Improved Mesh Models of Images Through the Explicit Representation of Discontinuities

Xi Tu and Michael D. Adams*, *Senior Member, IEEE*

***Abstract*—A new mesh model for images that explicitly represents image discontinuities (i.e., image edges), called ERD, is introduced. Then, two mesh-generation methods, known as ERDED and ERDGPI, that select the parameters of this new model for a given image are proposed. The proposed mesh-generation methods are shown to be capable of producing image approximations of higher quality than other competing strategies, both in terms of squared error and subjective quality.**

***Index Terms*—Image representation, nonuniform sampling, triangle mesh, mesh generation, constrained Delaunay triangulation.**

I. Introduction

Traditionally, images are most often represented using uniform sampling on a lattice. Unfortunately, uniform sampling is almost always suboptimal, placing too many sample points in regions of slow variation and too few sample points in regions of rapid change. This shortcoming has generated considerable interest in image representations based on nonuniform sampling [1]–[6]. Nonuniform sampling can produce much more compact representations of images, which is beneficial in many applications. Furthermore, representations based on nonuniform sampling are often better equipped to capture the geometric structure inherent in images (namely, image edges). Representations based on nonuniform sampling have proven to be useful for many tasks, such as feature detection [7], pattern recognition [8], computer vision [9], restoration [10], tomographic reconstruction [11], filtering [12], interpolation [13], and image coding [14].

Of the many classes of representations based on nonuniform sampling, (triangle) mesh models have become quite popular. With a mesh model, a triangulation is used to partition the image domain into triangle faces, and then an approximating function for the image is constructed over each face of the triangulation. Mesh models can be classified on the basis of the type of triangulation employed (e.g., Delaunay [15], [16], constrained Delaunay [17], or data dependent [16]) as well as how the approximating function is constructed over the triangulation (e.g., the continuity or smoothness of the approximating function, and whether it interpolates the original data). Traditionally, the overall approximating function is chosen to be continuous (e.g., [15]–[17]). Most images, however, tend to have a significant number of discontinuities (i.e., image edges). Therefore, there is reason to believe that a model that allows for discontinuities in the approximating function may be beneficial. In fact, this is one of the main motivations of the work described herein. In order to employ a mesh model, we need a means to select the model parameters so as to obtain the best possible approximation of a given image. This is the so called mesh-generation problem. Of the many mesh-generation methods proposed to date, two highly effective ones that are of particular interest herein are the error-diffusion (ED) scheme of Yang et al. [15] and the greedy point-insertion (GPI) scheme [18] (called "MGH" therein) which was inspired by the work of Garland and Heckbert in [16].

In this manuscript, we introduce a new mesh model for images, called ERD, that is based on constrained Delaunay triangulations [19], and then propose two mesh-generation methods, known as ERDED and ERDGPI, that employ this model. With our ERD mesh model, the approximating function is not required to be continuous everywhere, with discontinuities being permitted across (constrained) edges of faces in the triangulation. In this way, our model can explicitly represent discontinuities (i.e., image edges), which, as we will see, allows for much more compact image representations. Although our proposed methods require more computation time than the ED and GPI methods, this increase in time is relatively small in absolute terms (i.e., a fraction of a second), as shown later. This modest increase in computation time fits with the goal of our work, which was to develop methods that can produce better quality meshes than the ED and GPI schemes, while still only requiring on the order of a few seconds of computation time (as opposed to minutes/hours).

The remainder of this manuscript is structured as follows. Section II presents some background information on mesh models for images and briefly describes the (previously-proposed) ED and GPI mesh-generation methods. Our new ERD mesh model for images, which explicitly represents discontinuities (i.e., image edges), is introduced in Section III, and then Section IV proposes two mesh-generation methods that utilize this new model, the ERDED and ERDGPI methods. In Section V, we compare the performance of our proposed ERDED and ERDGPI methods to several other competing schemes. Through experimental results, our approaches are shown to yield image approximations of significantly better quality, in terms of squared error and subjective quality, relative to competing methods. Finally, Section VI concludes the manuscript with a summary of our key results. In passing, we note that the work described herein has been partially presented in our conference paper [20], which introduced an earlier version of our ERDED method. Herein, we build on this earlier work by proposing the entirely new ERDGPI method as well as further refining the earlier version of the ERDED

*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.

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

method from our conference paper.

II. Mesh Models of Images

Before proceeding further, we need to introduce the formal definition of a triangulation and some related concepts. Let $\mathbb{Z}$ and $\mathbb{R}$ denote the sets of integers and real numbers, respectively. A triangulation of a set V of points in $\mathbb{R}^2$ is a set T of triangles such that: 1) the union of the vertices of all triangles in T is V; 2) the interiors of any two triangles in T are disjoint; and 3) the union of the triangles in T is the convex hull of V. A triangulation is said to be Delaunay [21] if each triangle in the triangulation is such that the interior of its circumcircle contains no vertices. The Delaunay triangulation of a set of points is unique if certain degeneracies are handled appropriately (e.g., using a technique like preferred directions [22]). Sometimes, we would like a triangulation of a set of points to contain a certain prescribed set of line segments as edges, called edge constraints. A triangulation with such edge constraints is referred to as a constrained triangulation. In the context of constrained triangulations, two points are said to be visible to each other if the line segment joining them does not intersect an edge constraint. A triangulation of a set V of points with edge constraints E is said to be constrained Delaunay [19] if each triangle in the triangulation is such that: 1) the interior of the triangle does not intersect any constraining line segment; and 2) no vertex inside the triangle's circumcircle is visible from any point in the interior of the triangle. Essentially, a constrained Delaunay triangulation is a triangulation that is as close as possible to being Delaunay, subject to the constraint that certain edges must appear in the triangulation.

Having formally introduced triangulations, we can now explain how triangulations are used to build mesh models of images. Consider an image ϕ defined on the rectangular region $\Gamma = [0, W-1] \times [0, H-1]$. We assume that ϕ is known only at each of the points in $\Lambda = \{0,1,\ldots,W-1\} \times \{0,1,\ldots,H-1\}$ (i.e., the points on a rectangular grid of width W and height H). (Note that $\Lambda = \Gamma \cap \mathbb{Z}^2$ and Γ is the convex hull of Λ.) With a mesh model, we choose a set P of points in Γ, called sample points, where the points in P are not necessarily required to be in Λ (i.e., the points at which the function ϕ is known). Then, P is triangulated to partition the (continuous) image domain Γ into triangle faces, and an approximating function is constructed over each face in the triangulation. Finally, the approximating functions for all of the faces are combined to yield a function $\hat{\phi}$ that approximates ϕ over the entire image domain Γ. The set P must include the extreme convex hull points of Γ (i.e., the four corners of the image bounding box) so that the triangulation of P covers all of Γ. As a matter of terminology, the quantity $|P|/|\Lambda|$ is referred to as the sampling density of the mesh model. In our work, the mesh-generation problem can be succinctly stated as follows: Given an image ϕ and a desired number N of sample points, find a mesh approximation $\hat{\phi}$ of ϕ with N sample points (i.e., $|P| = N$) such that the difference between $\hat{\phi}$ and ϕ is minimized. In our work, this difference is measured using the mean squared error (MSE), which is given by $\mathsf{MSE} = |\Lambda|^{-1} \sum_{p \in \Lambda} \big(\hat{\phi}(p) - \phi(p)\big)^2$. The MSE is usually expressed in terms of the peak signal-to-noise ratio (PSNR), which is defined as $\mathsf{PSNR} = 20 \log_{10}\left(\frac{2^\rho - 1}{\sqrt{\mathsf{MSE}}}\right)$, where ρ is the sample precision in bits/sample. An increase in the PSNR (in dB) by Δp corresponds to a decrease in the root MSE by the factor $10^{\Delta p/20}$. For example, a 2 dB increase in the PSNR corresponds to a reduction in the root MSE by a factor of $10^{2/20} \approx 1.2589$.

