A Highly-Effective Incremental/Decremental Delaunay Mesh-Generation Strategy for Image Representation

Michael D. Adams*

Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada

Abstract

A flexible mesh-generation framework for image representation based on Delaunay triangulations is proposed. By fixing the various degrees of freedom available within this framework, two mesh-generation methods, known as ID1 and ID2, are derived. These two methods are shown to perform extremely well, producing meshes of significantly higher quality than state-of-the-art schemes at relatively low computational cost. Furthermore, the ID1 and ID2 methods each provide a mechanism whereby mesh quality can be increased (or decreased) in return for a corresponding increase (or decrease) in computational cost. Lastly, we demonstrate that one component of our proposed methods, called bad-point replacement, can be used as a postprocessing optimization step that, when added to other previously-proposed mesh-generation methods, yields meshes of much greater quality.

Keywords: Image representations, triangle meshes, Delaunay triangulations, mesh generation.

1. Introduction

In recent years, there has been a growing interest in image representations that employ adaptive (i.e., nonuniform) sampling [1–7] as such representations can, in many applications, have numerous advantages over traditional lattice-based sampling, including greater compactness and the ability to facilitate methods that yield higher quality results or have lower overall complexity. Some of the many applications that can benefit from adaptive sampling include: feature detection [8], pattern recognition [9], computer vision [10], restoration [11], tomographic reconstruction [12], filtering [13], and image/video coding [7, 14–19]. Although many classes of adaptively-sampled image representations have been proposed to date [20–23, 1, 24–27], those based on Delaunay triangulations have proven to be particularly effective

*__Corresponding author.__ Mailing address: Department of Electrical and Computer Engineering, University of Victoria, PO Box 3055 STN CSC, Victoria, BC, V8W 3P6, Canada; tel.: +1 250 721 6025; fax: +1 250 721 6052; e-mail: mdadams@ece.uvic.ca.

[6, 28, 2, 4, 3, 29, 30], and are the focus of our work herein. In order to employ a triangle-mesh representation of an image, a means is needed to construct such a representation given an arbitrary lattice-sampled image. That is, we must be able to select a good subset of the original sample points of an image from which to form a mesh model (of the image). This is the so called **mesh-generation problem**.

To date, numerous mesh-generation schemes have been proposed for Delaunay-triangulation-based mesh models of images. Some such methods include the MGH scheme [28] (inspired by the work of Garland and Heckbert [31]), the **error-diffusion (ED)** scheme of Yang et al. [29], the **greedy-point removal (GPR)** scheme of Demaret and Iske [3] (called "adaptive thinning" therein), and the **GPRFS-ED** scheme [6]. Of these methods, the GPR and GPRFS-ED schemes are particularly noteworthy as they represent state-of-the-art mesh-generation techniques. For example, in the recent paper [1, Figures 4 and 5], the GPR scheme (called "adaptive thinning" therein) was shown to yield meshes of vastly superior quality in comparison to all of the other methods considered. The main disadvantage of the GPR scheme is its extremely high computational and memory requirements. More recently, the GPRFS-ED scheme [6] was shown to produce, in many cases, meshes of only slightly lower quality than the GPR scheme, but at a very substantially reduced computational and memory cost.

In this manuscript, we propose a flexible new framework for mesh generation. By fixing the various degrees of freedom available within this framework, we derive two mesh-generation methods, known as ID1 and ID2 (where the "ID" in each name stands for incremental/decremental). As we shall see later, these two mesh-generation methods perform extremely well, producing meshes of higher quality than the state-of-the-art GPR and GPRFS-ED schemes at relatively low computational cost. Furthermore, our ID1 and ID2 methods each provide a mechanism whereby mesh quality can be increased (or decreased) in return for a corresponding increase (or decrease) in computational complexity. Moreover, we show that one component of our proposed mesh-generation methods, called bad-point replacement (BPR), can be used as a postprocessing step to improve upon the quality of meshes produced by other previously-proposed schemes such as the ED method. In passing, we note that the work described herein has been partially presented in the author's conference paper [32]. The ID1 and ID2 methods proposed herein, however, produce much higher quality meshes than the so-called **IDDT** method from this earlier conference paper, as demonstrated later by experimental results.

The remainder of this manuscript is organized as follows. First, Section 2 introduces the mesh model for image representation employed in our work. Then, Section 3 presents our proposed computational framework for mesh generation, with Section 4 proceeding to offer some guidance as to how this framework can be efficiently implemented. Our ID1 and ID2 mesh-generation methods (derived from the proposed frame-

work) are presented in Section 5, along with some details as to how they were obtained. In Section 6, our methods are evaluated and shown, through experimental results, to produce meshes of higher quality than state-of-the-art methods at relatively low computational cost. Finally, Section 7 concludes the manuscript with a summary of our key results.

Before proceeding further, a brief digression is in order regarding the test images used in our work. For the purposes of evaluation herein, we consider a set of 40 (grayscale) images, consisting of photographic, medical, and computer-generated imagery. For the most part, the images are taken from well-known image collections, with 24 taken from the Kodak test set [33], 11 taken from the JPEG-2000 test set [34], and 3 taken from the USC image database [35]. With the exception of statistical results taken over our entire test set, the experimental results presented herein focus on the small representative subset of images listed in Table 1, which were deliberately chosen to include photographic, medical, and computer-generated imagery.

[Table 1 about here.]

Lastly, a brief comment is appropriate regarding some of the notation employed in this manuscript. The cardinality of a set S is denoted $|S|$. Also, when presenting algorithms, the symbol ":=" is used to denote variable assignment.

2. Mesh Model for Image Representation

In the context of this work, a mesh model of an image ϕ defined on $\Lambda = \{0, 1, \ldots, W-1\} \times \{0, 1, \ldots, H-1\}$ (i.e., a rectangular grid of width W and height H) is completely characterized by: 1) a set $P = \{p_i\}_{i=1}^{|P|}$ of sample points; and 2) the set $Z = \{z_i\}_{i=1}^{|P|}$ of the corresponding sample values (i.e., $z_i = \phi(p_i)$). The set P is always chosen to include the extreme convex hull points of Λ (i.e., the four corner points of the image bounding box) so that the triangulation of P covers the entire image domain Λ. From P and Z, a unique interpolant $\hat{\phi}_P$ is constructed as follows. First, we form the Delaunay triangulation of P, which is ensured to be uniquely determined (from P alone) by employing the preferred-directions scheme [37]. Then, for each face f of the triangulation, we construct the unique linear function that interpolates ϕ at the three vertices of f. By combining the interpolants from each of the faces, we obtain the continuous piecewise-planar interpolant $\tilde{\phi}_P$. As a practical matter, since the sample values of the final image approximation must be integer, the final (integer-valued) interpolant $\hat{\phi}_P$ at the point p is given by $\hat{\phi}_P(p) = \text{round}(\tilde{\phi}_P(p))$, where round denotes rounding to the nearest integer. Since the vertices of the triangulation are nothing more than sample points, often the terms "sample point" and "vertex" are used interchangeably herein. As a matter of terminology, the **sampling density** of the mesh model is defined as $|P| / |\Lambda|$. In passing,

we note that our interest in Delaunay triangulations arises from their good properties for approximation. The Delaunay triangulation maximizes the minimum interior angle of all triangles in the triangulation, thus avoiding sliver triangles to whatever extent is possible. This leads to the Delaunay triangulation being optimal for controlling error bounds in approximation applications [38] such as the one considered herein.

Having defined the above mesh model, the mesh-generation problem that we address herein can be succinctly stated as follows: For a given target number N of sample points (where $N \in [4, |\Lambda|]$), choose $P \subset \Lambda$ such that $|P| = N$ and the mesh approximation error $\epsilon(P)$ is as small as possible—ideally, a global minimum. In our work, the mean squared error (MSE) is used as the error metric, so that $\epsilon(P) = |\Lambda|^{-1} \sum_{p \in \Lambda} (\hat{\phi}_P(p) - \phi(p))^2$. Herein, the MSE is typically expressed in terms of the peak-signal-to-noise ratio (PSNR), which is defined as $\mathsf{PSNR} = 20 \log_{10}(M/\sqrt{\mathsf{MSE}})$, where $M = 2^\rho - 1$, and ρ is the sample precision in bits/sample. Finding good *computationally-efficient* methods for solving the above problem is quite challenging, since problems like this are known to be NP hard [39].

3. Proposed Mesh-Generation Framework

Having introduced the particular mesh model for image representation assumed in our work, we are ready to present our proposed computational framework for mesh generation. Let ϕ denote an image sampled at the points Λ (which form a rectangular grid). Our framework is iterative in nature and works by making repeated updates to a mesh approximation of ϕ by alternating between two distinct processing phases: one that adds sample points to the mesh and one that deletes sample points from the mesh. As input, our framework takes the following information, in addition to ϕ: 1) a subset Γ of the sample points Λ from which to form an initial mesh approximation of ϕ; 2) the target number N of sample points that should be present in the mesh to be generated (where $N \in [4, |\Lambda|]$); and 3) a sequence $\{\eta_i\}_{i=0}^{L}$ (where $L \geq 1$) with $\eta_0 = |\Gamma|$, and $\eta_L = N$, known as a **growth setpoint sequence**, which specifies a **growth schedule** for the approximating mesh (i.e., how the size of the mesh should evolve by growing and shrinking as the framework is applied). As will be seen later, the growth setpoint sequence essentially specifies the mesh size at the end of each sample-point addition/deletion phase (mentioned above). Note that this sequence does not have to be (and normally is not) monotonically increasing (or monotonically decreasing). In order to either force or prevent certain points in Λ from appearing in the final mesh, the notion of mutability of a point is introduced. A point $p \in \Lambda$ is said to be **mutable** if it is permitted to be added to or deleted from the mesh as our framework is applied. A point that is not mutable (i.e., a point in the mesh that cannot be deleted from the mesh or a point not in the mesh that cannot be added to the mesh) is said to be **immutable**.

