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

Badr Eddine El Marzouki and Michael D. Adams
Dept. of Electrical and Computer Engineering, University of Victoria
Victoria, BC, V8P 5C2, Canada
marzouki@uvic.ca and mdadams@ece.uvic.ca

***Abstract*—Two highly effective content-adaptive methods for generating Delaunay mesh models of digital images, known as IID1 and IID2, are proposed. The methods are derived from the incremental/decremental mesh-generation framework of Adams by using a non-trivial initial mesh and an improved growth schedule. The higher complexity IID2 method is shown to yield mesh models of superior reconstruction quality compared to competing methods, both in terms of squared error and subjective quality, while the lower complexity IID1 method trades mesh quality in return for a decrease in computational cost.**

I. Introduction

Despite being the most common form of image sampling, lattice-based sampling is far from optimal. Typically, images are nonstationary, which inevitably leads to oversampling in some regions of the sampled domain and undersampling in others when uniform sampling is used. Nonuniform sampling addresses this issue by adapting the sampling density to the content of the image. A more intelligent placement of the sample points leads to greater efficiency, thereby significantly reducing the number of samples needed while keeping the visual quality at acceptable levels.

Several approaches to image representation based on nonuniform sampling have been proposed over the years. Triangle mesh models in particular have garnered significant interest from researchers in the past few years, due to their simplicity and their ability to model a wide range of image content as well as some features inherent in images such as sharp edges. With mesh models, the image domain is partitioned into triangle faces whose vertices are the sample points, and then an interpolant is constructed over each face. This approach takes advantage of the geometric structure in images and the correlation between adjacent image pixels. The so called mesh-generation problem is concerned with creating, for a given image, a mesh approximation that minimizes some error criteria. Such problems are often NP hard [1], which highlights the importance of finding good computationally efficient methods for the solution of these problems.

Over the years, a great number of methods have been developed for generating Delaunay-triangulation-based mesh models of images. In mesh-simplification schemes, such as GPR [2] and GPRFS [3], sample points are gradually removed from the mesh until the desired number of samples or approximation error has been reached. This can be done with one or more sample points removed at a time, starting from a mesh containing many or all of the sample points in the image domain. Conversely, mesh refinement methods start with a low-density mesh consisting of the extreme convex-hull points of the image domain, and possibly a few additional sample points as well. The mesh is then refined by adding one or more points at a time until some termination criteria, such as mesh size or mesh approximation error, is satisfied. Mesh-generation methods that use refinement, such as the error-diffusion scheme of Yang et al. [4], generally have lower computational and memory costs than simplification methods, but at the expense of mesh quality. A better balance between approximation quality and computational cost compared to the above approaches is achieved by the IDDT [5], ID1, and ID2 [3] schemes of Adams by using a combination of mesh refinement and simplification. The methods alternate between phases where points are added to the mesh and phases where points are removed from the mesh.

In this paper, we propose two new mesh-generation methods, known as IID1 and IID2, which are based on a common algorithmic framework. These two methods make different trade-offs between mesh quality and computational cost, with the IID2 method favouring mesh quality while the IID1 method favours computational cost. Through experimental results, we show our methods to outperform other state-of-the-art approaches, when mesh quality and computational cost are both considered.

The remainder of this paper is organized as follows. In Sections II and III, we introduce background material on triangle mesh models for image representation, Floyd-Steinberg error diffusion, and the mesh-generation framework of Adams on which our methods are based. Then, our proposed mesh-generation methods, IID1 and IID2, are presented in Section IV. The performance of the IID1 and IID2 methods in terms of approximation quality and computational cost is evaluated in Section V and compared to state-of-the-art methods. Finally, Section VI concludes with a summary of the key results of our work.