What distinguishes different types of mesh models from one another is: 1) which points in Γ are permitted to be chosen as points in P; 2) how the triangulation of P is formed; and 3) how the approximating function is determined over each face of the triangulation. The most commonly used mesh model, which we refer to as the basic model herein, is quite simple and employs Delaunay triangulations. With the basic model, the sample points P are chosen as a subset of Λ, and P is triangulated using the Delaunay triangulation. Over each (triangle) face in the triangulation, $\hat{\phi}$ is defined as the unique linear function that interpolates ϕ at the three vertices of the face. Thus, the approximating function $\hat{\phi}$ is continuous and interpolates ϕ at each point in P. Furthermore, this model is completely characterized by the sample points P and $\phi(p)$ for $p \in P$.

Although many mesh-generation methods have been proposed for selecting the parameters of the basic model for a given image, two particularly effective ones are the ED and GPI methods mentioned earlier. The ED method [15] uses Floyd-Steinberg error diffusion to distribute sample points such that the local density of these points is proportional to the maximum magnitude second-order directional derivative (MMSODD) of the image. (The formula for the MMSODD appears as equation (6) in [15].) Since the ED method has a number of degrees of freedom, we note that, in our work, we consider the variant of this method that employs a third-order binomial filter for noise removal during MMSODD estimation, a contrast sensitivity parameter γ of 1, and the serpentine scan order for error diffusion. Also, the first- and second-order derivative operators used in the calculation of the MMSODD were approximated by filters with transfer functions $\frac{1}{2}z - \frac{1}{2}z^{-1}$ and $z - 2 + z^{-1}$, respectively. Lastly, we note that, during filtering, signal boundaries are handled by zero extension (i.e., padding with zeros).

The GPI method [18] is iterative in nature. It starts with P selected as the four corner points of the image bounding box. Then, in each iteration, a new sample point is selected to be added to P. The new sample point is selected in two steps. First, the face in the triangulation with the greatest squared error is selected. Then, the point in that face at which the absolute error is greatest is chosen for addition.

III. Proposed Mesh Model

The basic mesh model used by the ED and GPI schemes (introduced earlier) is always associated with an approximating function that is continuous. Images, however, often contain a significant number of discontinuities (i.e., image edges). This observation motivated us to propose a new mesh model that explicitly represents discontinuities in images, known as the ERD model (where ERD stands for "explicit representation of

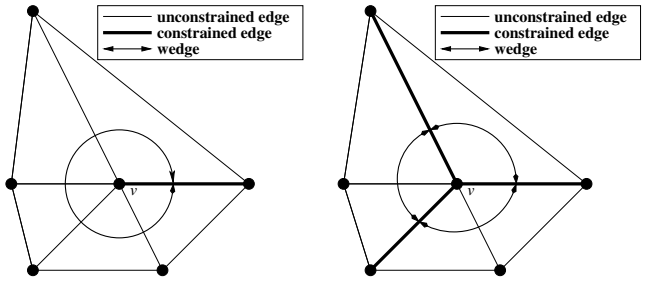


Fig. 1. The relationship between vertices, constrained edges, and wedges. The (a) single-wedge and (b) multiple-wedge cases.

discontinuities"). Our ERD model makes use of constrained Delaunay triangulations. Our model for the image ϕ is completely characterized by: 1) a set $P = \{p_i\}$ of sample points, where $p_i = (x_i, y_i) \in \Gamma \cap \frac{1}{2}\mathbb{Z}^2$; 2) a set E of edge constraints (i.e., a set of pairs of sample points from P); and 3) for each sample point p_i, one or more wedge values, where the term "wedge value" will be defined precisely later. The quantities P and E along with the associated wedge values are used to determine the function $\hat{\phi}$ defined on Γ which approximates ϕ. Note that, with our model, the sample points in P are chosen on twice as fine a grid as the original image being represented (i.e., $\Gamma \cap \frac{1}{2}\mathbb{Z}^2$ as opposed to $\Gamma \cap \mathbb{Z}^2$). This is done in order to allow for more accurate representation of image edges. As we will see, $\hat{\phi}$ is chosen to interpolate ϕ at each point in $\mathbb{Z}^2 \cap P$. In what follows, we explain how $\hat{\phi}$ is defined in terms of P, E, and the wedge values.

First, we construct a constrained Delaunay triangulation of P with the constrained edges E, which serves to partition the image domain Γ into triangle faces. The constrained edges are chosen to correspond to image edges. For each vertex $v \in P$, the set of faces incident on v is partitioned into what are called wedges. In particular, a wedge is a set of consecutive faces in a loop around a vertex v that are not separated by any constrained edge. This definition is illustrated in Fig. 1. If the number of constrained edges incident on the vertex v is zero or one, all faces incident on v form a single wedge, as shown in Fig. 1(a). Otherwise, if n constrained edges are incident on v (where $n \geq 2$), the faces incident on v form n wedges, as shown in Fig. 1(b). Wedges are used to facilitate the modelling of discontinuities (i.e., image edges). Since constrained edges are chosen to correspond to image edges, a vertex $v \in P$ that has more than one wedge must be located along a discontinuity (i.e., image edge). Each wedge of a vertex has associated with it what is called a wedge value. The wedge value z of the wedge w belonging to vertex v specifies the limit of $\hat{\phi}(p)$ as p approaches v from points inside the wedge w.

Now, we specify precisely how the function $\hat{\phi}$ is defined at each point $p \in \Gamma$. There are two cases to consider: 1) p is not on a constrained edge; 2) p is on a constrained edge.

Case 1. First, let us consider the case that p is not on a constrained edge. Let f denote a face of the triangulation with vertices $p_i = (x_i, y_i)$, $p_j = (x_j, y_j)$, and $p_k = (x_k, y_k)$ that contains the point p. Let z_i, z_j, and z_k denote the wedge values for the face f corresponding to the vertices p_i, p_j, and p_k, respectively. Then, $\hat{\phi}(p) = g(p)$, where the function g is the unique linear interpolant (i.e., plane) that passes through the points (x_i, y_i, z_i), (x_j, y_j, z_j), and (x_k, y_k, z_k).

