An Effective Mesh-Generation Strategy for Image Representation using Data-Dependent Triangulation

Ping Li and Michael D. Adams
Dept. of Electrical and Computer Engineering, University of Victoria
Victoria, BC, V8W 3P6, Canada
pingli@ece.uvic.ca and mdadams@ece.uvic.ca

***Abstract*—A new mesh-generation method for image representation based on data-dependent triangulations (DDTs) is proposed. The proposed method is shown to produce meshes of higher quality (both in terms of mean squared error and subjectively) than those generated by two competing approaches, namely the greedy-point removal scheme of Demaret and Iske and a DDT-based scheme of Garland and Heckbert. Furthermore, our method is shown to achieve these excellent results at a relatively modest computational and memory cost.**

I. Introduction

In recent years, there has been a growing interest in (triangle) mesh representations of images. Such representations facilitate the use of nonuniform sampling and have proven beneficial in many applications, such as feature detection, pattern recognition, tomographic reconstruction, and image coding. With a mesh model of an image, the image domain is partitioned by a triangulation into a set of (triangle) faces, and then over each face of the triangulation an approximating function is constructed. Of the many classes of mesh representations proposed to date, the class based on Delaunay triangulations is extremely popular. In the case of a Delaunay triangulation, the connectivity of the triangulation (i.e., how the points in the triangulation are connected by edges) is determined solely by the geometry (i.e., position) of the points being triangulated. Another class of mesh representations is the class based on data-dependent triangulations (DDTs). In the case of DDTs, the connectivity of the triangulation is chosen in a way that depends on the data set from which points to be triangulated originated (and not just the geometry of those points). In this way, DDTs offer much greater flexibility, and theoretically can perform better than their Delaunay counterparts *if well chosen*. In practice, however, due to this increased flexibility, it is much more difficult to develop *computationally-efficient* mesh-generation schemes that are based on DDTs (compared to the Delaunay case).

Of the many Delaunay-based mesh-generation schemes proposed to date, one of the very best is the **greedy point removal (GPR)** method proposed by Demaret and Iske [1]. This method starts with a triangulation containing all sample points of the original image, and then iteratively removes sample points using some optimality criterion. While the method produces very high-quality meshes, it unfortunately has very high computational and memory costs. Although numerous DDT-based mesh-generation methods have been proposed, many have impractically high computation times. Of those schemes that are relatively fast, a particularly good one was proposed by Garland and Heckbert [2, p. 18, Algorithm IV] (with the quality threshold parameter `qthresh` chosen as 0.5 and an L^2 error measure). We henceforth refer to this method as the **Garland-Heckbert (GH)** method.

In this paper, we propose a new DDT-based mesh-generation method. The proposed method produces mesh representations of images with lower approximation error than than those generated by both the GPR and GH methods. At the same time, the computational and memory costs of the proposed method are comparable to the GH method and much lower than the GPR method. In passing, we note that the work presented herein corresponds to a preliminary version of the mesh-generation method proposed in our journal article [3]. Consequently, the work described herein has also been partially presented in [3]. Since the approach to final-connectivity adjustment used herein is simpler and requires less computation than the one in [3], this preliminary work is also of interest to the research community.

The remainder of this paper is organized as follows. Section II provides some background information on meshes for image representation. In Section III, our new mesh-generation method is presented. Then, in Section IV, the performance of our proposed method is evaluated by comparing it to the GPR and GH methods. Finally, Section V concludes the paper with a summary of our key results.

II. Meshes for Image Representation

In what follows, for a set S, $|S|$ denotes the cardinality of S. Consider an integer-valued image function 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). With a mesh model of an image, the image domain is partitioned by a triangulation into a set of (triangle) faces and then over each face of the triangulation an approximating function is constructed. A mesh model is completely characterized by: 1) the **sample points** $P = \{p_i\}_{i=0}^{|P|-1} \subset \Lambda$, and their corresponding function values $Z = \{z_i\}_{i=0}^{|P|-1}$, where $z_i = \phi(p_i)$; and 2) the set F of (triangle) faces formed by a triangulation of P (i.e., the connectivity of

