framework proposed by Adams [20]. More specifically, we are concerned with mesh models that are based on preferred-directions Delaunay triangulations. The choice is motivated by the fact that, as stated previously, Delaunay triangulations have some attractive properties for approximation purposes. Furthermore, by using Delaunay triangulations, we can focus on optimizing the selection of the sample points without the additional burden of selecting their connectivity. The two mesh-generation methods proposed, IID1 and IID2, are derived from the incremental/decremental computational framework based on experimental results. The methods make different trade offs between the quality of the image approximation, and computational complexity. Nevertheless, both methods are highly effective in terms of objective and subjective approximation quality compared to state-of-the-art methods of similar computational complexity.

The remainder of this thesis is organized into four chapters and an appendix. In what follows, we provide and overview of these chapters and the appendix.

In Chapter 2, we introduce background material to aid with understanding the work presented in this thesis. The notation and terminology used are presented first. Subsequently, we describe concepts from computational geometry (e.g., Delaunay triangulations) and image processing (e.g., image filtering). Triangle-mesh models of images and the mesh-generation problem addressed in our work are also formally defined. A grid-point to face mapping scheme, and some of the techniques used for selecting the sample points of the initial mesh are then described. Further, we define the computational framework of [20] which forms the foundation for our work. Finally, we introduce some highly effective schemes that were previously proposed for generating triangle-mesh models of images. The first of the methods is the greedy point removal (GPR) scheme of Demaret and Iske [17], which is often considered a good benchmark for mesh quality.

The remaining methods are the IDDT [20], ID1, and ID2 [21] methods of Adams, all of which are based on the computational framework of [20].

Chapter 3 presents our proposed mesh-generation methods and the process by which these methods were developed. First, we describe in more detail the free parameters of the incremental/decremental computational framework, as well as the choices considered for experimentation for each of them. The effect of the different choices on various performance metrics is then determined. Finally, the mesh-generation methods, IID1 and IID2, are proposed based on the results of the experimental analysis. The IID2 method is a high complexity variant that achieves higher mesh quality at a higher computational cost, while the IID1 method is the low complexity faster variant that yields slightly lower quality meshes.

In Chapter 4, the performance of the IID1 and IID2 methods in terms of approximation quality and computational cost is evaluated. Each of the two methods is compared to other mesh generation schemes of similar complexity. It is shown that the IID2 method consistently outperforms the GPR method on all counts, and produces meshes of comparable quality to the ID2 method at a lower computational cost. The IID1 method is comparable to the IDDT method in terms of execution time but typically yields meshes of higher quality. For example, the IID2 method performs better than the GPR method by an average of 0.44 dB in terms of PSNR while managing to be up to 11 times faster, and use substantially less memory. Furthermore, the quality of meshes generated using the IID1 method are better in PSNR by an average of 0.89 dB than those produced by the IDDT method, at an equivalent or lesser computational cost.

Chapter 5 concludes by summarizing the key results of our work. Some suggestions for future work are also given.

In Appendix A, the software developed during the course of our research is documented. The documentation includes a description of the programs and their options as well as the file formats used. Some examples of how to use the software are also provided.

Chapter 2

Preliminaries

2.1 Overview

In this chapter, background information essential for understanding the work presented in this thesis is introduced, beginning with the notation and terminology used throughout this document. Some basic image processing and computational geometry concepts are then presented. Triangle-mesh models for image representation as well as some related topics, such as scan conversion and mesh evaluation are discussed. The chapter concludes with a description of error diffusion and the incremental/decremental mesh-generation framework, which are central to the work of this thesis.

2.2 Notation and Terminology

Before proceeding further, a brief digression is in order concerning the notation and terminology used herein. The sets of integers and real numbers are denoted $\mathbb{Z}$ and $\mathbb{R}$, respectively. For $a, b \in \mathbb{R}$, $(a, b), [a, b), (a, b]$, and $[a, b]$ denote, respectively, the open interval $\{x \in \mathbb{R} : a < x < b\}$, the half-closed half-open interval $\{x \in \mathbb{R} : a \leqslant x < b\}$, the half-open half-closed interval $\{x \in \mathbb{R} : a < x \leqslant b\}$, and the closed interval $\{x \in \mathbb{R} : a \leqslant x \leqslant b\}$. For $a \in \mathbb{R}$, $\lceil a \rceil$ denotes the ceiling of a (i.e. the smallest integer no less than a, and $\lfloor a \rfloor$ denotes the floor of a (i.e. the largest integer no more than a). The cardinality of a set S is denoted $|S|$. The 2-norm of a vector $v = (v_1, v_2, ..., v_n)$ in $\mathbb{R}^n$, denoted $\|v\|$, is defined as

$$\|v\| = \sqrt{v_1^2 + v_2^2 + ... + v_n^2}.$$

The assignment operator, which assigns the value of the right-hand argument to the left-hand argument, is denoted ":=". Hence, $a := b$ denotes the assignment of the value of b to a.

2.3 Image Processing

Binomial filters are low-pass filters that approximate Gaussian filtering [45]. Their simplicity and efficiency makes them particularly useful for smoothing in image processing applications. The transfer function $H_n(z)$ of a one-dimensional n-th order binomial filter with unity DC gain and zero phase 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 two-dimensional binomial filter can be constructed from the tensor product of two one-dimensional binomial filters. For example, the non-zero coefficients of a two-dimensional third-order 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 can be equivalently applied in a separable fashion using the one-dimensional third-order binomial filter with non-zero coefficients $\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$.

The bilateral filter is a nonlinear edge-preserving smoothing filter [46, 47]. It splits the image into large scale features and small scale features (e.g., texture), which allows for selective filtering of the latter, unlike a linear filter which smoothes image features indiscriminately. With $I_\mathbf{p}$ and $I_\mathbf{q}$ denoting the intensities of the pixels $\mathbf{p}$ and $\mathbf{q}$, respectively, the filter is defined as

$$BF[I]_\mathbf{p} = \frac{1}{W_\mathbf{p}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(\|I_\mathbf{p} - I_\mathbf{q}\|) I_\mathbf{q},$$

where $W_{\mathbf{p}}$ is a normalization factor given by

$$W_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)G_{\sigma_r}(\|I_{\mathbf{p}}-I_{\mathbf{q}}\|),$$

G_{σ_s} is the spatial Gaussian, and G_{σ_r} is the range Gaussian. The filter equation is effectively a normalized weighted average of pixel intensities. The spatial parameter σ_s and the range parameter σ_r constitute the window sizes of the G_{σ_s} and G_{σ_r} Gaussians, respectively, and characterize the bilateral filter. The spatial Gaussian G_{σ_s} reduces the influence of distant pixels. As the spatial parameter σ_s is increased, larger features of the image are smoothed. The range Gaussian G_{σ_r} reduces the influence of distant pixels $\mathbf{q}$ that have an intensity $I_{\mathbf{q}}$ different from $I_{\mathbf{p}}$. Increasing the range parameter σ_r has the effect of bringing the bilateral filter closer to a true Gaussian filter. The impact of varying the σ_s and σ_r parameters on the filtering result is illustrated in Figure 2.1.

The maximum-magnitude second-order directional derivative (MMSODD) [36] d for a function f defined on $\mathbb{R}$ is defined as

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

where

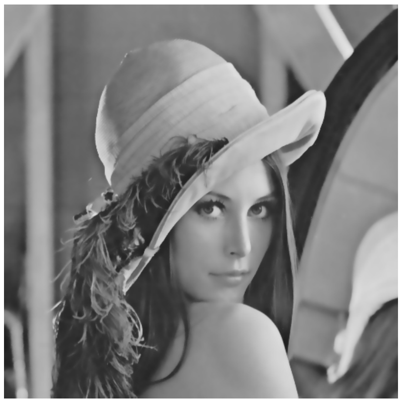
$$\alpha(x,y) = \frac{1}{2}\left[\frac{\partial^2}{\partial x^2}f(x,y) + \frac{\partial^2}{\partial y^2}f(x,y)\right]$$

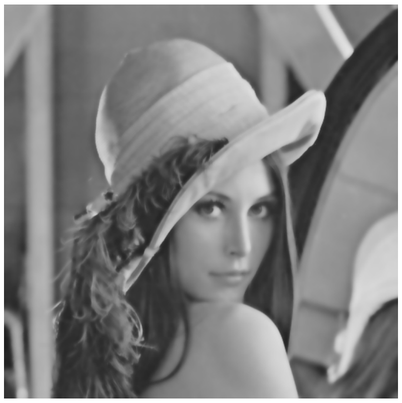
and

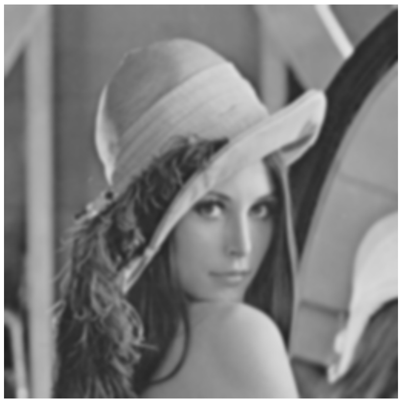
$$\beta(x,y) = \sqrt{\frac{1}{4}\left[\frac{\partial^2}{\partial x^2}f(x,y) - \frac{\partial^2}{\partial y^2}f(x,y)\right]^2 + \left[\frac{\partial^2}{\partial x\partial y}f(x,y)\right]^2}.$$

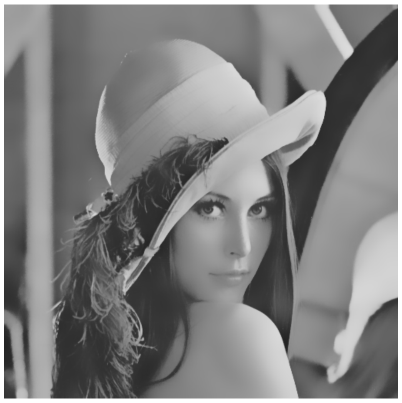
The partial-derivative operators in the preceding equation are formed from the tensor product of one-dimensional derivative operators, where the discrete-time approximations of the one-dimensional first- and second-order derivative operators are computed using the filters with transfer functions $\frac{1}{2}z - \frac{1}{2}z^{-1}$ and $z-2+z^{-1}$, respectively. The MMSODD has the special property that an edge produces a double response which makes it particularly useful in our work. An example of an image and its MMSODD are shown in Figures 2.2(a) and 2.2(b), respectively.

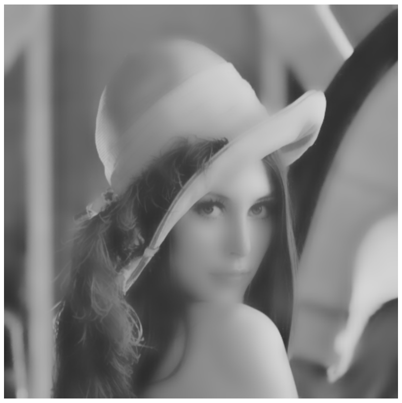
The magnitude of the gradient of an image, I_G, can be approximated using the Sobel

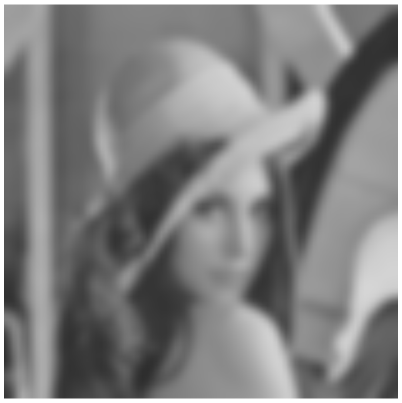

(a)

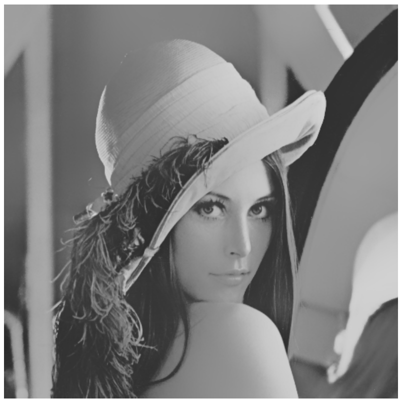

(b)

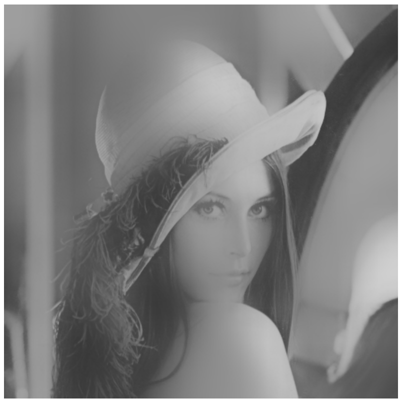

(c)


(d)


(e)

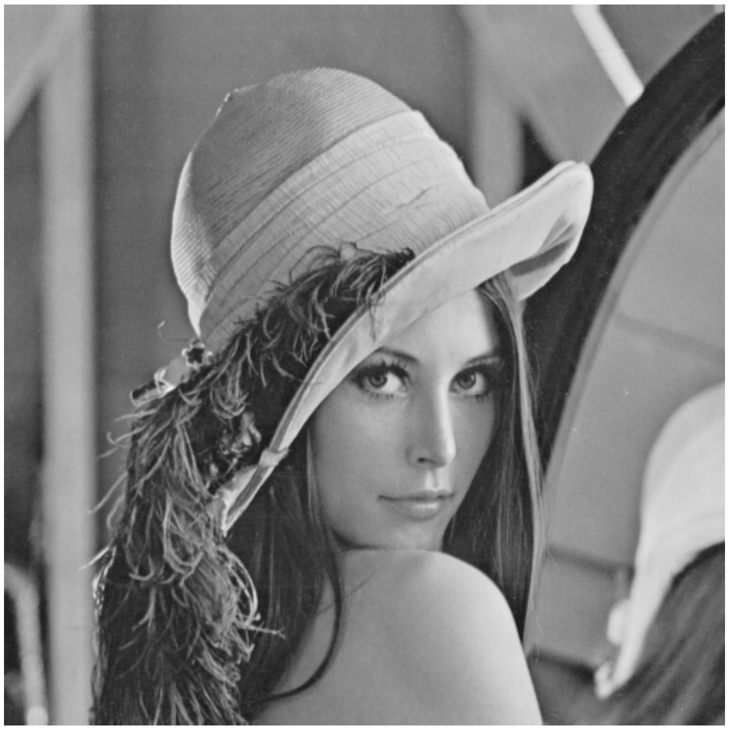

(f)


(g)

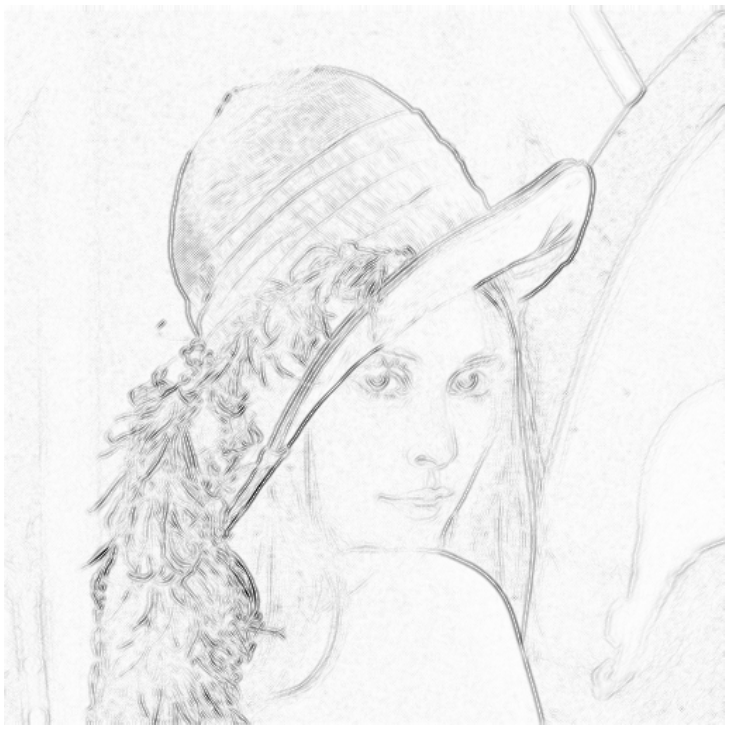

(h)

(i)

Figure 2.1: Effect of the space (σ_s) and range (σ_r) parameters on the output of the bilateral filter. The filter approximates a Gaussian for $\sigma_r \approx \infty$.
(a) $\sigma_s = 2$ and $\sigma_r = 0.1$, (b) $\sigma_s = 2$ and $\sigma_r = 0.25$, (c) $\sigma_s = 2$ and $\sigma_r \approx \infty$,
(d) $\sigma_s = 6$ and $\sigma_r = 0.1$, (e) $\sigma_s = 6$ and $\sigma_r = 0.25$, (f) $\sigma_s = 6$ and $\sigma_r \approx \infty$,
(g) $\sigma_s = 18$ and $\sigma_r = 0.1$, (h) $\sigma_s = 18$ and $\sigma_r = 0.25$, and (i) $\sigma_s = 18$ and $\sigma_r \approx \infty$



(a)



(b)

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

operator [48]. It is computed as

$$|I_G| = \sqrt{I_{Gx}^2 + I_{Gy}^2},$$

where I_{Gx} and I_{Gy} are obtained by filtering using the convolution kernels Gx and Gy, respectively, which are given by

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

An approximation of the Laplacian of an image [48] can be calculated in a separable fashion using the 1-D filter

$$L = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}.$$

The gradient magnitude and the Laplacian of an image computed using the filters above are illustrated in Figure 2.3. The gradient of the image in Figure 2.3(a) is shown in Figure 2.3(b). Similarly, the Laplacian of the image in Figure 2.3(a) is shown in Figure 2.3(c).

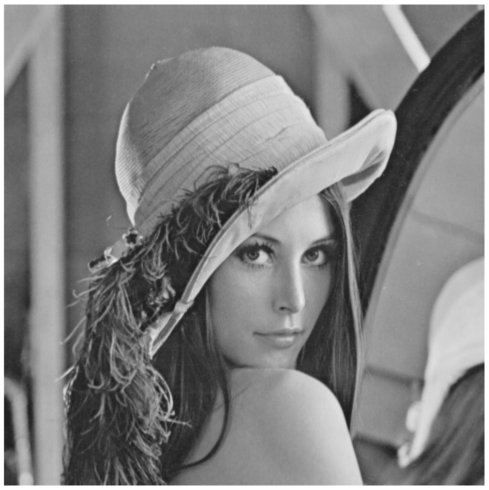
2.4 Computational Geometry

The mesh-generation methods proposed later in this thesis are based on Delaunay triangulations. An introduction to some computational geometry concepts is therefore in order. Such concepts include triangulations and Delaunay triangulations.

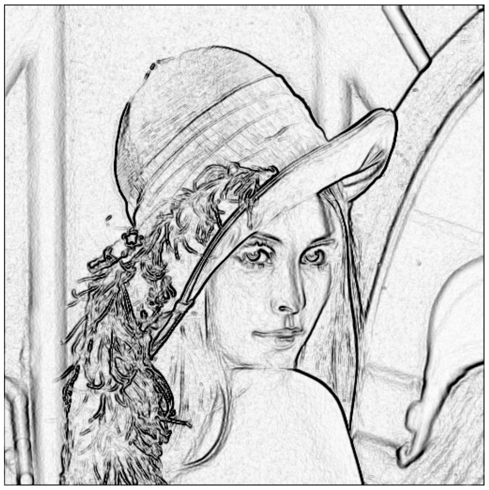
Before we can present the definition of a triangulation, we must first introduce the notion of a convex set and a convex hull.

Definition 2.1 (Convex set)**.** *A set P of points in $\mathbb{R}^2$ is said to be convex if, for every pair of points $x, y \in P$, the line segment $\overline{xy}$ is completely contained in P.*

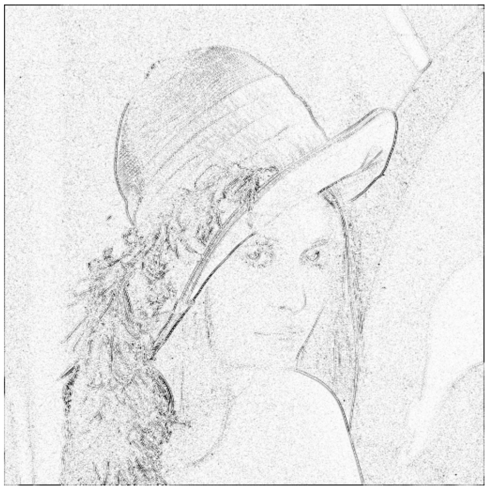
Two different sets are depicted in Figure 2.4 to better illustrate the notion of a convex set. In Figure 2.4(a), the line segment $\overline{xy}$ connecting the points x and y is also in the set. Similarly, we can see that any pair of two points from the set can be connected using a line segment that is also completely contained in the set. On the other hand, the set



(a)



(b)



(c)

Figure 2.3: Example of gradient magnitude and Laplacian of an image. (a) An image and its (b) gradient magnitude and (c) Laplacian.

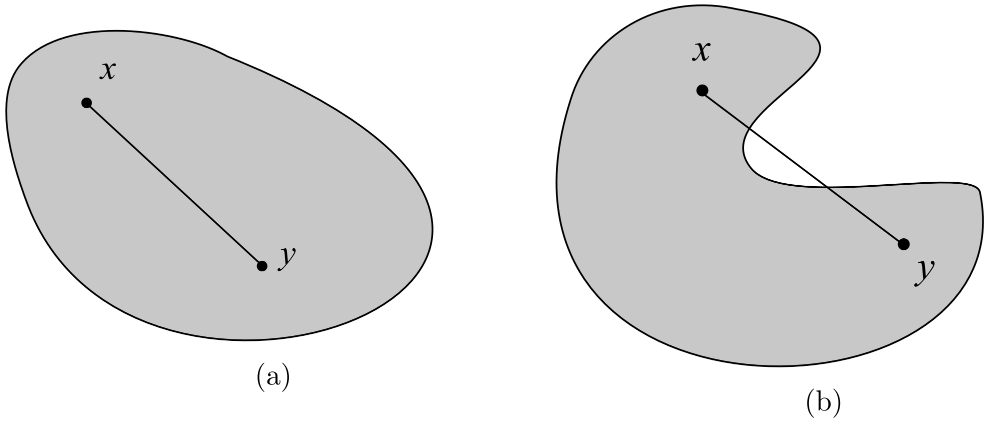


Figure 2.4: Examples of a (a) convex set, and (b) nonconvex set.

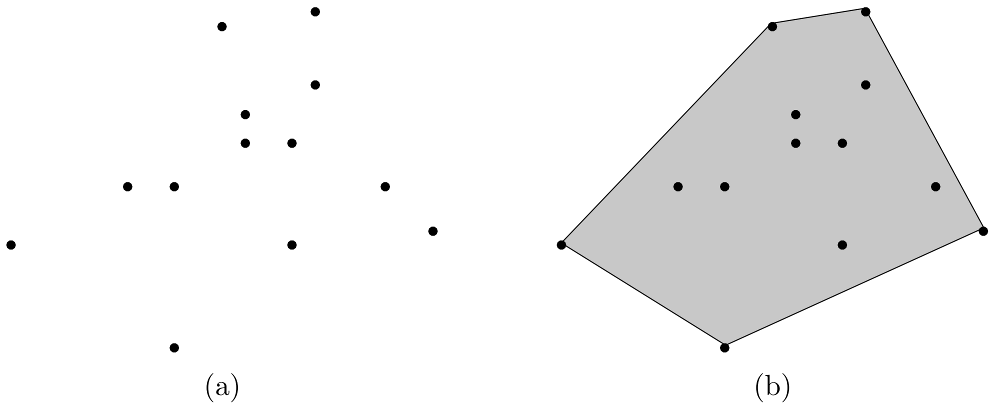


Figure 2.5: Convex-hull example. (a) A set of points and (b) its convex hull.

shown in Figure 2.4(b) is nonconvex because, for the two points x and y which are in the set, the line segment $\overline{xy}$ shown is not completely contained in the set.

Definition 2.2 (Convex hull)**.** *The convex hull of a set P of points is the intersection of all convex sets that contain P (i.e., the smallest convex set containing P).*

Figure 2.5 illustrates the preceding definition. For a set of points P in $\mathbb{R}^2$ as shown in Figure 2.5(a), the convex hull is the set represented by the shaded area in Figure 2.5(b). If the locations of the points in P are represented by pins sticking out of a plane, the convex hull of P may be viewed as the area enclosed by an elastic band that was stretched around the pins then released such that it contains all the pins in its interior. Having introduced the concept of convex hull, we can now define the concept of triangulation that is fundamental in this thesis.

Definition 2.3 (Triangulation)**.** *A triangulation T of the finite set P of points in $\mathbb{R}^2$ is*

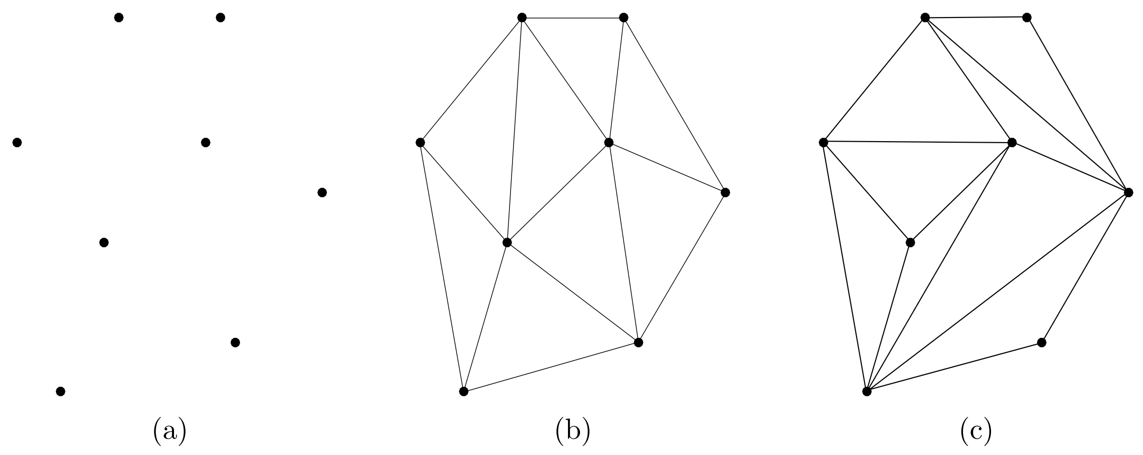


Figure 2.6: Example of triangulations of a set of points. (a) A set P of points, (b) a triangulation of P, and (c) a different triangulation of P.

a set T of non-degenerate triangles that satisfies the following conditions:

1. *the set of all vertices of triangles in T is P;*

2. *the interiors of any two triangles in T do not intersect;*

3. *the union of all triangles in T is the convex hull of P; and*

4. *every edge of a triangle in T only contains two points from P.*

In other words, a triangulation of a set P of points partitions the convex hull of P into a set of triangles such that no two distinct triangles overlap, and all the vertices are in P. As a matter of notation, for a triangulation T, the set of vertices, edges, and faces of T are denoted as $\mathcal{V}(T)$, $\mathcal{E}(T)$, and $\mathcal{F}(T)$ respectively.