Case 2. Next, let us consider the case that p is on a constrained edge. If p is not an endpoint of a constrained edge, $\hat{\phi}(p)$ is the average of the values on the two sides of the image discontinuity (computed as in case 1). On the other hand, if p is an endpoint of a constrained edge (i.e., a vertex in the triangulation), $\hat{\phi}(p)$ is the average of all wedge values for (the vertex) p.

From the mesh model $\hat{\phi}$, a lattice-sampled image can be reconstructed by straightforward rasterization algorithms. Due to the fact that our ERD model explicitly represents discontinuities, image edges could produce undesirable aliasing effects if the samples of the (discrete) image reconstruction were generated by simply evaluating $\hat{\phi}$ at points in Λ. Consequently, in the case of our ERD model, rasterization is performed using the well-known 4×4 supersampling technique [23], as this approach avoids such aliasing effects.

IV. Proposed Mesh-Generation Methods

Having introduced our ERD mesh model, we now propose two mesh-generation methods, called ERDED and ERDGPI, to be used in conjunction with this model. For a given image ϕ sampled at the points of the rectangular grid Λ, each of these methods selects the parameters of the model (i.e., P, E, and wedge values) so as to obtain the best possible approximation of ϕ for a specified target number N of sample points (i.e., $|P| = N$). Since the ERDED and ERDGPI methods fit into the same general algorithmic framework, we first introduce this framework. Then, we give the specifics of each of these methods. The algorithmic framework employed by both methods consists of the following steps:

1) *Initial triangulation.* Select initial values for P and E. This determines the initial triangulation (i.e., the constrained Delaunay triangulation of P with edge constraints E). Let $N_0 = |P|$ (i.e., N_0 is the initial mesh size).
2) *Initial wedge values.* For each vertex $v \in P$, calculate the wedge value for each wedge of v.
3) *Point selection.* Select a new sample point p^* to add to the mesh.
4) *Point insertion.* Insert the point p^* in the (constrained Delaunay) triangulation. If p^* is on a constrained edge, split the edge at p^* into two constrained edges, and compute the wedge value for each wedge of the vertex p^*. If p^* is not on a constrained edge, the wedges remain the same and no wedge values need to be recomputed.
5) *Stopping criterion.* If $|P| < N$, go to step 3 (i.e., add another sample point to the mesh).

In the framework above, steps 1 and 2 choose an initial coarse mesh of size N_0, and steps 3 to 5 iteratively refine the mesh by adding $N - N_0$ sample points to the mesh. The initial coarse mesh selection (i.e., steps 1 and 2) is performed in an identical manner for both the ERDED and ERDGPI methods. The two methods differ only in the approach used to refine the mesh (i.e., steps 3 to 5). First, we describe in more detail steps 1 and 2, which are identical for both the ERDED and ERDGPI methods.

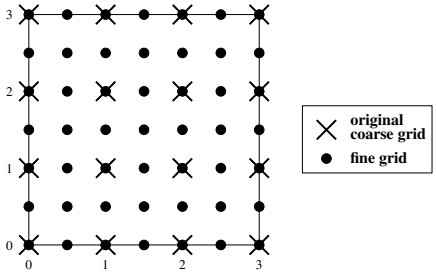


Fig. 2. Relationship between grids for a 4×4 image.

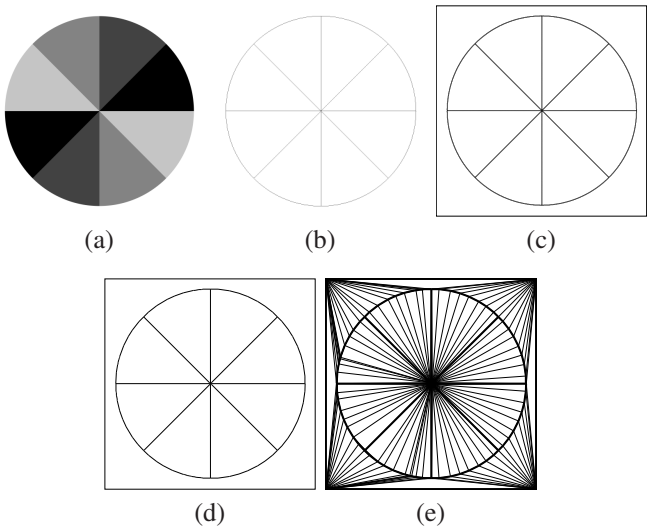


Fig. 3. Selection of the initial triangulation. (a) Original image. (b) Binary edge map. (c) Unsimplified polylines representing image edges. (d) Simplified polylines representing image edges. (e) Initial triangulation (with constrained edges denoted by thick lines).

A. Initial Coarse Mesh Selection

Recall that, with our ERD model, the sample points P are chosen as a subset of $\Gamma \cap \frac{1}{2}\mathbb{Z}^2$. That is, the grid on which the sample points lie is a grid with its spacing in the horizontal and vertical directions each reduced by half relative to the grid Λ on which ϕ is originally sampled. Since the original image ϕ is sampled on a W by H grid, this implies that the sample points (in P) are chosen to fall on a $(2W-1)$ by $(2H-1)$ grid. The relationship between these two grids is illustrated in Fig. 2 for the case of a 4×4 image (i.e., $W = H = 4$). In what follows, let $\bar{\phi}$ denote the function defined on Γ formed by the bilinear interpolation [24] of ϕ. By definition, $\bar{\phi}$ satisfies $\bar{\phi}(p) = \phi(p)$ for all $p \in \Lambda$.

Step 1. In step 1 of our framework, the selection of the initial triangulation consists of four substeps, which are numbered 1.1 to 1.4 below. These substeps are described in detail in the paragraphs that follow and are also illustrated by way of the example shown in Fig. 3.

1.1) Locate edges. First, we employ the modified Canny edge detector in [25] to locate edges in the image ϕ with half-pixel resolution. To accomplish this, we apply the edge detector to $\bar{\phi}$ sampled on the rectangular grid $\Gamma \cap \frac{1}{2}\mathbb{Z}^2$ to produce a binary edge map of dimensions $(2W-1) \times (2H-1)$. (An entry in the edge map is one if it corresponds to an edge pixel, and zero otherwise.) Note that the grid on which $\bar{\phi}$ is sampled here is twice as fine (in each dimension) as the grid Λ on which the original image ϕ is sampled. By applying the edge detector to this higher resolution version of the original image, we can locate edges with half-pixel accuracy. The edge detector works by computing the gradient magnitude and direction and then using this information in conjunction with hysteresis thresholding to select edge pixels. Two parameters must be specified as input to the edge detector, namely, the low and high thresholds for hysteresis thresholding, denoted herein as τ_{low} and τ_{high}, respectively. In our method, these edge-detector thresholds are controlled by the parameters β and r. The quantity τ_{high} is chosen such that the fraction of pixels (from $\bar{\phi}$) whose corresponding gradient magnitude is greater than or equal to τ_{high} is β (i.e., the edge detector will nominally produce at least $\beta \left|\Gamma \cap \frac{1}{2}\mathbb{Z}\right|$ edge pixels). Then, τ_{low} is selected as $\tau_{\mathsf{low}} = r\tau_{\mathsf{high}}$. To reduce the effects of noise, a smoothing operation (i.e., lowpass filter) is included in the convolution kernel used for gradient calculation. That is, the filter used to estimate each partial derivative is the composition of a first-order derivative operator and smoothing operator, where the first-order derivative operator is a filter with transfer function $\frac{1}{2}z - \frac{1}{2}z^{-1}$ and the smoothing operator is a fifth-order binomial filter [26]. Since edges in the edge map can be more than one pixel wide, we apply the line thinning algorithm from [27] to reduce the thickness of edges. The edge detection process is illustrated in Fig. 3. In particular, given the input image in Fig. 3(a), edge detection would produce a binary image resembling that shown in Fig. 3(b), where edge pixels are shown in black.