II. Background

In what follows, we introduce some background material necessary for understanding our proposed mesh-generation methods. As a matter of notation, the cardinality of a set S is denoted as $|S|$. In the context of our work,

an image ϕ of width W and height H is a function defined on the truncated 2-dimensional integer lattice $\Lambda = \{0, 1, ..., W-1\} \times \{0, 1, ..., H-1\}$. A triangle mesh model of ϕ is characterized by: 1) a set $P = \{p_i\} \subset \Lambda$ of sample points, 2) their corresponding sample values $\{z_i = \phi(p_i)\}$, and 3) a triangulation T of P. The triangulation T is uniquely determined from P using preferred-directions Delaunay triangulation. In order to ensure that all of Λ is covered by the triangulation, P is chosen to always include the extreme convex-hull points of the image domain. A continuous piecewise-linear function $\tilde{\phi}_P$ that approximates ϕ is associated with the mesh model. The function $\tilde{\phi}_P$ is constructed from $\{p_i\}$ and $\{z_i\}$ by combining the linear interpolants over each of the faces in T. The integer-valued image approximation function $\hat{\phi}_P$ is obtained from $\tilde{\phi}_P$ using rounding as $\hat{\phi}_P = \text{round}\left(\tilde{\phi}_P\right)$. As a matter of terminology, the size and the sampling density of the mesh model are defined as $|P|$ and $|P|/|\Lambda|$, respectively.

With the mesh model defined above, the mesh-generation problem addressed in our work can be stated as follows: given a target number N of sample points (where $N \in [4, |\Lambda|]$), choose $P \subset \Lambda$ with $|P| = N$ such that the mesh approximation error is minimized. The error metric used to measure the quality of the approximation is the mean squared error (MSE) ε defined as $\varepsilon = |\Lambda|^{-1} \sum_{p \in \Lambda} (\hat{\phi}(p) - \phi(p))^2$, which is typically expressed in terms of the PSNR for convenience. The PSNR is given by $\mathsf{PSNR} = 20 \log_{10}\left(\frac{2^\rho - 1}{\sqrt{\varepsilon}}\right)$, where ρ is the sample precision in bits.

The maximum-magnitude second-order directional derivative (MMSODD) [4] d for a function f defined on $\mathbb{R}^2$ is defined as

$$d(x,y) = \max\Big\{|\alpha(x,y)+\beta(x,y)|, |\alpha(x,y)-\beta(x,y)|\Big\}, \quad (1)$$

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

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

The partial-derivative operators in the preceding equation are formed from the tensor product of one-dimensional derivative operators, where the discrete-time approximations of the one-dimensional first- and second-order derivative operators are computed using the filters with transfer functions $\frac{1}{2}z - \frac{1}{2}z^{-1}$ and $z - 2 + z^{-1}$, respectively.

Error diffusion is a technique commonly used for halftoning. Yang et al. [4] proposed using Floyd-Steinberg error diffusion with a feature map based on the MMSODD of an image to adaptively distribute points in the image domain. The set of points generated have a spatial density that is approximately proportional to the amount of detail in a given region of the image. Given an image ϕ of width W and height H sampled on the set Λ of points (where $|\Lambda| = WH$), and a desired number N of points, error diffusion selects a subset P_N from Λ, where $|P_N| \approx N$, as follows:

1) Compute the MMSODD d of the image using (1).
2) Compute the feature map σ from the MMSODD as $\sigma(x,y) = \left(\frac{d(x,y)}{A}\right)^\gamma$, where $A = \max[d(\phi)]$ and γ is a (positive constant) sensitivity parameter.
3) In order to select approximately N sample points, compute the error diffusion threshold ρ as $\rho = \frac{1}{2N}\left[\sum_{i=0}^{W-1}\sum_{j=0}^{H-1}\sigma(i,j)\right]$.
4) Generate a binary image b defined on the same image domain as ϕ from the feature map σ using error diffusion with the threshold value ρ.
5) Select each point (x,y) of the image b that satisfies the condition $b(x,y) \neq 0$ as a sample point in P_N.