A given set P of points typically has many possible triangulations. This is illustrated by the example in Figure 2.6. For the set P of points in Figure 2.6(a), one possible triangulation of P is shown in Figure 2.6(b). In Figure 2.6(c), we can see that the same set P of points has a different triangulation from that shown in Figure 2.6(b).

A number of triangulation types have been proposed over the years. One widely used type that is of particular interest to our work is the Delaunay triangulation [43]. Before we can define the Delaunay triangulation, however, we must first introduce the concept of circumcircle.

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

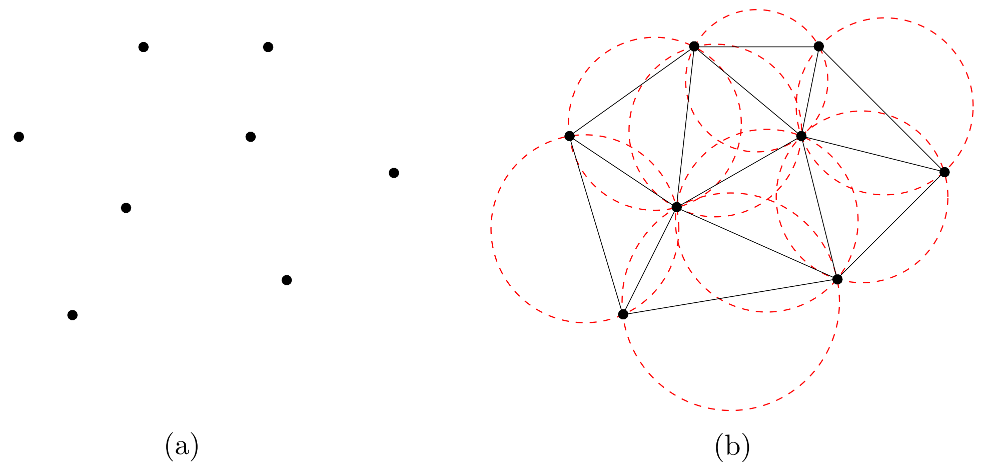


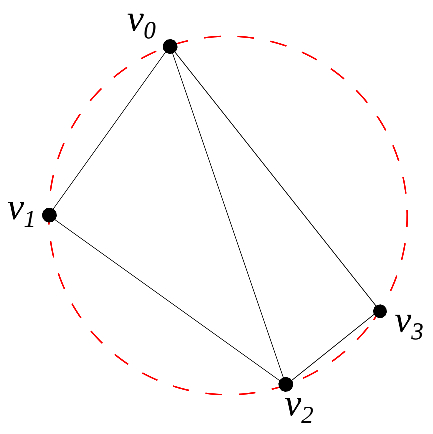
Figure 2.7: Example of a Delaunay triangulation. (a) A set of points, and (b) its Delaunay triangulation.

With the definition of circumcircle in place, we can define a Delaunay triangulation as given below.

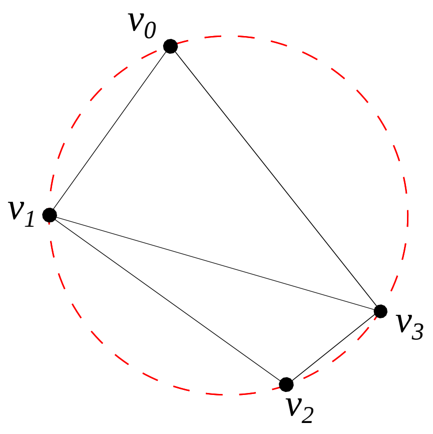
Definition 2.5 (Delaunay triangulation)**.** *A triangulation T of a set P of points in $\mathbb{R}^2$ is said to be Delaunay if each triangle in T is such that the interior of its circumcircle contains no vertices of T.*

An example of a Delaunay triangulation is illustrated in Figure 2.7. A set P of points is shown in Figure 2.7(a) along with its Delaunay triangulation in Figure 2.7(b). The circumcircle of each triangle is displayed using dashed lines. Clearly, no vertices are strictly in the interior of the circumcircle of any triangle in T.

Delaunay triangulations have numerous properties that make them particularly useful. They maximize the minimum interior angle of all of the triangles in the triangulation. Consequently, the incidence of sliver (i.e., long and thin) triangles is greatly reduced, which is often desirable. Due to the fact that multiple points may be cocircular, the Delaunay triangulation of a set of points is not guaranteed to be unique. In fact, when points are constrained to being on an integer lattice such as is the case in this work, the Delaunay triangulation is almost guaranteed not to be unique. The nonuniqueness of Delaunay triangulations is illustrated by the example in Figure 2.8 where, for a set of cocircular points, the two different triangulations in Figures 2.8(a) and 2.8(b) equally meet the Delaunay conditions. Several methods have been proposed for resolving the nonuniqueness of Delaunay triangulation by deterministically selecting one of the many possible choices. Such techniques include the symbolic perturbation [49–51] and preferred



(a)



(b)

Figure 2.8: Examples of Delaunay triangulations with cocircular points. (a) A Delaunay triangulation of a set of points, and (b) a different Delaunay triangulation of the same set of points.

directions [44] methods. When a Delaunay triangulation is used in conjunction with a method such as preferred directions, the triangulation connectivity is determined solely by the set P of points being triangulated. This property is desirable in many applications because it leads to more compact representations: the position and value of the sample points are sufficient for reconstructing a triangulation.

An advantageous property of Delaunay triangulations is that point deletion is local [52]. In other words, when a vertex is removed from the triangulation, it only impacts faces in the immediate vicinity of the removed vertex. Figure 2.9 illustrates this property of Delaunay triangulations. The vertex p of the Delaunay triangulation in Figure 2.9(a) is to be removed from the mesh. The shaded faces in the figure are in the immediate vicinity of the vertex p. Figure 2.9(b) shows the updated triangulation after p is removed from the triangulation. We can see that the deletion of p from the triangulation only affected the shaded region. The deletion of points from a Delaunay triangulation and the ensuing connectivity update are therefore efficient. Mesh simplification schemes that are based on Delaunay triangulations can potentially exploit this property.

2.5 Mesh Models of Images

In the context of our work, an image of width W and height H is an integer-valued function ϕ defined on $D = [0, W-1] \times [0, H-1]$ and sampled on the two-dimensional

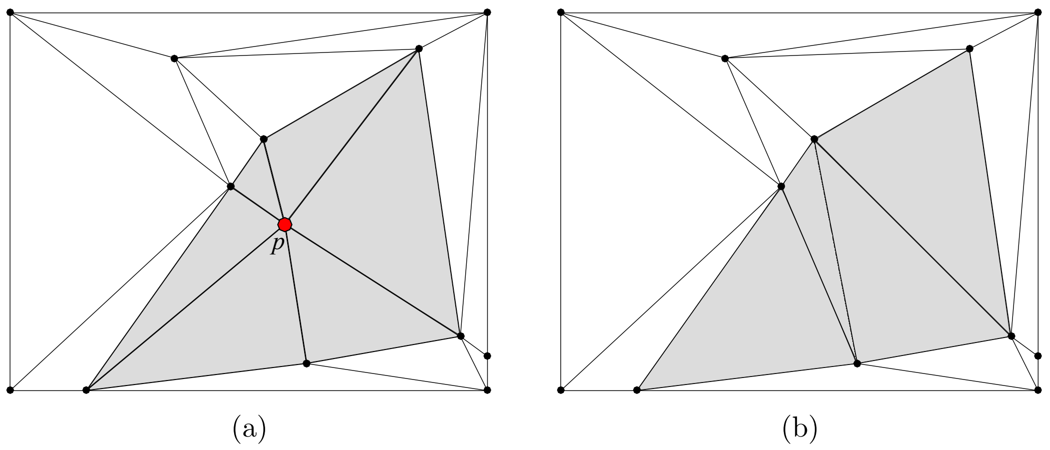


Figure 2.9: Example of the local effect of vertex deletion in Delaunay triangulations. (a) A Delaunay triangulation of a set of points, and the vertex p to be removed. (b) The Delaunay triangulation after removing the vertex.

integer lattice $\Lambda = \{0, 1, ..., W-1\} \times \{0, 1, ..., H-1\}$, where the function value corresponds to the brightness. The set of sample points for ϕ is denoted $P_\phi = \{p_i\}_{i=0}^{|\Lambda|-1}$, while the set of corresponding sample values is $Z_\phi = \{z_i\}_{i=0}^{|\Lambda|-1}$ where $z_i = \phi(p_i)$. Triangle mesh-based image representation is an approach that aims to model the image function ϕ by partitioning the entire image domain into a set of triangular regions. A mesh model of an image ϕ is completely characterized by:

1. a set $P = \{p_i\}_{i=0}^{|P|-1}$ of sample points;
2. a triangulation T of P; and
3. a set $Z = \{z_i\}_{i=0}^{|P|-1}$ of corresponding sample-point function values.

The set P is chosen to always include the extreme convex-hull points of the image domain. This ensures that all of Λ is covered by the triangulation. The quantity $|P|$ is known as the size of the mesh model, while the sampling density of the mesh model is defined as $|P|/|\Lambda|$.

The above mesh model is associated with a continuous piecewise-linear function $\tilde{\phi}_P$ that approximates ϕ and is defined on the entire domain Λ. The function $\tilde{\phi}_P$ is constructed from P and Z by combining the linear interpolants over each of the faces in the triangulation T. The integer-valued image approximation function $\hat{\phi}_P$ is obtained from $\tilde{\phi}_P$ using rounding as $\hat{\phi}_P = \text{round}\left(\tilde{\phi}_P\right)$. Standard rasterization techniques [53] can be used to generate $\hat{\phi}_P$ from the mesh model.

Triangle-mesh modelling of images is illustrated in Figure 2.10. An image such as that of Figure 2.10(a) may be viewed as a surface where the brightness of the sample points corresponds to their height above the plane as shown in Figure 2.10(b). In Figure 2.10(c), a mesh model of the image is created by selecting a subset of the original sample points and triangulating them, which creates a partition of the image domain. The resulting triangle-mesh model of the image is shown in Figure 2.10(d). Figure 2.10(e) shows the reconstruction of the image obtained by rasterizing the mesh model.

The metric used to measure error between the original image ϕ and its approximation $\hat{\phi}$ that we aim to minimize is the mean squared error (MSE) ε defined as

$$\varepsilon = |\Lambda|^{-1} \sum_{p \in \Lambda} (\hat{\phi}(p) - \phi(p))^2.$$

The MSE is typically expressed in terms of the PSNR for convenience, which is given by

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

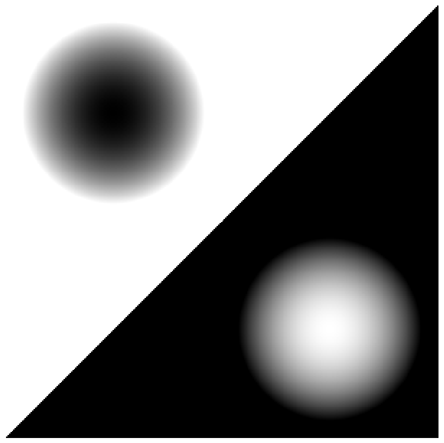
where ρ is the sample precision in bits. The PSNR is a logarithmic representation of the MSE relative to the dynamic range of the signal, with higher PSNR values corresponding to better quality.

2.6 Grid-Point to Face Mapping

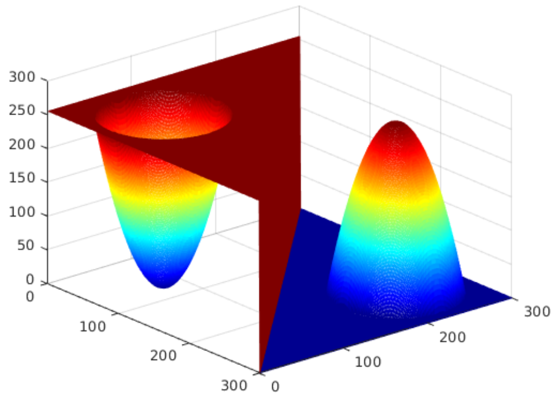
Let $\Gamma(T)$ denote the set of all integer grid points falling inside or on the boundary of a triangulation T. For various reasons that will become clear later, a mapping between points of the grid and faces of the triangulation is needed. The scheme from [53] was modified slightly for this purpose in our work.

Given an image ϕ and a triangulation T of the image domain, the scheme uniquely maps each point $p \in \Gamma(T)$ to a face $f \in \mathcal{F}(T)$ as follows:

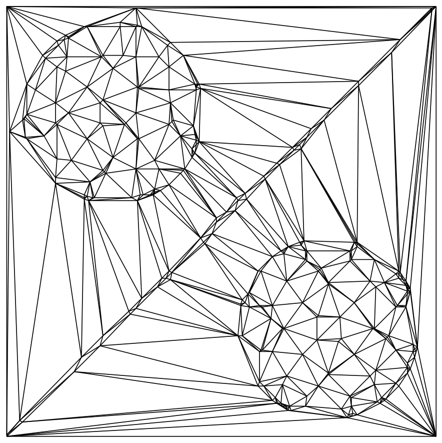
1. If p is strictly inside a face f, map p to the face f.
2. If p is on an edge e, excluding the endpoints of e:
   (a) If e is horizontal, map p to the face below e unless no such face exists, in which case p is mapped to the face above e.



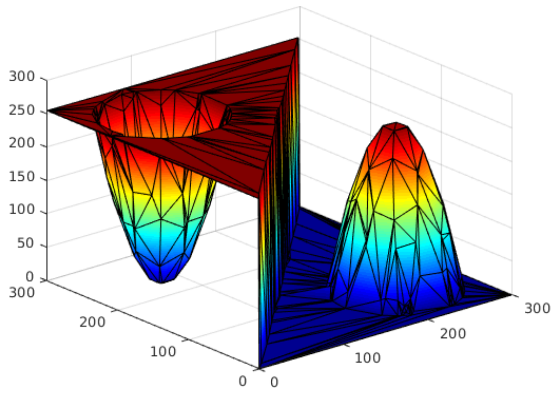
(a)



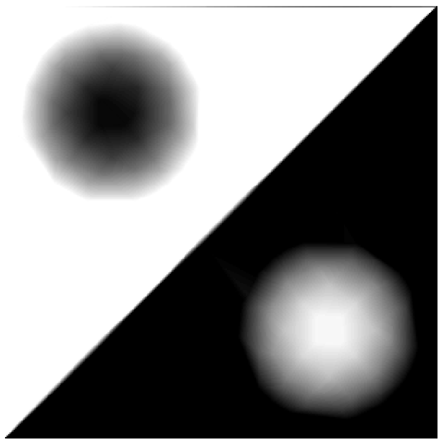
(b)



(c)



(d)



(e)

Figure 2.10: Mesh model of an image. (a) The original (raster) image. (b) The original image viewed as a surface. (c) A triangulation of the image domain. (d) The resulting triangle mesh. (e) The reconstructed image obtained by rasterizing the mesh model.

(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.

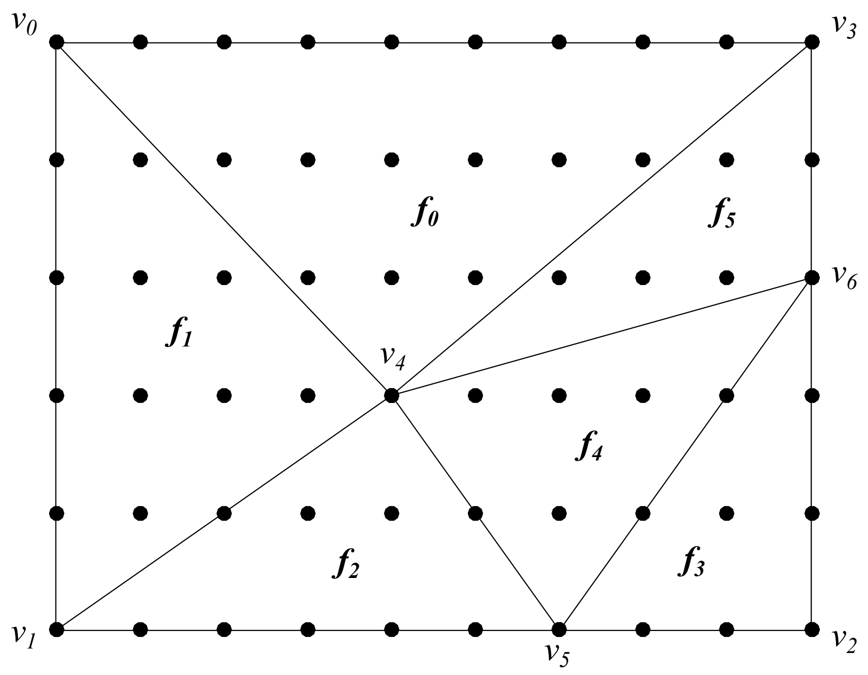
   (b) If p is not the right endpoint of any horizontal edge, map p to the face to the left of p, unless no such face exists, in which case map p to the face to the right of p.

Figure 2.11 shows an example to better illustrate the above mapping rules. In Figures 2.11(a) and 2.11(b), an image ϕ is defined on a rectangular grid $\{0...9\} \times \{0...5\}$. A subset $\{v_i\}_{i=0}^{6}$ of the points of the rectangular grid is selected and a triangulation of these points is constructed. The triangulation is shown superimposed on the grid. In order to help with understanding the mapping strategy used, each of the grid points in Figure 2.11(b) is marked with a symbol showing to which face it is mapped. For example, the grid point p_0 is mapped to the face f_0 as per rule 1 since the point is strictly inside the face f_0. The point p_1 is on the horizontal edge $\overline{v_2 v_5}$ but is not one of its endpoints. Since no face exists below the edge $\overline{v_2 v_5}$, the point p_1 is mapped according to rule 2a, to the face f_3 that is above the edge. The point p_2 is on the non-horizontal edge $\overline{v_4 v_5}$ which means that rule 2b applies, and the point is mapped to the face f_2 on the left of the edge $\overline{v_4 v_5}$. Rules 3a and 3b apply to the grid points v_2 and v_6, respectively. Hence, the points v_2 and v_6 are mapped to the faces f_3 and f_5, respectively.

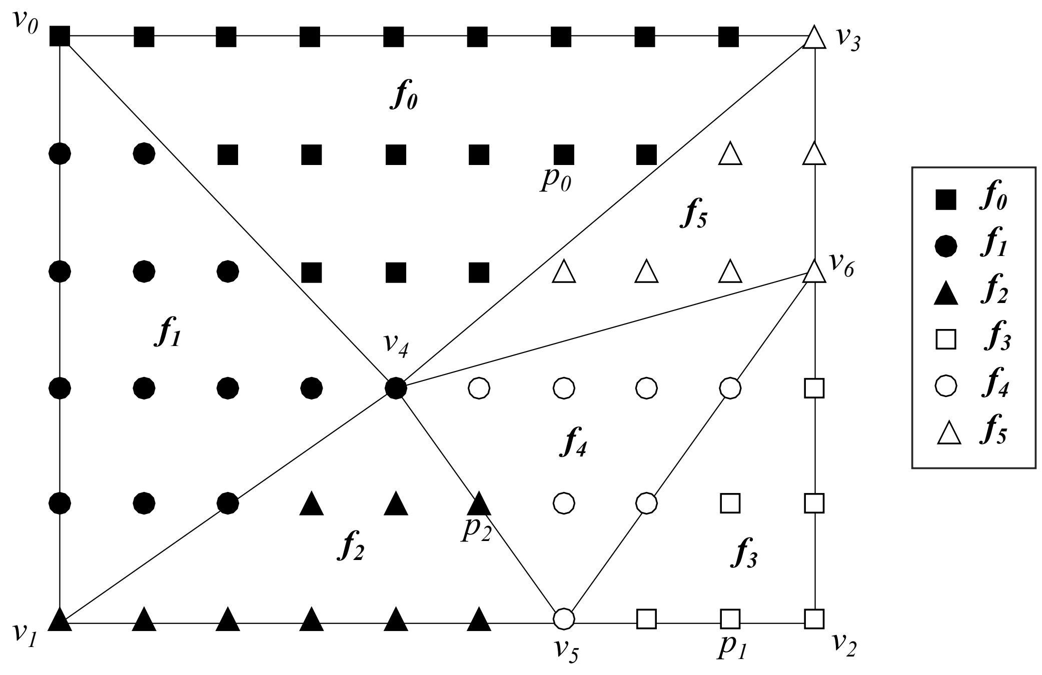
2.7 Error Diffusion

Error diffusion is a technique originally introduced by Floyd and Steinberg [38] for the purpose of image dithering. Yang et al. proposed using Floyd-Steinberg error diffusion with a feature map based on the MMSODD of an image (introduced in Section 2.3) to adaptively distribute points in the image domain [36]. The set of points generated have a spatial density that is approximately proportional to the amount of detail in a given region of the image. Given an image ϕ of width W and height H sampled on the set Λ of points (where $|\Lambda| = WH$), and a desired number N of points, error diffusion selects a subset P_N from Λ, where $|P_N| \approx N$, as follows:

1. Compute the MMSODD $d(\phi)$ of the image using (2.1).



(a)



(b)

Figure 2.11: Example of 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.

2. Compute the feature map σ from the MMSODD as

$$\sigma(x, y) = \left(\frac{d(x, y)}{A}\right)^{\gamma},$$

where $A = \max\left[d(\phi)\right]$ and γ is a positive constant sensitivity parameter.

3. In order to select approximately N sample points, compute the error diffusion threshold ρ as

$$\rho = \frac{1}{2N}\left[\sum_{i=1}^{W}\sum_{j=1}^{H}\sigma(i, j)\right].$$

4. Generate a binary image b defined on the same image domain as ϕ from the feature map σ using error diffusion with the threshold value ρ.

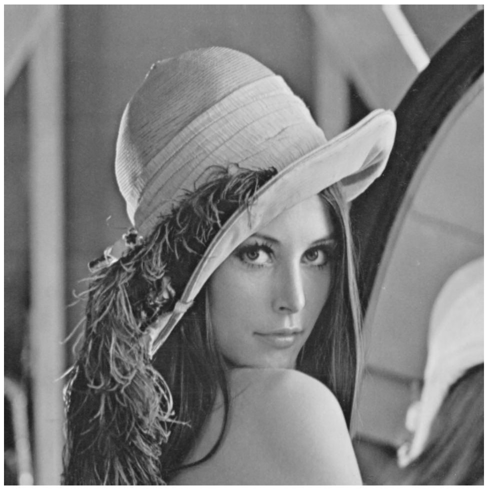
5. Select each sample point (x, y) of the image b that satisfies the condition $b(x, y) \neq 0$.

The error diffusion method above has some degrees of freedom. In our work, they are fixed to specific choices based on the recommendations and conclusions of past work [33, 36, 42]. The feature map sensitivity parameter is set to $\gamma = 1$. Non-leaky error diffusion with the serpentine scan order is used. A smoothing filter is normally applied to the image during the MMSODD computation to reduce the effects of noise. Its selection was left as a free parameter for our work.

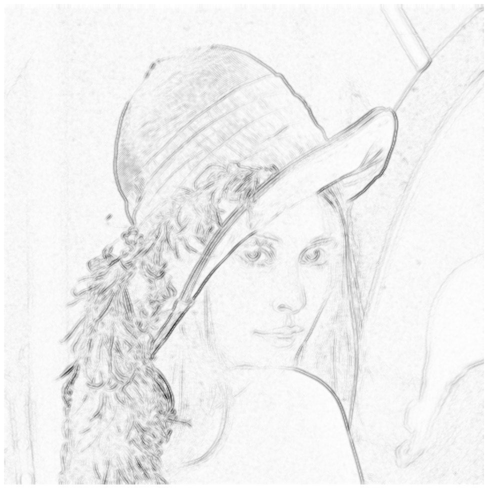
Figure 2.12 shows an example of the results produced by error diffusion. The MMSODD in Figure 2.12(b) is first (i.e., in step 1) computed from the image in Figure 2.12(a). Floyd-Steinberg error diffusion is then (i.e., in steps 4 and 5) used to select the sample points shown in Figure 2.12(c). The density of the points is clearly higher around image edges which correspond to areas in the image content with more detail.

2.8 Poisson Disk Sampling

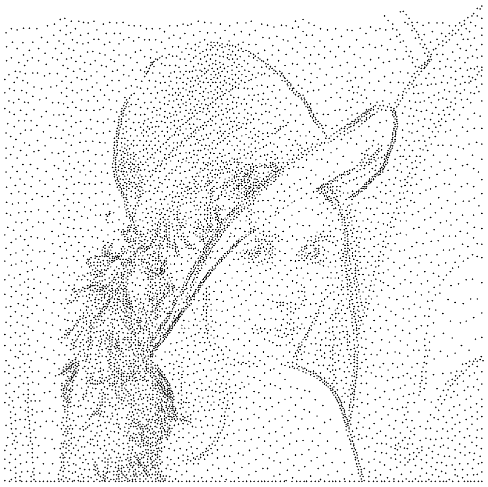
A random distribution of sample points with white noise properties can lead to clustering of points, or large gaps with few samples. On the other hand, blue noise samples are randomly and uniformly distributed in the spatial domain. Blue noise is valuable in a variety of imaging and rendering applications due to the fact that it does not exhibit



(a)



(b)



(c)

Figure 2.12: (a) An image, (b) its MMSODD, and (c) the sample points selected by error diffusion.

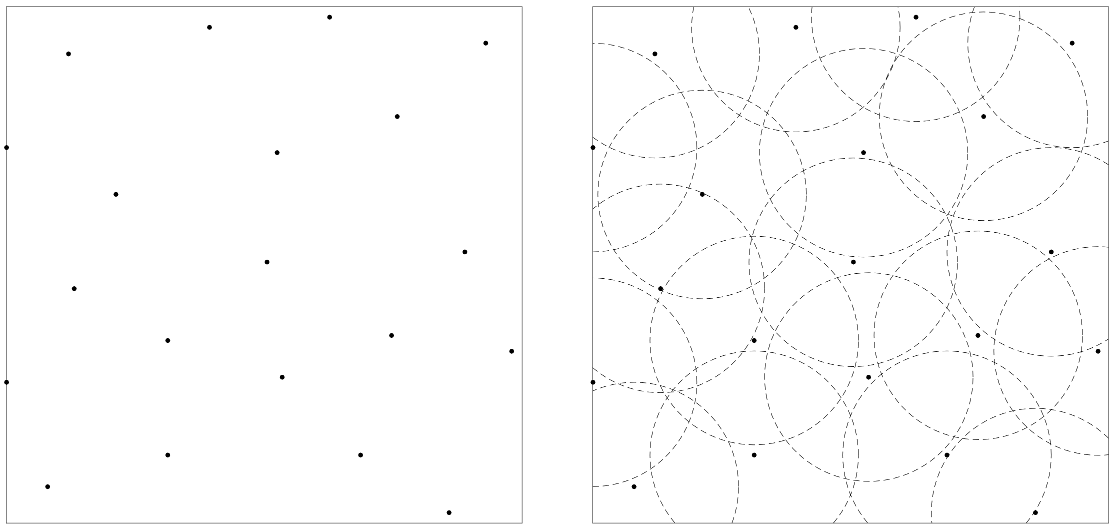

(a) (b)

Figure 2.13: Example of uniform Poisson disk sampling. (a) Sample points generated using uniform Poisson disk sampling, and (b) the boundaries of the exclusion disks.

aliasing artifacts, unlike uniform sampling [54–59]. Instead, frequencies above the Nyquist limit appear as noise of the correct average intensity, which is much more acceptable to the human visual system [54, 58, 60, 61].