Before we can proceed further, it is necessary to introduce some additional definitions and notation. For a set S of points in Λ, mutable(S) denotes the set of all mutable points in S. 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., $r_S(p) = \hat{\phi}_S(p) - \phi(p)$). In what follows, let P denote the sample points in the current mesh approximation of ϕ. Each point $p \in \Lambda$ is assigned to *exactly one* face in the Delaunay triangulation of P, which is denoted face(p). If p is strictly inside a face f, we define face(p) = f; otherwise (for points on edges or vertices), the method of [40] is used to uniquely assign p to *exactly one* face. The set of all points p satisfying face(p) = f (i.e., all points belonging to the face f) is denoted points(f). The squared error computed over all points in the face f is denoted as faceErr(f) (i.e., faceErr(f) = $\sum_{p \in \Lambda \cap \text{points}(f)} r_P^2(p)$). For a given face f, the set of all mutable points in the face that are not currently in the mesh are called **candidate points** (of the face f) and is denoted cands(f) (i.e., cands(f) = mutable($(\Lambda \setminus P) \cap$ points(f))).

With the above definitions in place, our computational framework then consists of the following steps:

1. **Initialize mesh**. Let $i := 0$. Let $P := \Gamma$. Mark all extreme convex hull points of Λ as immutable (so that they cannot be later deleted from the mesh) and mark all other points in Λ as mutable.

2. **Top of main loop**. If $i = L$ (i.e., no more setpoints are available), output P as the sample points of the final mesh, and stop. If $|P| < \eta_{i+1}$, go to step 3 (i.e., increase mesh size); if $|P| > \eta_{i+1}$, go to step 4 (i.e., decrease mesh size); otherwise, go to step 5 (i.e., bottom of main loop).

3. **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 containing at least one candidate point. That is, of all faces in which a point could be inserted, f^* is the one with the greatest squared error. Next, select a (mutable) point p^* in f^* to add to the mesh as given by

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

   where selCand is a function that embodies the candidate-selection process and is a free parameter of our framework. As for how selCand might be chosen, we defer this discussion until later in Section 5.1.

   (b) Let $P := P \cup \{p^*\}$ (i.e., add p^* to the mesh).

Go to step 5 (i.e., bottom of main loop).

4. **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

$$\text{sigDel}(p) = \sum_{q \in R \cap \Lambda} \left(r^2_{P \setminus \{p\}}(q) - r^2_P(q) \right), \tag{1}$$

   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. Then, select the point p^* to delete from the mesh as

$$p^* = \underset{p \in \text{mutable}(P)}{\arg\min} \ \text{sigDel}(p). \tag{2}$$

   That is, we choose to delete the mutable point having the least significance (i.e., the point that, when deleted, results in the least increase in squared error).

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

5. **Bottom of main loop**. Let $i := i + 1$. Go to step 2 (i.e., top of main loop).

Our framework, as specified above, provides a number of degrees of freedom by leaving open the choice of each of the following: 1) the initial mesh Γ, 2) the growth schedule $\{\eta_i\}_{i=0}^{L}$, and 3) the candidate-selection policy selCand. Of course, in order to produce a concrete (i.e., fully-specified) mesh-generation method, each of the above choices needs to be fixed. As for what specific choices might be effective, we defer this discussion until later in Section 5.

Computational Considerations. In passing, we would like to note that there is a fundamental asymmetry in the manner in which the optAdd and optDel operations are defined in our framework. In particular, a higher degree of symmetry could have been obtained by simply choosing the point p^* to insert as

$$p^* = \underset{p \in \text{mutable}(\Lambda \setminus P)}{\arg\max} \ \text{sigAdd}(p), \tag{3}$$

where sigAdd(p) is the amount by which the squared error decreases when the point p is added to the mesh. Such a choice, however, would be problematic from a computational standpoint. Because the evaluation of the sigAdd function in (3) requires a significant amount of computation and the formula in (3) requires the repeated evaluation of sigAdd for all points in the set mutable($\Lambda \setminus P$) which is typically very large in practice (on the order of the number of samples in the original image), the amount of computation associated with the choice of p^* given by (3) would be much too large to lead to a practical method. So, in short, the asymmetry in the optAdd and optDel operations has been arrived at very deliberately for computational reasons.

Additional Remarks. It is important to note that our proposed framework is fundamentally greedy in nature. That is, the choice of which point to add/delete in a given step is made without regard to how this choice affects *later* steps. Consequently, this framework does not guarantee a globally optimal solution. Practically speaking, however, no *computationally-tractable* algorithm likely exists for producing a globally optimal solution, since mesh-generation problems like the one addressed in this manuscript are NP hard. Next, it is important to understand that the (probably suboptimal) result produced from this framework is *very heavily dependent* on the choice of initial mesh, growth schedule, and candidate-selection policy. Furthermore, computational complexity is also strongly influenced by the preceding choices.

For a fixed initial mesh and candidate-selection policy, a practically limitless number of growth schedules is possible. Different choices of growth schedule lead to different mesh quality and computational complexity tradeoffs, however. For example, suppose that we are given an initial mesh having sample points Γ, where $|\Gamma| = 4$, and we wish to produce a mesh with $N = 104$ sample points. Two possible growth schedules that could be used to accomplish this correspond to the following: 1) perform two optAdd operations followed by one optDel operation, repeated 100 times; or 2) perform 200 optAdd operations followed by 100 optDel operations, only once. Although both scenarios have the same total number of optAdd and optDel operations, the quality of the resulting mesh in each case could be radically different. Also, for reasons that will become clearer later in Section 4, it is more computationally efficient to group optAdd/optDel operations of the same type together. Therefore, scenario 2 from above would most likely require less computation.

In passing, we note that the family of methods generated by our proposed framework includes as very-trivial special cases the GPR and GPRFS-ED schemes mentioned earlier. The GPR method is obtained by choosing $\Gamma = \Lambda$ and the growth setpoint sequence as $\{|\Gamma|, N\}$ (where $N < |\Gamma|$). The GPRFS-ED method is obtained by choosing Γ with the ED scheme such that $|\Gamma| = \min\{4N, |\Lambda|\}$ and the growth setpoint sequence as $\{|\Gamma|, N\}$ (where $N < |\Gamma|$). The preceding special cases are very trivial in the sense that they only involve point deletion, since in both cases $N < |\Gamma|$. The major benefit of our proposed framework and the contribution of the work presented herein, however, come from allowing both point addition and deletion.

4. Implementation of Proposed Mesh-Generation Framework

Although our proposed mesh-generation framework is conceptually simple, implementing it in a computationally efficient manner is tricky and requires careful software design. A naive implementation could easily require several orders of magnitude more computation than is strictly necessary. Below, we offer

some guidance as to how our mesh-generation framework can be efficiently implemented, by describing the particular implementation strategy that we employed.

The mesh-generator state consists primarily of the following: 1) the Delaunay-triangulation data structure, which maintains the mesh geometry and connectivity information; 2) the **vertex priority queue**, a heap-based priority queue with an entry for each mutable point currently in the mesh, where the entry for the point p has priority $-\text{sigDel}(p)$; 3) the **face priority queue**, a heap-based priority queue with an entry for each face f in the mesh satisfying $\text{faceErr}(f) > 0$ (i.e., faces with a strictly positive error), where the entry for face f has priority $\text{faceErr}(f)$; 4) the **vertex scan list**, a doubly-linked list with an entry for each vertex whose priority requires updating due to changes in the mesh. In what follows, we now describe how the optAdd and optDel operations can be implemented.

To perform an optAdd operation, we proceed as follows: 1) Remove the face with the highest priority from the face priority queue, letting f denote the face removed. The point p to be added to the mesh is then $\text{selCand}(f)$. 2) Insert p in the triangulation, letting R denote the region in the triangulation affected by the insertion of p. 3) For each face f in R, recompute $\text{faceErr}(f)$ and update accordingly the priority of f on the face priority queue. 4) For each mutable vertex p in R, add p to the vertex scan list for (possible) later updating of its priority.

To perform an optDel operation, we proceed as follows: 1) For each vertex p on the vertex scan list, remove p from the list, recompute $\text{sigDel}(p)$, and update accordingly the priority of the vertex p on the vertex priority queue. 2) Remove the vertex with the highest priority from the vertex priority queue, letting p denote the vertex removed. The vertex to be deleted is then p. 3) Delete p from the triangulation, letting R denote the region affected by the deletion of p (namely, the faces incident on p). 4) For each face f in R, recompute $\text{faceErr}(f)$ and update accordingly the priority of f on the face priority queue. 5) For each mutable vertex p in R, add p to the vertex scan list for (possible) later updating of its priority. To recompute $\text{sigDel}(p)$ for a given mutable vertex p, we temporarily delete p from the triangulation, and compute the resulting change in the mesh approximation error over the region affected by point deletion (namely, the faces incident on p).

As the description above implies, the computation associated with updating the vertex priority queue (e.g., re-evaluating sigDel values for vertices) is deferred until the result is absolutely needed (namely, when an optDel operation is about to be performed). In situations where multiple optAdd operations are performed without an intervening optDel operation, this deferred processing can save a considerable amount of computation. This savings results from avoiding vertex-priority updates that would later be rendered unnecessary by other optAdd operations. Lastly, in order to further reduce computational complexity, we

employ two additional optimizations: 1) the face priority queue is not initialized until the first optAdd operation; and 2) the vertex priority queue is not initialized until the first optDel operation. These optimizations save considerable time when the mesh-generation process begins with 1) a large number of optAdd operations without an intervening optDel operation; or 2) a large number of optDel operations without an intervening optAdd operation.

5. Proposed Mesh-Generation Methods and Their Development

As seen earlier, our proposed computational framework for mesh generation provides several degrees of freedom by leaving open the choices of initial mesh, growth schedule, and candidate-selection policy. Thus, in order to arrive at a concrete mesh-generation method, we need to fix these degrees of freedom by making specific choices for each of the preceding items. Rather than simply stating the final choices that were made in the case of the concrete mesh-generation methods proposed in this manuscript (which have yet to be introduced), we will instead describe several of the choices that were considered in the development of these methods, along with the rationale of how we selected from amongst them. In so doing, we hope to provide the reader with additional insight into both our proposed mesh-generation framework (introduced earlier) as well as our concrete mesh-generation methods (yet to be introduced).