III. Mesh-Generation Framework

Given that the computational framework of [3] constitutes the foundation of our work, an introduction to the framework is worthwhile. The framework is iterative in nature, alternating between a mesh refinement phase and a mesh simplification phase according to a predetermined sequence of setpoints $\{\eta_i\}_{i=0}^{L-1}$ of length L, referred to as a growth schedule. As the mesh-generation process proceeds, the mesh size tracks the values in the growth schedule.

Before we can proceed further, some additional notation and terminology need to be introduced. The face f to which a point $p \in \mathbb{Z}^2$ is uniquely mapped is denoted face(p). The set of all points in $\mathbb{Z}^2$ that are mapped to a face f (i.e., the points satisfying face(p)=f) is denoted points(f). For the mesh with sample points S, let $\hat{\phi}_S$ be the corresponding interpolant and let $r_S(p)$ denote the approximation error at the point p (i.e., $r_S(p) = \hat{\phi}_S(p) - \phi(p)$). Let P denote the sample points in the current mesh approximation of ϕ and let $r(p)$ denote the corresponding approximation error at the point p (i.e., $r(p) = \hat{\phi}_P(p) - \phi(p)$). Consequently, the error over a face f, denoted faceErr(f), is the sum of the errors at the points contained in f (i.e., faceErr(f)= $\sum_{p \in \Lambda \cap \text{points}(f)} r(p)$), and the total mesh model approximation error is $\sum_{p \in \Lambda} r(p)$. A point is said to be immutable if it is not allowed to be removed from or added to the mesh during the mesh-generation process; otherwise it is mutable. The subset of mutable points in a given set S is denoted mutable(S). A point $p \in \Lambda$ that is mutable but is not currently in the mesh is said to be a candidate point. The set of candidate points for a given face f are denoted cands(f) (i.e., cands(f) = mutable($(\Lambda \setminus P) \cap$ points(f)).

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

1) Select initial mesh points. Obtain the initial subset P of the sample points as $P := \Gamma$.
2) Initialize mesh. Create a mesh consisting of the extreme convex-hull points of Λ and mark the points as immutable so that they cannot be removed from the mesh. Then insert the points in P (from step 1) into the mesh and mark them as mutable.

3) If $i = L$ (i.e., no more points in the growth schedule remain), go to step 7; otherwise, proceed. If $|P| < \eta_{i+1}$, go to step 4 to increase the mesh size. If $|P| > \eta_{i+1}$, go to step 5 to decrease the mesh size. If $|P| = \eta_{i+1}$, go to step 6 (bottom of the main loop).
4) Increase mesh size. While $|P| < \eta_{i+1}$, add a point to the mesh by performing an optimal add (optAdd) operation which consists of the following steps:
   a) Select a face f^* in which to insert a new point as given by $f^* = \arg\max_{f \in \mho} \text{faceErr}(f)$, where $\mho$ is the set of all faces that have at least one candidate point.
   b) Select a point p^* in the face f^* to add to the mesh, as given by $p^* = \text{selCand}(f^*)$, where selCand is a function that embodies the candidate-selection policy, and is a free parameter of the framework.
   c) Add p^* to the mesh (i.e., let $P := P \cup \{p^*\}$).
   d) Go to step 6.
5) Decrease mesh size. While $|P| > \eta_{i+1}$, delete a point from the mesh by performing an optimal delete (optDel) operation, which consists of the following steps:
   a) Let the significance (with respect to deletion) of a (mutable) point $p \in P$, denoted sigDel(p), be defined as $\text{sigDel}(p) = \sum_{q \in (R \cap \Lambda)} (r^2_{P \setminus \{p\}}(q) - r^2_P(q))$, where R is the region in the triangulation affected by the deletion of p. That is, sigDel(p) is the amount by which the squared error increases if p were deleted from the mesh. The point p^* to delete from the mesh is then selected as $p^* = \arg\min_{p \in \text{mutable}(P)} \text{sigDel}(p)$.
   b) Delete p^* from the mesh (i.e., let $P := P \setminus \{p^*\}$)
   c) Go to step 6.
6) Proceed to next iteration. Let $i := i + 1$. Go to step 3 (i.e., the top of the main loop).
7) Postprocess mesh. Optionally, perform some postprocessing steps; then stop.