Poisson disk sampling is a method for generating a set of points with blue noise properties. The basic idea is that constraints are placed on the minimum distance between any two samples in the set of points. For a constant minimum distance value d, the constraints result in a disk of exclusion with radius d around each sample point, where another sample cannot be placed. Figure 2.13 shows an example of sample points with blue noise properties selected using Poisson disk sampling. The circles in Figure 2.13(b) illustrate the boundary of the exclusion disks around the sample points in 2.13(a). Several techniques have been proposed for Poisson disk sample generation [62]. The two that are most well know are the dart throwing [54, 63, 64], and relaxation [65] methods.

The choice of the radius value clearly affects the total number of sample points that may be selected from the image domain. A smaller radius results in samples that are closer together, which leads to a larger set of sample points. Conversely, the samples are spaced further apart with a larger radius, which leads to fewer sample points. If the samples are not all chosen to have the same minimum distance, the distribution of points may be adjusted for different regions of the sampling domain. For this purpose, a radii function may be used during the evaluation of the suitability of candidates for addition to the set of sample points. Such an approach is often referred to as adaptive Poisson disk sampling [40, 66, 67]. An example is shown in Figure 2.14 where the radius

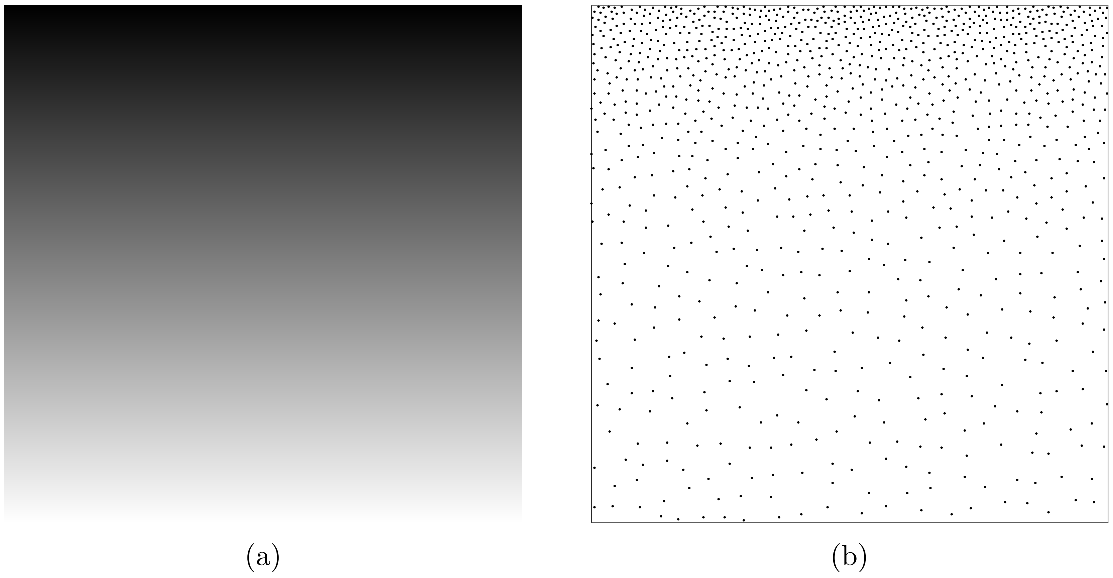


Figure 2.14: Example of adaptive Poisson disk sampling. (a) Visual representation of the radii function used, and (b) sample points generated based on the radii function.

is chosen to be proportional to the y-coordinate (but does not change along the x-axis). The function may be visualized as the value map in Figure 2.14(a) where a darker value represents a smaller radius. The sample points generated from the radii function are shown in Figure 2.14(b).

2.9 Incremental/Decremental Mesh-Generation Framework

In [20], Adams proposed a highly-flexible computational framework for generating triangle-mesh models of images with Delaunay connectivity. It is iterative in nature, alternating between two distinct processing phases: a mesh refinement phase that adds sample points to the mesh, and another mesh simplification phase whereby sample points are removed from the mesh. The alternation is performed according to a predetermined sequence $\{\eta_i\}_{i=0}^{L-1}$ of length L, referred to as a growth schedule. The number of vertices in the mesh tracks the values in the growth schedule at each iteration.

Before we can proceed further, some additional notation and terminology need to be introduced. The face f to which a point p is uniquely mapped, based on the rules described in Section 2.6, is denoted $\text{face}(p)$. The set of all points that are mapped to a face f (i.e. the points satisfying $\text{face}(p){=}f$) is denoted $\text{points}(f)$. Let $\hat{\phi}_S$ be the interpolant corresponding to the mesh with sample points S. Let $r_S(p)$ denote the approximation error at the point p for the mesh with sample points S (i.e., $r_S(p) = \hat{\phi}_S(p) - \phi(p)$). Let P denote the sample points in the current mesh approximation of ϕ. Consequently, the error over a face f, denoted faceErr(f), is the sum of the errors of the points contained in f (i.e. faceErr(f)= $\sum_{p \in \Lambda \cap \text{points}(f)} r(p)$), and the total mesh model approximation error is $\sum\limits_{p \in \Lambda} r(p)$. A point is said to be immutable if it is not allowed to be removed from or added to the mesh during the mesh-generation process; otherwise it is mutable. The subset of mutable points in a given set S is denoted mutable(S). A point $p \in \Lambda$ that is mutable but is not currently in the mesh is said to be a candidate point. The set of candidate points for a given face f are denoted cands(f) (i.e., cands(f) = mutable(($\Lambda \setminus P$) $\cap$ points(f)).

With the necessary background in place, we can now describe the computational framework. Given an image ϕ, a subset Γ of the sample points from which to construct the initial mesh, a growth schedule $\{\eta_i\}_{i=0}^{L-1}$, and a desired mesh size N (where $N \in [4, |\Gamma|]$), as input, the framework generates a triangle-mesh model of ϕ of the specified size that minimizes the total approximation error. The framework consists of the following steps:

1. Select initial mesh points. Obtain the initial subset P of the sample points as $P := \Gamma$.

2. Initialize mesh. Create a mesh consisting of the extreme convex-hull points of Λ. Then insert the points in P (from step 1) into the mesh. Mark the extreme convex-hull points as immutable so that they cannot be removed from the mesh, and mark all of the other points as mutable. Let $i := 0$.

3. If $i = L$ (i.e. no more points in the growth schedule remain), go to step 7; otherwise, proceed. If $|P| < \eta_{i+1}$, go to step 4 to increase the mesh size. If $|P| > \eta_{i+1}$, go to step 5 to decrease the mesh size. If $|P| = \eta_{i+1}$, go to step 6 (bottom of the main loop).

4. Increase mesh size. While $|P| < \eta_{i+1}$, add a point to the mesh by performing an optimal add (optAdd) operation which consists of the following steps:

   (a) Select a face f^* in which to insert a new point as given by
   $$f^* = \arg\max_{f \in \mho} \text{faceErr}(f),$$
   where $\mho$ is the set of all faces that have at least one candidate point.

   (b) Select a point p^* in the face f^* to add to the mesh, as given by
   $$p^* = \operatorname{selCand}(f^*),$$
   where selCand is a function that embodies the candidate-selection policy, and is a free parameter of the framework.

   (c) Add p^* to the mesh (i.e. let $P := P \cup \{p^*\}$).

   (d) Go to step 6.

5. Decrease mesh size. While $|P| > \eta_{i+1}$, delete a point from the mesh by performing an optimal delete (optDel) operation, which consists of the following steps:

   (a) Let the significance (with respect to deletion) of a (mutable) point $p \in P$, denoted sigDel(p), be defined as
   $$\operatorname{sigDel}(p) = \sum_{q \in (R \cap \Lambda)} (r^2_{P \setminus \{p\}}(q) - r^2_P(q)),$$
   where R is the region in the triangulation affected by the deletion of p. That is, sigDel(p) is the amount by which the squared error increases if p were deleted from the mesh. The point p^* to delete from the mesh is then selected as
   $$p^* = \operatorname*{arg\,min}_{p \in \operatorname{mutable}(P)} \operatorname{sigDel}(p).$$

   (b) Delete p^* from the mesh (i.e. let $P := P \setminus \{p^*\}$)

   (c) Go to step 6.

6. Restart main loop. Let $i := i + 1$. Go to step 3 (i.e. the top of the main loop).

7. Postprocess mesh. Optionally, perform some postprocessing steps; then stop.

As introduced above, the framework has several free parameters. A more detailed discussion of the parameters is deferred to Section 3.2.

2.10 Previously-Proposed Mesh-Generation Methods

The greedy point removal (GPR) scheme of Demaret and Iske [17] is a mesh-generation method based on simplification that starts from a mesh containing all the sample points in the image domain. Points are then removed iteratively from the mesh, one at a time, until the desired mesh size is achieved. The method takes advantage of the locality property of deletion in Delaunay triangulations (described in Section 2.4). The point selected for removal from the mesh in each iteration is the one that causes the mesh approximation error to increase the least following its removal from the mesh. The point is therefore chosen optimally from the whole mesh at a given iteration. Nevertheless, the point deletion policy is greedy.

The IDDT scheme proposed in [20] is an incremental/decremental method derived from the computational framework previously described in Section 2.9. It starts from a mesh containing only the four extreme convex-hull points of the image domain, and then alternates between mesh refinement and mesh simplification. The particular sequence of inserting points into, and deleting points from, the mesh is determined by the growth schedule. The growth schedule oscillates between values that are equal to the target mesh size, and values that are below it. The amplitude of the oscillations below the target mesh size decay exponentially according to a parameter of the method denoted by α. More concretely, the growth-schedule setpoints are given by

$$\eta_i = \begin{cases} N - \lfloor \alpha^{i/2}(N - |\mathit{\Gamma}|) \rfloor & i \text{ even} \\ N & i \text{ odd}, \end{cases}$$

where $\mathit{\Gamma}$ is the set of initial mesh points, $\alpha \in (0,1)$, and $L = 1 + 2\lfloor -\log_\alpha(N - |\mathit{\Gamma}|) \rfloor$. The policy for selecting the point to insert into a given face f (i.e. $\mathrm{selCand}(f)$) is known as the peak-weighted-absolute-error (PWAE), which is defined as

$$\operatorname*{arg\,max}_{p \in \mathrm{cands}(f)} d(p) |\hat{\phi}_P(p) - \phi(p)|,$$

where $d(p)$ is the MMSODD of ϕ at the point p.

Based on the same computational framework as the IDDT scheme, the ID1 and ID2 methods are a more effective alternative, yielding higher quality meshes. The ID1 and ID2 methods use what is known as growth schedule A, which has setpoints that alternate between values that are above the desired mesh size and the desired mesh size. The initial mesh is chosen as the trivial case of the extreme convex-hull points. Then, the mesh size oscillates between values that are equal to the desired mesh size, and values that are greater with an exponentially decaying amplitude. Growth schedule A is given by

$$\eta_i = \begin{cases} |\mathit{\Gamma}| & i = 0 \\ N & i \text{ even}, i \neq 0 \\ N + \lfloor \alpha^{(i-1)/2}(N - |\mathit{\Gamma}|) \rfloor & i \text{ odd}, \end{cases}$$

where $\alpha \in (0,1)$ and the growth schedule length is $L = 2 + 2\lfloor -\log_\alpha(N - |\mathit{\Gamma}|)\rfloor$.

As for selecting the insertion candidate point, the ID1 and ID2 methods use the so-called hybrid candidate-selection policy and the approximate local squared-error minimizer (ALSEM) policy, respectively. The ALSEM policy selects the candidate point iteratively by estimating the effect of multiple point insertions. A more detailed description is deferred to Section 3.2.4. The hybrid candidate selection chooses the candidate point using the PWAE policy during the initial part of the growth schedule, and later switches to the ALSEM policy, which is more computationally expensive. This allows the ID1 method to be faster, at a small penalty in terms of quality of the generated mesh. On the other hand, the ID2 method uses the ALSEM policy exclusively which makes the method slower. Nevertheless, both the ID1 and ID2 schemes perform significantly better than competing schemes such as the GPR method in terms of mesh model quality and computation cost.

The IDDT, ID1, and ID2 methods use a postprocessing step known as bad point replacement (BPR) that replaces bad points from the mesh after the application of the growth schedule is over. A bad point is a mutable point in the mesh whose deletion from the mesh does not cause the approximation error to increase (i.e., $\mathrm{sigDel}(p) \leq 0$). The BPR technique iteratively replaces bad points with new ones so that the total number of points in the mesh remains unchanged.

Chapter 3

Proposed Mesh-Generation Methods and Their Development

3.1 Overview

In this chapter, we introduce some aspects of Adams' framework in more detail, including its free parameters. We also describe the various choices for the free parameters that were considered for experimentation, including new choices that we proposed. Next, the image data used for analysis and evaluation purposes is briefly discussed. Subsequently, we evaluate the effect of the different choices for the free parameters on the quality of the generated mesh model. Finally, we proposed the IID1 and IID2 methods, which have lower and higher complexity, respectively, by fixing the framework's degrees of freedom based on the results of the experimental analysis.

3.2 Free parameters and Options Considered

As mentioned earlier in Section 2.9, the computational framework used as the basis for our mesh-generation methods has several free parameters. In what follows, we describe the framework's free parameters and the different choices considered for each of them.

Before proceeding further, some comments regarding the computational framework are in order. First, the framework is fundamentally greedy in that it only considers the current state and the immediate consequences of actions when selecting points to add to, or remove from, the mesh. As a result, the generated mesh is not guaranteed, or expected, to be a globally optimal solution. The fact that a non-global optimum is produced is an acceptable compromise given that the mesh-generation problem addressed in our work is known to be NP hard.

Additionally, it is clear that the manner in which the framework selects points for insertion and deletion is asymmetric. The point selected during a point deletion phase is the one that causes the error to increase the least. This is a greedy approach that yields a locally optimal solution for a particular iteration. On the other hand, the process by which the insertion candidate point is selected is not even likely to be locally optimal. Evidently, the framework could be made more symmetric by selecting for insertion the point that causes the approximation error to decrease the most as given by

$$p^* = \underset{p \in \text{mutable}(\Lambda \setminus P)}{\arg\max} \Big[\text{sigAdd}(p) \Big],$$

where sigAdd(p) denotes the amount by which the mesh approximation error decreases when the point p is added to the mesh. Unfortunately, computational considerations make such a choice problematic. First, the connectivity changes following the insertion of a point into a Delaunay triangulation are typically not local, unlike the case of point deletion. As a result, on average, a larger proportion of the image needs to be scan converted to compute the change in approximation error following the insertion of a given point. Moreover, the set of mutable points (i.e., mutable$(\Lambda \setminus P)$) for which sigAdd is evaluated, is typically much larger than the set of points evaluated for deletion using sigDel. Furthermore, given the greedy nature of the framework, such a drastic increase in computational cost is highly unlikely to yield a proportionate increase in the quality of the generated mesh.

The specific sequence of point deletions and insertions, which is determined from the growth schedule, has a significant impact on the performance. That is, the effect of two growth schedules that have the same total number of point insertions and deletions is typically quite different. For example, consider two growth schedules X and Y. With X, 2000 points are inserted into the mesh, and then 1000 points are removed. With Y, points are inserted by alternating between adding 20 points and removing 10 points for a total of 100 times. Applying both growth schedules to an initial mesh Γ consisting of the four extreme convex-hull points of the image domain will typically yield completely different results in terms of quality of the generated mesh and computational cost, even though the growth schedules have the same numbers of point insertions and deletions. From an implementation viewpoint, grouping point insertion operations together and grouping point deletions together is normally more efficient as we shall see later in Section 3.4.4, for reasons discussed in [21].

Another factor that greatly influences performance is the choice of the initial mesh. As stated earlier, the way insertion candidate points are selected tends to be sub-optimal in the long run due to the nature of the selection process and the greediness of the algorithm. Starting with a coarse initial mesh exacerbates this problem because the point-insertion policy has a larger window wherein it can insert multiple sub-optimal choices, which can steer the mesh-generation process into a bad direction from which it is unable to recover. Insertion-candidate selection policies that choose a point based on some measure relating to the approximation error are particularly prone to this problem. This is due to the fact that the error throughout the mesh is generally so large initially that the approximation error for a particular point is not indicative of how a good a choice it is. On the other hand, a non-trivial initial mesh can have significant benefits. Seen from an optimization perspective, the closer the starting point for the search process is to an optimal solution, the more likely it is that optimal solution will be reached and the less likely it is that the algorithm will be trapped in a bad local minimum. Another advantage of using a non-trivial initial mesh is that the computational cost of inserting and deleting points is higher the smaller the mesh size is. With fewer vertices in the triangulation, the faces are larger in proportion to the size of the image domain. The face-scanning operations to update the errors and select candidate points are therefore more computationally costly.

Given the above comments, the motivation behind certain choices of the framework's free parameters considered for experimentation will hopefully be clearer, and we are able to proceed.

3.2.1 Selection of Initial Mesh

The choices considered for selecting the point set Γ from which to construct the initial mesh fall into two categories. The first category of policies are primarily meant to achieve improvements in terms of computational cost compared to using a trivial mesh of the extreme convex-hull points. This is achieved by using an initial mesh with a size that is close to the target mesh size, and with a point distribution that avoids large faces throughout the image domain. The second category of policies is such that the initial mesh points are chosen to give a good initial approximation of the image using a content-adaptive approach, while computational cost is secondary.

The desired initial mesh size passed to the initial point-set selection policy is defined as a proportion of the target mesh density for the mesh-generation framework in the form of a ratio α. More formally, we have

$$\Gamma = \text{initMesh}(\phi, \alpha D),$$

where initMesh is the initial point-set selection policy and D is the target density. This particular way of choosing the initial mesh size is due to the fact that good choices for the value of this parameter typically depend on the target mesh size. It also allows the computational cost of the initial mesh to be more consistently proportional to the overall computational effort for mesh generation. Finally, the value of the parameter can be fixed more easily when deriving the methods to propose. It should be noted that the desired initial mesh density parameter is only meant as a rough guide to the policies. Consequently, depending on the particular policy chosen and rounding considerations, $|\Gamma|$ may only be approximately equal to $|\alpha D \Lambda|$.

In what follows, we introduce the choices considered for the initial point-set selection policy. All the choices considered for the initial mesh are newly proposed for the mesh-generation framework we are using. The random selection and error diffusion-based policies have been used in some other previously-proposed mesh-generation methods such as [42] and [33]. The other policies are new choices we adapted from other applications for the purpose of our work. The choices are as follows:

- Random. Points are selected randomly from a uniform distribution within the image domain. Some mesh-generation methods have shown this to be a poor choice [42] when used with a mesh-simplification only approach. Consequently, it is primarily used as a point of reference when evaluating the performance of the other choices.

- Uniform grid. Sample points are selected such that they are spaced uniformly in the image domain, subject to the limitations imposed by integer coordinates. Effectively, this corresponds to uniform subsampling of the image domain. This approach results in an initial mesh with faces that are roughly equal in area, and therefore have a similar number of mutable points in them. Since the error of a face may be influenced by the number of points it contains, this ensures that the effect of the size of a face on its error is reduced. Consequently, the first point insertions and deletions during mesh generation can be directed at the regions in the mesh with the most error and which would benefit most.
  Essentially, the image domain $D = [0, W-1] \times [0, H-1]$ is split into M equally sized blocks. Then, the corners of the blocks are selected as the sample points. Let w_M and h_M be the width and height of each block. Then the image width

and height will be split into $\frac{W}{w_M}$ and $\frac{H}{h_M}$ equally sized intervals, respectively. The number of blocks is therefore

$$M = \frac{W}{w_M}\frac{H}{h_M},$$

and the number of points corresponding to the corner points of the blocks is

$$N = \text{round}\left[\left(\frac{W}{w_M}+1\right)\left(\frac{H}{h_M}+1\right)\right].$$

If the blocks dimensions are chosen to have the same aspect ratio as the image, the number of points is

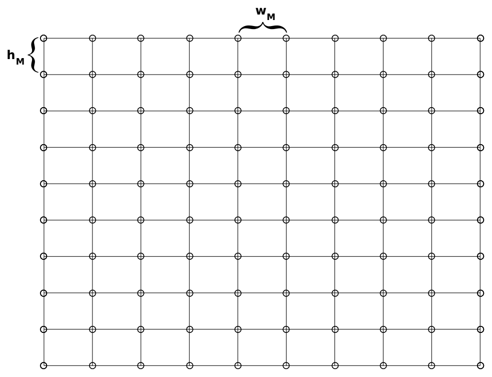
$$N = \text{round}\left[\left(\frac{W}{w_M}+1\right)\left(\frac{H}{\frac{H}{W}w_M}+1\right)\right] = \text{round}\left[\left(\frac{W}{w_M}+1\right)^2\right].$$

Hence, we have that

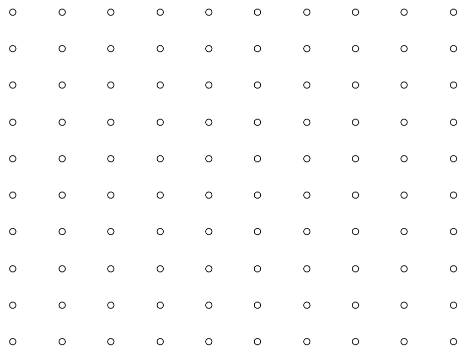
$$\begin{cases} w_m = \frac{W}{\sqrt{N}-1} \\ h_m = \frac{H}{W}w_m. \end{cases},$$

where N is the desired number of initial sample points (i.e., $N = |\alpha D\Lambda|$). Consequently, spacing points along the x and y directions by w_M and h_M, respectively, as defined above, would approximately give the total number of points desired. An example is shown in Figure 3.1 to better illustrate the process. In Figure 3.1(a), the image domain is split into M equally sized blocks. The width and height of each block are w_M and h_M, respectively. The corner points of the blocks are marked with circles. The corner points are selected as the sample points leading to the initial point set shown in Figure 3.1(b).

- Jittered grid. Points are placed approximately uniformly across the image domain with the approach used in uniform sampling above. Each point p_i is then displaced along the x and y directions by x_i and y_i ($x_i \in \mathbb{Z}$ and $y_i \in \mathbb{Z}$) respectively. The displacement amounts x_i and y_i are random and small enough to ensure that a point p_i remains within the vicinity of its original position (i.e., within $\pm\frac{w_M}{2}$ and $\pm\frac{h_M}{2}$ where w_M and h_M are as defined previously for the uniform grid). This is a form of stochastic sampling that has applications in image-rendering. It is used as



(a)



(b)

Figure 3.1: Selection of the uniform grid sample points. (a) The image domain divided into equally sized blocks and (a) the sample points selected.

an approximation to a uniform Poisson disk distribution due to its computational efficiency [54]. As a result, it has the anti-aliasing properties of stochastic sampling in addition to the benefits associated with a uniform grid.

- Error diffusion. Use the process described earlier in Section 2.7. The choices considered for the smoothing filter applied prior to the computation of the MMSODD are the binomial filter and the bilateral filter.

- Adaptive Poisson-disk sampling. Use the process described in Section 2.8. The mapping function that creates the radii map ϱ from the input image for adaptive Poisson disk sampling consists of the following steps:

  1. Apply a 7th-order binomial smoothing filter to the input image ϕ to reduce the effect of noise in subsequent filtering operations.
  2. Apply an image transform f to the smoothed image to create a map (i.e. bivariate function) whose values correlate with the level of detail. The options considered are:
     - gradient
     - Laplacian
     - maximum magnitude second order directional derivative (MMSODD)

3. Compute the normalized map $\tilde{f}$ as

$$\tilde{f}(p) = \frac{f(p) - \min\limits_{q \in \Lambda}[f(q)]}{\max\limits_{q \in \Lambda}[f(q)] - \min\limits_{q \in \Lambda}[f(q)] + \varepsilon},$$

   for $p \in \Lambda$, so that all values are in the range $[0, 1]$. The value ε is a very small decimal (i.e., $\varepsilon \approx 0$) that prevents division by zero in the unlikely case where $\max\limits_{q \in \Lambda}[f(q)] = \min\limits_{q \in \Lambda}[f(q)]$.

4. Apply a range inversion transform that flips the normalized values from the previous step within the range $[0, 1]$. The transform is defined as

$$g(p) = 1 - \tilde{f}(p).$$

   The reason for this range inversion is that larger values in the normalized map $\tilde{f}$, which was computed in the previous step, correspond to more detail. On the other hand, a smaller radius is needed in order to place more points in a region of the image with more detail using Poisson disk sampling.

5. Apply a sensitivity adjustment function $h(p) = [g(p)]^\gamma$ to the values in the map, where γ is a sensitivity adjustment parameter. This causes the difference between the values in the radii map ϱ to increase or decrease, thereby adjusting the number of points that are selected.

- Subset of points with greatest detail. For a desired number of sample points $N = |\alpha D\Lambda|$, a subset of the sample points is selected based on the input image as follows:

  1. Apply a 7-th order binomial smoothing filter to the input image ϕ to reduce the effect of noise in subsequent filtering operations.
  2. Derive a detail image f from the smoothed image using one of the following transforms:
     - gradient
     - Laplacian
     - MMSODD
  3. Create a histogram H of the detail image with a sufficiently large bin count β in order to cover the whole range of values without lumping too many distinct values together in a single bin (e.g., $\beta = 1024$). The histogram is organized such that bin 0 contains the largest values,

4. Initialize the index of the current bin $i := 0$ and the current sample count $C := H[0]$.
5. If $C \geq N$, go to step 7.
6. Increment i and update the sample count $C := C + H[i+1]$. Go to step 5.
7. Set the threshold θ to the value associated with the histogram bin at index i.
8. Select all sample points from the detail image f that have a value greater than or equal to the threshold θ.

The three policies are collectively referred to as the subset policies for short. The recursive part of the algorithm above basically tries to extract the N sample points that have the highest values (i.e., the most detail) from the detail image. Clearly, the actual number of sample points selected will depend on the number of bins used as well as the extent to which values overlap.

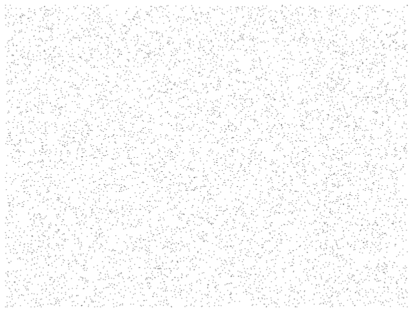
An illustration of the set of sample points selected by the various choices of policies considered is shown in Figure 3.2.

3.2.2 Growth Schedule

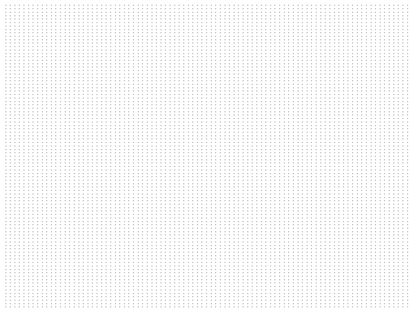
In [21], four growth schedules named A, B, C, and I were considered. The A growth schedule was selected by the author for the ID1 and ID2 methods, while the B growth schedule was used in the earlier IDDT method. These growth schedules work reasonably well and they can be used to cover a large number of test cases that may be of interest by changing their α parameter. Nevertheless, these growth schedules are not well suited for the larger non-trivial initial meshes that are being considered in our work. Consequently, we chose to create four new growth schedules that are able to cover an even wider range of test cases, while still keeping with the idea behind each of the original schedules. One difference is that our definitions of the growth schedules do not consider the size of the initial mesh. In addition, the length of the growth schedule is manually set and determines the number of alternations between sequences of point insertions and deletions before convergence to the target mesh size.