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

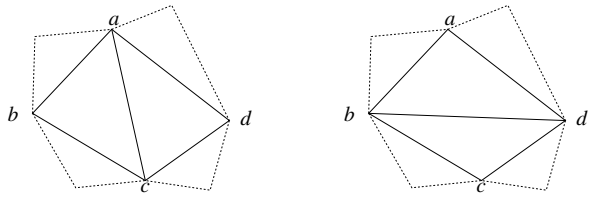

Fig. 1. Edge flip example. Part of (a) a triangulation with an edge *ac* and (b) the new triangulation obtained after the edge *ac* is transformed to the edge *bd* by an edge flip.

the vertices in a triangulation of P). Given P and F, a function $\hat{\phi}_{P,F}$ that interpolates ϕ at the points in P is constructed as follows. First, we form a continuous piecewise-linear function $\tilde{\phi}_{P,F}$. For each (triangle) face $f \in F$, $\tilde{\phi}_{P,F}$ is chosen as the unique linear function that interpolates ϕ at the three vertices of f. To ensure that $\hat{\phi}_{P,F}$ is integer valued (just like ϕ), we choose $\hat{\phi}_{P,F}$ as $\hat{\phi}_{P,F}(p) = \text{round}(\tilde{\phi}_{P,F}(p))$ for all $p \in \Lambda$, where round denotes an operator that rounds to an integer value. The set P must always include the extreme convex-hull points of Λ (i.e., the four corners of the image bounding box) so that the triangulation of P covers all points in Λ. As a matter of terminology, the **sampling density** of the mesh model is defined as $|P| \,/\, |\Lambda|$.

The mesh-generation problem that we address in this paper can be succinctly stated as follows: Given ϕ and a desired number N of sample points, find the mesh model of ϕ (i.e., P and F) with $|P| = N$ that minimizes the measure ϵ of the difference between ϕ and the approximation $\hat{\phi}_{P,F}$. In our work, the **mean squared error (MSE)** is used as the error measure, so that

$$\epsilon = |\Lambda|^{-1} \sum_{p \in \Lambda} \left(\hat{\phi}_{P,F}(p) - \phi(p) \right)^2 .$$

Herein, the MSE is typically expressed in terms of the **peak signal-to-noise ratio (PSNR)**, which is defined as $\text{PSNR} = 20 \log_{10} \left(\frac{2^\rho - 1}{\sqrt{\epsilon}} \right)$, where ρ is the number of bits/sample in the image ϕ.

Before proceeding to introduce our proposed method, we must first provide some additional background regarding triangulations. An edge e in a triangulation is said to be **flippable** if it has two incident faces (i.e., is not on the triangulation boundary) and the union of the two faces incident on e is a *convex* quadrilateral q (with no interior angles equal to 180°). If e is flippable, a valid triangulation is obtained if e is deleted from the triangulation and replaced by the other diagonal of the quadrilateral q. This transformation is known as an **edge flip**. For example, in Fig. 1, the edge *ac* is transformed to the edge *bd* by an edge flip. Any triangulation of a set of points can be reached from any other triangulation of the same set of points by a finite sequence of edge flips.

III. Proposed Mesh-Generation Method

Our proposed mesh-generation method is iterative in nature. It starts with a nearly empty mesh and adds points to the mesh until the desired sampling density is achieved. More specifically, our method consists of the following steps (in order): 1) Construct an initial triangulation, consisting of the extreme convex-hull points of the image domain Λ (i.e., the four corner points of the image bounding box). 2) Select a new point p^* to add to the triangulation, using an optimality criterion to be described shortly. 3) Insert p^* into the triangulation. 4) Update the connectivity of the triangulation by performing a sequence of edge flips, using an optimization procedure [4] with an edge-flipping criterion to be described shortly. 5) If the target sampling density has not yet been achieved, go to step 2 (i.e., add another point). 6) Further update the connectivity of the triangulation using a simple postprocessing scheme. In the preceding algorithmic summary, some details were necessarily omitted in the interest of simplicity. In what follows, we will now fill in these missing details.