1.2) Construct polylines for edges. Having generated the edge map, we next construct a polyline representation of each edge in the edge map. To accomplish this, 8-connected edge pixels in the edge map are joined (by line segments) to form polylines. In cases where a polyline has one or more self-intersections (excluding loops), the polyline is split at each intersection point. In this way, the final set of polylines is guaranteed not to have any self-intersections (excluding loops). This process is illustrated in Fig. 3. Given the edge map shown in Fig. 3(b), we would produce a set of polylines like that shown in Fig. 3(c).

1.3) Simplify polylines. After polylines have been constructed to represent image edges, we need to simplify these polylines. In other words, for each polyline we find a new polyline with fewer vertices (i.e., control points) that well approximates the original polyline. To perform polyline simplification, we employ the well known Douglas-Peucker (DP) [28] algorithm. For a given polyline, the DP scheme repeatedly (using a greedy approach) adds points to a trivial two-point approximation of the polyline until the resulting approximation error is less than a prespecified tolerance. In our method, each polyline is simplified using the DP algorithm with tolerance ε. Then, we discard any polylines with fewer than ℓ vertices, where ℓ is a parameter of our method. Polylines with only a few points are eliminated, as such polylines tend to be associated with false edges introduced by noise and degrade mesh quality. This process is illustrated in Fig. 3. Given the set of polylines shown in Fig. 3(c), we would produce a set of simplified polylines like that shown in Fig. 3(d).

1.4) Select P and E from polylines. Having obtained the set of simplified polylines, we now use those polylines in order to select P and E. Since the extreme convex-hull points of Λ (i.e., the four corner points of image bounding box) must be included in P, these four points are always forced to be included in P. We choose P as the union of all of the polyline vertices and select E as the set of line-segments from all of the polylines. Then, we form the constrained Delaunay triangulation of P with edge constraints E. This process is illustrated in Fig. 3. Given the simplified polylines shown in Fig. 3(d), we would produce the triangulation shown in Fig. 3(e), where constrained edges are denoted by thick lines.

Step 2. Having selected the initial triangulation, we now need to choose the wedge values. In particular, for each wedge w of each vertex $v \in P$, we must select the corresponding wedge value z. The selection of z is performed in one of two ways, depending on the number n of wedges associated with the vertex v. If $n = 1$, we simply choose z as $z = \bar{\phi}(v)$. Otherwise (i.e., if $n \geq 2$), we proceed as follows. Let f denote the MMSODD of ϕ (which is calculated according to equation (6) in [15]) and let d denote a unit vector in the direction of the ray originating from v and bisecting the wedge w. We select z as $z = \bar{\phi}(v')$, where $v' = v + \alpha^* d$ and $\alpha^* = \arg\max_{\alpha \in [1,1.5]} f(v + \alpha d)$. In other words, v' is chosen as the result of a local line search that maximizes the MMSODD along the ray bisecting w. The line search is restricted to $\alpha \in [1, 1.5]$ in order to prevent v' from falling far outside of the (triangle) face with which w is associated. As a practical matter, f (as defined above) is calculated with a fifth-order binomial filter for smoothing.

Selection of β*,* r*,* ℓ*, and* ε*.* As seen above, the selection of the initial triangulation (in step 1 of our framework) requires the specification of the parameters β, r, ℓ, and ε. Since the best choice for some of these parameters is dependent on both the image being processed and the sampling density, the manual selection of these parameters is quite tedious. For this reason, we propose an automated scheme for parameter selection, which was developed based on extensive experimentation with many images (including all 41 images in our test set) over a wide range of sampling densities. This automated scheme is described below.

In our framework, we always choose $r = 0.4$. The remaining parameters β, ℓ, and ε are chosen as described below. In what follows, let N and D denote the number of sample points (i.e., $N = |P|$) and the sampling density of the mesh model, respectively.

As a matter of terminology, we refer to an image as simple if it contains an abnormally low amount of edges. How we select β, ℓ, and ε depends on whether the image is simple. First, we make a determination of whether the image is simple. To do this, we perform step 1 of our framework with the fixed choices of $\beta = 0.055$, $\ell = 5$, and $\varepsilon = 1$. If this results in an initial triangulation where the number of vertices that are endpoints of constrained edges is less than $0.001\left|\Gamma \cap \frac{1}{2}\mathbb{Z}^2\right|$, the image is deemed to be simple; otherwise, it is deemed not to be simple.

Next, we make an initial choice for β, ℓ, and ε. If the image is not simple, we proceed as follows. Set $\ell = 5$. If $D > 0.01$, set $\varepsilon = 1$; otherwise, set $\varepsilon = 2$. Finally, make the initial choice of β based on the sampling density as given by Tables I and II for the ERDED and ERDGPI methods, respectively. If the image is simple, we make the initial choice of β, ℓ, and ε as $\beta = 0.03$, $\ell = 1$, and $\varepsilon = 1$.

TABLE I
CHOICE OF β FOR THE ERDED METHOD

Samp. Density (%)	β
$[0.00, 0.20)$	0.0095
$[0.20, 0.35)$	0.0095
$[0.35, 0.75)$	0.0118
$[0.75, 1.20)$	0.0280
$[1.20, 2.50)$	0.0550
$[2.50, 3.50)$	0.0830
$[3.50, 5.00)$	0.0950

TABLE II
CHOICE OF β FOR THE ERDGPI METHOD

Samp. Density (%)	β
$[0.00, 0.20)$	0.0110
$[0.20, 0.35)$	0.0115
$[0.35, 0.75)$	0.0118
$[0.75, 1.20)$	0.0280
$[1.20, 2.50)$	0.0550
$[2.50, 3.50)$	0.0830
$[3.50, 5.00)$	0.0950

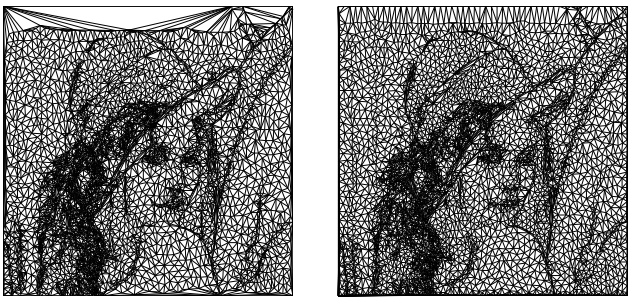
Next, we iteratively update β, ℓ, and ε. This is accomplished by the following steps: 1) Perform edge detection (i.e., step 1.1 which uses β and r). 2) Perform polyline simplification (i.e., steps 1.2 and 1.3 which uses ε and ℓ). If the number of sample points on constrained edges is less than $0.35N$, (i.e., too few sample points are obtained), set $\varepsilon = 1$ and redo polyline simplification. 3) If the actual number of constrained sample points is greater than $0.7N$, set $\beta := 0.75\beta$, set $\ell := 2\ell$, and go to step 1 (i.e., the start of the loop). 4) Output the current values of β, r, ℓ, and ε as the final selected values.