In what follows, we describe our four new growth schedules.

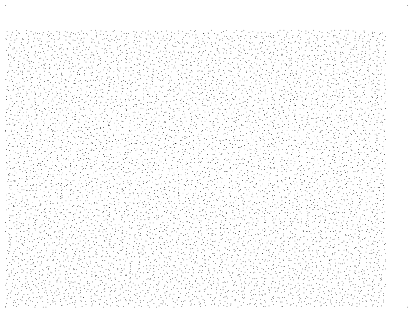
- Incremental (I′). Depending on the size of the initial mesh, $\Big| N - |\Gamma| \Big|$ consecutive point additions or point deletions are performed to reach the target mesh size N.



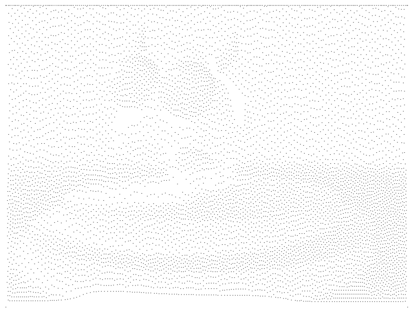
(a)



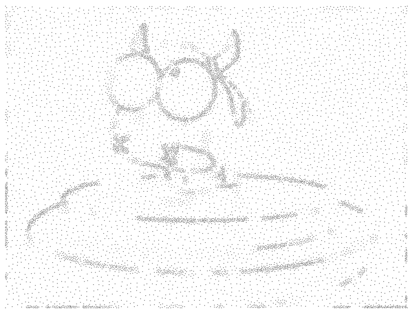
(b)



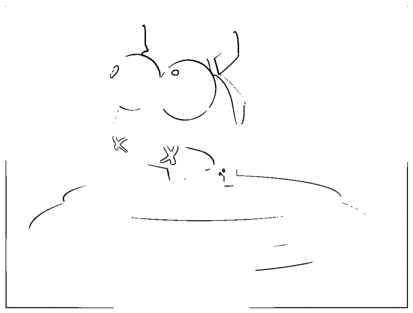
(c)



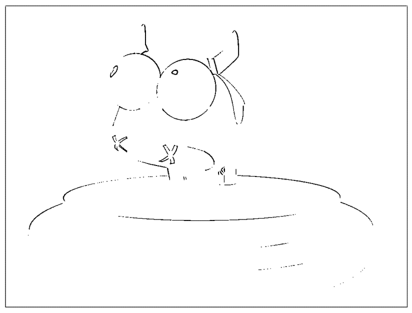
(d)



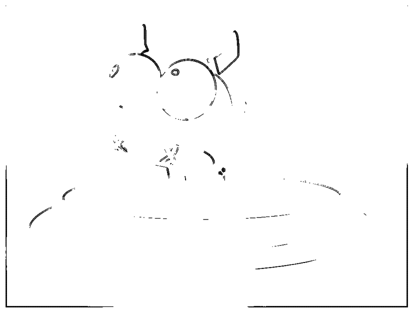
(e)



(f)



(g)



(h)

Figure 3.2: Examples of the sample points generated by each of the choices considered for generating the initial set of points. (a) Random, (b) uniform grid, (c) jittered grid, (d) error diffusion, and (e) Poisson-disk sampling policies. Subset of points with greatest detail policies with (f) gradient, (g) Laplacian, and (h) MMSODD detail image.

The growth schedule has a fixed length $L = 2$ and its setpoints are given by

$$\begin{cases} \eta_0 = |\Gamma| \\ \eta_1 = N. \end{cases}$$

Examples are shown in Figure 3.3 for two cases of growth schedule I′. The horizontal dashed line represents the target mesh size to which the size converges and the circular markers show the setpoints of the growth schedule. Two cases are shown: 1) when the initial mesh is smaller than the target mesh (i.e., $\alpha < 1$), and 2) when the initial mesh is larger than the target mesh (i.e., $\alpha > 1$).

- Circa (C′). The mesh size oscillates about the target mesh size N with exponential decay. The number of oscillations is equal to the growth schedule length L. The growth schedule setpoints are given by

$$\eta_i = N + (-1)^i \left\lfloor NAe^{\frac{-4i}{L-1}} \right\rfloor,$$

where A is an amplitude parameter and L is the growth schedule length. The parameters of the growth schedule will be discussed later in this section because they are common to types A′, B′, and C′. Figure 3.4 illustrates the evolution of the mesh size and the setpoints (circular markers) for two cases of the C′ growth schedule: with an initial mesh size that is 1) smaller and 2) larger than the target mesh size.

- Above (A′). The mesh size oscillates between the target mesh size N and exponentially decaying values above N. The number of oscillations above N is equal to the growth schedule length L. The growth schedule setpoints are given by

$$\eta_i = \begin{cases} N + \left\lfloor NAe^{\frac{-4i}{L-1}} \right\rfloor & i \text{ even} \\ N & i \text{ odd.} \end{cases}$$

Examples of the evolution of the mesh size for growth schedule A′ is shown in Figure 3.5. The two cases illustrated are for an initial mesh obtained with: 1) $\alpha < 1$ and 2) $\alpha > 1$.

- Below (B′). The mesh size oscillates between the target mesh size N and values below it, converging to N. The growth schedule setpoints are given by

$$\eta_i = \begin{cases} N - \left\lfloor NAe^{\frac{-4i}{L-1}} \right\rfloor & i \text{ even} \\ N & i \text{ odd.} \end{cases}$$

Examples of the evolution of the mesh size for growth schedule B′ is shown in Figure 3.6. The two cases illustrated are for an initial mesh obtained with: 1) $\alpha < 1$ and 2) $\alpha > 1$.

The amplitude parameter A of growth schedules A′, B′, and C′ defines the amplitude of the oscillations. A higher value leads to more points being added to/removed from the mesh during each phase of point insertions/deletions in the growth schedule. The amplitude of the oscillations is determined by NA (i.e., proportional to the target mesh size) in the equations, so that the value of A can be fixed more easily when deriving a method to work consistently with different image sizes. The length parameter L determines the number of setpoints in the growth schedule sequence. With the two parameters A and L, the number of insertion/deletion phases, and the number of operations in each phase can be controlled independently. Such an approach is advantageous because, as previously mentioned, the length and number of insertion/deletion phases impact the quality of the generated mesh and the computational cost in different ways. This differs from the original growth schedules A, B, and C with which the number of setpoints and the amplitude of oscillations could not be controlled directly or independently.

The formulas for growth schedules A′, B′, and C′ are set such that the oscillations have decayed considerably by the end. The last setpoint is designed to be within less than 2% of the initial amplitude. As seen in what follows, the decaying oscillations term becomes (for $i = L - 1$)

$$Ae^{\frac{-4i}{L-1}} = Ae^{\frac{-4(L-1)}{L-1}} = Ae^{-4} = 0.0183A = \frac{1.83}{100}A.$$

The decision to have the growth schedule end within 2% of the target mesh size is due to efficiency considerations. This is based on the fact that operations beyond that point have a noticeable computational cost with minimal impact on the quality of the generated mesh. This is to be expected given that frequent alternation between insertions and deletions is inefficient. The value of the exponent term was rounded to 4 for convenience. In passing, we note that, since we aim for a final mesh size that is exactly equal to N, growth schedules A′ and B′ are constrained to length values that are even so that the

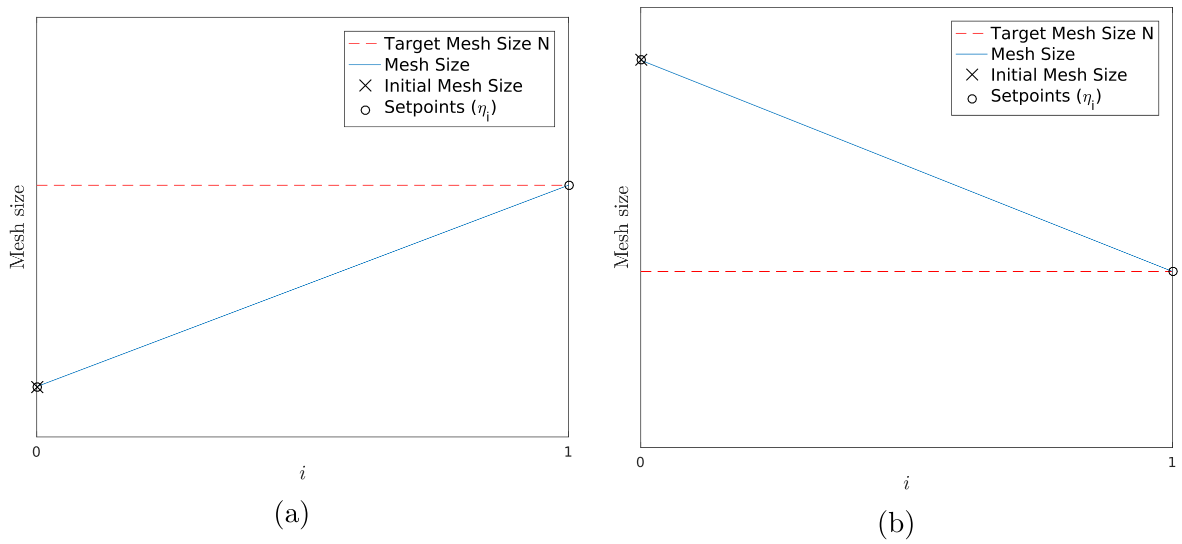


Figure 3.3: Evolution of the mesh size, and the setpoints for I′ growth schedule with (a) $\alpha < 1$ and (b) $\alpha > 1$.

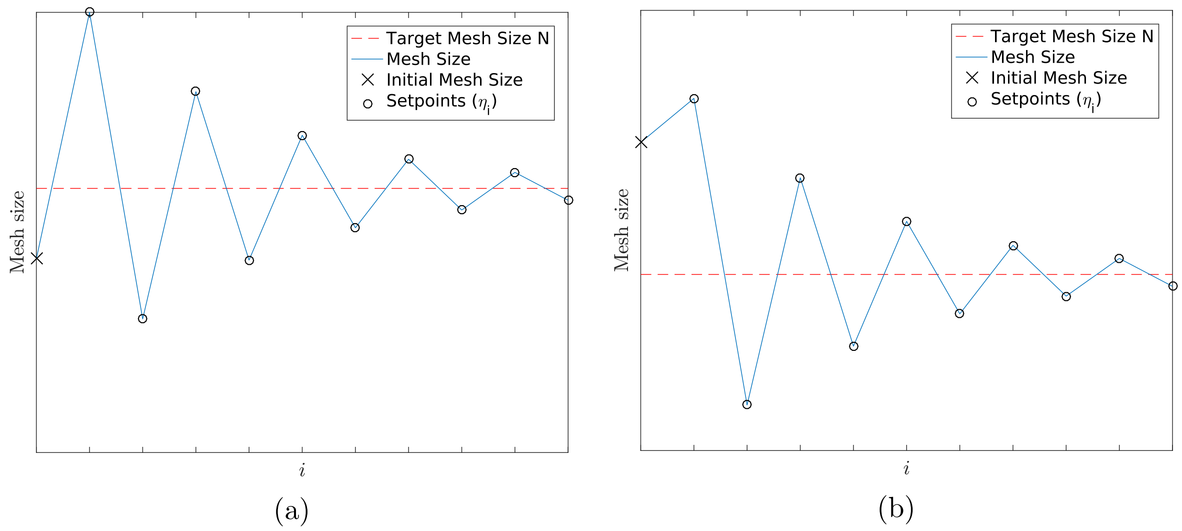
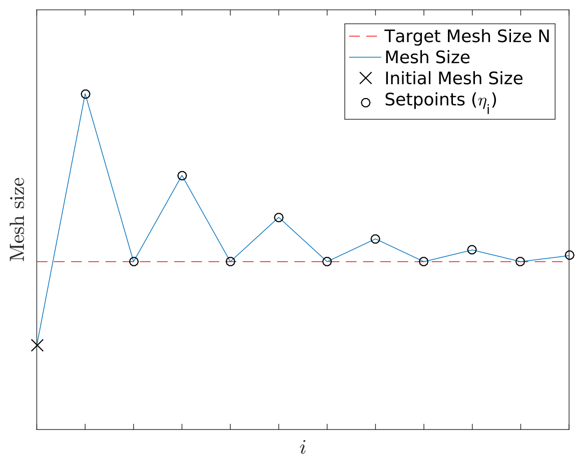
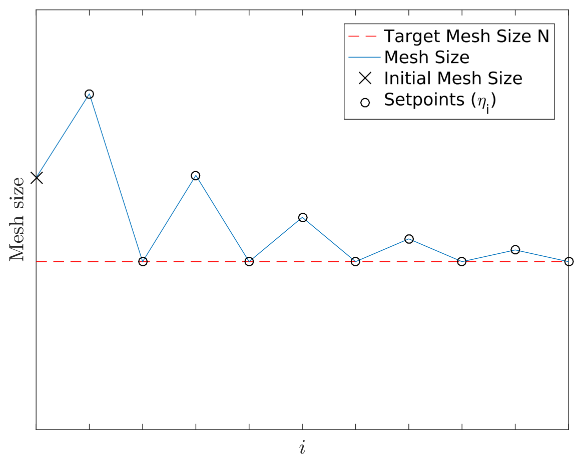


Figure 3.4: Evolution of the mesh size, and the setpoints for C′ growth schedule with (a) $\alpha < 1$ and (b) $\alpha > 1$.

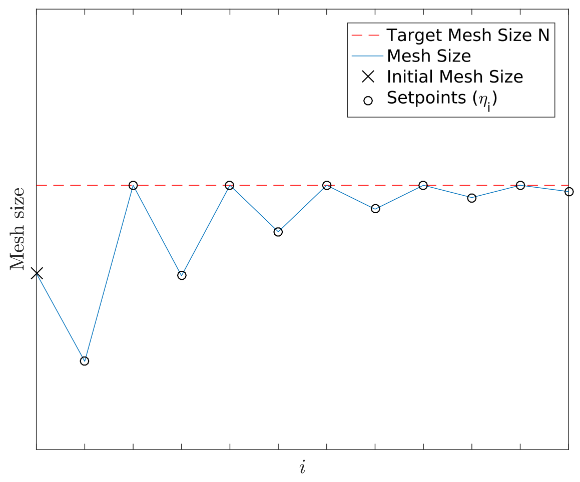


(a)

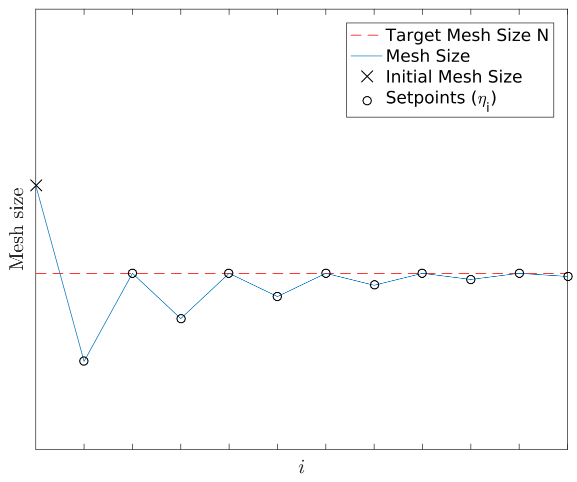


(b)

Figure 3.5: Evolution of the mesh size, and the setpoints for A′ growth schedule with (a) $\alpha < 1$ and (b) $\alpha > 1$.



(a)



(b)

Figure 3.6: Evolution of the mesh size, and the setpoints for B′ growth schedule with (a) $\alpha < 1$, and (b) $\alpha > 1$.

last setpoint is $\eta_{L-1} = N$ (i.e., $i = L - 1$ is odd). In the case of growth schedule C$'$, a final round of point insertions or deletions is performed to bring the mesh size to N since the $Ae^{\frac{-4i}{L-1}}$ term is not zero for $i = L - 1$.

3.2.3 Face Selection Policy

The policy used for selecting the face in which a point is to be inserted is fixed in the original framework formulation. We elected to consider the following choices as well in a bid to improve the quality of the mesh generated:

- Sum of squared-errors (SSE). The chosen face f^* is the one with the highest approximation error measured in SSE. The SSE forms the basis of the MSE which is used to measure the total mesh approximation error. As a result, it seems intuitive that it should be used for the face selection policy as well, since that would have the most impact on the total mesh error. This policy is also used in the original framework and selects f^* as

  $$f^* = \arg\max_{f \in \mho} \Big[\text{faceErr}(f) \Big],$$

  where $\mho$ is the set of all faces that have at least one candidate point, and the face error faceErr is defined as

  $$\text{faceErr}(f) = \sum_{p \in \text{points}(f)} \Big(\tilde{\phi}(p) - \phi(p) \Big)^2 .$$

- Sum of absolute errors (SAE). The chosen face f^* is the one with the highest approximation error measured in terms of SAE. The SSE above is used as a measure of approximation error in many modelling applications due to some of its mathematical properties. One of its weaknesses, however, is that outlier values such as noise can easily affect the quality of the model. In the case of photographic images, noise is frequently present which means the quality of the mesh model can suffer. The SAE is a good alternative to the SSE because it is less susceptible to the effect of outliers. The policy selects f^* as

  $$f^* = \arg\max_{f \in \mho} \Big[\text{faceErr}(f) \Big],$$

  where $\mho$ is the set of all faces that have at least one candidate point, and the face error faceErr is defined as

$$\text{faceErr}(f) = \sum_{p \in \text{points}(f)} \left| \tilde{\phi}(p) - \phi(p) \right|.$$

- Highest level of detail (HLoD). A detail image D_ϕ is derived from the original image using an image transform. The values in D_ϕ correlate with the amount of detail in corresponding regions of the original image. The level of detail for each face is given by

  $$\text{detailLevel}(f) = \sum_{p \in \text{points}(f)} \left| D_\phi(p) \right|.$$

  The face f^* is selected as

  $$f^* = \arg\max_{f \in \mho} \Big[\text{detailLevel}(f) \Big],$$

  where $\mho$ is the set of all faces that have at least one candidate point. The choices considered for creating the detail image are:

  1. gradient;
  2. Laplacian; and
  3. MMSODD.

  The motivation behind this policy is the observation that faces and regions of the mesh model that benefit the most from inserting more points are those with more detail (i.e., edges, texture). As more points are inserted using this policy, the faces in regions of high detail become increasingly smaller thereby improving the approximation.

3.2.4 Candidate Selection Policy

The policy selCand chooses a candidate point p^* from within the face f^* to insert into the mesh. The policies considered are as follows

- Peak absolute error (PAE). Of all the candidate points in the face, the policy selects the point that has the highest absolute error. The candidate selection function selCand is defined as

$$\text{selCand}(f) = \underset{p \in \text{cands}(f)}{\arg\max} \left[|\tilde{\phi}_P(p) - \phi_P(p)| \right].$$

This is considered a baseline since it is a commonly used policy in mesh-generation schemes.

- Approximate local squared-error minimizer (ALSEM), proposed in [21]. The function selCand that embodies the policy is defined as

$$\text{selCand}(f) = \underset{p \in S}{\arg\max} \sum_{q \in \text{points}(f)} \left(r_P^2(q) - r_{P \cup \{p\}}^2(q) \right), \tag{3.1}$$

  where S is a subset of cands(f). With $d(p)$ denoting the MMSODD of the point p, S is chosen as follows:

  - If $|\text{cands}(f)| > 18$, S is chosen as the 9 points $p \in \text{cands}(f)$ for which $d(p)|\hat{\phi}_P(p) - \phi(p)|$ is greatest, in addition to 9 other randomly-chosen (distinct) points.
  - Otherwise, $S = \text{cands}(f)$.

  In (3.1), the summation corresponds to the reduction in the squared error if p were inserted into the mesh, computed only locally over the points that are in points(f). In other words, a simulation of the insertion of point p into the mesh is performed with the assumption that no changes to the mesh occur outside the face f. The connectivity update is therefore assumed to be a simple splitting of the face by connecting the newly added vertex with the three pre-existing vertices of the face f. In the case where p is on an edge e, the insertion of p is assumed to result in the splitting of f into two triangles by connecting with a new edge the newly added vertex and the existing vertex of f that is opposite e. Figure 3.7 shows an example of the ALSEM policy. In Figure 3.7(a), the point p is a candidate point for the face f that is shaded. When computing the local change error change over f, the ALSEM policy assumes that the insertion of p only results in the creation of the new edges that are in dashed lines. In Figure 3.7(b), the correct connectivity change following the insertion of p into the mesh is shown. The face f, which no longer exists in the triangulation, and into which the point p was inserted is represented by the shaded area for reference.

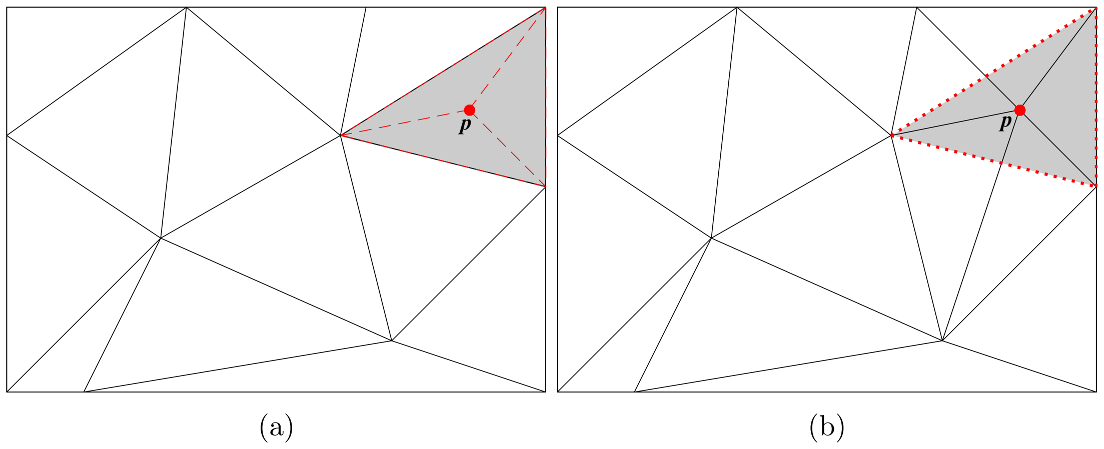


Figure 3.7: Assumed connectivity change in the ALSEM candidate selection policy. (a) The point considered for insertion into the shaded face, and the connectivity update assumed by ALSEM (dashed lines). (b) The actual Delaunay connectivity after inserting the point into the face.

- Peak gradient magnitude. Selects the candidate point that has the highest gradient value in the face. With $G(p)$ denoting the magnitude of the gradient of the image at point p, the function selCand is defined as

$$\operatorname{selCand}(f) = \underset{p \in \operatorname{cands}(f)}{\operatorname{arg\,max}} \Big[G(p) \Big].$$

- Peak Laplacian magnitude. Selects the candidate point that has the highest Laplacian magnitude in the face. With $\nabla(p)$ denoting the value of the Laplacian of the image at point p, the function is defined as

$$\operatorname{selCand}(f) = \underset{p \in \operatorname{cands}(f)}{\operatorname{arg\,max}} \Big| \nabla(p) \Big|.$$

- Peak MMSODD. Selects the candidate point that has the highest MMSODD value in the face. With $d(p)$ denoting the (nonnegative-valued) MMSODD of the point p, the policy selCand function is defined as

$$\operatorname{selCand}(f) = \underset{p \in \operatorname{cands}(f)}{\operatorname{arg\,max}} \Big[d(p) \Big].$$

The motivation behind the gradient, Laplacian, and MMSODD-based policies is the observation that points are best inserted around strong edges of the image. This is due to the fact that, when a sufficient number of points is placed in this manner, edges of the triangulation coincide with edges in the image which helps to reduce the approximation error.

3.2.5 Postprocessing

In what follows, we consider two choices for the postprocessing step of the mesh-generation framework. The two choices are aimed at dealing with bad points in the final mesh:

- Bad point replacement (BPR). This strategy, which was introduced in [21] and briefly described in Section 2.10, removes bad points from the mesh and inserts new points so that the total number of points remains unchanged. It consists of the following steps:

    1. Let $n_{old} := \infty$ and let $c := 0$.
    2. Let $n := 0$; while the point p that would be deleted from the mesh by the next optDel operation satisfies sigDel(p)≤ 0, perform an optDel operation (to delete p), mark p as immutable and, and let $n := n + 1$.
    3. If $n > 0$, perform n optAdd operations.
    4. If $n \geq n_{old}$, let $c := c + 1$.
    5. Let $n_{old} := n$; if $n = 0$ or $c \geq 3$, stop; otherwise, go to step 2.

- Bad point deletion (BPD). We consider a postprocessing algorithm known as bad point deletion. This is a simplified version of the bad point replacement strategy. The modified policy removes bad points, which are the result of the suboptimality of the candidate-selection policies, without replacing them with new ones. The motivation behind this simplification is the fact that point insertions, which are performed in BPR, do not necessarily decrease the error, and may in fact increase it. On the other hand, point deletion operations in bad point replacement are guaranteed to not increase the approximation error. As BPD only removes points from the mesh, the mesh density of the result is less than or equal to the target mesh density. As a result, it is only suitable for instances where small gains in mesh quality are deemed more important than obtaining a mesh with an exact size. The bad point deletion algorithm consists of the following steps:

1. If $\text{mutable}(P) = \emptyset$, go to step 5. Otherwise, update the significance with respect to deletion of the mutable points that are in the mesh (i.e., $\text{sigDel}(p)$ for $p \in \text{mutable}(P)$). As stated previously, $\text{sigDel}(p)$ is the amount by which the squared error increases if p were deleted from the mesh.
2. Select the point p^* defined as

$$p^* = \underset{p \in \text{mutable}(P)}{\arg\min} \ \text{sigDel}(p).$$

3. If $\text{sigDel}(p^*) \leq 0$, go to step 4. Otherwise, go to step 5.
4. Remove p^* from the mesh. Go to step 1.
5. Return P and terminate.

3.3 Test Data

Before proceeding to present experimental results, a brief digression is in order regarding the test images we used. In our work, we employed 50 images, taken mostly from standard data sets, such as the Kodak test set [68], JPEG-2000 test set [69], and USC image database [70]. During the analysis of the free parameters, we use a subset of the data set consisting of 40 images for statistical analysis. For the evaluation of the proposed methods, the full set of 50 images listed in Table 3.1 is used, with the extra 10 images serving as a validation data set. Such an approach allows us to reduce the likelihood of overfitting the data when developing the mesh-generation methods. In the remainder of the thesis, when presenting results for individual test case, we focus on the small representative subset of the test images listed in Table 3.2. The images in the set represent a variety of image types: photographic, medical, and computer-generated imagery.