Before proceeding further, some comments regarding the above computational framework are in order. First, this framework is fundamentally greedy in that it only considers the current state and the immediate consequences of actions when selecting points to add to, or remove from, the mesh. As a result, the meshes generated by methods that are based on this framework are not guaranteed to be globally optimal. This is an acceptable compromise given that the mesh-generation problem addressed in our work is known to be NP hard. Moreover, performance is greatly influenced by the choice of the initial mesh. Both computational cost and the quality of the result are affected by the size and the choice of the initial mesh. Seen from an optimization perspective, when the starting point for the search process is close to a good optimal solution, the optimal solution will be reached faster and the probability of the algorithm becoming trapped in a bad local minimum is greatly reduced. The above observations led to the motivation for our work to place a strong emphasis on the selection of the initial mesh and growth schedule.

IV. Proposed Mesh-Generation Methods

Having introduced the background material and the mesh-generation framework, we can proceed to define our proposed methods. The methods, named IID1 and IID2, offer different trade-offs between mesh quality and computational cost. In particular, the IID1 method aims for shorter execution times, while the IID2 method prioritizes mesh quality. These methods can be viewed as improved versions of the ID1 and ID2 methods from [3], which are based on the same computational framework used herein. Unlike the cases of the ID1 and ID2 methods, our methods use a non-trivial initial mesh as well as a novel growth schedule. Additionally, our IID1 method uses a different candidate-selection policy.

The IID1 and IID2 methods use a new growth schedule A′, which is a contribution of our work. Similar to growth schedule A employed in the ID1 and ID2 methods, the mesh size in growth schedule A′ oscillates between the target mesh size N and exponentially decaying values above N. The new growth schedule was developed to be more flexible than growth schedule A and to overcome some of its limitations. One of the main differences is that the definition of growth schedule A′ does not consider the size of the initial mesh. Additionally, its length is manually set and determines the number of alternations between sequences of point insertions and deletions before convergence to the target mesh size. The L setpoints $\{\eta_i\}_{i=0}^{L-1}$ of growth schedule A′ are given by

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

where A is a positive real constant. The lower complexity IID1 method uses $A = 3$ and $L = 4$, while the IID2 method, which favours approximation quality, uses the values $A = 3$ and $L = 6$.

The initial point set in step 1 of the framework is selected using non-leaky Floyd-Steinberg error diffusion with a sampling density of 1%. The serpentine scan order is used, and the feature map sensitivity parameter is set to $\gamma = 1$. Prior to the MMSODD calculation in step 1 of the error-diffusion algorithm, the image is smoothed using a 7th-order binomial filter. The transfer function $H_n(z)$ of a one-dimensional nth order binomial filter with unity DC gain and zero phase is given by $H_n(z) = z^{\frac{n-1}{2}} \left(\frac{1}{2} + \frac{1}{2}z^{-1}\right)^{n-1}$, where n is an odd integer. The two-dimensional binomial filter is constructed from the tensor product of two one-dimensional binomial filters.

In step 4b of the framework, the candidate-selection policy used for the IID1 method is based on the MMSODD. It selects, for insertion, the candidate point that has the highest MMSODD value in the face f^*. With $d(p)$ denoting the (nonnegative-valued) MMSODD at point p, the selCand function (which specifies the candidate-selection policy) is defined as

$$\text{selCand}(f) = \arg\max_{p \in \text{cands}(f)} \Big[d(p)\Big].$$

In the case of the IID2 method, the approximate local squared-error minimizer (ALSEM) [3] is employed for the candidate-

selection policy. The function selCand that embodies this policy is given by

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

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

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

The summation in (2) corresponds to the reduction in the squared error if p were inserted into the mesh, computed only locally over the points that are in points(f) (i.e., with the assumption that no changes to the mesh occur outside the face f).