Comment on edge detector. In passing, we make a brief comment with regard to the edge detector used in step 1.1 of our ERDED and ERDGPI methods. For the purposes of these methods, we only need to be able to detect relatively long strong image edges, as such image edges are the ones that are most advantageous to represent as explicit discontinuities with our ERD model. Our choice to use the modified Canny edge detector was motivated mainly by the desire to use the simplest possible edge detector that would lead to reasonably good results. This particular edge detector was found to work well for the many images employed in our work. Because of this, it is difficult to justify the use of a much more sophisticated edge detector, as this would increase the computational cost of our methods. The main advantage of the modified Canny edge detector over the original Canny edge detector (in [29]) is that image edge intersections are better handled.

B. Mesh Refinement for the ERDED Method

As mentioned above, mesh refinement is performed differently in our ERDED and ERDGPI methods. In particular, in step 3 of our framework, the strategy used to select the new point p^* to add to the mesh is different in these two cases. In what follows, we describe how step 3 is performed in the ERDED case.

Let S denote the set of sample points to be added to the mesh. Since the initial mesh has size N_0, we require that $|S| = N - N_0$. With the ERDED method, S is selected all at once. So, if step 3 is being encountered for the first time, S is chosen (in its entirety) before any other processing is performed. Then (with S having been initialized), we arbitrarily assign a point from S to p^* and then let $S := S \setminus \{p^*\}$. As



(a) (b)

Fig. 4. Startup effect in error diffusion. Triangulation obtained (a) without mirroring and (b) with mirroring.

for how S is initially chosen, we will describe this shortly. Theoretically, it is possible for one or more of the points in S to fall on a constrained edge. To avoid unnecessarily complicating our ERDED method, we discard any such points. Since it is extremely rare for this situation to arise, the impact on the target sampling density is negligible.

The set S is chosen using the error-diffusion technique from the ED method [15] as described earlier. In order to permit the error-diffusion technique to work more effectively with our ERDED method, we made several modifications to this technique. First, the edge sensitivity parameter γ was chosen as $\gamma = 0.5$ and the smoothing operator employed (for MMSODD calculation) was selected as a fifth-order binomial filter. Second, the density function d used for error diffusion was modified. Instead of simply choosing d as the MMSODD, d was chosen such that it equals the MMSODD at points where the corresponding edge map entry (obtained from edge location in step 1.1 of our framework) is zero, and zero otherwise.

The third modification to the error diffusion scheme serves to eliminate an undesirable startup effect. In particular, when the number of sample points to be chosen is sufficiently low (i.e., at low sampling densities), error diffusion will often result in an abnormally low number of sample points being selected in the region of the image processed first (namely, the top of the image). This abnormally low number of sample points leads to very high distortion in this region, degrading overall performance. To eliminate this startup effect, we extend the image to be processed by mirroring it about its first row so as to obtain an image of twice the original height. Then, we apply error diffusion to extended image, discarding any sample points that are chosen in the mirrored region. To demonstrate the benefit of this mirroring process, we provide an example in Fig. 4. In particular, Figs. 4(a) and (b) show the triangulation obtained without and with the use of mirroring. Observe that in the no-mirroring case, an abnormally low number of sample points is selected in the first few rows (i.e., top) of the image, which ultimately leads to higher approximation error. In contrast, the mirroring case does not suffer from this problem.

C. Mesh Refinement for the ERDGPI Method

Now, we describe how step 3 of our proposed framework is performed in the ERDGPI case. In the ERDGPI case, a new point p^* to add to the mesh is selected using the process described in what follows. During mesh refinement (i.e., steps 3 to 5 of our framework), we maintain the image approximation $\hat{\phi}$ generated from the current mesh model. This image approximation $\hat{\phi}$ is generated from the mesh model parameters as specified in Section III. Each time a new point is added to the mesh, the image approximation $\hat{\phi}$ is updated to reflect the change in the mesh. For a face f in the triangulation, let points(f) denote all points in Λ belonging to f. Using the current image approximation $\hat{\phi}$, we choose p^* in two steps. First, we select the face f^* with the greatest squared error. That is,

$$f^* = \arg\max_{f \in F} \sum_{p \in \text{points}(f)} (\hat{\phi}(p) - \phi(p))^2,$$

where F is the set of all faces in the triangulation. Then, we select the point p^* in f^* with the greatest absolute error. That is,

$$p^* = \arg\max_{p \in \text{points}(f^*)} \left|\hat{\phi}(p) - \phi(p)\right|.$$

TABLE III
TEST IMAGES

Image	Size, Bits/Sample	Description
bull	1024×768, 8	computer-generated bull [33]
ct	512×512, 12	CT scan of head [30]
glasses2	1024×768, 8	raytraced glasses [33]
lena	512×512, 8	woman [32]
mri	256×256, 11	MRI scan of head [30]
peppers	512×512, 8	collection of peppers [32]

V. Evaluation of Proposed Methods

Before proceeding further, a brief digression is in order concerning the test data used herein. In our work, we employed 41 (grayscale) images that were taken mostly from standard test sets, such as the JPEG-2000 test set [30], Kodak test set [31], and USC image database [32]. Herein, we focus our attention on the representative subset of six images listed in Table III, which were deliberately chosen to include computer-generated, medical, and photographic imagery.

Error diffusion startup in ERDED method. Earlier, in the context of our ERDED method, we mentioned that error diffusion can often exhibit an undesirable startup behavior and to combat this problem we introduced a mirroring scheme. Now, we present some results to demonstrate the effectiveness of this mirroring scheme. For several test images and sampling densities, we generated a mesh using the ERDED method with and without mirroring and measured the resulting approximation error in terms of peak signal-to-noise ratio (PSNR). The results obtained are shown in Table IV, with the best result in each case being highlighted in bold. Clearly, the mirroring scheme employed in our ERDED method is highly effective, outperforming the approach without mirroring in 11/12 of the test cases by a margin of up to 2.71 dB. It was due to this excellent performance of mirroring that we chose to include it in our ERDED method (as introduced earlier).