3.4 Analysis of Options Available

We introduced the mesh-generation framework earlier in Section 2.9 as well as its degrees of freedom and the choices considered for each of them in Section 3.2. In what follows, we examine the effect the various choices have on the framework's performance. Based on the results of this analysis, we make recommendations regarding a particular set of choices for the parameters, leading to the two specific mesh-generation methods proposed herein.

Table 3.1: Images in the test data set

Image	Dimensions	Bit depth	Description
aerial2	2048×2048	8	Aerial photography
bike	2048×2560	8	Photographic image
cats	3072×2048	8	Photographic image
chart	1688×2347	8	Computer-generated document
cmpnd1	512×768	8	Computer-generated document
cr	1744×2048	10	CR scan of abdomen
elev	1201×1201	12	Elevation map
gold	720×576	8	Photographic image
hotel	720×576	8	Photographic image
mat	1528×1146	8	Photographic image
mri	256×256	11	MRI scan of head
tools	1524×1200	8	Photographic image
us	512×448	8	Abdominal ultrasound
water	1465×1999	8	Photographic image
woman	2048×2560	8	Photographic image
x_ray	2048×1680	12	X-ray
kodim01	768×512	8	Photographic image
kodim02	768×512	8	Photographic image
kodim03	768×512	8	Photographic image
kodim04	512×768	8	Photographic image
kodim05	768×512	8	Photographic image
kodim06	768×512	8	Photographic image
kodim07	768×512	8	Photographic image
kodim08	768×512	8	Photographic image
kodim09	512×768	8	Photographic image
kodim10	512×768	8	Photographic image
kodim11	768×512	8	Photographic image
kodim12	768×512	8	Photographic image
kodim13	768×512	8	Photographic image
kodim14	768×512	8	Photographic image
kodim15	768×512	8	Photographic image
kodim16	768×512	8	Photographic image
kodim17	512×768	8	Photographic image
kodim18	512×768	8	Photographic image
kodim19	512×768	8	Photographic image
kodim20	768×512	8	Photographic image
kodim21	768×512	8	Photographic image
kodim22	768×512	8	Photographic image
kodim23	768×512	8	Photographic image
kodim24	768×512	8	Photographic image
animal	1238×1195	8	Computer-generated
bull	1024×768	8	Computer-generated image
checkerboard	512×512	8	Computer-generated image
checkerboard_aa	512×512	8	Computer-generated image
ct	512×512	12	CT scan of head
glasses2	1024×768	8	Computer-generated scene
lena	512×512	8	Photographic image
muttart	1912×761	8	Photographic image
peppers	512×512	8	Photographic image
wheel	512×512	8	Computer-generated image

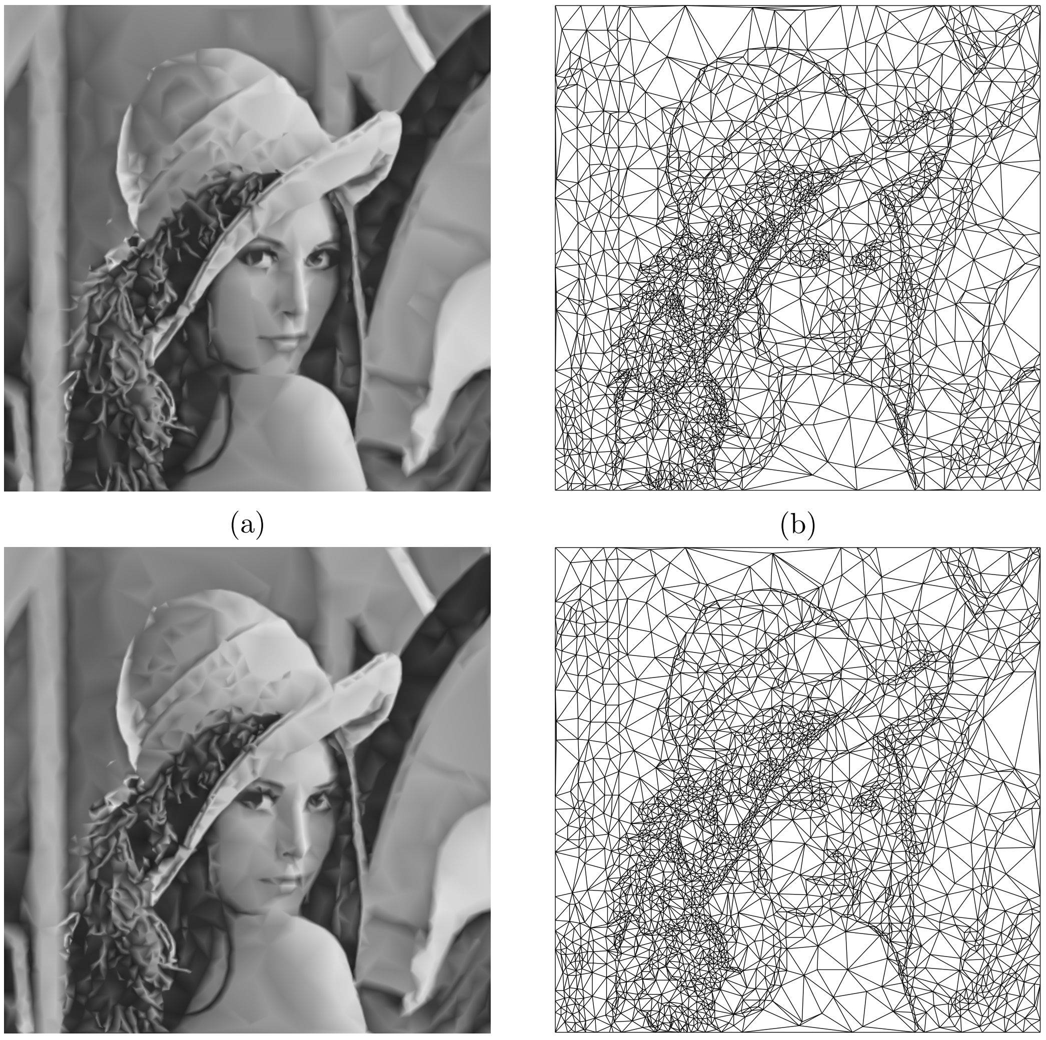
Table 3.2: Images in the representative data set

Image	Dataset	Dimensions	Bit depth	Description
`bull`	Misc.	1024×768	8	Computer-generated bull
`ct`	Misc.	512×512	12	CT scan of head
`lena`	Misc.	512×512	8	Photographic image of woman

3.4.1 Face-Selection Policy

First, we study the effect of the face-selection policy in step 4 of the framework from Section 2.9 on the quality of the mesh. To do this, we fix the initial mesh to the extreme convex-hull points of the image domain, the candidate-selection policy to be PAE, the growth schedule to be A′ with $L = 15$ and $A = 1$, and disable the use of BPR/BPD. Then, we select from amongst the five face-selection policies under consideration, namely, SSE, SAE, and gradient-, Laplacian-, and MMSODD-based HLoD. For each of the 40 images in our test set and six sampling densities per image (for a total of 240 test cases), we generated a mesh using each of the face-selection policies, and measured the approximation error of the reconstruction in terms of PSNR. Individual results for the three images in Table 3.2 are given in Table 3.3(a). In addition, the PSNR results obtained with each of the policies over the entire test set were ranked from 1 (best) to 5 (worst). Subsequently, the average and standard deviation of the ranks were computed across each sampling density as well as overall. The rankings are given in Table 3.3(b). The best result in each of the test cases is typeset in bold.

Examination of the statistical results shows that SSE clearly performs best with an average overall rank of 1.09, followed by SAE with an overall rank of 1.94. The remaining policies are on approximately equal footing, although the MMSODD and gradient policies do slightly better than the Laplacian policy. A more detailed analysis shows that SSE is best in 221/240 (92%) of the test cases. Since the goal is to minimize the approximation error which is measured using the squared error, it is not surprising that the error-based face-selection policies together (i.e., SSE and SAE) perform best in 238/240 (99%) of the test cases. Nevertheless, the idea of selecting faces in regions of the image with high levels of detail in order to partition them into smaller triangles appears to have been well founded. This is demonstrated by the fact that the PSNR for the gradient, Laplacian, and MMSODD policies is not far behind the results obtained with SSE and SAE in



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

Figure 3.8: Comparison of the subjective mesh quality obtained for the lena image at a sampling density of 1% with the (a) SSE and (c) SAE policies, and the corresponding meshes (b) and (d), respectively.

some test cases. The individual results shown in Table 3.3(a) are consistent with the statistical results. For example, SSE outperforms SAE in 16/18 (89%) of the test cases with a margin of 0.01 dB to 1.52 dB. We should also mention that the PSNR was found to correlate reasonably well with subjective image quality. Furthermore, all five policies have similar computational complexity so the quality of the approximation is the deciding factor. Consequently, we deem the SSE policy to be most effective and recommend its use in the framework.

Examples of the results obtained with each of the choices considered for the face-selection policy and the corresponding triangulations are shown in Figures 3.8 and 3.9.

Table 3.3: Comparison of the mesh quality obtained with the various choices of face-selection policies. (a) PSNR for three specific images. (b) Average rankings across the 40 images in the data set.

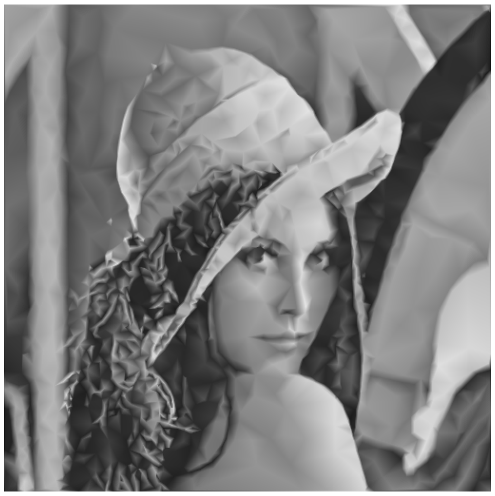
(a)

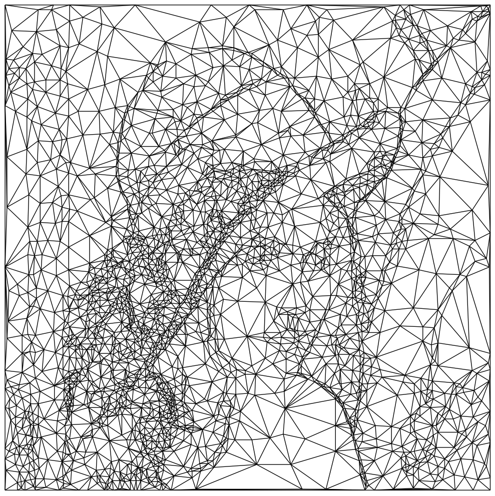
Image	**Sampling density (%)**	**PSNR (dB)**				
		SSE	SAE	Gradient	Laplacian	MMSODD
`bull`	0.125	**33.69**	33.19	31.51	31.50	31.56
	0.250	**38.36**	36.83	35.31	35.52	35.71
	0.500	**41.26**	39.81	39.74	38.50	39.98
	1.000	**43.17**	42.07	42.02	42.12	42.20
	2.000	**45.15**	44.19	44.10	44.46	44.46
	3.000	**46.56**	45.96	45.45	45.97	45.89
`ct`	0.125	**28.53**	28.51	27.81	27.43	27.67
	0.250	32.73	**33.02**	31.90	30.73	31.18
	0.500	**37.73**	37.31	36.31	36.11	36.25
	1.000	**41.40**	41.11	40.11	40.36	39.98
	2.000	**45.44**	45.13	43.94	44.41	44.24
	3.000	**47.97**	47.69	46.35	45.61	45.48
`lena`	0.125	21.41	**21.44**	20.90	20.30	20.19
	0.250	**23.99**	23.84	23.12	22.63	22.86
	0.500	**26.32**	25.94	25.55	25.05	25.25
	1.000	**28.87**	28.50	28.15	27.68	28.24
	2.000	**31.55**	31.21	30.96	30.58	31.05
	3.000	**33.01**	32.70	32.41	32.13	32.53

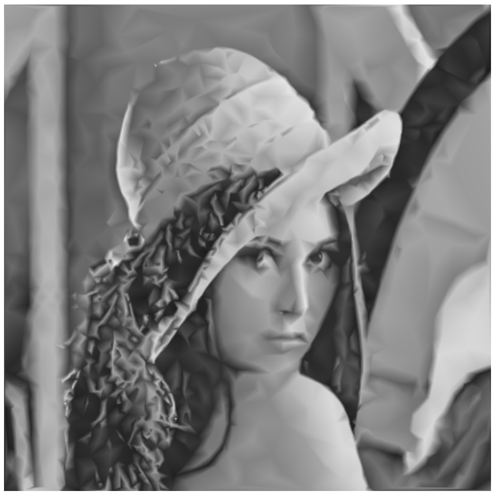
(b)

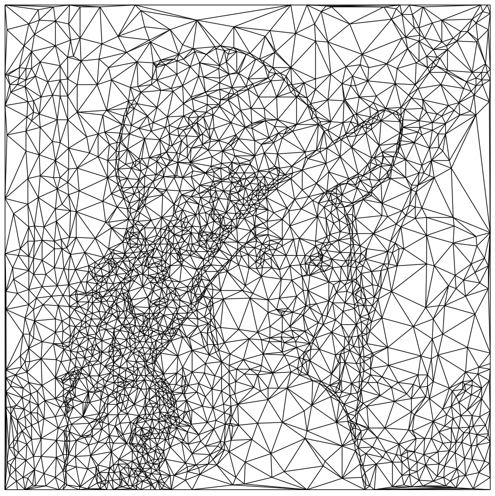
Sampling density (%)	**Mean Rank**[a]				
	SSE	SAE	Gradient	Laplacian	MMSODD
0.125	**1.28** (0.60)	1.95 (0.71)	4.03 (1.00)	4.05 (0.81)	3.70 (0.91)
0.250	**1.15** (0.36)	1.85 (0.36)	3.85 (0.83)	4.25 (0.84)	3.90 (0.74)
0.500	**1.08** (0.27)	1.95 (0.32)	3.65 (0.74)	4.50 (0.68)	3.83 (0.84)
1.000	**1.03** (0.16)	1.98 (0.42)	3.73 (0.85)	4.30 (0.72)	3.93 (0.86)
2.000	**1.00** (0.00)	1.98 (0.42)	3.95 (0.81)	4.18 (0.90)	3.83 (0.78)
3.000	**1.00** (0.00)	1.95 (0.32)	4.08 (0.73)	4.13 (1.04)	3.73 (0.78)
Overall	**1.09** (0.32)	1.94 (0.44)	3.88 (0.84)	4.23 (0.85)	3.82 (0.82)

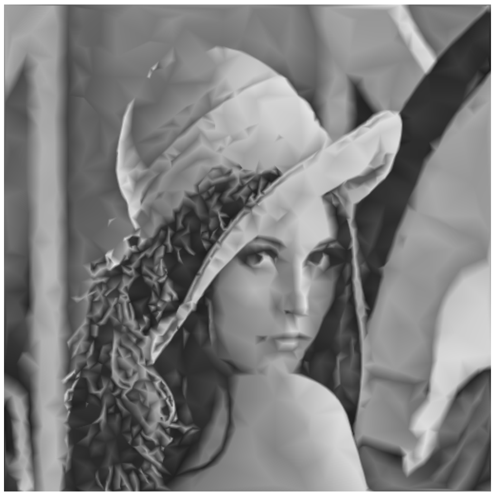
[a]The standard deviation is given in parentheses.


(a)


(b)


(c)


(d)


(e)

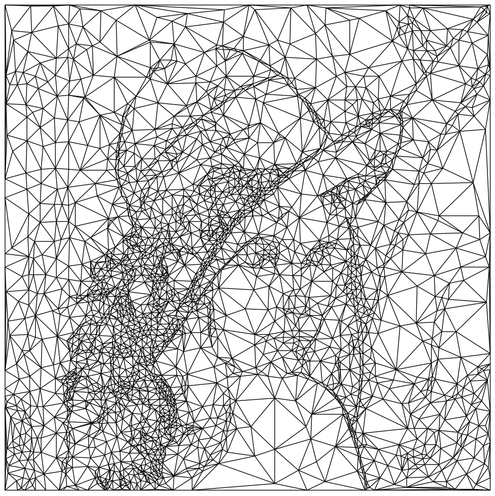

(f)

Figure 3.9: Comparison of the subjective mesh quality obtained for the lena image at a sampling density of 1% with the (a) gradient, (c) Laplacian, (e) and MMSODD policies, and the corresponding meshes (b), (d), and (f), respectively.

3.4.2 Candidate-Selection Policy

Now, we evaluate the choices for the candidate-selection policy in step 4 of the framework from Section 2.9, and how they affect performance. We fix the free parameters of the framework as follows: the extreme convex-hull points of the image domain for the initial mesh, SSE for the face-selection policy, growth schedule A$'$ with $L = 4$ and $A = 4$, and disable the use of BPR/BPD. Then, we select amongst the choices under consideration for the candidate selection policy which are PAE, ALSEM, gradient, Laplacian and MMSODD. The test cases used are the same as previously described: 40 images with six mesh densities per image between 0.125% and 3%. The mesh-quality results for individual test cases as well as the statistical results for the full data set are shown in Table 3.4. In each case, the best and second best results are typeset in bold and italic fonts, respectively.

We begin by examining the individual mesh-quality results obtained with each of the candidate-selection policies as given by Table 3.4(a). The ALSEM policy performs best in 15/18 (83%) of the test cases. For lower mesh densities, the margin by which the ALSEM policy outperforms the other choices can be quite large, even compared to the second-best performing choice. The difference in PSNR gradually decreases as the size of the target mesh increases. Based on the individual PSNR results, we cannot discern a clear choice for the second best performer. The PAE policy consistently ranks second for densities of 0.25% and lower, and is not too far behind the ALSEM policy for higher densities. On the other hand, the level-of-detail policies have mixed performance depending on the mesh density and the image used. Nevertheless, the MMSODD policy is relatively consistent compared to the gradient and Laplacian policies when it comes to mesh quality relative to the ALSEM policy.

Now, we examine the statistical ranking results in Table 3.4(a). The previous conclusion that the ALSEM policy performs best is confirmed by its average rank of 1.19 with a low standard deviation value of 0.60. The previous observation that the PAE policy performs acceptably at low densities only is confirmed by its average rank that gradually declines from 2.95, for test cases with a density of 0.125%, to 3.63 at 3%. The statistical results for the MMSODD policy show that it is consistently the second-best choice for mesh densities of 0.25% and higher. Its rank gradually improves as the density increases which was also observed in the individual results. Despite the clearly superior performance of the ALSEM policy, the recommendation of a choice for the candidate-selection policy is actually not quite as straightforward as it may appear. The reason is that the ALSEM policy has the downside of noticeably higher computational cost compared to the other choices. The selection of one candidate-selection policy for the framework is even more problematic when we consider the fact that, in certain cases, the MMSODD policy is able to achieve results that are very close or better than the ALSEM policy, with a typically much smaller computational cost. For these reasons, we recommend both choices and the task of selecting which one to use will fall upon the user depending on whether mesh quality or computational cost is more important.

3.4.3 Growth Schedule

Next, we examine how the choice of growth schedule affects performance. We fix the initial mesh to the extreme convex-hull points of the image domain, the face-selection policy to be SSE, the candidate-selection policy to be PAE, and disable the use of BPR/BPD. Then, we select from amongst the growth schedules under consideration, namely, I′, A′, B′, and C′. The growth schedules will normally have a significantly different number of operations if the growth schedule length and amplitude parameters, when applicable, are chosen the same. Consequently, we try to put the growth schedule on equal footing by fixing the amplitude parameter A to 0.5, and adjusting the length parameter L so that they have an approximately equal total number of operations. Growth schedule I′ has no parameters, but it does have the same target mesh size. The length value set for each growth schedule type and an example of the corresponding count of insertion and deletion operations for a test case with a target mesh size of 10004 are listed in Table 3.5. We can see that the individual as well as total operations counts are suitably close for growth schedules A′, B′, and C′, especially considering that we are constrained to an integer schedule length value.

Let us consider the statistical results for the rank of each of the growth schedules in Table 3.6(b). The clear winner is growth schedule A′ with an average rank of 1.02 across the 240 test cases. It is followed by growth schedules C′, B′ then I′, with ranks of 2.00, 2.88 and 3.94, respectively. The standard deviations are all very small for growth schedule A′ (i.e., less than or equal to 0.34), and only marginally higher for the other choices, confirming that these rankings are consistent across different images in the data set as well as the different mesh densities. Looking at the results in more detail, we find that growth schedule A′ performs best and second best in 235/240 (98%) and 4/240 (1.7%) of the test cases, respectively. The results for the individual test cases are shown in Table 3.6(a). The PSNR data in this table clearly validates the observations from the statistical data, with growth schedule A′ outperforming the B′, C′, and I′ growth

Table 3.4: Comparison of the mesh quality obtained with the various choices of candidate-selection policies. (a) PSNR for three specific images, and (b) average rankings across the 40 images in the data set.

(a)

Image	**Sampling density (%)**	**PSNR (dB)**				
		PAE	ALSEM	Gradient	Laplacian	MMSODD
`bull`	0.125	*33.55*	**34.60**	27.11	32.32	32.16
	0.250	*38.20*	**39.07**	31.88	36.89	37.43
	0.500	*41.53*	**42.37**	38.93	40.63	41.18
	1.000	43.50	**44.30**	43.81	42.91	*43.89*
	2.000	45.44	**46.13**	*46.05*	44.92	45.80
	3.000	46.87	**47.41**	*47.35*	46.33	47.08
`ct`	0.125	*28.16*	**28.62**	25.54	25.28	27.54
	0.250	*32.65*	**32.89**	28.37	30.00	32.54
	0.500	37.52	*37.78*	32.78	35.95	**37.80**
	1.000	41.33	*41.76*	38.85	41.31	**42.05**
	2.000	45.24	*45.68*	44.82	45.52	**45.74**
	3.000	47.95	**48.24**	47.71	48.08	*48.09*
`lena`	0.125	*21.79*	**22.57**	20.14	20.41	20.64
	0.250	*24.33*	**24.72**	22.25	22.71	23.39
	0.500	26.58	**27.08**	25.11	24.98	*26.64*
	1.000	28.88	**29.61**	27.98	27.87	*29.43*
	2.000	31.57	**32.20**	31.25	30.96	*32.15*
	3.000	33.16	**33.72**	33.25	32.68	*33.64*

(b)

Sampling density (%)	**Mean Rank**[a]				
	PAE	ALSEM	Gradient	Laplacian	MMSODD
0.125	2.95 (0.93)	**1.15** (0.58)	*2.90* (1.13)	4.73 (0.60)	3.28 (0.88)
0.250	3.30 (0.99)	**1.20** (0.61)	2.90 (1.17)	4.73 (0.75)	*2.85* (0.74)
0.500	3.48 (0.78)	**1.33** (0.89)	3.00 (1.15)	4.70 (0.82)	*2.43* (0.78)
1.000	3.53 (0.78)	**1.25** (0.71)	2.93 (1.14)	4.58 (1.08)	*2.20* (0.79)
2.000	3.60 (0.84)	**1.10** (0.30)	3.00 (1.04)	4.55 (1.11)	*2.00* (0.60)
3.000	3.63 (0.84)	**1.13** (0.33)	2.93 (1.00)	4.50 (1.18)	*2.08* (0.80)
Overall	3.41 (0.89)	**1.19** (0.60)	2.94 (1.10)	4.63 (0.94)	*2.47* (0.89)

[a]The standard deviation is given in parentheses.

Table 3.5: Length value used for evaluating growth schedules A′, B′, C′, and I′, and the associated number of operations with a target mesh size of 10000.

Growth schedule	Schedule length	Insertions count	Deletions count	Total count
I′	-	10000	0	10000
A′	9	17862	7767	25629
B′	17	17479	7566	25045
C′	5	17862	7767	25629

schedules by margins of 0.13–0.62 dB, 0.11–0.71 dB and 1.44–2.97 dB, respectively. The performance of growth schedule I′ is substantially worse than all of the other choices, and this was found to be generally true regardless of the number of operations growth schedule I′ is able to perform. This is somewhat expected considering growth schedule I′ exclusively performs either point insertions or point deletions, which is known to be less effective. Since growth schedule A′ is the clear winner, we recommend it for the growth-schedule free parameter of the framework.

3.4.4 Growth Schedule Length and Amplitude

The choice of the length and amplitude parameters of growth schedules A′, B′, and C′ is best considered jointly since the two parameters are interdependent. The initial mesh is fixed to the extreme convex-hull points of the image domain, the face-selection policy to SSE, the candidate-selection policy to MMSODD, the growth schedule to A′ and disable the use of BPR/BPD. The growth schedule length L and amplitude A parameters are then varied. For reasons of simplicity, we will focus on one particular test case of the lena image with a target mesh density of 1%. We note, however, that the trends observed were found to be consistent across the entire data set at the various densities. In Figure 3.10, each test case with a particular combination of values for L and A is indicated by a marker (i.e., square, circle, triangle, and asterisk). The x-coordinate of a marker corresponds to the value of L used in the test case that is represented by the marker. In Figure 3.10(a), the y-coordinate of a marker corresponds to the PSNR of the reconstruction of the mesh generated in the test case represented by the marker. In Figure 3.10(b), the y-coordinate corresponds to the total execution time for the test case represented by the marker. In both figures, test cases that are represented by the same marker symbol (e.g., asterisk) are those for which A was set to the same value. The lines connecting the markers are

Table 3.6: Comparison of the mesh quality obtained with growth schedules I′, A′, B′, and C′. (a) PSNR for three specific images. (b) Average rankings across the 40 images in the data set.

(a)

Image	**Sampling density (%)**	**PSNR (dB)**[a]			
		A′ (9)	B′ (17)	C′ (5)	I′
`bull`	0.125	**33.49**	33.34	33.37	30.74
	0.250	**37.84**	37.44	37.57	35.29
	0.500	**40.81**	40.46	40.57	38.77
	1	**42.76**	42.31	42.52	41.07
	2	**44.78**	44.30	44.49	43.07
	3	**46.27**	45.69	45.96	44.54
`ct`	0.125	**28.24**	27.64	27.79	25.27
	0.250	**32.52**	32.39	31.80	30.00
	0.500	**37.61**	37.19	37.33	35.18
	1.000	**41.34**	41.06	41.22	39.67
	2.000	**45.25**	44.98	45.13	43.79
	3.000	**47.88**	47.54	47.74	46.30
`lena`	0.125	**21.04**	20.61	20.79	19.01
	0.250	**23.47**	22.85	23.16	21.66
	0.500	**25.9**	25.50	25.78	24.27
	1.000	**28.49**	27.98	28.32	26.87
	2.000	**31.32**	30.80	31.04	29.75
	3.000	**32.77**	32.29	32.51	31.33

[a]The schedule length is given in parentheses next to the schedule type.