As for the face-selection policy in step 4a of the mesh-generation algorithm, it is chosen to be the sum of squared-errors (SSE). The policy selects the face f^* in which to insert a new point as given by

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

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

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

In the post-processing step of the framework, a technique named bad point replacement (BPR), which was introduced in [3], iteratively replaces bad points in the mesh with new ones so that the total number of points in the mesh remains unchanged. A bad point is a mutable point in the mesh whose deletion from the mesh does not cause the approximation error to increase (i.e., sigDel(p) ≤ 0). Bad point replacement consists of the following steps:

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

V. Results

With our proposed IID1 and IID2 mesh-generation methods introduced, we now evaluate their performance in terms of mesh quality and computational cost by comparing them to other well-known schemes. The implementations used for this purpose were developed by the authors and were written in C++. Our performance comparison focuses, in particular, on the GPR [2], IDDT [5], ID1, and ID2 [3] methods because they produce Delaunay meshes and use a linear interpolant, similar to our methods. In passing, we note that the ID1 and ID2 methods have a parameter α for adjusting the length of the growth schedule, thereby increasing or decreasing the computational cost with a corresponding increase or decrease in mesh quality. In our comparison, we use the value $\alpha = 0.4$ advocated in [3]. We also note that, by comparing the performance of our methods against that of the state-of-the-art IDDT, ID1, and ID2 schemes, we are also indirectly showing that our methods outperform schemes such as the ED [4], MGH, OMGH, OED [5], and GPRFS-ED [3] methods, to which the IDDT, ID1, and ID2 methods have been shown to be superior [3], [5].

Mesh Quality. In order to evaluate mesh quality, we employ a set of 50 test images, including images from several popular test sets (such as lena from the USC image database). For all 50 images in our test set and 7 sampling densities per image (between 0.125% and 4%), we used each of the methods under consideration to generate a mesh, and then measured the resulting approximation error in terms of PSNR. The individual results for a representative subset of the images are shown in Table I(a). For each of the $50 \times 7 = 350$ test cases (i.e., 50 images with 7 sampling densities per image), the PSNR performance of the six methods (i.e., IID1, IID2, ID1, ID2, IDDT, and GPR) was ranked from 1 (best) to 6 (worst). The average and standard deviation were then calculated for each sampling density as well as overall with the results presented in Table I(b). The best and second best result in each case are typeset in bold and italic, respectively.

Proposed vs. IDDT. We begin by comparing the IID1 and IID2 methods to the IDDT scheme. The overall statistical results in Table I(b) show that our IID1 and IID2 methods outperform the IDDT scheme at all sampling densities. In fact, the IDDT method has the worst average rank of 5.69 amongst all six methods. Looking at the full results in more detail, we find that our IID1 and IID2 methods beat the IDDT scheme in 332/350 (95%) and 342/350 (98%) of the test cases, respectively.

Proposed vs. GPR. Next, we compare the performance of the proposed IID1 and IID2 methods to the GPR method. Examining the statistical results in Table I(b), we observe that our IID2 method outperforms the GPR method consistently across all of the sampling densities, whereas our IID1 method ranks similarly or better on average than the GPR scheme for target densities of 0.5% and higher. More detailed analysis of the results shows that our IID2 and IID1 methods beat the GPR method in 346/350 (99%) and 221/350 (63%) of the test cases, respectively. Now, we consider the individual results in Table I(a). We see that these individual results are consistent with the overall statistical results. Our IID2 method outperforms the GPR scheme in all cases, while our IID1 method has mixed results in comparison but typically performs better at higher densities and with medical and photographic images. Such performance from our IID1 and IID2 methods is particularly impressive considering that, as we will see later, they are substantially less computationally expensive than the GPR method.