The quantities Λ, ϕ, and $\hat{\phi}_{\cdot,\cdot}$ are as defined previously in Section II, and P and F denote, respectively, the set of sample points and set of faces currently in the mesh. For convenience, $\hat{\phi}$ is used as an abbreviation for $\hat{\phi}_{P,F}$ (i.e., the current approximating function for ϕ). Each point in the image domain Λ is assigned to *exactly one* face in the triangulation of P. If a point is strictly inside a face, the point is assigned to that face. If a point is on an edge or is a vertex in the triangulation, a scheme similar to [5] is used to uniquely assign the point to a face. The set of all points belonging to the face f is denoted $\text{points}(f)$. The **face error** of the face f, denoted $\text{faceErr}(f)$, is defined as

$$\text{faceErr}(f) = \sum_{p \in \text{points}(f)} \left(\hat{\phi}(p) - \phi(p) \right)^2 ,$$

(i.e., $\text{faceErr}(f)$ is the squared error summed over the points in the face f). The **approximate diameter** of the face f, denoted $\text{diam}(f)$, is defined as the length of the longest side of the (smallest) bounding box of the face f. The **shape quality** of a face f, denoted $\text{quality}(f)$, is defined as

$$\text{quality}(f) = \text{area}(f) / \,\text{diam}(f),$$

where $\text{area}(f)$ denotes the area of the face f.

Point insertion (step 3). In step 3 of our method, the new point p^* needs to be inserted into the triangulation, which is accomplished as illustrated in Fig. 2. If p^* is strictly inside a face in the triangulation, say face *abc*, we connect p^* by new edges to each vertex of its containing face, as shown in Fig. 2(a). If p^* is on an edge in the triangulation, say on edge *ac*, we split this edge at p^* and, for each face f incident on *ac*, we connect p^* with a new edge to the vertex in f that is opposite the edge *ac*, as shown in Fig. 2(b).

Point selection (step 2). In step 2 of our method, we must select the new point p^* to insert into the triangulation. This is accomplished in two stages. First, we select the face f^* in the triangulation into which a new point is to be inserted. Then, we choose a point p^* in the face f^* for insertion. The face f^* is selected according to

$$f^* = \arg\max_{f \in F} \text{faceErr}(f),$$

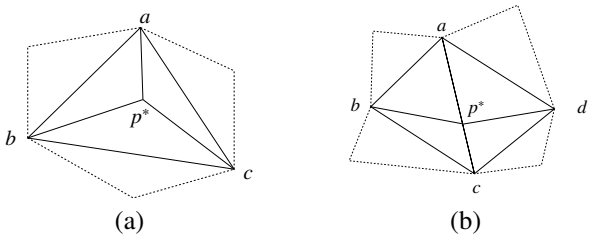

Fig. 2. Point insertion example. Part of a triangulation showing how the new vertex p^* is inserted (a) inside a triangle abc and (b) on an edge ac.

where F is the set of all faces in the triangulation (i.e., f^* is the face with the largest face error). Having chosen the face f^*, we select the point p^* to insert. This is done in two stages. First, we choose a set T of test points to consider as candidates for insertion. We then choose p^* from T. If $\left|\text{points}(f^*)\right| > 8$, T is chosen as the 8 points p in points(f^*) for which $\left|\hat{\phi}(p) - \phi(p)\right|$ is largest. (As an aside, we note that the value of 8 here was chosen based on experimentation.) Otherwise, we choose T = points(f^*). Given T, we choose p^* as

$$p^* = \arg\min_{t \in T} \sum_{p \in \text{points}(f^*)} (\hat{\phi}_{P \cup \{t\}, (F \setminus f^*) \cup F^*}(p) - \phi(p))^2, \qquad (1)$$

where F^* denotes the set of faces that would replace the face f^* if the point t were inserted into the triangulation (of P) as explained in the description of step 3 above (i.e., we choose p^* as the point in T whose insertion would result in the greatest decrease in the approximation error over the face f^*).