(b)

Sampling density (%)	**Mean Rank**[a]			
	A′	B′	C′	I′
0.125	**1.05** (0.22)	2.95 (0.31)	3.98 (0.15)	3.98 (0.15)
0.250	**1.07** (0.34)	2.90 (0.37)	4.00 (0.00)	4.00 (0.00)
0.500	**1.02** (0.15)	2.90 (0.37)	4.00 (0.00)	4.00 (0.00)
1.000	**1.00** (0.00)	2.86 (0.47)	4.00 (0.00)	4.00 (0.00)
2.000	**1.00** (0.00)	2.83 (0.54)	3.86 (0.65)	3.86 (0.65)
3.000	**1.00** (0.00)	2.86 (0.52)	3.79 (0.78)	3.79 (0.78)
Overall	**1.02** (0.18)	2.88 (0.44)	2.00 (0.30)	3.94 (0.42)

[a]The standard deviation is given in parentheses.

merely added to help with visualization of the trends. That is, the lines do not represent continuity in the data and they are not necessarily indicative of expected results for test cases that are in-between data markers.

By examining Figure 3.10(a), we can make several observations. First, for a fixed growth schedule amplitude A, increasing the growth schedule length L typically leads to an increase in the PSNR of the approximation. We obtain diminishing returns for the mesh quality, however, as the rate of increase of the PSNR that results from a higher value of L diminishes. For example, the gradients (i.e., rate of increase of the PSNR) for $L = 8$ are typically steeper than the gradients for $L = 35$. Furthermore, the rate of increase of the PSNR resulting from increasing the value of L decreases with A. That is, for a given L, the gradients are steeper with smaller values of A. For example, the gradients for $A = 0.5$ are much higher than they are for $A = 2$ throughout the range of L in our test cases. The second observation is that, for a fixed L, increasing A typically improves the PSNR. As with the first observation, we have diminishing returns and the benefits are smaller the greater the value of L. Increasing A from 0.5 to 2, for instance, leads to a greater increase in PSNR when $L = 4$ than it does when $L = 40$.

Now looking at Figure 3.10(b), we see that computational cost increases with A and L. The rate of increase of the computational cost when L is varied depends on the value of A. For example, the gradient (i.e. rate of increase of the execution time) for $L = 35$ is higher with $A = 3$ than it is with $A = 0.5$. A similar observation can be made regarding the effect of changing A.

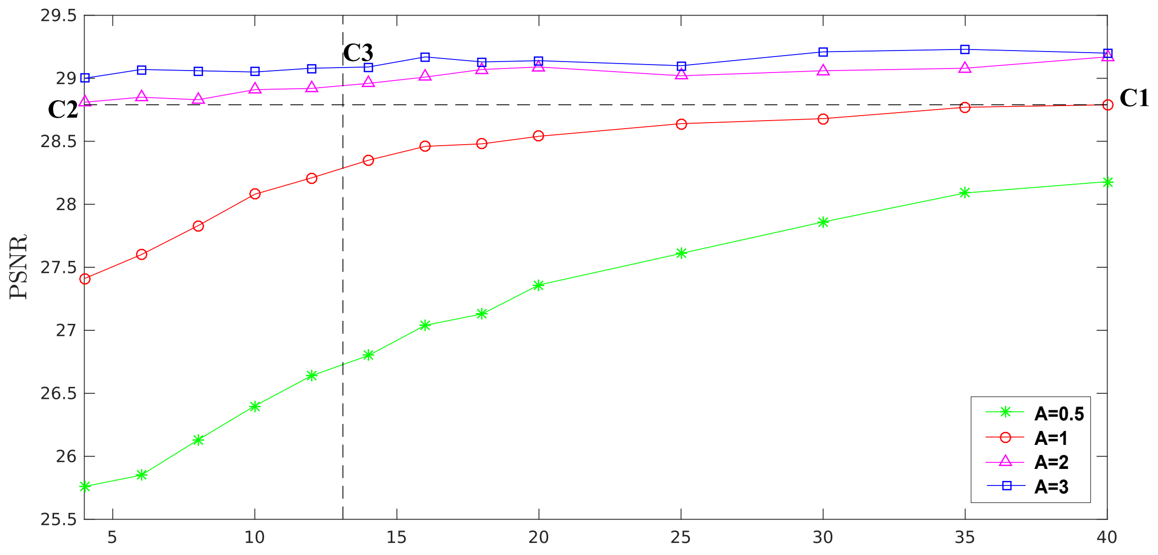
When both figures are considered together, we can make the observation that computational cost keeps increasing as the two parameters are increased, while the PSNR eventually saturates. Consequently, the ratio of computational cost to PSNR of the generated mesh will vary greatly depending on the choice of the parameters. For instance, let us examine the test case where $A = 1$ and $L = 40$, we will refer to it as C1. The mesh generated has a PSNR of 28.79 dB with an associated execution time of 5.88s. In the PSNR plot (i.e. Figure 3.10(a)), using the horizontal dashed line as a guide, we can see that a mesh with a very similar PSNR (28.81 dB) to that of C1 can be obtained using a $L = 4$ and $A = 2$. We will refer to this second case as C2. Now, let us examine the timing plot (i.e. Figure 3.10(b)). We find that C2 has a computational cost of 2.75s. Consequently, we are able to achieve a similar mesh quality to C1 in less than half the computation time of C1 by using different values for the parameters of the growth schedule. In some instances, a higher mesh quality is of more importance than reducing computational cost. Using the horizontal and vertical dashed lines as guides, we can also see that for the case C3 where the $A = 3$ and $L = 13$, a higher quality mesh associated with a PSNR of 29.09 dB can be obtained.

The reason for these results is that grouping point insertions together and point deletions together is typically more effective than frequently alternating between the two. The specific cause is an implementation detail into which we will not delve, but an explanation can be found in the paper introducing the framework [20]. The observations are crucial and are used to our advantage in deciding on what particular values should be chosen for the parameters. Based on further experimentation, we determined that, for growth schedule A′ which we recommended earlier, and parameter values of $A = 3$ and $L = 4$ or $L = 6$ generally provide a good compromise between mesh quality and performance.

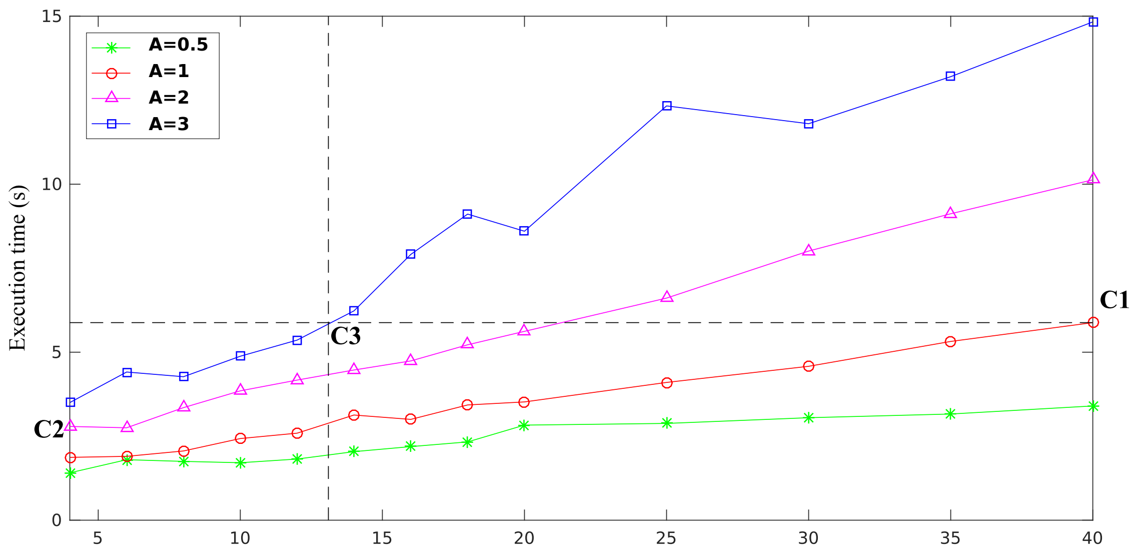
3.4.5 Initial Mesh

In what follows, we consider the effectiveness of the various choices of the initial mesh in step 1 of the mesh-generation framework from Section 2.9. For this purpose, we perform an experiment. In this experiment, the face-selection policy is fixed to SSE, the candidate-selection policy to PAE, the growth schedule to A′ with $A = 3$ and $L = 4$, and the use of BPR/BPD is disabled. Subsequently, we select amongst the choices of initial mesh point selection policies. A gradient-based radii map is selected for use with adaptive Poisson disk sampling following a preselection process. For the three images in our representative test set and six sampling densities per image, we generated a mesh using each of the eight policies considered for the selection of the initial mesh points, and then we measured the approximation error of the reconstruction in terms of PSNR. The mesh-quality results for individual test cases are presented in Table 3.7.

By examining the results in Table 3.7, we observe that the variation between the PSNR values is relatively small. In fact, if we exclude the results of the adaptive Poisson-disk sampling choice for reasons that we will explain shortly, the difference between the PSNRs of the highest and worst performing choice has median and average values of 0.22 dB and 0.23 dB, respectively. This shows that the effect of the choice of initial mesh on the quality of the results is not as significant as that of the choices for other free parameters. In fact, choices that select the initial points randomly, or based on a uniform or jittered grid in a non-adaptive way, often perform similar to adaptive point selection such gradient-based choices. As long as the points are reasonably scattered across the



(a)



(b)

Figure 3.10: Comparison of the effect of L and A on (a) the PSNR of the approximation and (b) the execution time.

Table 3.7: Comparison of the mesh quality obtained for three specific images with the various choices of initial mesh considered.

Image	**Samp. density (%)**	**PSNR (dB)**							
		ED	Subset[a]			Random	Uniform Grid	Jittered Grid	Poisson Disk
			G.	L.	M.				
`bull`	0.125	33.75	33.70	33.46	33.51	33.46	33.48	33.59	**33.81**
	0.250	**38.45**	38.21	38.36	38.31	38.18	38.39	38.27	38.00
	0.500	41.55	41.51	**41.61**	41.55	41.53	41.60	41.51	40.79
	1.000	43.55	43.45	43.43	43.49	43.55	43.55	**43.56**	42.41
	2.000	45.44	45.39	45.35	45.40	45.45	45.40	45.43	**45.51**
	3.000	**46.92**	46.81	46.82	46.79	46.86	46.85	46.82	46.89
`ct`	0.125	28.52	28.48	**28.86**	28.38	28.46	28.32	28.55	28.14
	0.250	32.64	32.78	**32.86**	32.68	32.64	32.67	32.73	32.84
	0.500	37.30	37.35	37.51	37.41	37.50	37.50	37.27	**37.54**
	1.000	41.27	41.35	41.28	41.26	**41.48**	41.44	41.43	41.41
	2.000	45.37	**45.41**	45.24	45.28	45.27	45.36	45.31	45.00
	3.000	**47.87**	47.86	47.82	47.84	47.85	47.84	47.85	47.37
`lena`	0.125	21.77	21.46	21.68	21.79	21.84	21.75	21.63	**22.16**
	0.250	24.17	23.98	24.18	24.04	**24.36**	24.21	24.29	**24.36**
	0.500	26.53	26.34	26.45	26.54	26.60	26.53	26.61	**26.73**
	1.000	28.99	28.69	29.01	29.01	29.05	29.01	29.03	**29.24**
	2.000	31.69	31.55	31.62	31.65	31.62	31.70	**31.72**	31.58
	3.000	33.24	33.05	33.13	33.06	33.20	**33.25**	33.13	32.88

[a]G., L. and M. stand for the gradient, Laplacian and MMSODD-based variants, respectively.

image domain, performance is typically good. In passing, we note that these observations are valid for growth schedules with large amplitudes and small length values such as those used in the test cases in this section. The impact of the initial mesh varies with the specific growth schedule used, and generally the choice of initial mesh does affect performance. In the case of adaptive Poisson-disk sampling, no direct mechanism exists for controlling the number of points selected in the implementation used in our work. In the test cases of Table 3.7, the number of samples selected by adaptive Poisson disk sampling were typically much larger than those generated by the other choices of initial mesh policies despite careful tuning of its free parameters. Furthermore, the computational cost of Poisson disk sampling is prohibitive even in the cases where it performs best, resulting in execution time penalties of up to an order of magnitude. This leads us to recommend Floyd-Steinberg error diffusion for the selection of the initial mesh points, as we found it during additional testing to be the most consistent across different image types and densities.

3.4.6 Postprocessing

In what follows, we evaluate the impact of choosing the bad point deletion (BPD) technique for step 7 of the mesh-generation framework. To do this, we perform an experiment where we fix the free parameters of the framework as follows: face-selection policy to SSE, the candidate-selection policy to MMSODD, the growth schedule to A$'$ with $A = 4$ and $L = 4$. For all 40 images in our test set and for six sampling densities (totalling 240 test cases), we generated a mesh both with and without BPD. Then, we computed the difference, in terms of PSNR, between the approximation error of the reconstruction for each test case with BPD and the corresponding test case without it. For each density as well as the overall results, we calculate statistics relating to the percentage of cases that BPD increases the quality of the mesh, in addition to the minimum, median, and maximum difference values. The results are shown in Table 3.8.

By examining the results, we find that BPD typically only changes the PSNR of a small proportion of the test cases. Although not immediately clear, we note that BPD affects fewer of the test cases as the target mesh density increases. Furthermore, the extent of change generally also decreases as mesh density increases. The results are somewhat to be expected with the growth schedule of type A$'$ that we are seeing as the growth schedule always ends with a series of points deletions. Bad points, by definition, have a negative deletion significance value (i.e., sigDel < 0). Consequently, they are the first to be removed from the mesh when a sequence of point deletions starts. As the density or A increases, the last sequence of point deletions becomes longer, which decreases the likelihood that bad points are left when the mesh-generation framework reaches the step where BPD is applied.

We evaluate the effect of enabling bad point replacement (BPR) on mesh quality using an experimental procedure similar to that used for BPD. We fix the free parameters of the framework as follows: face-selection policy to SSE, the candidate-selection policy to MMSODD, the growth schedule to A$'$ with $A = 4$ and $L = 4$. For each of the 40 images in our test set and for six sampling densities, we generated a mesh both with and without BPR. Then, we computed the PSNR differences between the approximation error of the reconstructions for the test cases with BPR and those without it. For each density as well as the overall results, we calculate statistics relating to the percentage of test cases where the difference is not zero (i.e., the cases where BPR leads to a change in the mesh quality), in addition to the minimum, median, and maximum difference. The statistical results are shown in Table 3.9,

Table 3.8: Statistical results for the change in PSNR when bad point deletion (BPD) is applied.

Sampling density (%)	**PSNR Change**			
	Incidence (%)	Minimum (dB)	Median (dB)	Maximum (dB)
0.125	30.0	0.0005	0.0042	0.1469
0.250	27.5	0.0006	0.0031	0.4043
0.500	30.0	0.0002	0.0016	0.0192
1.000	25.0	0.0001	0.0009	0.039
2.000	22.5	0.0001	0.0002	0.0192
3.000	22.5	0.0001	0.0002	0.0008
Overall	26.25	0.0001	0.0012	0.4043

A first observation is that, for some test cases, the use of BPR leads to a decrease in the PSNR of the approximation. This is the result of inserting new points, which is not guaranteed to decrease the error in BPR. Furthermore, we note that the number of test cases for each mesh density as well as overall where the PSNR changed after applying BPR is higher than with BPD even though both apply to the same test cases (i.e., test cases that have bad points). The reason is that BPD only removes points that do not lead to a decrease in PSNR. As a result, it changes the PSNR of a smaller subset of the test cases to which it is applied. Looking at the minimum, median, and maximum of the PSNR change, we can see that the impact of BPR on the quality of the approximation typically reduces as the mesh density increases. This is due to the fact that a single point tends to have a lower incidence on the overall mesh connectivity and, subsequently, the approximation quality as the size of the mesh increases. Finally, we observe that, for test cases where the mesh quality improves, the PSNR increase is typically higher than with BPD. Given that the increase in mesh quality in many test cases is quite significant relative to the potential decrease in other test cases, we recommend the use of the BPR technique in the postprocessing step of the framework.

3.4.7 Image Filter

A smoothing filter needs to be applied to the image before the computation of the MMSODD, gradient and Laplacian to reduce the effect of noise. The choice of filter simultaneously affects error diffusion used for selecting the initial mesh, as well as the MMSODD-based and ALSEM candidate-selection policies. In what follows, we evaluate

Table 3.9: Statistical results for the change in PSNR when bad point replacement (BPR) is applied

Sampling density (%)	**PSNR Change**			
	Incidence (%)	Minimum (dB)	Median (dB)	Maximum (dB)
0.125	35	-1.1696	-0.0904	2.1261
0.250	37.5	-0.4102	-0.0615	7.9005
0.500	35	-0.1953	-0.0201	1.0997
1.000	45	-0.659	-0.0218	0.0131
2.000	75	-0.1238	0.0002	0.0202
3.000	37.5	0.0004	0.0040	0.0127
Overall	44.17	-1.1696	-0.00625	7.9005

the impact of the choice of filter as well as filter parameters. To this effect, we perform an experiment. In this experiment, we fix the candidate-selection and the face-selection policies to be PAE and SSE, respectively, the growth schedule to be A$'$ with $A = 3$ and $L = 4$, and disable the use of BPD. We select amongst the choices of filter type as well as filter parameters, and generate a mesh for each of the 40 images in our data set with six sampling densities. We evaluate in terms of PSNR the approximation error of the reconstruction of the mesh generated in each of the test cases. For the binomial filter, we vary the filter order, and for the bilateral filter we vary the range and space parameters simultaneously. A preliminary selection round was used to select a subset of parameters to consider for each filter type. Binomial filters of orders 5, 7, 9, 11, and 13 were evaluated from which the values 5, 7, and 9 were selected. Similarly, nine combinations of parameter values for the bilateral filter were evaluated and narrowed down to three. The subset of results selected for each filter were compiled into a single list, ranked from 1 (best) to 6 (worst) for each test based on the PSNR, and then the ranks were averaged. The ranking results for each density and overall are presented in Table 3.10.

Upon examination of the results, we notice that the binomial filters consistently outperform the bilateral filters in all the test cases. This is the case for the averages over all images as well as per density. The binomial filters have similar overall average ranks and standard deviation values. Looking at the results in more detail, we are not able to discern any clear trends. We can conclude that there is no consistent best choice, and an order value in the range 5–9 for the binomial filter should perform reasonably well in most cases. Since the binomial filter of order 7 ranks slightly better than the other choices, we recommend it as the smoothing filter in our work.

Table 3.10: Statistical comparison of the choices for the filter type and parameter values

Samp. density (%)	**Mean Rank**[a]					
	Binomial[b]			Bilateral[c]		
	5	7	9	0.10	0.25	0.25
				12	12	18
0.125	2.43 (1.35)	2.75 (1.45)	2.55 (1.36)	4.73 (1.32)	3.93 (1.61)	4.63 (1.46)
0.250	2.65 (1.08)	2.15 (1.37)	2.25 (1.19)	4.75 (1.24)	4.30 (1.42)	4.90 (1.22)
0.500	2.45 (1.50)	2.55 (1.30)	2.80 (1.40)	4.48 (1.30)	4.10 (1.85)	4.63 (1.31)
1.000	2.55 (1.57)	2.80 (1.43)	2.75 (1.45)	4.65 (1.41)	4.03 (1.94)	4.15 (1.33)
2.000	2.85 (1.29)	2.38 (1.69)	3.00 (1.59)	4.35 (1.66)	4.05 (1.92)	3.78 (1.59)
3.000	2.75 (1.78)	2.75 (1.35)	2.55 (1.43)	4.15 (1.73)	4.10 (2.05)	3.70 (1.52)
Overall	2.61 (1.45)	2.56 (1.43)	2.65 (1.41)	4.52 (1.45)	4.08 (1.80)	4.30 (1.47)

[a]The standard deviation is given in parentheses.

[b]5, 7, and 9 are the binomial filters' orders

[c]0.10 and 0.25 are values of the range parameter. 12 and 18 are values of the space parameter.

3.5 Proposed Methods

In the preceding sections, we studied how various choices of the free parameters of the mesh-generation framework affect performance. With the insight obtained from that analysis, we are able to introduce the two mesh-generation methods proposed herein. The growth schedule and face-selection policy A′ and SSE, respectively, were consistently found to be the most effective choices compared to the alternatives. Hence, unsurprisingly, they have been selected for the methods we are proposing. The recommendation of the ALSEM and MMSODD candidate-selection policies as two choices with different trade-offs between mesh quality and computational cost motivates us to propose two methods. The first method called IID1 uses the lower complexity MMSODD candidate-selection policy, and the second method called IID2 uses the higher complexity ALSEM policy. The IID1 method aims for faster execution times and so the values of its A′ growth schedule parameters A and L were chosen as $A = 3$ and $L = 4$. The higher complexity IID2 method, which prioritizes mesh quality, is identical to the IID1 method, except that it uses $L = 6$ and the ALSEM policy. Since in the case of an even L, the growth schedule A′ has one setpoint appended to it to reach the actual mesh size desired, a value of $L = 6$ is effectively only one step up from $L = 4$. As for the remainder of the free choices for the two methods, Floyd-Steinberg error diffusion is used to select the initial mesh points with a density of 1%, a 7th-order binomial filter is selected for smoothing prior to the MMSODD calculation, and the use of the bad point replacement technique is enabled.

Chapter 4

Evaluation of the Proposed Methods

4.1 Overview

4.2 Evaluation of the Proposed Methods

Having introduced the proposed IID1 and IID2 mesh-generation methods, we now evaluate their performance in terms of mesh quality and computational cost by comparing them to other well-known schemes. The implementations used for this purpose were developed by the author of this thesis and were written in C++. (For more details about the software, see Appendix A). Our performance comparison focuses, in particular, on the GPR, IDDT, ID1, and ID2 methods because they produce Delaunay meshes and use a linear interpolant, similar to our methods. Furthermore, the other methods can be easily derived from the same mesh-generation framework used herein by choosing the free parameters appropriately. This has the benefit of ensuring that the comparison that we are performing is fair. In passing, we note that the ID1 and ID2 methods have a parameter α for adjusting the length of the growth schedule, thereby increasing or decreasing the computational cost with a corresponding increase or decrease in mesh quality. In our comparison, we use $\alpha = 0.4$ which is the value advocated in [21]. We also note that, by comparing the performance of our methods against that of the state-of-the-art IDDT, ID1, and ID2 schemes, we are also indirectly showing that they outperform schemes such as the MGH [71], OMGH [20], ED [36], OED [20] and GPRFS-ED [42] methods, to which the IDDT, ID1, and ID2 methods have been shown to be superior [20, 21].

4.3 Mesh Quality

For mesh quality evaluation, we employ the full set of 50 images as introduced earlier in Section 3.3 with seven sampling densities (between 0.125% and 4%) per image for a total of $50 \times 7 = 350$ test cases. We used each of the methods under consideration to generate a mesh model for each of the images in our data set. Then, we measured the approximation error of the mesh reconstruction in terms of PSNR. The individual results for a representative subset of the images (namely, those listed in Table 3.2) are shown in Table 4.1(a). For each of the 350 test cases, the PSNR performance of the six methods (i.e. IID1, IID2, ID1, ID2, IDDT, GPR) was ranked from 1 (best) to 6 (worst). The average and standard deviation were then calculated for each density as well as overall with the result presented in Table 4.1(b). The best and second best result in each case are typeset in bold and italic, respectively.

We begin by comparing the IID1 and IID2 methods to the IDDT scheme. The overall statistical results in Table 4.1(b) show that the IID1 and IID2 methods outperform the IDDT scheme at all sampling densities. In fact, the IDDT method has the worst average rank of 5.69 amongst all six methods. Looking at the full results in more detail, we find that the IID1 and IID2 methods beats the IDDT scheme in 332/350 (95%) and 342/350 (98%) of the test cases, respectively. By examining the results for the individual test cases in Table 4.1(a), we see that the IID1 method beats the IDDT scheme in 19/21 of the test cases by a margin of 0.04–1.31 dB, and the IID2 method beats the IDDT scheme in 21/21 of the test cases by 0.33–2.36 dB. This is consistent with the overall statistical results.

Next, we compare the performance of the proposed methods to the GPR method. Examining the statistical results in Table 4.1(b), we observe that the IID2 method outperforms the GPR method consistently across all of the sampling densities, whereas the IID1 method ranks similarly or better on average than the GPR scheme for target densities of 0.5% and higher. More detailed analysis of the results shows that the IID2 method beats the GPR method in 346/350 (99%) of the test cases. The IID1 method outperforms the GPR scheme in 221/350 (63%) of the test cases across all densities. As expected, the performance of the IID1 method improves considerably for higher densities. It is able to beat the GPR method in 117/150 (78%) of the test cases for densities of 2% and higher. Now, we consider the individual results in Table 4.1(a). We see that these individual results are consistent with the overall statistical results. The IID2 method outperforms the GPR scheme in all cases, while the IID1 method has mixed results in comparison but

Table 4.1: Comparison of the mesh quality for the various methods. (a) PSNRs for three specific images. (b) Rankings averaged across 50 images.

(a)

Image	**Sampling density (%)**	**PSNR (dB)**					
		IID1	IID2	ID1	ID2	IDDT	GPR
`bull`	0.125	31.14	34.18	**34.90**	*34.56*	33.85	33.51
	0.250	37.63	**39.15**	*38.87*	38.76	37.51	38.18
	0.500	41.45	**42.24**	41.84	**42.24**	40.42	41.89
	1.000	43.81	*44.22*	43.91	**44.27**	42.50	43.97
	2.000	45.73	*46.09*	45.79	**46.13**	44.46	45.83
	3.000	47.08	**47.37**	47.15	*47.37*	45.78	47.14
	4.000	48.23	**48.44**	48.26	*48.44*	46.97	48.24
`ct`	0.125	27.71	**28.88**	*28.62*	28.60	27.52	28.22
	0.250	32.30	*33.09*	**33.27**	32.99	32.43	32.38
	0.500	37.77	37.87	**38.20**	*37.88*	37.44	37.44
	1.000	**42.07**	41.79	*41.97*	41.74	41.37	41.45
	2.000	45.62	45.59	**45.83**	*45.69*	45.25	45.32
	3.000	47.96	48.10	**48.30**	*48.17*	47.74	47.88
	4.000	49.91	49.99	**50.16**	*50.07*	49.63	49.80
`lena`	0.125	20.43	**22.76**	22.03	*22.50*	20.39	22.08
	0.250	23.73	**24.90**	24.68	*24.84*	23.18	24.38
	0.500	26.75	**27.19**	26.93	*27.10*	25.82	26.59
	1.000	29.40	*29.58*	29.44	**29.62**	28.46	29.09
	2.000	32.10	**32.22**	32.15	*32.17*	31.05	31.78
	3.000	33.63	**33.73**	33.59	*33.64*	32.50	33.37
	4.000	34.66	34.71	34.62	**34.72**	33.49	34.42

(b)

Sampling density (%)	**Mean Rank** [a]					
	IID1	IID2	ID1	ID2	IDDT	GPR
0.125	4.92 (0.57)	**1.36** (0.94)	3.72 (0.86)	2.08 (0.57)	5.64 (1.16)	3.28 (0.83)
0.250	4.74 (0.69)	**1.38** (0.81)	3.68 (1.04)	2.06 (0.68)	5.70 (1.04)	3.44 (0.93)
0.500	4.02 (0.84)	**1.42** (0.84)	3.70 (1.36)	2.04 (0.70)	5.78 (0.91)	4.02 (0.94)
1.000	3.58 (0.95)	**1.38** (0.67)	3.46 (1.28)	2.02 (0.74)	5.70 (1.20)	4.46 (0.99)
2.000	3.12 (0.77)	**1.32** (0.65)	3.38 (1.18)	1.96 (0.73)	5.68 (1.20)	4.64 (1.05)
3.000	3.10 (0.81)	**1.40** (0.83)	3.22 (1.23)	1.98 (0.65)	5.68 (1.20)	4.70 (0.99)
4.000	3.18 (0.80)	**1.38** (0.67)	3.20 (1.26)	1.96 (0.67)	5.68 (1.20)	4.70 (0.99)
Overall	3.81 (1.06)	**1.38** (0.77)	3.48 (1.19)	2.01 (0.67)	5.69 (1.13)	4.18 (1.11)

[a]The standard deviation is given in parentheses.

typically performs better at higher densities and with medical and photographic images. Such performance from the IID1 and IID2 methods is particularly impressive considering that, as we will see later, they are noticeably less computationally expensive than the GPR method.