Proposed vs. ID1 and ID2. Now, let us compare our IID1 and IID2 methods to the ID1 and ID2 methods. To do this, we examine the statistical results in Table I(b). The IID2

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

(a)

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

(b)

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

* The standard deviation is given in parentheses.

method is clearly best overall with a rank of 1.38 and a relatively small standard deviation of 0.77. The competing ID2 method is second best with a rank of 2.01 and a small standard deviation as well (0.67). The results for the IID2 and ID2 methods are consistent across all sampling densities. The ID1 method is third best overall followed closely by the IID1 method in fourth position, with average ranks of 3.48 and 3.81, respectively. The performance of the ID1 and IID1 methods varies with sampling density. The average ranking of both is typically worse at low sampling densities and then gradually improves as the density increases, eventually superseding the GPR method. A more detailed analysis of the results shows that the IID2 method beats the ID1 and ID2 methods in 310/350 (89%) and 281/350 (80%) of the test cases, respectively. The IID1 method performs better than the ID1 method in 165/350 (47%) of the test cases, which improves to 65% for densities of 2% and higher. This performance from the IID1 method is deemed acceptable given that, as will be seen later, it typically has a lower computational cost. The individual PSNR results in Table I(a) are consistent with the conclusions from the statistical analysis. They also show that in the test cases where the ID2 method is able to beat our IID2 method, the difference in mesh quality is typically marginal. For instance, in the individual PSNR results, out of the ten test cases where the ID2 method yields better results, the margin does not exceed 0.08 dB in 8/10 (80%) of such test cases.

Summary of results. Based on the preceding analysis, we conclude that, in terms of mesh quality measured by PSNR, our IID2 method consistently performs best compared to the other methods in our evaluation. On the other hand, our IID1 method, which trades off mesh quality for computational cost, has performance that is on a par with that of the ID1 method. As we will see later, our proposed IID1 and IID2 methods compare even more favorably to the other methods under evaluation when computational cost is considered. That is, as it turns out, our proposed methods are often able to produce higher quality meshes than other schemes for a given computational cost.

In the above evaluation, PSNR was found to correlate reasonably well with subjective quality. For the benefit of the reader, however, we include some examples illustrating the subjective quality achieved by the four best-ranking methods overall. Fig. 1 shows a set of examples which are for the photographic lena image with a sampling density of 2%. The PSNR values for these test cases, which are fairly close, are consistent with the subjective quality since the results appear similar visually. The execution times that correspond to these test cases, however, are quite different as we will see shortly.

Computational Cost. Next, we briefly evaluate the computational cost (i.e., execution time) of the methods in our comparison. For this purpose, we provide a representative subset of timing measurements collected using a 5 year old laptop with a 2.4GHz Intel Core i7 processor and 4GB of RAM. For three representative images and six sampling densities per image, the time required for mesh generation for each of the methods was measured. Then, the median value of five runs for each test case was determined. The results are presented in Table II.

Examining the results, we see that the computational cost of our two methods is typically lower than that of competing methods. For example, in the case of the lena image with a sampling density of 2%, the execution time of the IID1 method is 47%, 13%, 52% and 89% lower than that of the IID2, ID1, ID2 and GPR methods, respectively, but the PSNR of the corresponding approximation is only marginally lower (0.05–0.12 dB) compared to the first three methods, and 0.32 dB higher compared to the GPR method. As for the IID2 method, its computational cost is lower than that of the competing ID2