Comparison with ED and GPI methods. Having introduced our proposed ERDED and ERDGPI methods, we now compare

TABLE IV
EFFECTIVENESS OF THE STRATEGY FOR MITIGATING THE STARTUP EFFECT IN ERROR DIFFUSION

Image	Samp. Density (%)	PSNR (dB)	
		No Mirroring	Mirroring
bull	0.125	21.97	**24.68**
	0.250	28.28	**28.86**
	0.500	34.69	**35.15**
	1.000	**39.22**	39.02
lena	0.500	20.19	**20.56**
	1.000	24.65	**25.82**
	2.000	28.09	**29.28**
	3.000	30.64	**31.31**
mri	0.250	11.56	**12.55**
	0.500	22.46	**24.79**
	1.000	28.69	**30.33**
	2.000	32.13	**33.14**

their performance to that of two competing methods, namely the ED and GPI schemes (described earlier). In terms of computational complexity, the ERDED method is most comparable to the ED scheme, and the ERDGPI method is most comparable to the GPI scheme. Therefore, we compare the ERDED method to the ED scheme and the ERDGPI method to the GPI scheme. For all 41 images in our test set and 7 sampling densities per image (namely. 0.125, 0.25, 0.5, 1, 2, 3, and 4%), we used each of the ERDED, ED, ERDGPI, and GPI methods to generate a mesh and then measured the resulting approximation error in terms of PSNR. A representative subset of the results obtained is shown in Table V for the ERDED and ED methods and in Table VI for the ERDGPI and GPI methods. In these tables, the best result in each test case is highlighted in bold. The sampling densities for which results are presented are chosen to be representative of the range that would be typically used for each image in practice (which may differ from image to image). Due to space constraints, it is not possible to present results for all $41 \cdot 7 = 287$ test cases herein.

ERDED versus ED. First, we compare the ERDED and ED methods. From the results of Table V, we can see that the ERDED method outperforms the ED scheme in 24/24 of the test cases, by a margin of 1.12 to 15.69 dB (with a median of 3.77 dB). Subjective image quality was found to correlate reasonably well with PSNR. As an example to illustrate subjective quality, the image approximations for one of the test cases from Table V is shown in Fig. 5. The corresponding image-domain triangulations are also shown, with constrained edges (in the ERDED case) denoted by thick lines. Clearly, our ERDED method produces a vastly superior image approximation (relative to the ED scheme), preserving image edges much more faithfully. In passing, we note that in our full set of 287 test cases (i.e., 41 images and 7 sampling densities per image), our ERDED method produces a higher quality mesh than the ED scheme in 284/287 (i.e., 99%) of the test cases. So, when the complete set of test cases is considered, our ERDED method clearly outperforms the ED scheme overall.

ERDGPI versus GPI. Now, we compare the ERDGPI and GPI methods. From the results of Table VI, we can see that the ERDGPI method outperforms the GPI scheme in 23/24 of the test cases, by margin of up to 4.91 dB (with a median of 1.08 dB). Again, subjective quality was found to correlate well with PSNR. As an example to illustrate subjective quality, the image approximations for one of the test cases from Table VI is shown in Fig. 6. The corresponding image-domain triangulations are also shown, with constrained edges (in the ERDGPI case) denoted by thick lines. A close inspection of the two image approximations shows that the ERDGPI method more faithfully reproduces image edges and generally has less significant distortion, relative to the GPI scheme. With the ERDGPI method, the constrained edges in the triangulation align well with image edges, allowing for better image-edge reproduction. In passing, we note that in our full set of 287 test cases (i.e., 41 images and 7 sampling densities per image), our ERDGPI method produced a higher quality mesh than the GPI scheme in 250/287 (i.e., 87%) of the test cases. So, when the complete set of test cases is considered, our ERDGPI method clearly outperforms the GPI scheme overall.

ERDED/ERDGPI versus ED/GPI. Generally speaking, the margin by which our proposed ERDED and ERDGPI methods outperform their respective counterparts (i.e., the ED and GPI schemes) tends to be higher when: 1) the sampling density is lower; and/or 2) the image being approximated is close to being piecewise smooth (i.e., has relatively little texture). The first of these two trends (i.e., item 1) is explained as follows. As the sampling density increases, all methods (regardless of how efficient they are) will ultimately converge to the same MSE (namely, zero). Therefore, differences in efficiency of the image representations being employed (i.e., the basic mesh model versus the ERD mesh model) will tend to be more pronounced at lower sampling densities. (In cases where the ED method performs particularly poorly for an image, this trend is sometimes not observed unless the sampling density is chosen to have a value that is much higher than would be used in practice.) The second of these two trends (i.e., item 2) is explained as follows. If an image is close to being piecewise smooth, relatively more sample points are needed to well capture image edges, making the approach used to represent image edges more important. Since our proposed methods use a model that more efficiently handles image edges (i.e., the ERD model), our methods can more efficiently handle images that are approximately piecewise smooth.

Comparison with GVS method. As an additional point of reference, we compare our ERDED and ERDGPI methods to the Garcia-Vintimilla-Sappa (GVS) scheme from [17], which is based on constrained Delaunay triangulations. In [17], some mesh-generation results are provided for three standard test images (from the USC image database), namely, the `house`, `lena`, and `peppers` images. Using our ERDED and ERDGPI methods, we generated meshes to match the sizes of the meshes produced in [17] and then measured the resulting approximation error in terms of PSNR. Our results along with the ones from [17] are given in Table VII for comparison. Examining the results of this table, we can see that our ERDED and ERDGPI methods each outperform the GVS scheme in every case by a margin of 1.39 to 2.34 dB and 3.46 to 4.64 dB, respectively. Clearly, our ERDED and ERDGPI methods are both superior to the GVS scheme.

TABLE V
COMPARISON OF THE MESH QUALITY OBTAINED WITH THE ERDED AND ED METHODS

Image	Samp. Density (%)	PSNR (dB) ERDED	PSNR (dB) ED
bull	0.125	**24.68**	14.66
	0.250	**28.86**	17.36
	0.500	**35.15**	27.79
	1.000	**39.02**	34.16
ct	0.125	**15.63**	12.99
	0.250	**25.97**	13.23
	0.500	**30.06**	17.47
	1.000	**36.51**	20.82
glasses2	0.500	**20.54**	16.55
	1.000	**24.80**	21.78
	2.000	**28.71**	26.02
	3.000	**30.85**	27.98
lena	0.500	**20.56**	17.60
	1.000	**25.82**	22.45
	2.000	**29.28**	26.90
	3.000	**31.31**	28.54
mri	0.250	**12.55**	10.87
	0.500	**24.79**	16.10
	1.000	**30.33**	15.55
	2.000	**33.14**	19.94
peppers	0.500	**22.14**	17.53
	1.000	**25.97**	22.42
	2.000	**29.00**	27.10
	3.000	**30.18**	29.06