The remainder of Section 5 is structured as follows. To begin, Section 5.1 presents several choices of growth schedules and candidate-selection policies considered in our work. Next, Section 5.2 introduces an algorithm that can be employed as a postprocessing step to mesh generation in order to further improve mesh quality. In Section 5.3, we proceed to study the relative merits of the various choices/ideas introduced in Sections 5.1 and 5.2. This leads us to recommend particular combinations of choices that correspond to our two proposed concrete mesh-generation methods, which are formally introduced in Section 5.4.

5.1. Initial Mesh, Growth Schedules, and Candidate-Selection Policies

With regard to the choice of initial mesh, we limit our attention herein to schemes that start from a mesh consisting of the (four) extreme convex hull points of Λ. We have found many such schemes to be highly effective with relatively low computational cost.

Growth Schedules. Of the many growth schedules studied in our work, we consider only four herein. Of these four schedules, the most basic is the **incremental (I) growth schedule**, which has a setpoint sequence $\{\eta_i\}_{i=0}^{1}$ given by

$$\eta_0 = |\Gamma| \quad \text{and} \quad \eta_1 = N.$$

This schedule simply results in $N - |\Gamma|$ optAdd operations (and no optDel operations) being performed. The remaining growth schedules are somewhat more complicated, involving both optAdd and optDel operations. The **below (B) growth schedule** has the setpoint sequence $\{\eta_i\}_{i=0}^{L}$ given by

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

where $\alpha \in (0,1)$ and $L = 1 + 2\lfloor -\log_\alpha(N - |\Gamma|)\rfloor$. In the case of this schedule, the mesh size oscillates between N and values below N, with the amplitude of oscillation decaying exponentially at a rate given by the damping parameter α. The **about/circa (C) growth schedule** has a setpoint sequence $\{\eta_i\}_{i=0}^{L}$ given by

$$\eta_i = N + (-1)^{i+1} \left\lfloor \alpha^{\lfloor i/2 \rfloor}(N - |\Gamma|) \right\rfloor,$$

where $\alpha \in (0,1)$ and $L = 2 + 2\lfloor -\log_\alpha(N - |\Gamma|)\rfloor$. In the case of this schedule, the mesh size oscillates about N (i.e., both above and below N), with the amplitude of oscillation decaying exponentially at a rate given by the damping parameter α. Lastly, the **above (A) growth schedule** has a setpoint sequence $\{\eta_i\}_{i=0}^{L}$ given by

$$\eta_i = \begin{cases} |\Gamma| & i = 0 \\ N & i \text{ even, } i \neq 0 \\ N + \left\lfloor \alpha^{(i-1)/2}(N - |\Gamma|) \right\rfloor & i \text{ odd,} \end{cases} \tag{4}$$

where $\alpha \in (0,1)$ and $L = 2 + 2\lfloor -\log_\alpha(N - |\Gamma|)\rfloor$. In the case of this schedule, the mesh size oscillates between N and values above N, with the amplitude of oscillation decaying exponentially at a rate given by the damping parameter α. For comparison purposes, the number of optAdd operations, the number of optDel operations, and the total operation count are given in Table 2 for each of the growth schedules. The information in this table will be used later in order to guide the selection of the damping parameter α.

[Table 2 about here.]

Candidate Selection Policies. Of the many candidate-selection policies studied in our work, we present only four herein, chosen on the basis of their simplicity and/or effectiveness. The first (and simplest) of these four policies is the **peak-absolute-error (PAE) policy**, which chooses the selCand function as

$$\text{selCand}(f) = \underset{p \in \text{cands}(f)}{\arg\max} \left| \hat{\phi}_P(p) - \phi(p) \right|. \tag{5}$$

That is, of all candidate points in the face, this policy selects the point at which the absolute error is greatest. The second policy, known as the **peak-weighted-absolute-error (PWAE) policy** chooses the selCand function as

$$\text{selCand}(f) = \underset{p \in \text{cands}(f)}{\arg\max}\, d(p) \left| \hat{\phi}_P(p) - \phi(p) \right|, \tag{6}$$

where $d(p)$ denotes the **maximum magnitude second-order directional derivative (MMSODD)** of ϕ at p. In other words, of all candidate points in the face, this policy selects the point at which the MMSODD-weighted absolute error is greatest. As a practical matter, in (6), d is computed as given by [29]

$$d\left[(x,y)\right] = \max\{|\alpha(x,y) + \beta(x,y)|\,, |\alpha(x,y) - \beta(x,y)|\}, \tag{7}$$

where $\alpha(x,y) = \frac{1}{2}[\frac{\partial^2}{\partial x^2}\phi(x,y) + \frac{\partial^2}{\partial y^2}\phi(x,y)]$ and $\beta(x,y) = \sqrt{\frac{1}{4}[\frac{\partial^2}{\partial x^2}\phi(x,y) - \frac{\partial^2}{\partial y^2}\phi(x,y)]^2 + [\frac{\partial^2}{\partial x \partial y}\phi(x,y)]^2}$. 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. Furthermore, the partial-derivative operators are applied to a smoothed version of ϕ (rather than ϕ directly), where the smoothing operator employed is the tensor product of two one-dimensional filters with transfer function $z^4(\frac{1}{2} + \frac{1}{2}z^{-1})^8$ (i.e., a ninth-order binomial lowpass filter with zero-phase and unity DC gain). During convolution, domain boundaries are handled by zero extension (i.e., the signal is padded with zeros).

In the PWAE policy, the weighting of the absolute error by d (i.e., the MMSODD) is motivated by the fact that the MMSODD is typically large in locations where the placement of a sample point would be desirable. For example, the MMSODD has a double response to image edges, attaining maxima just to each side of an image edge. This behavior is illustrated by the simple example in Figure 1. From the figure, it is evident that the locations where the MMSODD is large are likely to be good places to select a sample point. Due to this behavior, there is reason to believe that the MMSODD-weighted absolute error in (6) could be an effective means for candidate selection. As we will see later, this suspicion, in fact, turns out to be well founded.

[Figure 1 about here.]

The third candidate-selection policy, known as the **approximate local squared-error minimizer (ALSEM) policy**, chooses the selCand function as

$$\text{selCand}(f) = \arg\max_{p \in S} \sum_{q \in \text{points}(f)} \left(r_P^2(q) - r_{P\cup\{p\}}^2(q)\right) \tag{8}$$

where S is a subset of cands(f), chosen as follows. If $|\text{cands}(f)| > 18$, S is chosen as the 9 points $p \in \text{cands}(f)$ for which $d(p)\left|\hat{\phi}_P(p) - \phi(p)\right|$ is greatest (where $d(p)$ is as defined in (6)) in addition to 9 other randomly-chosen (distinct) points from cands(f); otherwise, $S = \text{cands}(f)$. (The values of 18 and 9 were chosen based on considerable experimentation involving a variety of images and sampling densities.) In (8), the summation corresponds to the reduction in the squared error if p were added to the mesh, computed only *locally* over the points in Λ that belong to the face f.

The PAE and PWAE candidate-selection policies have very low computational overhead. This follows from the simplicity of the expression being maximized in each of (5) and (6). Relative to the PAE and PWAE policies, the ALSEM policy has a much higher computational cost. This computational disparity motivates the last candidate-selection policy, known as the **hybrid policy**. The hybrid policy simply employs the PWAE policy until the first growth-schedule setpoint η_1 is reached, with the ALSEM policy being used thereafter. By using the less-computationally costly PWAE policy initially, computational cost can be significantly reduced (relative to the ALSEM policy).

5.2. Bad-Point Replacement (BPR)

In the preceding section, several options for growth schedules and candidate-selection policies were presented. Before proceeding further, we need to introduce a basic algorithm that is beneficial in the context of our proposed mesh-generation framework. As a matter of terminology, a (mutable) point p in the mesh is said to be **bad**, if $\text{sigDel}(p) \leq 0$ (i.e., the deletion of p would not cause an increase in the mesh approximation error). Clearly, bad points are undesirable since their inclusion in the mesh either increases the mesh approximation error (if $\text{sigDel}(p) < 0$) or leaves the mesh approximation error unchanged (if $\text{sigDel}(p) = 0$). As it turns out, with our framework, when the target number of points is finally achieved and the mesh-generation process would normally terminate, there is the possibility that some bad points will be present in the mesh. Depending on the choice of initial mesh, growth schedule, and candidate-selection policy, the number of bad points could, in fact, be quite large. At first, it might seem counterintuitive that deleting points from the mesh could actually *decrease* the approximation error. The mesh approximation error, however, depends not only on the sample points, but also the topology of the triangulation associated with these points. Therefore, removing a point could, for example, cause a triangulation edge that crosscuts an image edge to be eliminated, thus having the potential to reduce approximation error.

To combat the degradation in mesh quality caused by the presence of bad points, we have devised a technique for eliminating such points called the **bad-point-replacement (BPR)** method. This method works by deleting bad points and substituting other new points in their place. This is done in such a way as to not result in any net change in the number of points in the mesh (i.e., the number of optAdd operations and number of optDel operations employed are equal). This method is intended to be performed as a final postprocessing step in the mesh-generation process, after a mesh with the target number of points has been obtained. In more detail, the BPR method consists of the following steps: 1) Let $n_{\text{old}} := \infty$ 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 $\text{sigDel}(p) \leq 0$, perform an optDel operation (to delete p), mark p as immutable, and let $n := n + 1$. 3) If

$n > 0$, perform n optAdd operations. 4) If $n \geq n_{\text{old}}$, let $c := c + 1$. 5) Let $n_{\text{old}} := n$; if $n = 0$ or $c \geq 3$, stop; otherwise, go to step 2. In step 2, p is marked as immutable in order to prevent p from being added back to the mesh in subsequent iterations, which could cause the algorithm to become trapped in an infinite loop, repeatedly cycling through the same sequence of optAdd and optDel operations. The counter c is used to allow the algorithm to terminate early in the case that convergence is abnormally slow, which has been observed to occur occasionally for some very simple synthetic images.