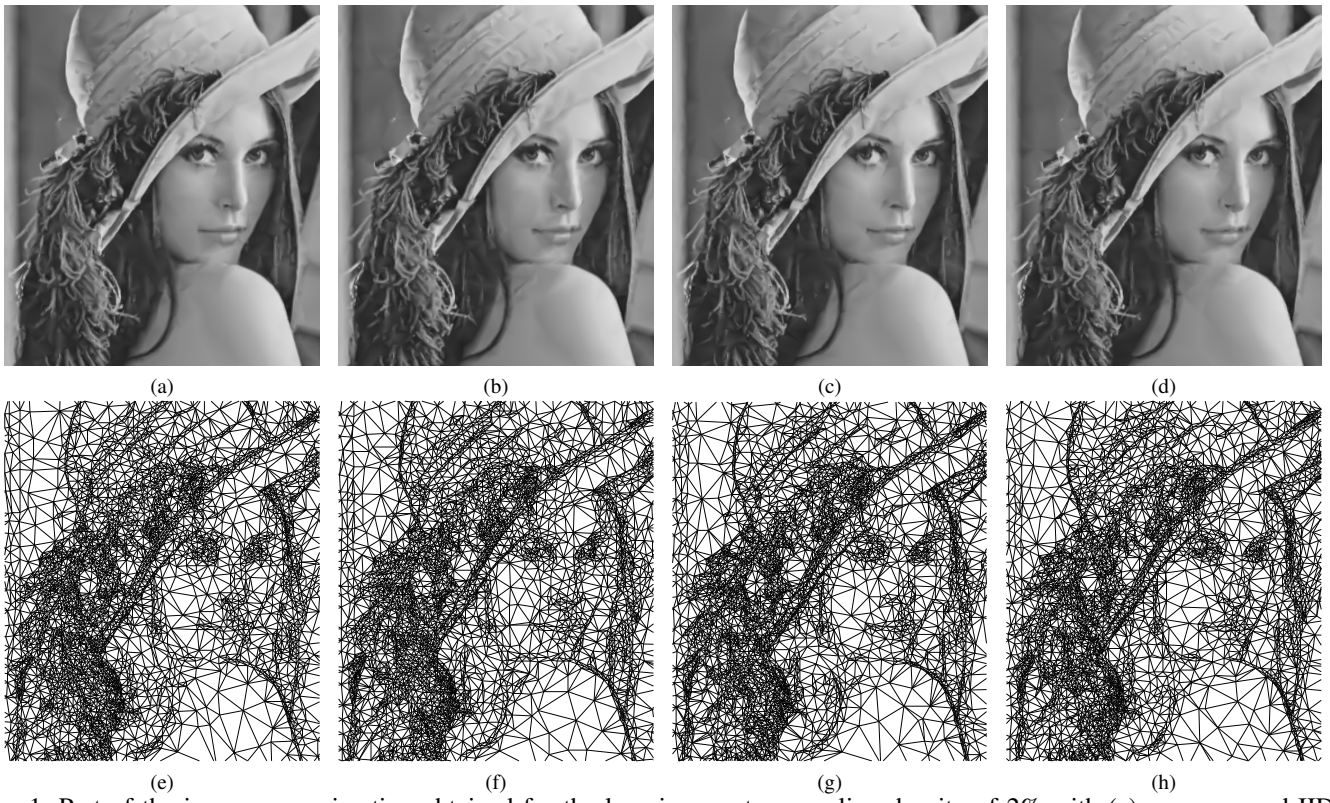

Fig. 1: Part of the image approximation obtained for the lena image at a sampling density of 2% with (a) our proposed IID1 method (32.1 dB), (b) the ID1 method (32.15 dB), (c) our proposed IID2 method (32.22 dB), and (d) the ID2 method (32.17 dB), and (e), (f), (g), and (h) their corresponding triangulations.

TABLE II: Comparison of the computational cost for the various methods.

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

and GPR methods in the vast majority of the test cases.

VI. Conclusions

In this paper, we proposed two new methods for generating mesh models of images, namely IID1 and IID2. Through experimental results, the IID1 and IID2 methods were shown to outperform state-of-the-art mesh-generation methods, namely the ID1, ID2, IDDT, and GPR methods, when mesh quality and computational cost are both considered. In particular, we demonstrated that our proposed methods produce higher quality image approximations for a given computational cost. The higher complexity IID2 method was shown to achieve better quality approximations than the ID2 and GPR methods, with lower or similar computational cost. The lower complexity IID1 method was shown to trade a typically small penalty in image approximation for lower computational cost. As a result, the IID2 method may be used in applications where mesh quality is prioritized, whereas the IID1 method is more suitable when execution times need to be optimized.

References

[1] P. K. Agarwal and S. Suri, "Surface approximation and geometric partitions," *SIAM Journal on Computing*, vol. 27, no. 4, pp. 1016–1035, 1998.

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

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

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

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