TABLE VI
COMPARISON OF THE MESH QUALITY OBTAINED WITH THE ERDGPI AND GPI METHODS

Image	Samp. Density (%)	PSNR (dB) ERDGPI	PSNR (dB) GPI
bull	0.125	**35.47**	30.56
	0.250	**38.33**	35.30
	0.500	**40.16**	38.72
	1.000	**41.39**	40.95
ct	0.125	**27.19**	24.69
	0.250	**32.11**	29.93
	0.500	**36.31**	35.12
	1.000	39.49	**39.82**
glasses2	0.500	**24.94**	23.65
	1.000	**28.56**	27.01
	2.000	**32.11**	31.00
	3.000	**33.86**	33.53
lena	0.500	**25.58**	24.22
	1.000	**28.35**	26.96
	2.000	**30.79**	29.74
	3.000	**32.30**	31.36
mri	0.250	**27.14**	26.34
	0.500	**29.55**	29.07
	1.000	**33.01**	32.14
	2.000	**35.21**	35.08
peppers	0.500	**25.99**	24.66
	1.000	**28.40**	27.49
	2.000	**30.42**	29.99
	3.000	**31.50**	31.19

TABLE VII
COMPARISON OF THE MESH QUALITY OBTAINED WITH THE ERDED, ERDGPI, AND GVS METHODS

Image	Mesh Size	PSNR (dB) ERDED	PSNR (dB) ERDGPI	PSNR (dB) GVS
house	1649	29.52	31.52	27.62
lena	7492	29.69	31.76	28.30
peppers	7224	28.82	31.12	26.48

TABLE VIII
TIME COMPLEXITY COMPARISON FOR ERDED AND ED METHODS FOR THE lena IMAGE

Samp. Density (%)	Time (s) ERDED	Time (s) ED
0.5	0.572	0.019
1.0	0.599	0.020
2.0	0.617	0.025
3.0	0.663	0.032

TABLE IX
TIME COMPLEXITY COMPARISON FOR ERDGPI AND GPI METHODS FOR THE lena IMAGE

Samp. Density (%)	Time (s) ERDGPI	Time (s) GPI
0.5	0.709	0.123
1.0	0.741	0.179
2.0	0.825	0.365
3.0	0.901	0.653

Computational complexity. In passing, we note that our ERDED and ERDGPI schemes are both quite modest in terms of their computational requirements. For example, on a notebook computer with a 2.7 GHz Intel Core i7 CPU and 8 GB of RAM, for the lena image and sampling densities in the range 0.5% to 3%, the ERDED and ERDGPI methods have the execution times given in Tables VIII and IX, respectively. From these results, we can see that our ERDED and ERDGPI methods require less than one second for an image like lena. This is in stark contrast to the numerous mesh-generation methods that take on the order of minutes to process an image of this size, such as the scheme in [2]. In the above tables, we have also included the execution times for the ED and GPI methods for comparison to our ERDED and ERDGPI methods, respectively. In this regard, it is important to note that, while our ED and GPI implementations were optimized for execution speed, our ERDED and ERDGPI implementations were not. Consequently, the ED and GPI methods have a very unfair advantage over our proposed methods in this execution-time comparison. In spite of this, our ERDED and ERDGPI methods still only require 0.553 to 0.631 seconds and 0.248 to 0.586 seconds longer than the ED and GPI schemes, respectively. Considering the significant improvement in mesh quality obtained with our methods, the fraction-of-a-second increase in execution time is quite reasonable.

Additional comments. As seen earlier, the ERDED and ERDGPI methods are able to produce considerably better meshes than the ED and GPI schemes, respectively. This better performance obtained with our methods is facilitated by the more effective mesh model that they employ. Recall that the ED and GPI schemes employ the basic model, which uses a Delaunay triangulation and a continuous approximating function, while our ERDED and ERDGPI methods employ the ERD model, which uses a constrained Delaunay triangulation and an approximating function that need not be continuous along triangulation edges. As a consequence, our ERDED and ERDGPI methods can represent image edges much more efficiently, requiring many fewer sample points in the vicinity of image edges. This ultimately allows our proposed methods to produce meshes of higher quality than the ED and GPI schemes. Also, as seen earlier, the ERDED and ERDGPI methods also significantly outperform the GVS scheme. Although the GVS scheme employs a mesh model that is based on a constrained Delaunay triangulation, the associated approximating function is continuous and each image edge is

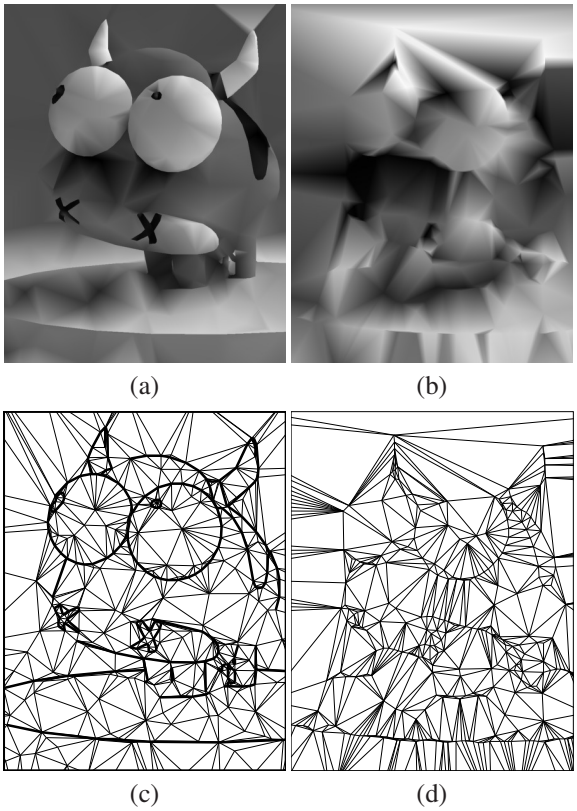


Fig. 5. Comparison of the ERDED and ED methods. Part of the image approximation obtained for the `bull` image at a sampling density of 0.125% with the (a) ERDED (24.68 dB) and (b) ED (14.66 dB) methods, and (c) and (d) their corresponding triangulations.

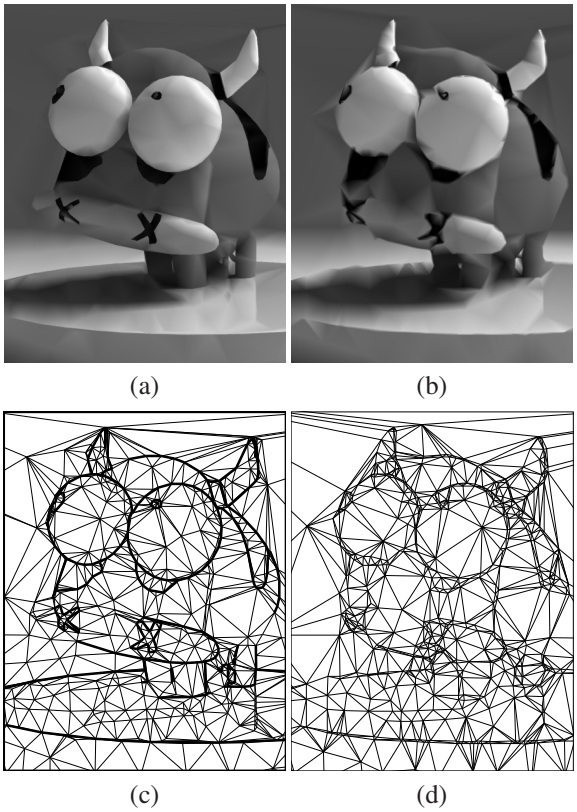


Fig. 6. Comparison of the ERDGPI and GPI methods. Part of the image approximation obtained for the `bull` image at a sampling density of 0.125% with the (a) ERDGPI (35.47 dB) (b) GPI (30.56 dB) methods, and (c) and (d) their corresponding image-domain triangulations.

represented using a two (approximately) parallel sets of edge constraints, which greatly reduces the efficiency with which image edges can be represented. Consequently, our ERDED and ERDGPI methods are also able to significantly outperform the GVS scheme.

