AN INCREMENTAL/DECREMENTAL DELAUNAY MESH-GENERATION FRAMEWORK FOR IMAGE REPRESENTATION

Michael D. Adams

Dept. of Elec. and Comp. Eng., University of Victoria, Victoria, BC, V8W 3P6, Canada

ABSTRACT

A flexible new mesh-generation framework for image representation is proposed, based on Delaunay triangulations. It is shown that this framework can be used to form a postprocessing optimization step that, when added to other previously-proposed methods, yields meshes of much greater quality. A new mesh-generation method, called IDDT, is also derived using the proposed framework, and shown to yield meshes of much better quality than those produced by previously-proposed methods as well as optimized versions of these methods constructed with the proposed framework.

***Index Terms*—** Image representations, mesh generation, Delaunay triangulations, triangle meshes.

1. INTRODUCTION

Image representations that employ adaptive (i.e., nonuniform) sampling have been receiving increasing attention in recent years, as such representations are useful in many applications, including filtering, feature detection, restoration, tomographic reconstruction [1], computer vision, pattern recognition, and image/video coding [2]. Although many classes of adaptively-sampled image representations have been proposed to date, those based on Delaunay triangulations have proven to be particularly effective, and are the focus of the work described herein. Since image acquisition systems typically produce data that is sampled on a (truncated) lattice, in order to generate an adaptively-sampled representation, a means is needed for selecting a good subset of the original sample points of the image from which to form such a representation. This is the so called mesh-generation problem.

Two highly effective mesh-generation methods proposed to date are the **MGH** scheme proposed in [2] and the **error-diffusion (ED)** scheme proposed in [1]. In this paper, we propose a flexible new mesh-generation framework and two methods derived from it, known by the names IDDT and BPR. We explain how our BPR scheme can be used to form a postprocessing optimization step that can be added to the MGH and ED methods in order to obtain meshes of much greater quality. Also, the IDDT method is shown to produce meshes of vastly superior quality relative to the MGH and ED schemes and their optimized versions constructed with the BPR scheme.

The remainder of this paper is structured as follows. Section 2 begins by providing some background information on mesh models for image representation. In Section 3, our new mesh-generation framework is introduced along with our IDDT and BPR methods. Then, Section 4 offers some guidance regarding the efficient implementation of our framework. To demonstrate the effectiveness of the proposed framework and methods, some experimental results are presented in Section 5. Finally, Section 6 concludes the paper with a summary of the key results.

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

2. MESHES FOR IMAGE REPRESENTATION

In the context of this work, a mesh model of an image ϕ defined on $\Lambda = \{0, 1, \ldots, W-1\} \times \{0, 1, \ldots, H-1\}$ (i.e., a rectangular grid of width W and height H) is completely characterized by: 1) a set $P = \{p_i\}_{i=1}^{|P|}$ of sample points; and 2) the set $Z = \{z_i\}_{i=1}^{|P|}$ of the corresponding sample values (i.e., $z_i = \phi(p_i)$). The set P is always chosen to include the extreme convex hull points of Λ (i.e., the four corner points of the image bounding box) so that the triangulation of P covers the entire image domain Λ. From P and Z, a unique interpolant $\hat{\phi}_P$ is constructed as follows. First, we form the Delaunay triangulation of P, which is ensured to be uniquely determined (from P) by employing the preferred-directions scheme [3]. Then, the interpolant $\hat{\phi}_P$ is constructed by forming, for each face of the triangulation, the unique plane that passes through the sample values at the three vertices of the face. Since the vertices of the triangulation are nothing more than the sample points, often the terms "sample point" and "vertex" are used interchangeably herein. As a matter of terminology, the **sampling density** of the mesh model is defined as $|P| \,/\, |\Lambda|$.

Having defined the above mesh model, the mesh-generation problem that we address herein can be succinctly stated as follows: For a given target number N of sample points (where $N < |\Lambda|$), choose $P \subset \Lambda$ such that $|P| = N$ and the mesh approximation error $\epsilon(P)$ is as small as possible (ideally, a global minimum). In our work, the mean squared error is used as the error metric, so that $\epsilon(P) = |\Lambda|^{-1} \sum_{p \in \Lambda} (\hat{\phi}_P(p) - \phi(p))^2$. Finding good practically-useful methods for solving the above problem is quite challenging, since problems like this are known to be NP hard.

3. PROPOSED MESH-GENERATION FRAMEWORK

Having introduced the particular mesh model for image representation assumed in our work, we are ready to present our proposed mesh-generation framework. On a very basic level, our framework takes the following information as input: 1) an image ϕ sampled at the points Λ (which form a rectangular grid), 2) a subset P_0 of the sample points Λ from which to form an initial mesh approximation of the image, and 3) the target number N of sample points that should be present in the mesh to be generated (where $N \in [4, |\Lambda|]$). The framework is iterative in nature. Let P denote the sample points of the mesh in the current iteration. To begin, P is initialized to P_0. Then, in each iteration, a single point is either added to or deleted from P. In order to either force or prevent certain points in Λ from appearing in the final mesh, the notion of mutability of a point is introduced. A point $p \in \Lambda$ is said to be **mutable** if it is permitted to be added to or deleted from the mesh in this iterative framework.