Now, let us compare the IID1 and IID2 methods to the ID1 and ID2 methods. To do this, we examine the statistical results in Table 4.1(b). The IID2 method is clearly best overall with rank 1.38 and a relatively small standard deviation of 0.77. The competing ID2 method is second best with rank 2.01 and a small standard deviation as well (0.67). The results for the IID2 and ID2 methods are consistent across all sampling densities with very little deviation from the mean. The ID1 method is third best overall followed closely by the IID1 method in fourth position, with average ranks of 3.48 and 3.81, respectively. The performance of the ID1 and IID1 methods varies with sampling density. The average ranking of both is typically worse at low densities and then gradually improves as the mesh density increases. The IID1 method, however, has a larger spread of ranking values across the range of mesh densities, going from noticeably worse than the ID1 method at 0.125% to performing better than the ID1 or equally well on average at sampling densities of 2% and higher. A more detailed analysis of the results shows that the IID2 method beats the ID1 and ID2 methods in 310/350 (89%) and 281/350 (80%) of the test cases, respectively. The IID1 method performs better than the ID1 method in 165/350 (47%) of the test cases, which increases to 65% for densities of 2% and higher. The individual PSNR results in Table 4.1(a) are consistent with the conclusions from the statistical analysis. We also note that, although the ID2 method beats our IID2 method in some test cases, the difference is typically small. In 8/10 (80%) of the test cases where the ID2 method beats our IID2 method, the difference in PSNR is 0.08 dB or less. We have found this to be a consistent occurrence across the test cases.

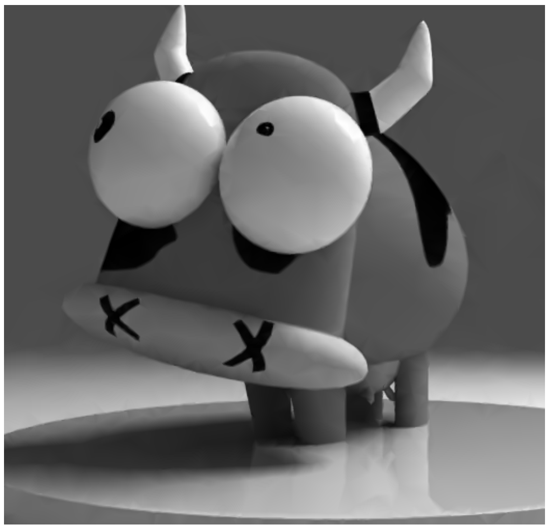
Finally, comparing the individual results of our methods in Table 4.1(a), we see that the IID2 method outperforms the IID1 scheme in 19/21 (90%) of the test cases. This is consistent with the statistical results in Table 4.1(b). Nevertheless, the IID1 method is able to beat, or comes very close to, the IID2 scheme at times. Examining the results in more detail, we find this to be the case in 20/350 (6%) of the test cases. This happens with images that have strong edges, such as photographic images, at densities of 1% and higher, which highlights one of the strengths of the MMSODD candidate-selection policy. Overall, we have found that one of our two method performs best out of all methods that were considered in our evaluation in 263/350 (75%) of the test cases.

In the above evaluation, PSNR was found to correlate reasonably well with subjective quality. For the benefit of the reader, however, we include some examples illustrating the subjective quality achieved by the various methods. The examples are chosen to contain a variety of image types. For each example, we show a small part of the image reconstruction under magnification as well as the corresponding triangulation. Since the IID2 and ID2 methods are higher complexity methods, while the IID1 and ID1 methods are lower in complexity, the examples are grouped together according to their complexity for easier comparison. We begin by examining one of the test cases from Table 4.1(a), specifically the computer-generated bull image with a sampling density of 0.5% shown in Figures 4.1 and 4.2. We can clearly see that the image approximations produced by our IID1 and IID2 methods are similar to those produced by the ID1 and ID2 methods, respectively. The PSNR difference of 0.39 dB between the approximations of the IID1 and ID1 methods are barely perceptible and are less significant than one would expect from the PSNR difference alone. Furthermore, the quality of the image approximations produced by the IID2 and ID2 methods are also clearly superior to those of the IID1 and ID1 methods, with smoother gradients and sharper, more accurate contours.

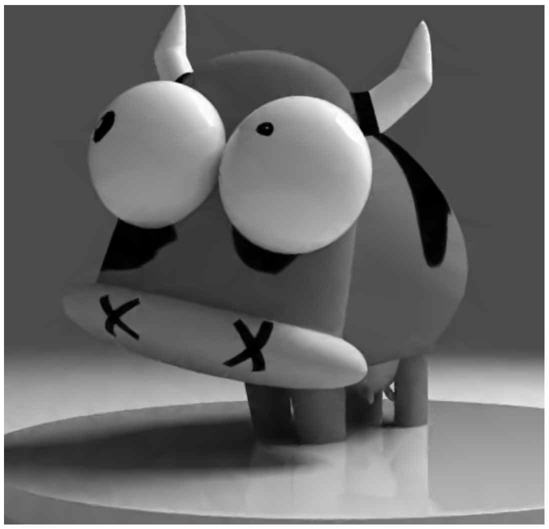
Figures 4.3 and 4.4 show another set of examples which are for the photographic lena image with a sampling density of 2%. The PSNR values for these test cases are very close, with a difference of 0.12 dB between the worst performing IID1 method and the best performing IID2 method. The image approximations are visually similar overall. Upon closer inspection, however, we notice slightly higher quality with the higher complexity IID2 and ID2 methods over the other two methods. The gradients over the skin are smoother and the contours are preserved better. We also observe that the approximation generated using our IID1 method, compared to the ID1 method, is able to capture facial features slightly more accurately despite being marginally worse by 0.05 dB in terms of PSNR. Consequently, we can say that the approximations produced by the IID1 method are typically visually similar to those produced using the ID1 scheme.

4.4 Computational Cost

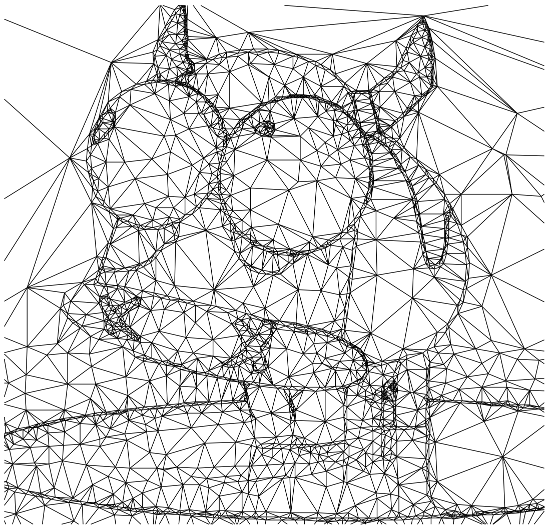
Next, we briefly evaluate the computational cost (i.e. execution time) of the methods in our comparison. For this purpose, we provide a representative subset of timing measurements collected using a 5 year old laptop with a 2.4GHz Intel Core i7 processor and 4GB of RAM. For the images in Table 3.2 and six sampling densities per image, the time required for mesh generation for each of the methods was measured. Five runs were performed in each case with the execution time for each one measured individually. Then,



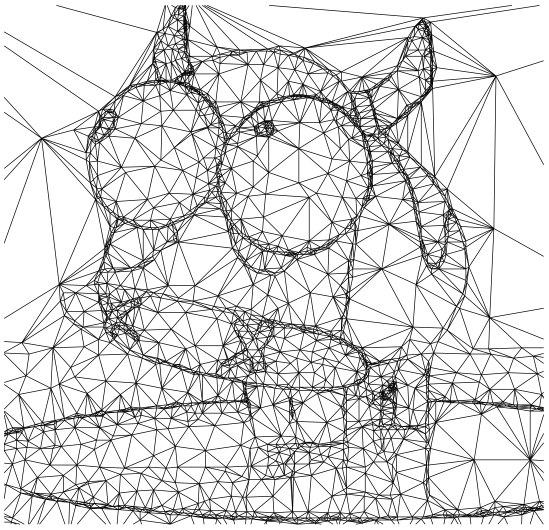
(a)



(b)

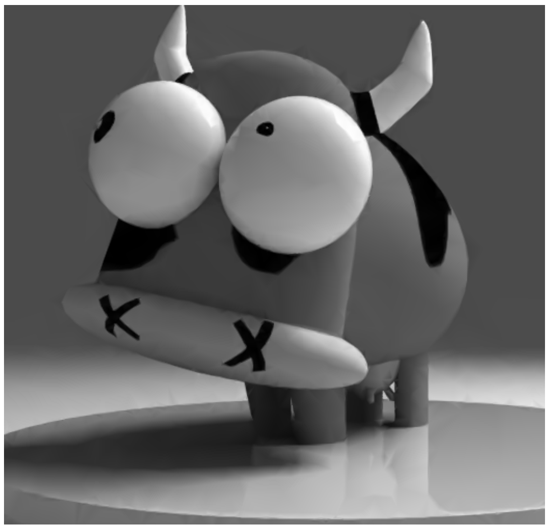


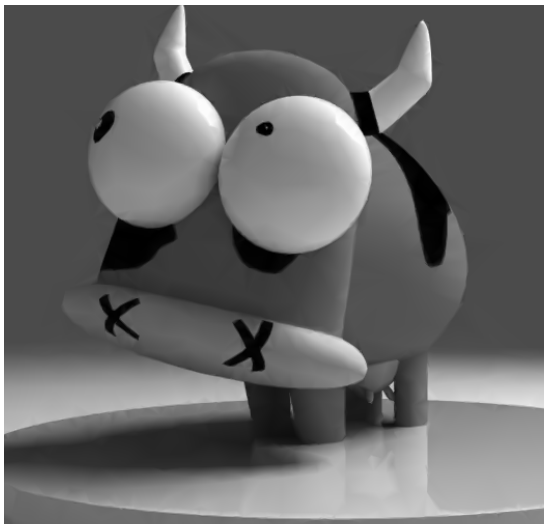
(c)

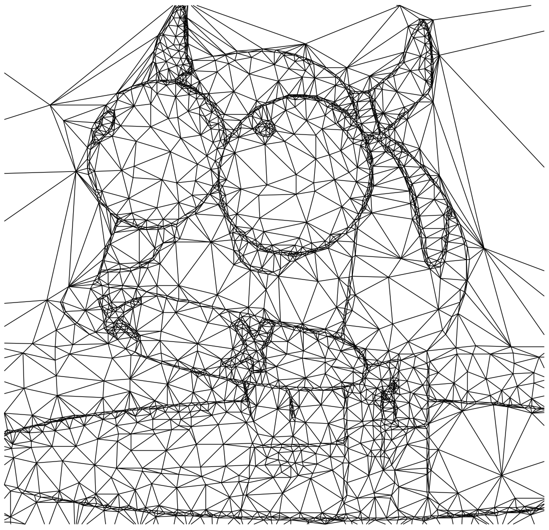


(d)

Figure 4.1: Part of the image approximation obtained for the bull image at a sampling density of 0.5% with the proposed (a) IID1 method (41.45 dB), and (b) the ID1 method (41.84 dB), and (c) and (d) their corresponding triangulations.


(a)


(b)


(c)

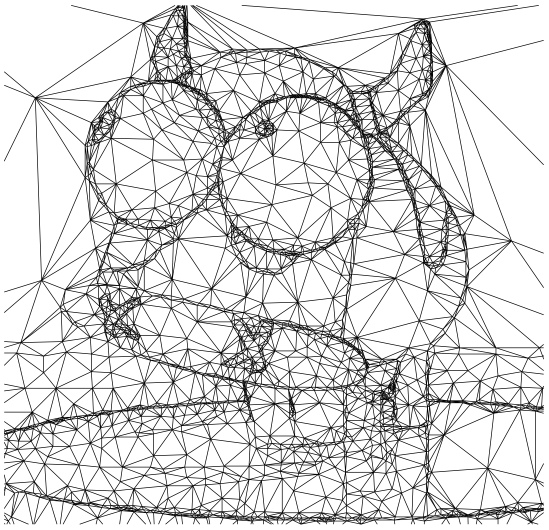
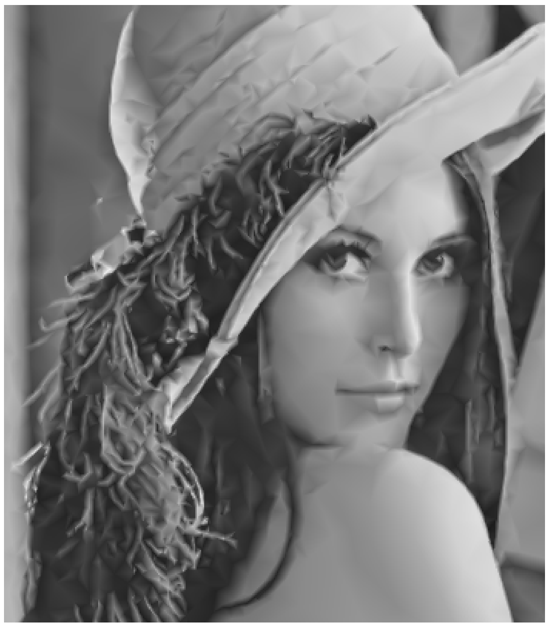
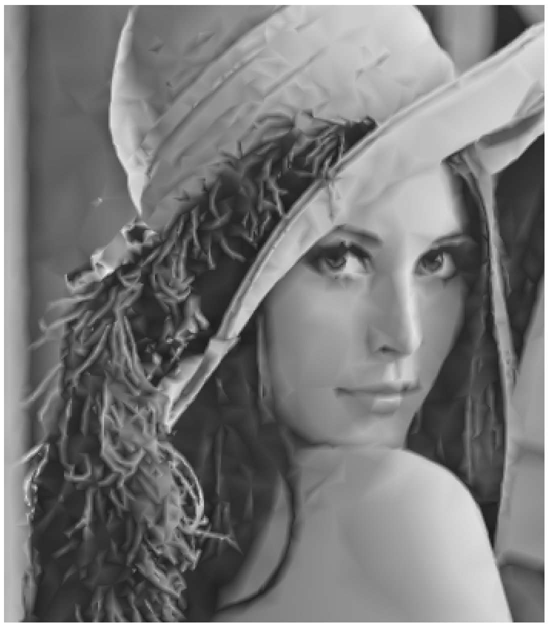
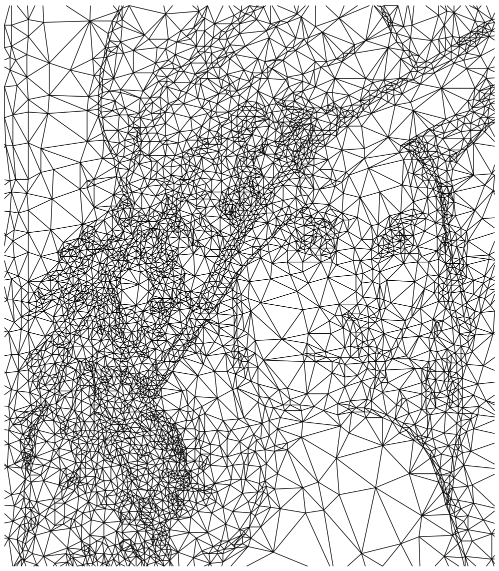

(d)

Figure 4.2: Part of the image approximation obtained for the bull image at a sampling density of 0.5% with the proposed (a) IID2 method (42.24 dB), and (b) the ID2 method (42.24 dB), and (c) and (d) their corresponding triangulations.


(a)


(b)


(c)

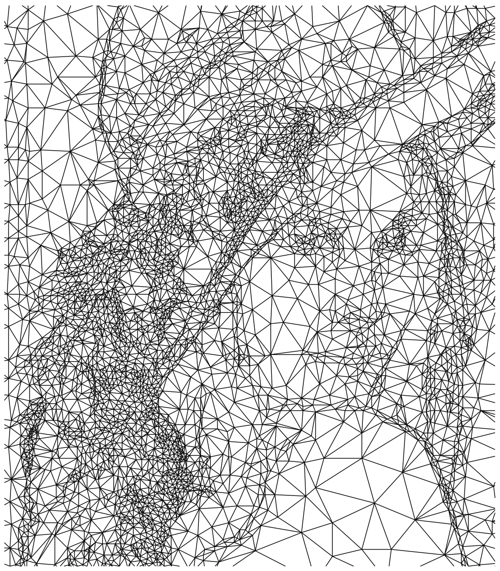
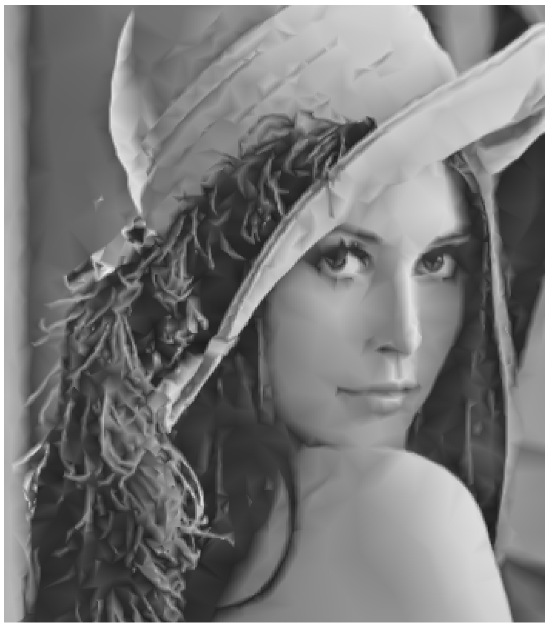
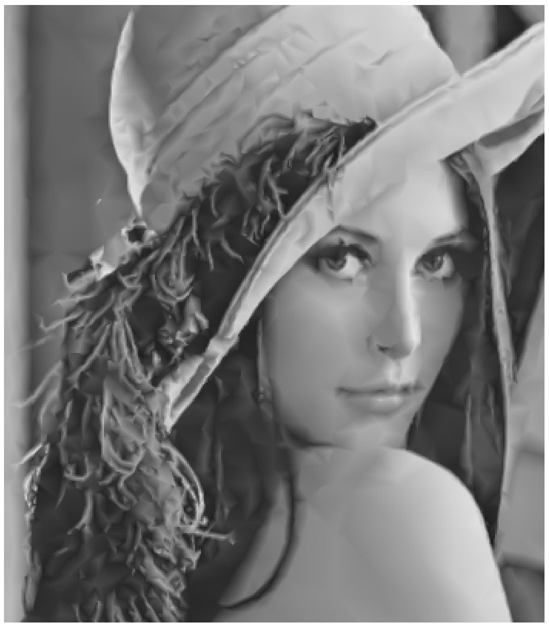

(d)

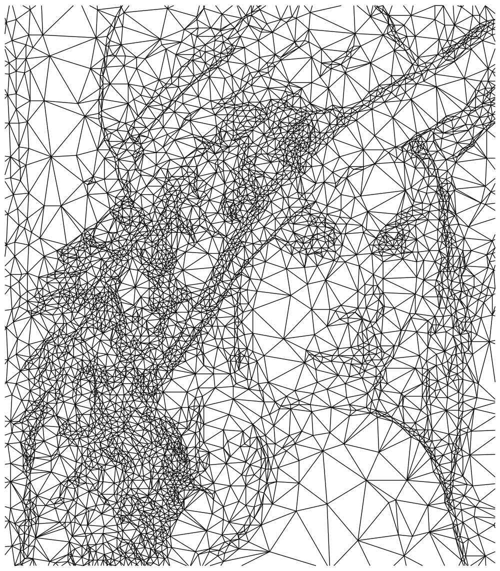
Figure 4.3: Part of the image approximation obtained for the lena image at a sampling density of 2% with the proposed (a) IID1 method (32.10 dB), and (b) the ID1 method (32.15 dB), and (c) and (d) their corresponding triangulations.



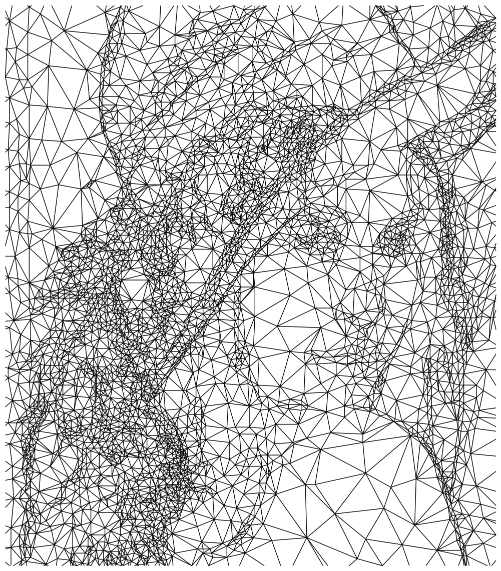
(a)



(b)



(c)



(d)

Figure 4.4: Part of the image approximation obtained for the lena image at a sampling density of 2% with the proposed (a) IID2 method (32.22 dB), and (b) the ID2 method (32.17 dB), and (c) and (d) their corresponding triangulations.

Table 4.2: Comparison of the computational cost for the various methods.

Image	**Sampling density (%)**	**Execution Time (s)**					
		IID1	IID2	ID1	ID2	IDDT	GPR
`bull`	0.125	4.0	8.8	5.0	11.5	3.2	125.4
	0.250	5.4	11.1	8.0	16.3	4.3	117.5
	0.500	7.2	16.0	16.1	26.7	7.5	116.1
	1.000	9.9	22.6	24.0	38.5	11.4	115.2
	2.000	14.8	31.4	30.3	45.4	16.1	112.3
	4.000	24.4	41.2	29.6	45.2	22.0	109.7
`ct`	0.125	1.1	2.7	1.4	3.4	0.9	38.9
	0.250	1.2	3.2	1.8	4.0	1.1	39.2
	0.500	1.8	4.0	2.4	5.2	1.5	36.6
	1.000	2.7	5.4	3.0	6.0	1.9	37.4
	2.000	4.2	7.2	4.1	7.6	2.6	36.8
	4.000	7.1	12.2	6.7	11.1	4.0	35.3
`lena`	0.125	1.1	3.0	1.6	3.6	1.2	39.1
	0.250	1.2	2.9	1.7	4.1	1.2	39.1
	0.500	1.6	3.9	2.5	4.9	1.4	38.7
	1.000	2.5	5.3	3.3	6.3	2.0	37.5
	2.000	4.1	7.8	4.7	8.6	2.7	37.3
	4.000	7.1	12.7	7.6	11.9	4.3	36.8

the median value of the runs for each case was determined. The results are presented in Table 4.2.

At first glance, we can see that the execution time for all methods is monotonically increasing with the sampling density except for the GPR scheme for which has the opposite. The result is expected given that the GPR method is a mesh-simplification scheme and therefore requires more operations for a smaller target mesh size. Furthermore, the performance of the GPR method is clearly the worst. It is slower than all of the other methods in all cases, and its execution time is high even at very low densities. Examining the results more closely, we see that the computational cost of our two methods is typically lower than that of competing methods of similar complexity. In particular, our IID1 method typically beats the ID1 and ID2 methods at most densities in our test cases. Compared to the IDDT method, our IID1 method alternates between being faster and slower depending on the mesh density and particular image. This performance is particularly noteworthy when we recall that the IID1 method is able to achieve approximation qualities that are typically noticeably better than the IDDT method, and often close to those of higher complexity methods. For example, in the case of the lena image with a mesh density of 2%, the execution time of the IID1 method is 47%, 13%, 52%

and 89% lower than that of the IID2, ID1, ID2 and GPR methods, respectively, but the PSNR of the corresponding approximation is only marginally lower (0.05–0.12 dB) compared to the first three methods, and 0.32 dB higher compared to the GPR method. As for the IID2 method, it has an execution cost lower than that of the GPR method by margins of 62–93% across the test cases. Additionally, the computational cost of the IID2 method is lower than or similar to that of the competing ID2 method in all of the test cases. More specifically, the IID2 method is able to achieve a reduction in execution time of 9–41% in 15/18 (83%) of the test cases. It is able to achieve the most savings in the test cases with low densities and those with large images. Furthermore, the IID2 method has lower or similar computational cost compared to the lower complexity ID1 method for the bull image with densities of 0.5–2%. The computational cost of our IID2 method is particularly impressive considering that it has the best ranking in terms of mesh quality by a decent margin. These computational cost results are consistent with the observations we made earlier in Section 3.4 that better efficiency can be achieved using growth schedules with a higher value for the amplitude parameter and a smaller value for the length parameter.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we have studied image representation using triangle-mesh models. In particular, we have proposed two new mesh-generation methods, derived by fixing the degrees of freedom of the incremental/decremental mesh-generation framework of [20]. As the framework has several free parameters, we first evaluated how different choices of these parameters affect mesh quality. The results of the analysis were used to recommend specific choices for these parameters, which led to the proposal of two new mesh-generation methods, known as IID1 and IID2.

Through experimental results, our proposed IID1 and IID2 methods were shown to outperform competing mesh-generation methods of similar complexity, namely the ID1, ID2, IDDT, and GPR methods. In particular, we demonstrated that our proposed methods are more efficient, yielding higher quality image approximations for a given computational cost. The higher complexity IID2 method was shown to achieve better quality approximations than the ID2 and GPR methods, with lower or similar computational cost. The lower complexity IID1 method was shown to trade a typically small penalty in image approximation for lower computational cost. The different trade off between computational cost and approximation quality allows our two methods to be useful in a wide 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 new growth schedule A′, a MMSODD-based candidate-selection policy, and Floyd-Steinberg error diffusion for the selection of the initial mesh points. These choices were shown to typically lead to an increase in mesh quality or reduction in computation time or both. The IID1 method uses all three of our newly proposed choices for the free parameters, while the IID2 method uses the newly proposed initial mesh and growth schedule.

5.2 Future Work

As part of the work presented in this thesis, we covered numerous possible choices for the various free parameters of the incremental/decremental mesh-generation framework of [20]. Nevertheless, further work on certain areas could still prove beneficial. In what follows, we discuss some of the areas that we feel are most promising.

In Section 3.2, we introduced adaptive Poisson-disk sampling (PDS) as a choice for selecting the initial mesh points. The radii map used with adaptive Poisson-disk sampling and its mapping function were developed by the author based on extensive experimentation. The performance of PDS was somewhat mixed and we opted not to select it as a choice due to its computational cost relative to the return on mesh quality. The implementation we used is based on a state-of-the-art algorithm [72], currently considered to be one of the most efficient methods for generating Poisson-disk samples. Nevertheless, we believe the computational cost of PDS could be reduced and the quality of its results could be improved and made more consistent by developing an improved choice for the mapping function. In particular, the normalization procedure we used is quite basic and does not account for the energy in the image. With fixed parameter values, this leads to too few samples being selected with certain images and too many with others, resulting in the mixed results that we observed. Alternatively, since the properties of true PDS are not quite needed in our application, another option worth exploring is the use of a different (and faster) approach for adaptive sampling with blue-noise properties.