Connectivity update (step 4). In step 4 of our method, the connectivity of the triangulation is updated using an optimization procedure. A cost function is used to measure the badness of each edge. The cost of the (flippable) edge e, denoted edgeCost(e), is defined as

$$\text{edgeCost}(e) = \frac{\text{faceErr}(f_1) + \text{faceErr}(f_2)}{\text{quality}(f_1)\,\text{quality}(f_2)},$$

where f_1 and f_2 are the two faces incident on e. As a matter of terminology, an edge is said to be **optimal** if: 1) it is not flippable; or 2) it is flippable and the cost of the edge after flipping is strictly less than the cost of the edge before flipping. An edge whose optimality is uncertain is said to be **suspect**. The connectivity update process essentially tests any suspect edges for optimality, and performs edge flips to eliminate any suspect edge that is not optimal.

Let S denote the current set of suspect edges. In step 3, we inserted the new point p^* into the triangulation. Initially, we set S to all flippable edges belonging to faces that are incident on p^*. For example, in Fig. 2(a), the edges ab, bc, and ca are suspect. The edges p^*a, p^*b, and p^*c are not suspect, since they are not flippable. As another example, in Fig. 2(b), the edges ab, bc, cd, da, and p^*a are suspect. The edges p^*b, p^*c, and p^*d are not suspect, as they are not flippable. The optimization procedure then proceeds as follows: 1) If $|S| = 0$ (i.e., S is empty), then stop. 2) Remove an edge e from S. 3) If e is not flippable, go to step 1. 4) Let q denote the convex quadrilateral for which e is a diagonal. Let e' denote the edge obtained by flipping e (i.e., the other diagonal of q). If edgeCost(e') < edgeCost(e), flip e and add all edges on the boundary of q to S. For example, for the scenario depicted in Fig. 1, after flipping the edge ac to bd, the edges ab, bc, cd, and da would each be added to S if flippable. 5) Go to step 1.

Final connectivity adjustment (step 6). In step 6 of our method, a final adjustment is made to the connectivity of the triangulation through a simple postprocessing scheme. This scheme works as follows. Let E denote the set of all edges in the triangulation. For each flippable edge $e \in E$, if flipping the edge e results in a strictly lower approximation error, the edge e is flipped. (Unlike the case of the optimization procedure used in step 4, the final connectivity adjustment scheme has no notion of suspect edges and edge flips cannot propagate.)

TABLE I. TEST IMAGES

Image	Size, Bits/Sample	Description
`bull`	1024×768, 8	computer-generated bull
`ct`	512×512, 12	CT scan of head [6]
`glasses`	1024×768, 8	raytraced glasses
`lena`	512×512, 8	woman [7]
`peppers`	512×512, 8	collection of peppers [7]
`x_ray`	2048×1680, 12	X-ray of pelvis [6]

IV. EXPERIMENTAL RESULTS

Having introduced our proposed method, we now evaluate its performance by comparing it to the GPR and GH methods introduced in Section I. In our evaluation, we consider both mesh quality (in terms of PSNR and subjective quality) and time/memory complexity. In our work, we employed more than 40 images as test data. Herein, we present results for a representative subset of these images, namely, those listed in Table I. This subset was deliberately chosen to contain a variety of image types including photographic, medical, and computer-generated imagery.

Mesh quality. For all images in our test set and several sampling densities, we used each of the methods under consideration to generate a mesh and then measured the resulting approximation error in terms of PSNR. The results obtained are shown in Table II, with the best result in each case (i.e., each row in the table) highlighted in **boldface**.

To begin, we compare our proposed method to the GPR scheme. From the results of Table II, we can see that our proposed method outperforms the GPR scheme in 23 out of 24 test cases by a margin of up to 2.85 dB with a median of 0.75 dB. The only case in which the GPR scheme fares better is the `x_ray` image at a sampling density of 1%. In this instance, the GPR scheme only beats our proposed method by 0.02 dB which is essentially negligible. The excellent relative performance of our proposed method is particularly noteworthy when one considers that, as we shall see later, the GPR scheme takes significantly more computation time and a few orders of magnitude more memory. Next, we compare our proposed method to the GH scheme. From the results of Table II, we can see that our proposed method outperforms the GH scheme in all 24 test cases by a margin ranging from 0.46 to 2.85 dB, with a median of 1.60 dB. Clearly, the above results show our