Although the BPR scheme was initially developed as a tool for potential use in the mesh-generation methods proposed later in this manuscript, it is important to note that our BPR method can be used as a postprocessing step added to other arbitrary (i.e., not necessarily derived from our framework) mesh-generation methods in order to improve the resulting mesh quality. That is, we can take a mesh M produced by another arbitrary method, use M as the initial mesh for our framework, and then simply invoke our BPR scheme to produce the new mesh M'. Provided that M had some bad points (which is the case for many methods), we can expect the new mesh M' to be of higher quality than the original mesh M. As we will see later, some previously proposed schemes, although quite effective, often produce meshes with a significant number of bad points.

5.3. Analysis of the Available Options

Having introduced numerous options (i.e., growth schedules, candidate-selection policies, and BPR) that could be used to construct a concrete mesh-generation method, we will now study each of these options more carefully. Through this analysis, a better understanding of the issues surrounding these options can be obtained, which will ultimately allow for the formulation of more effective mesh-generation methods.

Comparison of Growth Schedules. To begin, let us examine the impact of the choice of growth schedule on mesh quality. To do this, we fix the candidate-selection policy to be PWAE, disable the use of BPR, and then select from amongst the various growth schedules under consideration (namely, the I, B, C, and A growth schedules). In order to place the B, C, and A growth schedules on approximately equal footing, the damping parameter α is chosen in each case, through the use of Table 2 (for large d), such that the three growth schedules yield (approximately) the same total number of optAdd/optDel operations. In particular, we have chosen the damping parameter α as 0.625, 0.25, and 0.4 for the B, C, and A growth schedules, respectively.

For all 40 images in our test set and 6 sampling densities, we generated a mesh using each of the various growth schedules and measured the resulting mesh approximation error in terms of PSNR. Individual results for three specific images (namely, those listed in Table 1) are given in Table 3(a). Also, for each of the

240 test cases (i.e., 40 images with 6 sampling densities per image), the PSNR performance of the four approaches was ranked from 1 (best) to 4 (worst), and the average and standard deviation of the ranks were computed for each sampling density as well as overall, with the results shown in Table 3(b). (The standard deviations are the numbers shown in parentheses in the table.) To assist in visualizing trends, in each row of the tables, the best and second-best results are shown in **bold** and ***gray italic***, respectively. Examining the statistical results of Table 3(b), we can see that the standard deviations are all very small and in many cases zero, meaning that the rankings are very consistent across test cases. Clearly, from the table, the A growth schedule ranks best, followed (in order) by the C, B, and I growth schedules. The results for individual test cases shown in Table 3(a) can be seen to be consistent with the statistical results, with the A growth schedule faring best, outperforming the C, B, and I growth schedules by margins of up to 0.21 dB, 0.31 to 0.69 dB, and 1.91 to 4.61 dB, respectively. It is worth noting that the I growth schedule performs especially poorly, producing results that are very substantially worse (i.e., by more than 1 dB) than even its closest competitor (namely, the B growth schedule). Although we have elected to present results for one particular choice of candidate-selection policy (namely, PWAE), we have observed similar trends with other candidate-selection policies. Based on the above results, we recommend the use of the A growth schedule.

As we saw above, the I growth schedule performs quite poorly relative to the other growth schedules. There is, however, a good reason for this behavior, as we shall now explain. The I growth schedule starts with a nearly empty mesh (of only four sample points) and achieves the target mesh size *only* by adding points (i.e., points are *never* deleted). Because the underlying framework is greedy in nature, the preceding approach has a fundamental weakness. Since the framework is greedy, it will unavoidably make some bad decisions regarding points to add/delete, and such bad choices lead to a degradation in mesh quality. Therefore, in order to achieve the highest possible mesh quality, a mechanism is needed for allowing bad decisions to be reversed. In the case of the I growth schedule, no such mechanism exists. If the decision to add a particular point turns out later to have been a bad one, there is no way for the point to be deleted. For similar reasons, a growth schedule based solely on point deletion (without ever adding points) is also fundamentally weak. In contrast, the B, C, and A growth schedules alternate between the addition and deletion of points. Thus, if a bad choice is made in adding a point, the choice can effectively be undone by later deleting the same point. Similarly, if a bad choice is made in deleting a point, the choice can be effectively undone by later adding the same point. Consequently, growth schedules that alternate between the addition and deletion of points are fundamentally more robust to the bad choices inherent in a greedy framework. In fact, it was precisely the preceding observation that motivated our proposal of a mesh-generation framework that is based on *both* the addition and deletion of points.

To better illustrate the shortcomings of a growth schedule that only involves the addition of points, we consider an example. For a simple color-wheel image, a mesh with a sampling density of 0.09% was generated using each of the I and B growth schedules. The reconstructed images and corresponding image-domain triangulations are shown in Figure 2. The I growth schedule, starting from a nearly empty mesh (having only four sample points), simply adds points until the target number of sample points is achieved. From the resulting triangulation, shown in Figure 2(b), we can see that this process leads to an undesirable clustering of points. Although the B growth schedule also starts in an identical fashion to the I growth schedule, initially obtaining exactly the same triangulation as in Figure 2(b), instead of stopping as in the case of the I growth schedule, the B growth schedule continues to delete and add points, leading to the final triangulation shown in Figure 2(d). Observe that the poor decisions initially made, which led to an undesirable clustering of points, have been undone, resulting in a vastly superior triangulation. This example clearly demonstrates the importance of having a mechanism whereby earlier bad decisions to add/delete points can be reversed.

The above explains the poor performance of the I growth schedule. The relative performance of the remaining three growth schedules (namely, B, C, and A) is strongly influenced by the average mesh size associated with each growth schedule. In particular, in the case of the B, C, and A growth schedules, the average mesh size is below, approximately equal to, and above the target number N of points, respectively. Having a larger average mesh size is advantageous, as it offers more possibilities for the points that are eventually chosen for the final mesh. The A growth schedule, which has the highest average mesh size performs best, while the B growth schedule, which has the lowest average mesh size, performs worst.

[Table 3 about here.]

[Figure 2 about here.]

Recall that each of the B, C, and A growth schedules are associated with a damping parameter α. Therefore, one might reasonably wonder what effect this parameter has on mesh quality. In the case of all three growth schedules, the clear trend is for mesh quality to improve as α is increased. To more concretely demonstrate this behavior, we provide some experimental results in Table 4. This table shows the mesh quality obtained for the A growth schedule with three different choices of damping parameter α. The tendency for mesh quality to increase with α is clearly evident in these results. The above behavior can be easily explained. As α increases, the oscillations in mesh size decay more slowly, resulting in more alternations between adding and deleting points. This provides more opportunity for bad choices to be

undone, yielding higher mesh quality. Of course, the higher mesh quality comes at the cost of increased computational complexity. As a consequence of the above behavior, by making different choices for α, we can tradeoff mesh quality against computational complexity.

[Table 4 about here.]

Comparison of Candidate-Selection Policies. Next, we examine the impact of the choice of candidate-selection policy on mesh quality. To do this, we fix the growth schedule to be A with the damping parameter $\alpha = 0.4$, disable the use of BPR, and then select from amongst the various candidate-selection policies under consideration (namely, the PAE, PWAE, ALSEM, and hybrid policies). For all 40 images in our test set and 6 sampling densities, we generated a mesh using each of the various candidate-selection policies and then measured the resulting mesh approximation error in terms of PSNR. Individual results for three specific images (namely, those listed in Table 1) are given in Table 5(a). Also, for each of the 240 test cases (i.e., 40 images with 6 sampling densities per image), the PSNR performance of the four candidate-selection policies was ranked from 1 (best) to 4 (worst), and the average and standard deviation of the ranks were computed for each sampling density as well as overall, with the results shown in Table 5(b). (The standard deviations are the numbers shown in parentheses in the table.) Again, the best and second-best results in each case are shown in **bold** and ***gray italic***, respectively.

Examining the statistical results averaged across all images as shown in Table 5(b), we can see that the ALSEM policy clearly performs best, followed by (in order) the hybrid, PWAE, and PAE policies. A more detailed analysis shows that the PAE policy is worst in 233/240 (97%) of the test cases (with one rare exception being the case of the `bull` image at a sampling density of 1.0% as shown in Table 5(a)). Also, the hybrid policy was observed to perform better than the PAE and PWAE policies in 238/240 (99%) of the test cases. Looking at the results for individual test cases shown in Table 5(a), we can see that the best result is consistently produced by either the ALSEM or hybrid policy. In particular, the ALSEM policy beats the PAE and PWAE policies by margins of 0.20 to 1.07 dB and up to 1.10 dB, respectively, while the hybrid policy beats the PAE and PWAE policies by margins of 0.47 to 0.95 dB and 0.08 to 0.77 dB, respectively. Although best on average (as shown by the results of Table 5(b)), the ALSEM policy is not always best in every individual case. In particular, for some images and sampling densities, the hybrid policy can sometimes perform better than the ALSEM policy. Typically, this tends to happen in instances where the PWAE policy also beats the ALSEM policy (e.g., the cases of the `ct` and `bull` images in Table 5(a)). Although the above results are specifically for the A growth schedule, we should note that similar trends were observed for the other growth schedules as well. As we will see later, the ALSEM policy typically

requires about 25% to 50% more computation than the hybrid policy. For the above reasons, we recommend the use of both the ALSEM and hybrid policies. The ALSEM policy is best when (average) mesh quality is the only consideration, while the hybrid policy is best when computational complexity also needs to be taken into account. Unlike the PAE and PWAE policies, the ALSEM and hybrid policies directly consider the change in approximation error (i.e., squared error) that results from adding a new point to the mesh. By directly taking this error into account, the ALSEM and hybrid policies are able to make more effective choices (at candidate selection), leading to higher-quality meshes.

[Table 5 about here.]