VI. Conclusions

In this manuscript, we have introduced a new mesh model for images, called ERD, that explicitly represents discontinuities (i.e., image edges). Then, we proposed two different mesh-generation methods, called ERDED and ERDGPI, that select the ERD-model parameters for a given input image. To demonstrate the effectiveness of our proposed approach, we compared the quality of the image approximations obtained with our ERDED and ERDGPI methods to that produced by three other competing schemes, namely, the ED, GPI, and GVS methods. Experimental results showed our ERDED and ERDGPI methods to significantly outperform the ED and GPI schemes, respectively, in terms of both squared error and subjective quality. Moreover, our ERDED and ERDGPI methods were also found to outperform the GVS scheme, both in terms of squared error and subjective quality. Our proposed methods are relatively fast, typically requiring less than one second on modest hardware. As our methods yield image approximations of superior quality and have modest computational requirements, these methods can benefit the many applications in which mesh models of images are employed.

References

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

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

[3] M. Sarkis and K. Diepold, "Content adaptive mesh representation of images using binary space partitions," *IEEE Trans. on Image Processing*, vol. 18, no. 5, pp. 1069–1079, May 2009.

[4] M. Grundland, C. Gibbs, and N. A. Dogson, "Stylized multiresolution image representation," *Journal of Electronic Imaging*, vol. 17, no. 1, pp. 013 009.1–17, 2008.

[5] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Trans. on Image Processing*, vol. 6, no. 9, pp. 1305–1315, Sep. 1997.

[6] D. Terzopoulos and M. Vasilescu, "Sampling and reconstruction with adaptive meshes," in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Maui, HI, USA, 1991, pp. 70–75.

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

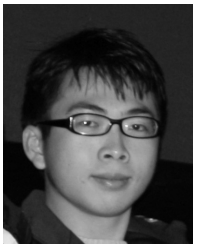
[8] M. Petrou, R. Piroddi, and A. Talebpour, "Texture recognition from sparsely and irregularly sampled data," *Computer Vision and Image Understanding*, vol. 102, pp. 95–104, 2006.

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

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

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

[12] M. A. Garcia and B. X. Vintimilla, "Acceleration of filtering and enhancement operations through geometric processing of gray-level images," in *Proc. of IEEE International Conference on Image Processing*, vol. 1, Vancouver, BC, Canada, 2000, pp. 97–100.
[13] D. Su and P. Willis, "Image interpolation by pixel-level data-dependent triangulation," *Computer Graphics Forum*, vol. 23, no. 2, pp. 189–201, 2004.
[14] M. D. Adams, "An efficient progressive coding method for arbitrarily-sampled image data," *IEEE Signal Processing Letters*, vol. 15, pp. 629–632, 2008.
[15] Y. Yang, M. N. Wernick, and J. G. Brankov, "A fast approach for accurate content-adaptive mesh generation," *IEEE Trans. on Image Processing*, vol. 12, no. 8, pp. 866–881, Aug. 2003.
[16] M. Garland and P. S. Heckbert, "Fast polygonal approximation of terrains and height fields," School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU-CS-95-181, Sep. 1995.
[17] M. A. Garcia, B. X. Vintimilla, and A. D. Sappa, "Approximation and processing of intensity images with discontinuity-preserving adaptive triangular meshes," in *Lecture Notes in Computer Science*, vol. 1842. Springer Verlag, Jul. 2000, pp. 844–855.
[18] 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*, San Diego, CA, USA, Oct. 2008, pp. 1041–1044.
[19] L. P. Chew, "Constrained Delaunay triangulations," *Algorithmica*, vol. 4, pp. 97–108, 1989.
[20] X. Tu and M. D. Adams, "Image representation using triangle meshes with explicit discontinuities," in *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, BC, Canada, Aug. 2011, pp. 97–101.
[21] B. Delaunay, "Sur la sphere vide," *Bulletin of the Academy of Sciences of the USSR, Classe des Sciences Mathematiques et Naturelle*, vol. 7, no. 6, pp. 793–800, 1934.
[22] C. Dyken and M. S. Floater, "Preferred directions for resolving the non-uniqueness of Delaunay triangulations," *Computational Geometry—Theory and Applications*, vol. 34, pp. 96–101, 2006.
[23] M. K. Agoston, *Computer Graphics and Geometric Modeling: Implementation and Algorithms*. London, UK: Springer, 2005.
[24] W. K. Pratt, *Digital Image Processing*, 4th ed. Hoboken, NJ, USA: Wiley, 2007.
[25] L. Ding and A. Goshtasby, "On the Canny edge detector," *Pattern Recognition*, vol. 34, no. 3, pp. 721–725, Mar. 2001.
[26] M. Aubury and W. Luk, "Binomial filters," *Journal of VLSI Signal Processing*, vol. 12, pp. 35–50, 1996.
[27] Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms," *Communications of the ACM*, vol. 32, no. 3, pp. 359–373, Mar. 1989.
[28] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.
[29] J. Canny, "A computational approach to edge detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, Nov. 1986.
[30] "JPEG-2000 test images," ISO/IEC JTC 1/SC 29/WG 1 N 545, Jul. 1997.
[31] "Kodak lossless true color image suite," 2011. [Online]. Available: http://r0k.us/graphics/kodak
[32] "USC-SIPI image database," 2011. [Online]. Available: http://sipi.usc.edu/database
[33] "Michael Adams' research datasets," 2011. [Online]. Available: http://www.ece.uvic.ca/ mdadams/datasets



Xi Tu was born in JiangXi, China. He received the Bachelor degree in electrical engineering from Shanghai Jiaotong University, Shanghai, China, and the M.A.Sc. degree in electrical engineering from the University of Victoria, Victoria, BC, Canada. He is currently with Amazon in Seattle, WA, USA. His research interests include signal/image processing, computational geometry, and algorithms.

Michael Adams received the B.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada, the M.A.Sc. degree in electrical engineering from the University of Victoria, Victoria, BC, Canada, and the Ph.D. degree in electrical engineering from the University of British Columbia, Vancouver, BC, Canada. Since 2003, Michael has been with the Department of Electrical and Computer of Engineering at the University of Victoria, Victoria, BC, Canada, where he is currently an Associate Professor. His research interests include signal processing, multimedia systems, computational geometry, and algorithms.