Another choice we considered for the initial mesh is Floyd-Steinberg error diffusion, which we eventually selected for our two proposed methods. Error diffusion has several degrees of freedom (e.g., scan order and sensitivity) which we fixed based on the conclusions of previous research. These choices are, however, not necessarily optimal in the context of the computational framework that we used. Mesh quality could potentially be improved with different choices for the free parameters. In particular, reduction of the start-up effect that often affects error diffusion is worth investigating.

Finally, for efficiency reasons, we generated the MMSODD of the image once and used it in all the choices that are based on the MMSODD. As a result, the binomial smoothing filter we applied to the image before computing the MMSODD affects all of the choices that are based on the MMSODD, such as the candidate-selection policies (e.g., ALSEM) and error diffusion for the selection of the initial mesh points. Performance of the mesh-generation methods is likely to benefit from using distinct versions of the MMSODD that are computed from copies of the image that are smoothed differently for the different applications. For example, an edge-preserving bilateral filter with $\sigma_r = 1$ and $\sigma_s = 1$ might be a good choice for smoothing the image while computing the MMSODD used in error diffusion, whereas a 9-th order binomial filter might be better suited for the MMSODD used in the ALSEM candidate-selection policy.

Appendix A

Software User Manual

A.1 Introduction

During the course of the research presented in this thesis, the author, with considerable help from his supervisor, Dr. Michael Adams, developed a software framework for the mesh-generation methods proposed herein. In addition, the software allows for other variations that were considered during experimentation, as well as some existing schemes such as the GPR [17], GPRFS [42], ID1 [21], and ID2 [21], and IDDT [20] methods, which are based on Delaunay triangulations. The software implementation was written in C++ and consists of over 9000 lines of code. It makes use of some fairly involved data structures and algorithms to optimize performance. The Poisson-disk sampling implementation was ported from Java [73] to C++ and modified for the purposes of our work. It is based on the Fast Poisson Disk Sampling algorithm of Bridson [72]. The bilateral filter implementation was obtained from [74]. Additionally, the software implementation relies heavily on the Computational Geometry Algorithm Library (CGAL) [75], the Signal Processing Library (SPL) [76] and its extension SPLEL, and some of the Boost libraries [77].

The software framework has several degrees of freedom, with many possible choices for each, in order to maximize flexibility. The software package consists primarily of two executable programs: `generate_mesh` and `rasterize_mesh`. The `generate_mesh` program reads a grayscale image from standard input or a file specified by the user, and creates a mesh model of the image with the desired sampling density based on the method and method-specific parameters that were passed to the program. The `rasterize_mesh` program takes, as input, the set of points and corresponding function values that comprise the mesh model, and then creates a raster image reconstruction. In the remainder of this appendix, information relating to the software will be provided in more detail including how to build and use the software.

A.2 Building the Software

The software needs to be built before it can be used. The standard `make` utility program is used for this purpose for the sake of simplicity and robustness. In order to create the executable files, `make` compiles and links the necessary source files based on a set of rules defined in the Makefiles provided. The software was developed and tested on Linux but does not use platform-specific features. Consequently, in theory, it is not restricted to UNIX-like platforms or specific hardware, but some of its dependencies (e.g., libraries) may have such constraints. The build process requires a C++ compiler that complies with the C++11 version of the standard, but C++14 is recommended. The GCC C++ compiler version 4.8 or greater is recommended. Alternative compilers supporting C++11 could also be used but they have not been tested. The Clang compiler may be used with a simple change of the compiler command in the Makefile, while other compilers may require more extensive changes. As stated previously, the software uses some libraries which need to be installed first: CGAL, Boost, SPL, and SPLEL. The following versions of these libraries were used by the author and are known to work:

- CGAL version 4.7
- Boost version 1.58
- SPL version 1.1.24
- SPLEL version 1.1.36

Other version combinations, especially newer releases, may very well work but this cannot be guaranteed. The libraries are assumed to be located in a standard system path. If this is not the case, the correct paths need to be manually set in the Makefile using the following variables:

- `CGAL_PATH` for CGAL
- `BOOST_PATH` for Boost

- `SPL_PATH` for SPL and SPLEL

Once the prerequisites are satisfied, the software may be built by setting the working directory to the top level of the source-code tree, and then running the following commands:

```
make clean
make
```

The `make clean` command is optional, although considered good practice. The build process may be accelerated by enabling multithreading for `make` using the `-j N` option where `N` is the number of threads desired. A value up of 4 is considered reasonable on a modern processor, for example: `make -j4`. The above commands should create the executables `generate_mesh` and `rasterize_mesh` in the directories in which the source code of the programs is located.

A.3 File Formats

Given that the `rasterize_mesh` program is meant to rasterize mesh models created by `generate_mesh`, the input of the former is the output of the latter. This format is referred to herein as the mesh model format. Conversely, the input to `generate_mesh` and the output of `rasterize_mesh` are images containing the original data and the reconstructed model, respectively. These images use the format described in what follows.

A.3.1 Image Format

The `generate_mesh` program takes as input an image that complies with the set of graphics formats defined by the Netpbm project [78], sometimes collectively named the portable anymap format (PNM). Black and white, grayscale, and color images are all supported in the PNM format, but the author's software only supports black and white images, and grayscale images of arbitrary bitdepths. The grayscale variants are often referred to portable as gray map (PGM) format, hence associated with the ".pgm" extension. The reconstructed model output of the `rasterize_mesh` program is also generated in the PGM format.

A.3.2 Mesh Model Format

A slightly modified version of the Object File Format (OFF) [79] is used for outputting the mesh model generated by the `generate_mesh` program, and used as input to the `rasterize_mesh` program. Given that the OFF format is commonly used for storing meshes, a major advantage of the format is the ability to use a number of existing software for visualization and manipulation of the mesh data (e.g., GeomView). The OFF format is a raw representation of the data (i.e., uncompressed) with space-separated values in plain text, following a header consisting of the keyword `OFF`. The modification to the original OFF format consists of the insertion by the `generate_mesh` program of some additional data before the `OFF` keyword. The data consists of the width, height, and maximum possible brightness value of the original image. The width and height allow for a reduction of the computational cost during rasterization. The maximum possible brightness value, which is extracted from the original image, is important because it allows support for multiple bit-depths, which the mesh-generation software supports. It ensures that the reconstruction output has the same brightness range as the original image. Given that the custom header only affects the beginning of the file, the rest of the data in the mesh file remains completely compatible with the original object file format. Consequently, the OFF-formatted data may be easily extracted for use with other software that is OFF-compatible. In summary, the mesh model has the following format:

```
WIDTH  HEIGHT
MAX_VALUE
{OFF_DATA}
```

where {`OFF_DATA`} represents the pure OFF data for the mesh model, the contents of which begin with the keyword `OFF`.

A.4 Detailed Software Description

As previously stated, the software consists of two programs: `generate_mesh` and `rasterize_mesh`. The `generate_mesh` program is used for mesh-generation, whereas the `rasterize_mesh` program is used for rasterization of the mesh model. In the sections that follow, a detailed description of the command line interface for each program is given.

A.4.1 The `generate_mesh` Program

Synopsis

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

Description

The `generate_mesh` program reads an image in the PNM format from standard input, and writes a mesh model of the image to standard output. The `generate_mesh` program consists of three main functional blocks: the initial mesh selection block, the mesh simplification and refinement block, and the postprocessing block. The first block selects the initial-mesh points using an approach that does not use meshes. In other words, a subset of the sample points of the original image is generated either adaptively based on the image content, or in an open-loop fashion (e.g., randomly). The set of sample points that constitute the output of this block are used to seed the initial mesh. The mesh simplification and refinement block which comes second works iteratively on the mesh to create a model of the image with the desired sampling density. The sequence of insertions and deletions is determined from the growth schedule setpoints which the size of the mesh tracks. The final block performs postprocessing on the generated mesh model. In the software, it implements bad point replacement and bad point deletion. If a mesh encoding method were used, it would be part of this block.

Due to the multitude of free parameters and their hierarchical structure, a naming convention that naturally shows this hierarchy was used. A sub-parameter "Y" of a parameter "X" is indicated by an option named "`X.Y`". For example, parameters relating to the initial mesh generation block are prefixed with "`init.`", and `init.prefilter.bilateral.range` denotes the "range" option sub-parameter associated with the `init.prefilter.bilateral` parameter.

Options

In what follows, `$arg` denotes the value of the argument of an option.

`--help`
: Print a help message listing all the free parameters and the supported values for these parameters.

`--density $arg`

Set the desired output mesh density as a percentage of the total number of samples in the original image to `$arg` (e.g., `$arg`=8 for 8%). Valid values for `$arg` are as follows: 0 < `$arg` $\leq$ 100

`--init.density.ratio $arg`

Set the initial mesh density scaling factor to `$arg`. The actual initial mesh density is the product of the target density value and the value of the initial mesh density scaling factor. Therefore, a value that less than 1 leads to an initial mesh with fewer points than the final mesh. Valid values for `$arg` are as follows: `density` $*$ `$arg` <= 100%

`--init.generator $arg`

Set the method used for generating the initial point set to `$arg`.
Valid values for `$arg` are:
- `none`: same effect as `init.density.ratio`=0. An initial-mesh generation method is not used. The iterative mesh generation starts with the trivial mesh consisting of the four extreme convex-hull points of the image domain.
- `all samples`: Creates an initial mesh that contains all the sample points of the input image. Used by mesh refinement methods such as the GPR method.
- `random`: The initial point set is selected randomly within the image domain. The x and y coordinates are generated independently from 2 distinct random values.
- `random2`: The initial point set is selected randomly within the image domain. The x and y coordinates are extracted from a single random value.
- `uniform`: Distributes points uniformly along the width and height of the image domain so that a coarse grid is created. The number of points and uniformness of the grid are limited by the constraints of the integer lattice.
- `jittered`: Distributes points evenly across the image domain. A uniform grid is first created then every point is moved by a small amount along the x and y coordinates.
- `ed`: Generates the initial mesh points using the error diffusion (ED) method.

- `subset`: Selects the points with the highest values in the subset image. Requires specification of the following: `init.subset.type`
- `poisson`: Selects the initial sample points based on adaptive Poisson disk sampling. The points density in a region are based on the radii map which is created from the base radii map using a mapping function. The mapping function converts the radii values into a suitable range and applies some transformations to improve their characteristics. Requires specification of the following:
`init.poisson.map.type`, `init.poisson.map.filter_order`,
`init.poisson.map.power`, `init.poisson.map.range_max`,
`init.poisson.map.min_dist`

`--init.prefilter $arg`

Set to `$arg` the type of smoothing filter to apply to the input image before it is used in operations that benefit from smoothing (e.g., gradient).
Valid values for `$arg` are as follows:
- `none`: no smoothing filter applied.
- `linear`: binomial filter.
Requires specification of the following:
`init.prefilter.linear.order`
- `bilateral`: bilateral filter.
Requires specification of the following:
`init.prefilter.bilateral.range`, `init.prefilter.bilateral.space`

`--init.prefilter.linear.order $arg`

Set the order of the binomial smoothing filter to `$arg`.
Valid values for `$arg` are as follows: positive odd integers

`--init.prefilter.bilateral.range $arg`

Set the range parameter of the bilateral smoothing filter to `$arg`.
Valid values for `$arg` are as follows: positive

`--init.prefilter.bilateral.space $arg`

Set the range parameter of the bilateral smoothing filter to `$arg`.
Valid values for `$arg` are as follows: positive

`--init.poisson.map.type $arg`

Set to `$arg` the type of transform to be applied to the input image to create the base radii map for the adaptive Poisson disk sampling initial mesh.

Valid values for `$arg` are as follows: `gradient`, `laplacian`, `mmsodd`

`--init.poisson.map.filter_order $arg`

Set to `$arg` the order of the binomial filter used to smooth the base radii map in adaptive Poisson disk sampling before the mapping function is applied to it.

Valid values for `$arg` are as follows:positive odd integers

`--init.poisson.map.power $arg`

Set the exponent parameter of the Poisson disk sampling radii map mapping function to `$arg`.

`--init.poisson.map.range_max $arg`

Set the maximum range parameter of the Poisson disk sampling radii map mapping function to `$arg`.

`--init.poisson.map.min_dist $arg`

Set the minimum distance range parameter of the Poisson disk sampling radii map mapping function to `$arg`.

`--init.subset.type $arg`

Set to `$arg` the type of transform to apply to the input image to create a value map to be used with threshold-based subset point selection.

Valid values for `$arg` are as follows: `gradient`, `laplacian`, `mmsodd`

`--initial_mesh_only`

Exit after creating the initial mesh. If mesh-generation parameters (e.g., growth schedule type) are set, they are ignored.

`--growth.schedule $arg`

Set the type of growth schedule to `$arg`.

Valid values for `$arg` are as follows:

- `A`: “Above” (A′) new growth schedule.

Requires specification of the following: `growth.schedule.length`, `growth.schedule.amp`

- `B`: “Below” (B′) new growth schedule.

Requires specification of the following: `growth.schedule.length`, `growth.schedule.amp`

- `C`: “Circa” (C′) new growth schedule.

Requires specification of the following: `growth.schedule.length`, `growth.schedule.amp`

- `I`: “Incremental” (I′) new growth schedule

- `IDDT_A`: Original A growth schedule used in the IDDT, ID1, and ID2 methods.

Requires specification of the following:

`growth.schedule.iddt.alpha`

- `IDDT_B`: Original B growth schedule used in the IDDT, ID1, and ID2 methods.

Requires specification of the following:

`growth.schedule.iddt.alpha`

- `IDDT_C`: Original C growth schedule used in the IDDT, ID1, and ID2 methods.

Requires specification of the following:

`growth.schedule.iddt.alpha`

- `IDDT_I`: Original I growth schedule used in the IDDT, ID1, and ID2 methods.

Requires specification of the following:

`growth.schedule.iddt.alpha`

`--growth.schedule.length $arg`

Set the growth schedule length to `$arg`.

Valid values for `$arg` are as follows: positive integers

`--growth.schedule.amp $arg`

Set the overshoot amplitude parameter of the growth schedule. It is specified as a percentage of the target density. That means that the first set point is: $(\texttt{\$arg} + 100\%) * \texttt{density}$. For example, the first setpoint for an amplitude value of `120` is 2.2 times the size of the target mesh size.

Valid values for `$arg` are as follows: positive

`--growth.schedule.iddt.alpha $arg`

Set the alpha parameter of the IDDT growth schedules to `$arg`.

Valid values for `$arg` are as follows: $0 < \texttt{\$arg} < 1$

`--select.face $arg`

Set the policy for selecting the face into which a point will be inserted to `$arg`.

Valid values for `$arg` are as follows:

- `sse`: Select the face with the highest sum-of-squared errors.
- `sae`: Select the face with the highest sum-of-absolute errors.

`--select.candidate $arg`

Set to `$arg` the policy for selecting the candidate point for insertion into the mesh from the points inside a given face.

Valid values for `$arg` are as follows:

- `pae`: Select the point with the highest (peak) absolute error.
- `pwae`: Select the point with the highest (peak) weighted absolute error.
- `alsem`: Select the point based on the approximate local squared-error minimizer (ALSEM) policy.
- `gradient`: Select the point with the highest gradient magnitude value.
- `laplacian`: Select the point with the highest Laplacian magnitude value.
- `mmsodd`: Select the point with the highest MMSODD value.

`--sig.del $arg`

Set the policy for selecting the candidate point for deletion from the mesh to `$arg`.

Valid values for `$arg` are as follows:

- `opt`: Optimal point that reduces the error the most (or increases it the least).

`--bpr`
: Enable bad point replacement. Mutually exclusive with the `--bpd` option.

`--bpd`
: Enable bad point deletion. Mutually exclusive with the `--bpr` option.

`--output_psnr`
: Enable outputting the PSNR of the generated mesh model instead of the actual model data to the standard output stream. If not set, the PSNR is written to the error stream.

A.4.2 The `rasterize_mesh` Program

Synopsis

```
rasterize_model   --input $input_mesh_model \
                  --output $output_reconstructed_model \
                 [--original_image $original_image]

rasterize_model  [--original_image $original_image] \
                 < $input_mesh_model \
                 > $output_reconstructed_model
```

Description

The `rasterize_mesh` program reconstructs a raster image from a mesh model using Delaunay triangulation with preferred directions. The input mesh model and output image conform to the formats described previously. The standard and the extended OFF mesh model formats are both supported. The data may be read from/written to standard input/output, from/to files, or a combination of the two. If a file path is specified for an option, the corresponding input/output from a standard stream is ignored. In all cases however, status information is printed to the standard error stream. The program can also, optionally, calculate the PSNR of the mesh model's rasterization result, and write it to the error stream. The original image needs to be set using the `--original_image` option. It should be noted that the mesh connectivity specified in the input data is ignored by the program. The reason being that Delaunay triangulation with preferred directions determines connectivity from the vertices solely.

Options

`--input $file`	Set the path of the file from which to read the input mesh model to `$file`. The parameter is optional, and if not set the data is read from standard input.
`--output $file`	Set the path of the file to which the mesh rasterization result should be written to `$file`. The parameter is optional, and if not set the data is output to standard output.
`--original_image $file`	Set the path of the original image from which the mesh model was created to `$file`. The parameter is optional, and if set the PSNR is written to the error stream

A.5 Examples of Mesh-Generation Methods

The methods proposed in the thesis as well as other previously proposed methods that were referenced in previous chapters can be derived from the framework by setting the program options appropriately. To guide the reader, we provide some examples in what follows.

1. For the image in the file `$image`, a mesh-model with a sampling density of 0.125% may be generated using the proposed IID2 method with the command:

```
generate_mesh --density 0.125 --init.density 1 --init.prefilter linear \
--init.prefilter.linear.order 9 --init.generator ed \
--growth.schedule A --growth.schedule.length 5 \
--growth.schedule.amp 300 --select.candidate alsem \
--select.face sse --sig.del opt --bpr < $image
```

2. For the image in the file `$image`, a mesh-model with a sampling density of 1% may be generated using the proposed IID1 method with the command:

```
generate_mesh --density 1 --init.density 1 --init.prefilter linear \
--init.prefilter.linear.order 7 --init.generator ed
--growth.schedule A --growth.schedule.length 13 \
--growth.schedule.amp 300 --select.candidate mmsodd \
--select.face sse --sig.del opt --bpr < $image
```

3. For the image in the file `$image`, a mesh model with a sampling density of 0.25% may be generated using the GPR method (described in Section 2.10) with the command:

```
generate_mesh --density 0.25 --init.density 1 --init.prefilter none \
--init.generator all_samples --growth.schedule I \
--growth.schedule.length 2 --growth.schedule.amp 1 \
--select.candidate pae --select.face sse --sig.del opt  < $image
```

4. For the image in the file `$image`, a mesh model with a sampling density of 0.5% may be generated using the IDDT method (described in Section 2.10) with the command:

```
generate_mesh --density 0.5 --init.density 0 --init.prefilter linear \
--init.prefilter.linear.order 9 --init.generator none \
--growth.schedule B --growth.schedule.length 14 \
--growth.schedule.amp 99.95 --select.candidate pwae --select.face sse \
--sig.del opt --bpr < $image
```

5. For the image in the file `$image`, a mesh model with a sampling density of 3% may be generated using the ID2 method (described in Section 2.10) with the command:

```
generate_mesh --density 3 --init.density 0 --init.prefilter linear \
--init.prefilter.linear.order 9 --init.generator none \
--growth.schedule IDDT_A --growth.schedule.iddt.alpha 0.4 \
--select.candidate alsem --select.face sse --sig.del opt \
--bpr < $image
```

6. The mesh-model contained in the file `$model` may be rasterized and written to standard output using the command:

```
rasterize_mesh < $model
```

or

```
rasterize_mesh --input  $model
```

7. The mesh-model contained in the file `$model` may be rasterized and written to a file `$output` output using the command:

   ```
   rasterize_mesh < $model > $output
   ```

   or:

   ```
   rasterize_mesh --output $output < $model
   ```

   or:

   ```
   rasterize_mesh --input $model --output $output
   ```

8. For a mesh-model of the image `$original` that is contained in the file `$model`, the PSNR of the reconstruction may be computed using the command:

   ```
   rasterize_mesh --input $model --original_image $original
   ```

Bibliography

[1] 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.

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

[3] 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*, pages 1017–1020, Apr. 2009.

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

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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, Feb. 2004.

[11] 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:II–621–II–624, 2002.

[12] 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.

[13] 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.

[14] I. Amidror. Scattered data interpolation methods for electronic imaging systems: a survey. *Journal of Electronic Imaging*, 11(2):157–176, Apr. 2002.

[15] R. Franke and G. M. Nielson. Scattered data interpolation and applications: A tutorial and survey. *Geometric Modeling: Methods and Applications*, pages 131–160, 1991.

[16] M. D. Adams. An evaluation of several mesh-generation methods using a simple mesh-based image coder. *Proceedings of IEEE International Conference on Image Processing*, pages 1041–1044, Oct. 2008.

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

[18] M. Garland and P. S. Heckbert. Fast polygonal approximation of terrains and height fields. *School of Computer Science, Carnegie Mellon University*, CMU-CS-95-181, Sep. 1995.

[19] K. Wang, C.-P. Lo, G. A. Brook, and H. R. Arabnia. Comparison of existing triangulation methods for regularly and irregularly spaced height fields. *International Journal of Geographical Information Science*, 15(8):743–762, 2001.

[20] M. D. Adams. An incremental/decremental Delaunay mesh-generation framework for image representation. *Proceedings of IEEE International Conference on Image Processing*, pages 189–192, Sep. 2011.

[21] M. D. Adams. A highly-effective incremental/decremental Delaunay mesh-generation strategy for image representation. *Signal Processing*, 93:749–764, 2013.

[22] M. Garland and P. S. Heckbert. Fast triangular approximation of terrains and height fields. *Draft manuscript*, dated May 2, 1997.

[23] S. Rippa. Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1123–1141, 1992.

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

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

[26] E. Quak and L. L. Schumaker. Curves and surfaces. chapter Least Squares Fitting by Linear Splines on Data Dependent Triangulations, pages 387–390. Academic Press Professional, Inc., San Diego, CA, USA, 1991.

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

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

[29] 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.

[30] N. Dyn, D. Levin, and S. Rippa. Algorithms for the construction of data dependent triangulations. *Algorithms for Approximation II*, pages 185–192, 1990.

[31] C. L. Lawson. Software for C^1 surface interpolation. *Mathematical Software III*, pages 161–194, 1977.

[32] J. L. Brown. Vertex based data dependent triangulations. *Computer Aided Geometric Design*, 8:239–251, 1991.

[33] X. Ma and M. D. Adams. An improved error-diffusion approach for generating mesh models of images. *Signal Processing*, 117:17–32, 2015.

[34] X. Tu and M. D. Adams. Improved mesh models of images through the explicit representation of discontinuities. *Canadian Journal of Electrical and Computer Engineering*, 36(2):78–86, 2013.

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

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

[37] P. K. Agarwal and S. Suri. Surface approximation and geometric partitions. *SIAM Journal on Computing*, 27(4):1016–1035, 1998.

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

[39] 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.

[40] D.-M. Yan and P. Wonka. Gap processing for adaptive maximal poisson-disk sampling. *ACM Transactions on Graphics*, 32(5):148:1–148:15, October 2013.

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

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

[43] B. Delaunay. Sur la sphere vide. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et Naturelle*, 7(6):793–800, 1934.

[44] 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.

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

[46] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *Proceedings of the International Conference on Computer Vision*, pages 839–846, 1998.

[47] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *Proceedings of the ACM SIGGRAPH'02 Conference*, pages 257–266, 2002.

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

[49] O. Devillers and M. Teillaud. Perturbations and vertex removal in a 3D Delaunay triangulation. *14th ACM-SIAM Symposium on Algorithms*, 34:313–319, 2003.

[50] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, Jan. 1990.

[51] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, NY, USA, 2nd edition, 1998.

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

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

[54] R. L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, 1986.

[55] M. A. Z. Dippé and E. H. Wold. Antialiasing through stochastic sampling. *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, 19(3):69–78, July 1985.

[56] R. Fattal. Blue-noise point sampling using kernel density model. *ACM Transactions on Graphics*, 30(4):48:1–48:12, July 2011.

[57] D. P. Mitchell. Generating antialiased images at low sampling densities. *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, 21(4):65–72, August 1987.

[58] D. P. Mitchell. The antialiasing problem in ray tracing. *SIGGRAPH '90 Course Notes*, 1990.

[59] H. Schechter and R. Bridson. Ghost SPH for animating water. *ACM Transactions on Graphics*, 31(4):61:1–61:8, July 2012.

[60] D. R. Williams and R. Collier. Consequences of spatial sampling by a human photoreceptor mosaic. *Science 221*, pages 385–387, 1983.

[61] J. I. Jr. Yellott. Spectral consequences of photoreceptor sampling in the rhesus retina. *Science 221*, pages 382–385, 1983.

[62] P. Dutr and A. Lagae. A comparison of methods for generating poisson disk distributions. *Computer Graphics Forum*, 27(1):114 – 129, 2008.

[63] D. Cline, S. Jeschke, K. White, A. Razdan, and P. Wonka. Dart throwing on surfaces. *Proceedings of the Twentieth Eurographics Conference on Rendering*, pages 1217–1226, 2009.

[64] K. B. White, D. Cline, and P. K. Egbert. Poisson disk point sets by hierarchical dart throwing. *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing*, pages 129–132, 2007.

[65] M. McCool and E. Fiume. Hierarchical poisson disk sampling distributions. *Proceedings of the Conference on Graphics Interface '92*, pages 94–105, 1992.

[66] D.-M. Yan and P. Wonka. Adaptive maximal poisson-disk sampling on surfaces. *SIGGRAPH Asia 2012 Technical Briefs*, pages 21:1–21:4, 2012.

[67] S. A. Mitchell, A. Rand, M. S. Ebeida, and C. Bajaj. Variable radii poisson disk sampling. *Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG)*, pages 185–190, 2012.

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

[69] JPEG-2000 test images, July 1997.

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

[71] M. D. Adams. An evaluation of several mesh-generation methods using a simple mesh-based image coder. *Proceedings of IEEE International Conference on Image Processing*, pages 1041–1044, Oct. 2008.

[72] R. Bridson. Fast poisson disk sampling in arbitrary dimensions. *ACM SIGGRAPH 2007 Sketches*, 2007.

[73] Poisson disk sampling java implementation. http://devmag.org.za/2009/05/03/poisson-disk-sampling/, 2016.

[74] S. Paris and F. Durand. Fast bilateral filter. http://people.csail.mit.edu/sparis/bf/, 2016.

[75] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org, 2016.

[76] M. D. Adams. SPL, Signal Processing Library. SPL manual page: http://www.ece.uvic.ca/~frodo/SPL/manual/current/html, 2016.

[77] Boost C++ libraries. http://www.boost.org, 2016.

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

[79] OFF, object file format. http://segeval.cs.princeton.edu/public/off_format.html, 2016.