TABLE II. COMPARISON OF THE MESH QUALITY OBTAINED WITH THE VARIOUS MESH-GENERATION METHODS

Image	Sampling Density (%)	PSNR (dB)		
		Proposed	GPR	GH
bull	0.125	**35.97**	33.12	33.12
	0.250	**40.14**	38.23	38.28
	0.500	**42.48**	41.87	40.73
	1.000	**44.19**	43.99	42.48
ct	0.250	**33.66**	32.15	32.22
	0.500	**38.14**	37.22	37.68
	1.000	**42.66**	41.35	42.01
	2.000	**47.39**	45.33	46.63
glasses	0.500	**26.47**	25.84	25.07
	1.000	**29.74**	28.88	28.87
	2.000	**33.73**	32.73	33.13
	3.000	**36.55**	35.24	35.94
lena	0.500	**27.43**	26.66	25.37
	1.000	**30.11**	29.12	28.51
	2.000	**32.55**	31.82	31.26
	3.000	**33.96**	33.37	32.78
peppers	0.250	**24.66**	23.93	22.58
	0.500	**27.71**	27.09	25.95
	1.000	**30.36**	30.05	28.77
	2.000	**32.48**	32.39	31.14
x_ray	0.125	**40.60**	39.79	38.61
	0.250	**42.23**	41.97	40.39
	0.500	**43.70**	43.66	41.82
	1.000	45.05	**45.07**	43.24

proposed method to be vastly superior to both the GPR and GH schemes in terms of mesh quality.

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 II), a small part of each image reconstruction is shown under magnification in Fig. 3, along with the corresponding image-domain triangulation. Examining the figure, we can see that our proposed method produces an approximation that better captures image details such as image edges, whereas the GPR and GH schemes tend to produce more severe artifacts in the vicinity of image edges.

Time complexity. Due to space constraints, we cannot provide a detailed comparison of the time complexities of the various methods under consideration herein. So, to give the reader some insight in this regard, we provide a representative subset of some timing results collected on very modest hardware (namely, a seven year old notebook computer with a 3.4 GHz Intel Pentium 4 and 1 GB of RAM). For the lena image and sampling densities in the range 0.5% to 3%, the proposed and GH methods each have an execution time of less than 5.5 seconds, while the GPR method has a much higher computational cost, requiring approximately 43 seconds. The proposed method typically has an execution time about 1.6 to 1.8 times that of the GH method, due largely to the cost of computing (1). This modest increase in computational cost is certainly quite reasonable considering that the proposed method still only takes a few seconds to execute and produces meshes of much higher quality than the GH method. Furthermore, the proposed method also produces meshes of higher quality than the state-of-the-art GPR method, in only a small fraction of the time. In this sense, our proposed method performs exceptionally well for its relatively modest execution-time cost.