Utility of Bad-Point Replacement (BPR). Now, we consider the effectiveness of the BPR scheme introduced earlier. To accomplish this, we fix the growth schedule to be I and the candidate-selection policy to be PWAE, and then allow BPR to be either used or not used. For all images in our test set and several sampling densities, we generated a mesh both with and without using BPR and measured the resulting approximation error in terms of PSNR. A representative subset of the results is shown in Table 6. From these results, it is clear that BPR has the potential to offer a significant improvement in mesh quality, relative to not using BPR. In particular, for the results given, BPR beats no BPR by a margin of about 0.81 to 2.90 dB. Although the results shown here are specifically for the I growth schedule and PWAE candidate-selection policy, BPR was also found to be beneficial for other combinations of growth schedule and candidate-selection policy as well. Essentially, BPR has the potential to improve mesh quality any time that bad points are present. For other choices of growth schedule and candidate-selection policy, the number of bad points is sometimes smaller, and the benefit of BPR is less pronounced. This said, however, there is no harm in always including BPR, as it incurs essentially no extra computational cost in the case that no bad points are present in mesh. For this reason, we recommend that BPR always be employed.

[Table 6 about here.]

5.4. Proposed Mesh-Generation Methods

Equipped with a good understanding of the effectiveness of the various growth schedules, candidate-selection policies, and BPR, we are now in a position to introduce the two concrete mesh-generation methods proposed herein. Earlier, we found the A growth schedule, the ALSEM and hybrid candidate-selection policies, and BPR to be most effective. So, not surprisingly, our methods utilize the preceding choices. The first of our proposed mesh-generation methods, known as ID1, employs the A growth schedule, hybrid candidate-selection policy, and BPR. The second method, known as ID2, is identical to the ID1 method,

except that the ALSEM candidate-selection policy is employed instead of the hybrid policy. Since the ID1 and ID2 methods both employ the A growth schedule, a choice must be made for the damping parameter α associated with this growth schedule. For both the ID1 and ID2 methods, we nominally choose $\alpha = 0.4$, as experimentation has shown this value (of α) to provide good performance at a reasonable computational cost. The ID1 and ID2 methods are intended to offer somewhat different tradeoffs between mesh quality and computational complexity, as will be seen later.

Having introduced our proposed ID1 and ID2 mesh-generation methods, we now comment on the differences between these methods and the GPR, GPRFS-ED, and IDDT schemes introduced earlier. As we have seen, the ID1 and ID2 methods are not based exclusively on mesh refinement or exclusively on mesh simplification. With these two methods, as part of the mesh-generation process, points are *both* added to and deleted from the mesh. In particular, the ID1 and ID2 schemes start with a trivial mesh consisting of the (four) extreme convex-hull points of the image domain, and then add and delete points according to a growth schedule that is carefully designed to yield high-quality meshes. Unlike the ID1 and ID2 schemes, the GPR and GPRFS-ED methods are based *exclusively* on mesh simplification. The GPR and GPRFS-ED methods start with very large meshes, and then *only* delete points until the desired sampling density is achieved. Points are *never* added to the mesh with these schemes. In the case of the GPR scheme, the initial mesh is chosen to consist of all sample points of the original image, while in the case of the GPRFS-ED scheme, the initial mesh is chosen as a subset of the sample points of the original image, using an error diffusion process (without any need for geometric algorithms). Since the GPR and GPRFS-ED methods only exclusively delete points, no mechanism exists for undoing bad decisions (of points deleted). As experimental results later confirm, this inability to reverse bad decisions limits the performance of these methods. In contrast, the ID1 and ID2 methods do not suffer from this shortcoming. Lastly, the IDDT method from our earlier conference paper [32] is obtained from the mesh-generation framework proposed herein by choosing the candidate-selection policy as PWAE, the growth schedule as B with $\alpha = \frac{1}{2}$, and the use of BPR. As one might suspect (and is later confirmed by experimental results), the IDDT method performs quite poorly relative to the ID1 and ID2 schemes, due to its choice of a poorer performing candidate-selection policy and growth schedule.

6. Evaluation of the Proposed Methods

Having introduced our ID1 and ID2 mesh-generation methods, we now compare their performance to that of the state-of-the-art GPRFS-ED and GPR schemes (mentioned earlier) in terms of both mesh quality and computational/memory complexity. For convenience, in what follows, we denote our ID1 and ID2

schemes with the damping parameter α as ID1(α) and ID2(α), respectively. In our evaluation, we consider the performance of our ID1(0.4) and ID2(0.4) schemes (i.e., the ID1 and ID2 methods with the damping parameter chosen as the nominal value 0.4). Since, as was demonstrated earlier, higher mesh quality can be obtained by increasing the damping parameter α used in the ID1 and ID2 methods, we also consider the ID1(0.9) scheme in our evaluation in order to show what range of mesh-quality performance is possible if one is willing to incur greater computational cost. The implementations of the GPRFS-ED and GPR methods used in this evaluation are taken from [6] (and are written in C++). The software implementing our ID1 and ID2 methods was developed by the author and is also written in C++. As an aside, we recall that, for the mesh-generation problem being addressed herein, the mesh approximation of an image is required to interpolate the original image at each of the mesh sample points. Consequently, the GPR method does not employ a scheme like the least squares approximation technique described in [3]. In passing, we note that although the GPR scheme is known for producing very high quality meshes and is used for comparison purposes herein, if one is willing to incur additional complexity, another state-of-the-art method proposed in [41], called AT*, can produce even much higher quality meshes than the original GPR scheme and possibly other methods considered herein.

Mesh Quality. For all 40 images in our test set and 7 sampling densities, we used each of the various methods under consideration to generate a mesh, and then measured the resulting approximation error in terms of PSNR. Individual results for three specific images (namely, those listed in Table 1) are given in Table 7(a). For each of the 280 test cases (i.e., 40 images with 7 sampling densities per image), the PSNR performance of the six methods was ranked from 1 (best) to 6 (worst), and the average and standard deviation of the ranks were computed for each sampling density as well as overall, with the results shown in Table 7(b). (The standard deviations are the numbers shown in parentheses in the table.) In the tables, the best and second best results in each row are indicated by **bold** and ***gray italic***, respectively. To demonstrate that the ID1 and ID2 methods proposed herein make a very substantial contribution beyond the IDDT method from the author's earlier conference paper [32] (mentioned in Section 1), results for the IDDT method are also included in these tables for comparison purposes.

To begin, we compare the ID1 and ID2 methods to the IDDT scheme. From the statistical results given in Table 7(b), we can see that the ID1 and ID2 methods outperform the IDDT scheme by a wide margin, with the IDDT scheme having a much poorer average ranking (with relatively low standard deviation). In fact, a more detailed analysis of the results shows that the ID1(0.9), ID2(0.4), and ID1(0.4) methods beat the IDDT scheme in 280/280 (100%), 279/280 (99%), and 280/280 (100%) of the test cases. Examining the results for the individual test cases in Table 7(a), we observe that the ID1 and ID2 methods beat the

IDDT scheme by a margin of 0.20 to 2.12 dB. Thus, the ID1 and ID2 methods represent a very substantial contribution beyond the author's conference paper which proposed the IDDT scheme. Since the ID1 and ID2 methods are clearly superior to the IDDT scheme, we will not consider the IDDT scheme further in our evaluation.

Now, we compare the ID1 and ID2 methods to the GPRFS-ED and GPR schemes. First, let us consider the statistical results taken across all 40 test images as given in Table 7(b). The small standard deviations in the table are indicative that the rankings are fairly consistent across test cases. Examining this table, we see that: 1) the ID1(0.9) method ranks best overall (with rank 1.04) and best at each sampling density; 2) the ID2(0.4) method ranks second best overall (with rank 2.16) and second best at each sampling density; and 3) the ID1(0.4) method ranks third best overall (with rank 3.46), third best at 4/7 of the sampling densities, and third best at all sampling densities above 0.5%. The fact that the ID1(0.9) and ID2(0.4) methods quite consistently rank in first and second place, respectively, is suggested by their corresponding low standard deviations (nearly all of which are below 0.5). Furthermore, a more detailed analysis shows that: 1) the ID1(0.9) method beats the GPRFS-ED and GPR schemes in 280/280 (100%) and 277/280 (99%) of the test cases, respectively; 2) the ID2(0.4) method beats the GPRFS-ED and GPR schemes in 277/280 (99%) and 269/280 (96%) of the test cases, respectively; 3) the ID1(0.4) method beats the GPRFS-ED and GPR schemes in 210/280 (75%) and 184/280 (66%) of the test cases, respectively. As the results of Table 7(b) suggest, the relative performance of the ID1(0.4) method relative to the GPR scheme improves as the sampling density increases. In fact, for sampling densities above 0.5%, the ID1(0.4) method beats the GPR scheme in 146/160 (91%) of the test cases. As we shall see later, at sampling densities below 0.5%, the GPR scheme requires over 23 times more computation time and over 100 times more memory relative to the ID1(0.4) method. So, in any cases where these lower sampling densities might be of practical interest, the extreme savings in (computational and memory) complexity offered by the ID1(0.4) method (relative to the GPR scheme) more than compensates for the performance difference between the ID1(0.4) and GPR methods. Thus, in terms of average behavior, our ID1(0.4), ID1(0.9) and ID2(0.4) methods are clearly superior to the state-of-the-art GPRFS-ED and GPR schemes. Moreover, in our ID1(α) method, as α is varied from 0.4 to 0.9, we obtain a very substantial improvement in mesh quality, at the cost of increased computational complexity. Thus, we can effectively tradeoff mesh quality against computation time by varying α.

Next, let us consider results for the individual test cases as given in Table 7(a). From these results, we can see that: 1) the ID1(0.9) method is the clear winner, performing best in 20/21 test cases, being beaten by only 0.02 dB in one instance (namely, the `ct` image at a sampling density of 4%); 2) the ID2(0.4) method

is second best overall, ranking second and third in 13/21 and 8/21 of the cases, respectively; 3) the ID1(0.4) method is third best overall, ranking first/second and third place in 8/21 and 11/21 of the cases, respectively. These observations are consistent with the statistical results above from Table 7(b). A close examination of the numbers reveals that: 1) the ID1(0.9) method beats the GPRFS-ED and GPR schemes by margins of 0.26 to 6.17 dB and 0.30 to 1.92 dB, respectively; 2) the ID2(0.4) method beats the GPRFS-ED and GPR schemes by margins of 0.11 to 5.49 dB and 0.16 to 1.24 dB, respectively; 3) the ID1(0.4) method beats the GPRFS-ED and GPR schemes by margins of 0.04 to 5.73 dB and up to 1.48 dB, respectively. So, in terms of the individual test cases, our ID1(0.4), ID1(0.9), and ID2(0.4) methods are also clearly superior to the GPRFS-ED and GPR schemes (sometimes by margins of a few dB). The excellent performance of our ID1(0.4) and ID2(0.4) methods relative to the GPR scheme is particularly impressive given that, as we will see later, they require very substantially less (often by more than an order of magnitude) computation and memory.