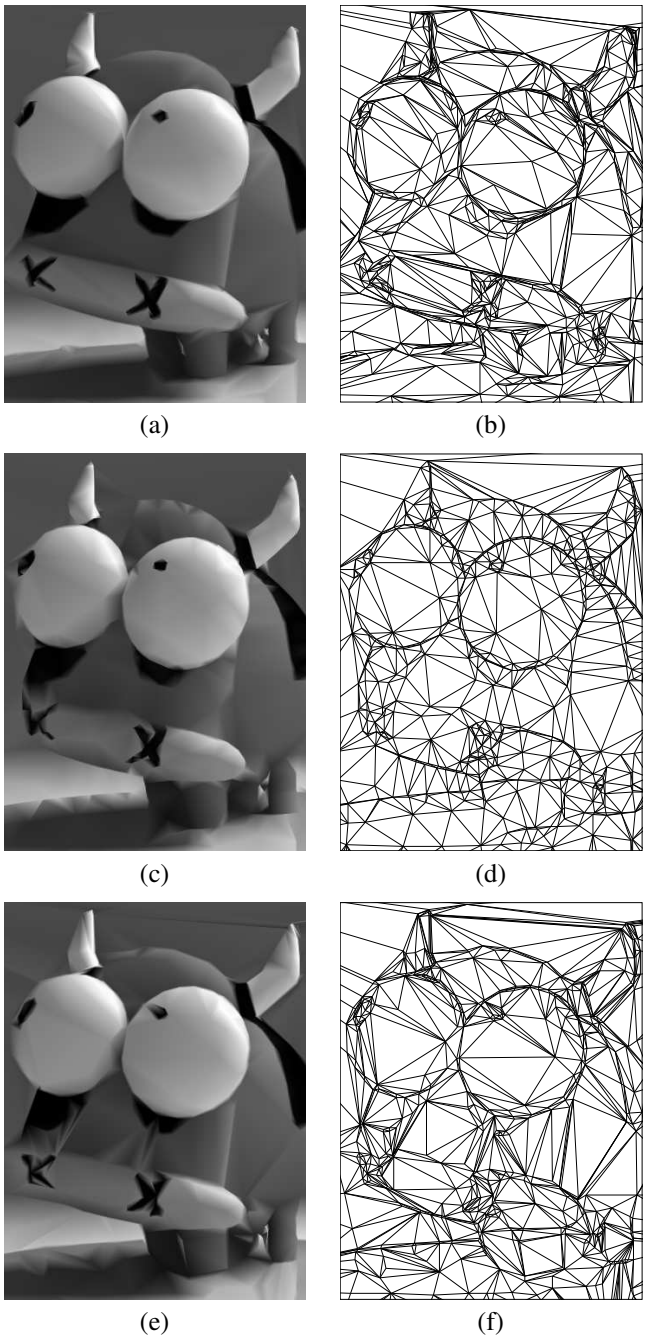

Fig. 3. Part of the image approximation obtained for the bull image at a sampling density of 0.125% with the (a) proposed (35.97 dB), (c) GPR (33.12 dB) and (e) GH (33.12 dB) methods and (b), (d), (f) their corresponding triangulations.

Memory complexity. For each of the methods under consideration, memory usage is dominated by the data structure employed to represent the mesh (i.e., triangulation). In all cases, the size of the mesh data structure is proportional to the number of vertices (i.e., sample points) in the mesh. Therefore, the peak memory usage is determined by the peak mesh size. For an image of width W, height H, and a given sampling

density D, the peak mesh size for each of the proposed, GH, and GPR methods is DWH, DWH, and WH, respectively. So, the memory usage for the proposed and GH methods is essentially the same, while, for values of D of practical interest, say $D \in [0.125\%, 3\%]$, the GPR method requires 33 to 800 times more memory than the proposed and GH methods. Thus, in terms of memory usage, our proposed method fares quite well (i.e., is tied for the best).

V. Conclusions

In this paper, we proposed a new mesh-generation method for image representation based on DDTs. The performance of our proposed scheme was compared to the GPR method, a leading approach based on Delaunay triangulations, as well as the GH method, a highly-effective technique based on DDTs. Through experimental results, our proposed method was shown to produce much higher quality meshes (both in terms of PSNR and subjective quality) than both the GPR and GH methods. Furthermore, our proposed method was found to require very substantially less computation and memory than the GPR scheme. Relative to the GH scheme, our proposed method was seen to have essentially the same memory cost and require only a modest increase in computational cost, considering the much higher quality meshes produced by our method. Our mesh-generation method can be of great benefit to the many applications that employ mesh models of images.

References

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

[2] 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.

[3] P. Li and M. D. Adams, "A tuned mesh-generation strategy for image representation based on data-dependent triangulation," *IEEE Trans. on Image Processing*, vol. 22, no. 5, pp. 2004–2018, May 2013.

[4] C. L. Lawson, "Software for C^1 surface interpolation," in *Mathematical Software III*, J. R. Rice, Ed. New York, NY, USA: Academic Press, 1977, pp. 161–194.

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

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

[7] "USC-SIPI image database," http://sipi.usc.edu/database, 2011.