In the above evaluation, PSNR was found to correlate reasonably well with subjective quality. For the benefit of the reader, however, we provide an example illustrating the subjective quality achieved by the various methods. In particular, for one of the test cases involving the `bull` image from Table 7(a), a small part of each image reconstruction is shown under magnification in Figure 3. Examining this figure closely, we can see that the reconstructions produced by the ID1(0.4), ID2(0.4), and ID1(0.9) methods in Figures 3(a) to (c) are better than those obtained from the GPRFS-ED and GPR schemes in Figures 3(d) and (e). In particular, the reconstructions from the GPRFS-ED and GPR schemes (especially the former) tend to have more disturbing triangle-tooth artifacts along image edges than the reconstructions produced by our proposed methods.

[Table 7 about here.]

[Figure 3 about here.]

Computational Complexity. Next, we consider the computational complexity of the various mesh-generation methods under consideration. For the purposes of this evaluation, computational complexity is measured in terms of execution time. Before proceeding to present any experimental results, we need to make one very important comment regarding the various software implementations used in our evaluation. In particular, the implementations of the GPRFS-ED and GPR methods from [6] (used in our evaluation) are much more highly optimized for execution speed than the software that implements our ID1 and ID2 methods. Although our software was carefully designed to be reasonably efficient, little effort was made to

optimize the code beyond this basic level of efficiency. Even more importantly, our software was designed to be general enough to handle any concrete mesh-generation method based on our framework (from Section 3). Because our code had to handle this very general case, the extent to which specific cases (such as the ID1 and ID2 methods) could be optimized was severely limited. So, for the above reasons, the implementation of the GPRFS-ED and GPR methods has a very unfair advantage in terms of execution time. This very important fact must be taken into consideration when interpreting the experimental results presented in what follows.

For the several test cases from Table 7(a) involving the `lena` image, the time required for mesh generation was measured, yielding the results shown in Table 8. (These measurements were made on relatively mediocre hardware, namely a seven-year old notebook computer with a 3.4 GHz Intel Pentium 4 and 1 GB of RAM.) As mentioned earlier, the GPRFS-ED and GPR methods can be viewed as very-trivial special cases of our framework. In fact, our software implementation of our computational framework also supports both of these methods. So, for the GPRFS-ED and GPR methods, we provide two sets of numbers. The first set was obtained with the implementation from [6], while the second set, given in parentheses, is obtained by using our implementation (i.e., as a special case of our very general code).

To begin, let us focus only on the first set of numbers for the GPRFS-ED and GPR methods. The main observations that we want to make are as follows. First, in spite of the ID1(0.4) and ID2(0.4) methods producing better quality meshes than the GPR scheme (as seen earlier), these better results are obtained with much less computation time. (Again, keep in mind that the GPR scheme has an unfair advantage, being more highly optimized.) In particular, the ID1(0.4) and ID2(0.4) methods have execution times about 0.04 to 0.56 times and 0.06 to 0.68 times those of the GPR scheme, respectively. Second, in spite of the ID1(0.4) and ID2(0.4) methods producing significantly better quality meshes than the GPRFS-ED scheme (as seen earlier), this better quality does not come at the cost of an order of magnitude (i.e., ten times) more computation. The computational cost is, in fact, quite reasonable. That is, our ID1(0.4) and ID2(0.4) methods have execution times about 1.27 to 2.02 times and 1.77 to 2.45 times those of the GPRFS-ED scheme, respectively. (Again, keep in mind that the GPRFS-ED scheme has an unfair advantage, being more highly optimized.) Third, we observe that for the ID1(α) method, as α increases (from 0.4 to 0.9), so too does computation time. A similar behavior is also exhibited in the case of the ID2(α) method. Thus, in our ID1 and ID2 methods, by varying α, we can tradeoff between mesh quality and computational complexity.

Now, let us also consider the second set of numbers (in parentheses) for the GPRFS-ED and GPR methods. Observe that the execution times for the GPRFS-ED and GPR schemes are significantly higher

in the case of our implementation (i.e., the numbers in parentheses) than in the case of the implementation from [6] (i.e., the numbers not in parentheses). This difference in execution times shows that the two implementations are not equally optimized, as we indicated above. Following this line of reasoning, we argue that if our ID1 and ID2 methods were implemented with a level of optimization equal to the GPRFS-ED and GPR schemes, it is quite likely that our ID1(0.4) method would be faster than the GPRFS-ED scheme, and our ID2(0.4) method would likely be only marginally slower than, or even comparable in speed to, the GPRFS-ED scheme. In this sense, our ID1(0.4) and ID2(0.4) schemes compare very favorably to the GPRFS-ED scheme in terms of computational complexity.

Memory Complexity. Lastly, it is worthwhile to briefly comment on the memory complexity of the various mesh-generation methods. The amount of memory required by each of these methods is largely determined by the peak mesh size (i.e., the maximum number of sample points in the mesh). For an image of width W and height H, and a sampling density D (where typically $D < \frac{8}{100} = 8\%$), the peak mesh size for each of the GPRFS-ED and GPR schemes is $4DWH$ and WH, respectively. In the case of our ID1 and ID2 methods, the peak mesh size is $2DWH$. So, all of our proposed methods have a peak mesh size that is smaller than in the cases of the GPRFS-ED and GPR schemes by factors of 2 times (independent of sampling density) and 6.25 to 400 times (for sampling densities from 8% to 0.125%), respectively.

[Table 8 about here.]

BPR and Other Mesh-Generation Methods. Earlier, we mentioned that our BPR scheme can be added as a postprocessing step to other previously-proposed mesh-generation methods in order to produce meshes of higher quality. Now, we provide an example to substantiate this claim. In particular, we consider the ED method (mentioned earlier). As it turns out, the ED method typically produces meshes where about 50% of the sample points are bad. (Note that, in this context, we mean "bad" in the specific sense introduced previously in Section 5.2.) Thus, the ED method can potentially benefit from the use of our BPR scheme. With this in mind, we propose a modified version of the ED method, called **optimized ED (OED)**, which includes our BPR scheme as a postprocessing step. That is, the OED method first employs the ED scheme to produce a mesh with the desired number of sample points, and then our BPR scheme is applied to the resulting mesh. For all 40 images in our test set and 7 sampling densities (280 test cases in total), each of the ED and OED methods was used to generate a mesh and the resulting mesh approximation error was measured in terms of PSNR. Individual results for three images are given in Table 9, with the best result in each case indicated in **bold**. Examining the results of the table, we find the OED method outperforms the ED scheme in each case, by margins of about 2.95 to 16.48 dB. Furthermore, a more detailed examination

of the results for all 280 test cases (i.e., 40 images with 7 sampling densities per image) shows that the OED method outperforms the ED scheme in 280/280 (100%) of the test cases, by margins varying from about 2 to 17 dB. Clearly, the addition of the BPR scheme to the ED method (resulting in our OED method) has led to a very marked improvement in mesh quality. From these results, it is clear that our BPR method has potential value in further optimizing meshes produced by other (previously-proposed) methods. In passing, we note that the BPR postprocessing step added (to the ED scheme) in the OED method only adds a few seconds of extra computation time for an image such as `lena`. So, the increase in mesh quality does not come at an exorbitant computational cost.

[Table 9 about here.]

7. Conclusions

In this manuscript, we have proposed a flexible new mesh-generation framework for image representation, along with two concrete methods derived from this framework, known by the names ID1 and ID2. Through experimental results, the ID1 and ID2 methods were shown to perform extremely well, producing meshes of significantly higher quality than the state-of-the-art GPRFS-ED and GPR methods at a reasonably modest computational cost. Furthermore, the ID1 and ID2 methods each provide a means whereby mesh quality can be increased (or decreased) in return for a corresponding increase (or decrease) in computational cost (i.e., by varying the damping parameter α). This computational scalability makes our methods suitable for a wide range of applications with differing computational constraints. One component of our proposed methods, called BPR, was shown to be highly effective as a postprocessing step to improve upon the results produced by other mesh-generation methods. In particular, this postprocessing strategy was shown to yield much higher quality meshes when applied to the previously-proposed ED scheme. In short, the methods that we have proposed can benefit the numerous applications where adaptively-sampled image representations are needed. Moreover, by further exploring the many other algorithmic possibilities that our proposed framework affords, we are optimistic that it will be possible to derive even more effective mesh-generation schemes.

References

[1] M. Sarkis, K. Diepold, Content adaptive mesh representation of images using binary space partitions, IEEE Trans. on Image Processing 18 (5) (2009) 1069–1079.

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

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

[4] L. Demaret, A. Iske, Scattered data coding in digital image compression, in: Curve and Surface Fitting: Saint-Malo 2002, Nashboro Press, Brentwood, TN, USA, 2003, pp. 107–117.

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

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

[7] M. D. Adams, Progressive lossy-to-lossless coding of arbitrarily-sampled image data using the modified scattered data coding method, in: Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2009, pp. 1017–1020.

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

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

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

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

[12] J. G. Brankov, Y. Yang, M. N. Wernick, Tomographic image reconstruction based on a content-adaptive mesh model, IEEE Trans. on Medical Imaging 23 (2) (2004) 202–212.

[13] M. A. Garcia, B. X. Vintimilla, Acceleration of filtering and enhancement operations through geometric processing of gray-level images, in: Proc. of IEEE International Conference on Image Processing, Vol. 1, 2000, pp. 97–100.

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

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

[16] Y. Wang, O. Lee, A. Vetro, Use of two-dimensional deformable mesh structures for video coding, part II–the analysis problem and a region-based coder employing an active mesh representation, IEEE Trans. on Circuits and Systems for Video Technology 6 (6) (1996) 647–659.

[17] F. Davoine, M. Antonini, J.-M. Chassery, M. Barlaud, Fractal image compression based on Delaunay triangulation and vector quantization, IEEE Trans. on Image Processing 5 (2) (1996) 338–346.

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

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

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

[21] N. Dyn, D. Levin, S. Rippa, Data dependent triangulations for piecewise linear interpolant, IMA Journal of Numerical Analysis 10 (1990) 137–154.

[22] L. Darsa, B. Costa, Multiresolution representation and reconstruction of adaptively sampled images, in: Proc. of SIBGRAPI, 1996, pp. 321–328.

[23] G. Wolberg, Nonuniform image reconstruction using multilevel surface interpolation, in: Proc. of IEEE International Conference on Image Processing, Vol. 1, 1997, pp. 909–912.

[24] Y. Eldar, M. Lindenbaum, M. Porat, Y. Y. Zeevi, The farthest point strategy for progressive image sampling, IEEE Trans. on Image Processing 6 (9) (1997) 1305–1315.

[25] M. Kashimura, Y. Sato, S. Ozawa, Image description for coding using triangular patch structure, in: Proc. of IEEE International Conference on Communications Systems, 1992, pp. 330–334.

[26] M. Grundland, C. Gibbs, N. A. Dogson, Stylized multiresolution image representation, Journal of Electronic Imaging 17 (1) (2008) 013009.1–17.

[27] D. Terzopoulos, M. Vasilescu, Sampling and reconstruction with adaptive meshes, in: Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, pp. 70–75.

[28] M. D. Adams, An evaluation of several mesh-generation methods using a simple mesh-based image coder, in: Proc. of IEEE International Conference on Image Processing, 2008, pp. 1041–1044.

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

[30] L. Rila, Image coding using irregular subsampling and Delaunay triangulation, in: Proc. of SIBGRAPI, 1998, pp. 167–173.

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

[32] M. D. Adams, An incremental/decremental Delaunay mesh-generation framework for image representation, in: Proc. of IEEE International Conference on Image Processing, 2011, pp. 193–196.

[33] Kodak lossless true color image suite, `http://r0k.us/graphics/kodak` (2011).

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

[35] USC-SIPI image database, `http://sipi.usc.edu/database` (2011).

[36] Michael Adams' research datasets, `http://www.ece.uvic.ca/~mdadams/datasets` (2011).

[37] C. Dyken, M. S. Floater, Preferred directions for resolving the non-uniqueness of Delaunay triangulations, Computational Geometry—Theory and Applications 34 (2006) 96–101.

[38] J. R. Shewchuk, What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures, Tech. rep., Department of Computer Science, University of California at Berkeley, Berkeley, CA, USA (Dec. 2002).

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

[40] K. Fleischer, D. Salesin, Accurate polygon scan conversion using half-open intervals, in: Graphics Gems III, 1995, pp. 362–365.

[41] L. Demaret, N. Dyn, A. Iske, Image compression by linear splines over adaptive triangulations, Signal Processing 86 (2006) 1604–1616.

List of Figures

(a) (b) (c)

Figure 1: MMSODD example. (a) Full image, showing a rectangular region of interest. (b) Region of interest under magnification and the (c) corresponding MMSODD.

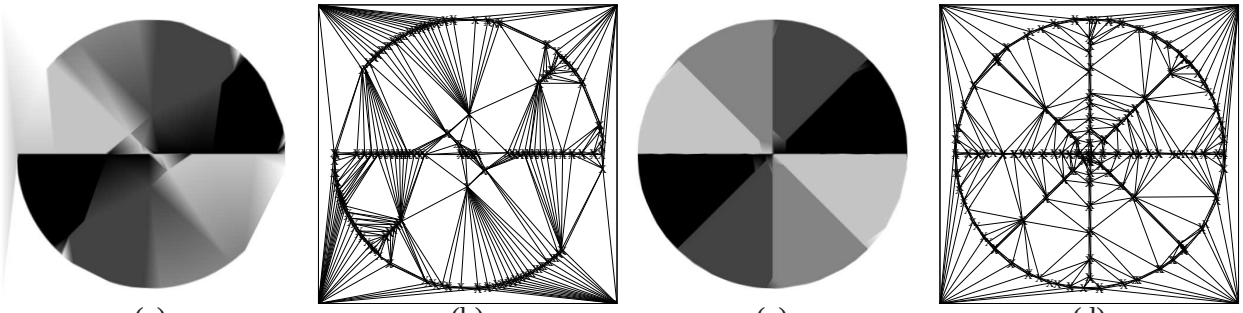

Figure 2: The reconstructed images obtained for the `wheel` image at a sampling density of 0.09% using the (a) I (21.49 dB) and (c) B (31.87 dB) growth schedules and (b) and (d) their corresponding triangulations.

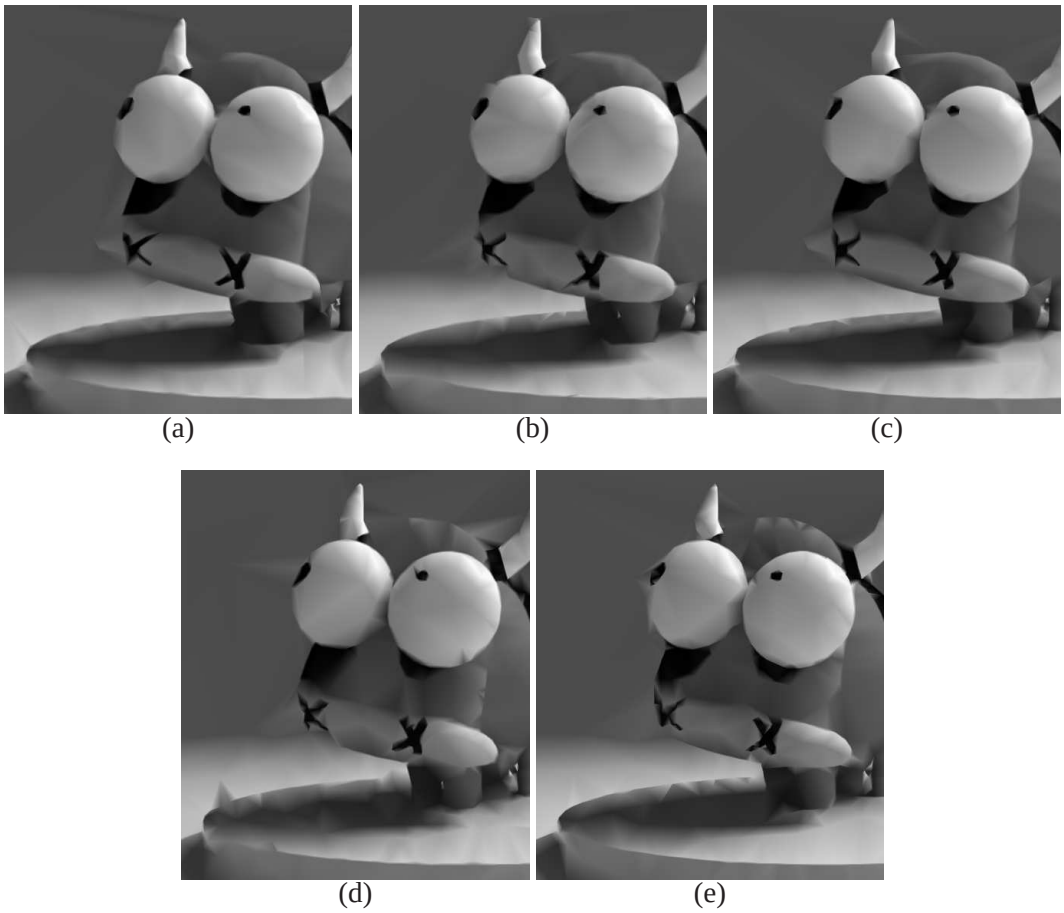

Figure 3: Comparison of the subjective mesh-quality for the various methods. Part of the reconstructed image obtained for the `bull` image at a sampling density of 0.125% using each of the (a) ID1(0.4) (34.60 dB), (b) ID2(0.4) (34.36 dB), (c) ID1(0.9) (35.04 dB), (d) GPRFS-ED (28.87 dB), and (e) GPR (33.12 dB) methods.

List of Tables

Table 1: Test images

Name	Size, Bits/Sample	Description
`bull`	1024×768, 8	computer-generated, cartoon bull [36]
`ct`	512×512, 12	tomography [34]
`lena`	512×512, 8	woman [35]

Table 2: Operation counts for the various growth schedules, where $d = N - |\Gamma|$, $\ell = \left\lfloor \log_{1/\alpha} d \right\rfloor$, and $\beta = \frac{1-\alpha^{\ell+1}}{1-\alpha}$.

Growth Schedule	optAdd Count	optDel Count	Total Count	Total Count, Large d
I	$= d$	$= 0$	$= d$	$= d$
B	$\approx \beta d$	$\approx (\beta - 1)d$	$\approx (2\beta - 1)d$	$\approx \left(\frac{1+\alpha}{1-\alpha}\right) d$
C	$\approx 2\beta d$	$\approx (\alpha + 1)\beta d$	$\approx (\alpha + 3)\beta d$	$\approx \left(\frac{3+\alpha}{1-\alpha}\right) d$
A	$\approx (\beta + 1)d$	$\approx \beta d$	$\approx (2\beta + 1)d$	$\approx \left(\frac{3-\alpha}{1-\alpha}\right) d$

Table 3: Comparison of the mesh quality obtained with the various growth schedules. (a) PSNRs for three specific images. (b) Rankings averaged across 40 images.

(a)

Image	Samp. Density (%)	PSNR (dB) I	B	C	A
`bull`	0.125	29.82	34.07	***34.37***	**34.43**
	0.250	34.74	37.75	***38.29***	**38.38**
	0.500	38.07	40.59	***41.04***	**41.25**
	1.000	40.65	42.51	***42.99***	**43.14**
`ct`	0.250	29.91	32.70	***32.97***	**33.01**
	0.500	35.37	37.54	***38.12***	**38.16**
	1.000	39.49	41.59	***41.85***	**41.91**
	2.000	43.40	45.37	**45.77**	***45.75***
`lena`	0.500	23.99	25.89	***26.37***	**26.51**
	1.000	27.19	28.46	***29.02***	**29.10**
	2.000	29.81	31.12	***31.60***	**31.78**
	3.000	31.23	32.51	***33.06***	**33.20**

(b)

Samp. Density (%)	Mean Rank† I	B	C	A
0.125	4.00 (0.00)	3.00 (0.00)	***1.98*** (0.16)	**1.02** (0.16)
0.250	4.00 (0.00)	3.00 (0.00)	***2.00*** (0.00)	**1.00** (0.00)
0.500	4.00 (0.00)	3.00 (0.00)	***1.98*** (0.16)	**1.02** (0.16)
1.000	4.00 (0.00)	3.00 (0.00)	***1.98*** (0.16)	**1.02** (0.16)
2.000	4.00 (0.00)	3.00 (0.00)	***1.98*** (0.16)	**1.02** (0.16)
3.000	4.00 (0.00)	3.00 (0.00)	***1.95*** (0.22)	**1.05** (0.22)
Overall	4.00 (0.00)	3.00 (0.00)	***1.98*** (0.16)	**1.02** (0.16)

†The standard deviation is given in parentheses.

Table 4: Effect of varying the damping parameter α in the A growth schedule

Image	Samp. Density (%)	PSNR (dB) α 0.4	0.6	0.9
`bull`	0.125	34.43	**34.54**	***34.54***
	0.250	38.38	***38.48***	**38.86**
	0.500	41.25	***41.36***	**41.45**
	1.000	43.14	***43.18***	**43.32**
`ct`	0.250	33.01	***33.02***	**33.25**
	0.500	38.16	***38.18***	**38.24**
	1.000	41.91	***41.96***	**42.01**
	2.000	45.75	***45.79***	**45.85**
`lena`	0.500	26.51	***26.52***	**26.63**
	1.000	29.10	***29.20***	**29.26**
	2.000	31.78	***31.83***	**31.90**
	3.000	33.20	***33.29***	**33.37**

Table 5: Comparison of the mesh quality obtained with the various candidate-selection policies. (a) PSNRs for three specific images. (b) Rankings averaged across 40 images.

(a)

Image	Samp. Density (%)	PSNR (dB) PAE	PWAE	ALSEM	Hybrid
`bull`	0.125	33.65	***34.43***	34.36	**34.60**
	0.250	38.23	38.38	**38.97**	***38.89***
	0.500	41.22	41.25	**42.25**	***41.95***
	1.000	43.17	43.14	**44.24**	***43.91***
`ct`	0.250	32.79	***33.01***	32.99	**33.26**
	0.500	37.58	***38.16***	37.97	**38.24**
	1.000	41.30	***41.91***	41.80	**42.05**
	2.000	45.33	***45.75***	45.59	**45.85**
`lena`	0.500	26.30	26.51	**27.05**	***26.99***
	1.000	28.83	29.10	**29.56**	***29.51***
	2.000	31.49	31.78	***32.12***	**32.15**
	3.000	32.98	33.20	**33.65**	***33.63***

(b)

Samp. Density (%)	Mean Rank† PAE	PWAE	ALSEM	Hybrid
0.125	3.88 (0.40)	3.08 (0.35)	**1.05** (0.32)	***2.00*** (0.23)
0.250	3.97 (0.16)	3.00 (0.23)	**1.05** (0.32)	***1.98*** (0.16)
0.500	4.00 (0.00)	2.97 (0.16)	**1.12** (0.40)	***1.90*** (0.30)
1.000	3.97 (0.16)	2.97 (0.28)	**1.20** (0.52)	***1.85*** (0.36)
2.000	3.95 (0.32)	2.92 (0.35)	**1.27** (0.64)	***1.85*** (0.43)
3.000	4.00 (0.00)	2.92 (0.27)	**1.30** (0.61)	***1.77*** (0.42)
Overall	3.96 (0.23)	2.98 (0.28)	**1.17** (0.49)	***1.89*** (0.34)

†The standard deviation is given in parentheses.

Table 6: Effect of using BPR on mesh quality

Image	Samp. Density (%)	PSNR (dB)	
		No BPR	BPR
`bull`	0.125	29.82	**32.72**
	0.250	34.74	**36.78**
	0.500	38.07	**39.92**
	1.000	40.65	**42.09**
`ct`	0.250	29.91	**31.11**
	0.500	35.37	**36.60**
	1.000	39.49	**40.76**
	2.000	43.40	**44.56**
`lena`	0.500	23.99	**25.34**
	1.000	27.19	**28.07**
	2.000	29.81	**30.62**
	3.000	31.23	**32.14**

Table 7: Comparison of the mesh quality for the various methods. (a) PSNRs for three specific images. (b) Rankings averaged across 40 images.

(a)

Image	Samp. Density (%)	PSNR (dB) ID1(0.4)	ID2(0.4)	ID1(0.9)	GPRFS-ED	GPR	IDDT
bull	0.125	***34.60***	34.36	**35.04**	28.87	33.12	33.86
	0.250	38.89	***38.97***	**39.43**	35.88	38.23	37.53
	0.500	41.95	***42.25***	**42.40**	39.78	41.87	40.48
	1.000	43.91	***44.24***	**44.37**	43.50	43.99	42.43
	2.000	45.80	***46.09***	**46.23**	45.65	45.81	44.39
	4.000	48.29	***48.42***	**48.55**	47.98	48.24	47.04
	8.000	52.03	***52.04***	**52.24**	51.48	51.88	51.14
ct	0.125	28.51	***28.73***	**29.15**	23.94	28.17	27.54
	0.250	***33.26***	32.99	**33.42**	30.38	32.15	32.25
	0.500	***38.24***	37.97	**38.39**	36.86	37.22	37.59
	1.000	***42.05***	41.80	**42.12**	40.73	41.35	41.42
	2.000	***45.85***	45.59	**45.93**	44.63	45.33	45.39
	4.000	**50.24**	50.05	***50.22***	49.62	49.79	49.72
	8.000	***55.18***	55.08	**55.19**	54.81	54.89	54.63
lena	0.125	22.26	***22.45***	**22.92**	19.70	21.90	20.80
	0.250	24.55	***24.67***	**25.30**	23.05	24.42	23.25
	0.500	26.99	***27.05***	**27.51**	26.21	26.66	25.81
	1.000	29.51	***29.56***	**29.90**	29.05	29.12	28.54
	2.000	***32.15***	32.12	**32.41**	31.95	31.82	31.09
	4.000	34.68	***34.69***	**34.95**	34.56	34.39	33.56
	8.000	37.23	***37.30***	**37.45**	37.19	36.99	36.06

(b)

Samp. Density (%)	Mean Rank† ID1(0.4)	ID2(0.4)	ID1(0.9)	GPRFS-ED	GPR	IDDT
0.125	3.88 (0.56)	***1.98*** (0.42)	**1.15** (0.36)	5.47 (0.64)	3.10 (0.67)	5.43 (0.55)
0.250	3.72 (0.64)	***2.05*** (0.32)	**1.05** (0.22)	5.20 (0.56)	3.28 (0.68)	5.70 (0.52)
0.500	3.62 (0.90)	***2.12*** (0.40)	**1.02** (0.16)	4.93 (0.62)	3.45 (0.64)	5.85 (0.43)
1.000	3.50 (0.96)	***2.20*** (0.46)	**1.00** (0.00)	4.10 (1.01)	4.30 (0.76)	5.90 (0.38)
2.000	3.38 (0.81)	***2.22*** (0.48)	**1.00** (0.00)	3.83 (0.93)	4.70 (0.72)	5.88 (0.46)
4.000	3.17 (0.87)	***2.25*** (0.49)	**1.02** (0.16)	3.92 (0.83)	4.68 (0.66)	5.95 (0.22)
8.000	2.92 (0.69)	***2.30*** (0.52)	**1.02** (0.16)	4.12 (0.88)	4.72 (0.45)	5.90 (0.50)
Overall	3.46 (0.84)	***2.16*** (0.46)	**1.04** (0.19)	4.51 (1.01)	4.03 (0.94)	5.80 (0.47)

†The standard deviation is given in parentheses.

Table 8: Comparison of the computation time for the various methods in the case of the `lena` image

Samp. Density (%)	Time (s)				
	ID1(0.4)	ID2(0.4)	ID1(0.9)	GPRFS-ED	GPR
0.500	1.847	2.769	9.050	1.205 (1.852)	42.996 (212.343)
1.000	2.576	3.734	14.623	1.862 (2.550)	42.670 (211.359)
2.000	4.143	5.660	26.201	3.196 (4.409)	42.126 (211.094)
4.000	7.377	10.576	56.260	5.766 (9.335)	41.049 (209.105)
8.000	22.097	26.891	240.891	10.935 (26.124)	39.305 (206.994)

Table 9: Comparison of the mesh quality for the ED and OED methods (for three specific images).

Image	Samp. Density (%)	PSNR (dB) ED	PSNR (dB) OED
`bull`	0.125	15.54	**32.02**
	0.250	20.59	**36.44**
	0.500	25.89	**40.15**
	1.000	33.34	**42.20**
	2.000	37.56	**44.06**
	4.000	41.51	**46.23**
	8.000	44.85	**49.76**
`ct`	0.125	16.61	**26.72**
	0.250	17.81	**30.70**
	0.500	21.61	**35.82**
	1.000	29.45	**40.44**
	2.000	35.62	**44.02**
	4.000	41.53	**48.13**
	8.000	46.74	**51.89**
`lena`	0.125	13.78	**20.37**
	0.250	14.49	**23.10**
	0.500	17.17	**25.37**
	1.000	21.13	**27.92**
	2.000	25.83	**30.61**
	4.000	29.59	**33.08**
	8.000	32.52	**35.47**