(A point that is not mutable is said to be **immutable**.) Since the mesh model must always include the four corner points of the image bounding box, these points are always included in P_0 and marked as immutable (so that they cannot be later deleted). All other points are initially marked as mutable.

Before proceeding further, it is necessary to introduce some additional definitions and notation. Let $\hat{\phi}$ be the interpolant associated with P. Let F denote the set of all faces in the Delaunay triangulation of P. Each point $p \in \Lambda$ is assigned to *exactly one* face in F, which is denoted $\mathrm{face}(p)$. If p is strictly inside a face f, $\mathrm{face}(p) = f$; otherwise (for points on edges or vertices), a method like the one in [4] is used to uniquely assign p to *exactly one* face. The set of all points p satisfying $\mathrm{face}(p) = f$ (i.e., all points belonging to the face f) is denoted $\mathrm{points}(f)$. For a point $p \in \Lambda$, $\mathrm{candErr}(p)$ denotes a local error measure that quantifies, in some way, the difference between $\hat{\phi}(p)$ and $\phi(p)$. Unless otherwise indicated, we choose $\mathrm{candErr}(p) = |\hat{\phi}(p) - \phi(p)|$ (i.e., absolute error). The **significance** of a mutable point $p \in P$, denoted $\mathrm{vertexSig}(p)$, is the amount by which the squared error increases if p were deleted from the mesh, computed over all points in $R \cap \Lambda$, where R is the region in the triangulation affected by the deletion of p. The squared error computed over all points $p \in \Lambda \cap \mathrm{points}(f)$ is denoted as $\mathrm{faceErr}(f)$ (i.e., $\mathrm{faceErr}(f) = \sum_{p \in \Lambda \cap \mathrm{points}(f)} (\hat{\phi}(p) - \phi(p))^2$). The **candidate point** for a face f, denoted $\mathrm{cand}(f)$, is defined as the mutable point $p \in (\Lambda \setminus P) \cap \mathrm{points}(f)$ that maximizes $\mathrm{candErr}(p)$.

With the above definitions in place, we can now specify in more concrete terms the process whereby points are added/deleted. In particular, in each iteration, the mesh can be modified by applying exactly one of the following two operations: 1) optimal add: this operation adds the (mutable) point $p = \mathrm{cand}(f)$ to the mesh, where f is the face in F that maximizes $\mathrm{faceErr}(f)$; 2) optimal delete: this operation deletes the point p from the mesh, where p is the mutable point in P that minimizes $\mathrm{vertexSig}(p)$. In simple (but less precise) terms, the optimal-add operation adds to the mesh the candidate point of the face with the highest squared error, while the optimal-delete operation deletes from the mesh the point that will cause the least increase in squared error. The iterative framework simply performs optimal add/delete operations until $|P| = N$ and no further optimal add/delete operations are desired.

A few comments are in order regarding the proposed framework. First, it is important to note that this framework is greedy in nature. That is, it makes the choice of which point to insert/delete in a given iteration without regard to how this choice affects *later* iterations. Consequently, this framework does not guarantee a globally optimal solution. Practically speaking, however, no *computationally-tractable* algorithm likely exists for producing a globally optimal solution, since mesh-generation problems like the one addressed in this paper are NP hard. Next, it is important to understand that the (probably suboptimal) result produced from this framework is *very heavily dependent* on the choice of P_0 as well as the specific sequence of points added/deleted that lead to the final mesh. For example, given a mesh, the number of sample points in the mesh could be increased by 100 by either: 1) performing two optimal-add operations followed by one optimal-delete operation, repeated 100 times; or 2) performing 200 optimal-add operations followed by 100 optimal-delete operations, only once. Although both scenarios have the same total number of optimal-add and optimal-delete operations, the quality of the resulting mesh in each case could be radically different. Furthermore, computational complexity is also strongly influenced by the choice of P_0 as well as the specific sequence of optimal-add/delete operations. For example, for reasons that will become clearer later in Section 4, it is more computationally efficient to group optimal add/delete operations of the same type together. Therefore, scenario 2 from above would most likely require less computation. The flexibility of our framework comes from the fact that there are a great many ways in which to choose P_0 as well as the precise sequence of optimal add/delete operations used to produce the final mesh.

BAD-POINT-REPLACEMENT (BPR) METHOD. As a matter of terminology, a (mutable) point p in the mesh is said to be **bad**, if $\mathrm{vertexSig}(p) \leq 0$ (i.e., the deletion of p would not cause an increase in the mesh approximation error). Clearly, bad points are undesirable since their inclusion in the mesh either increases the mesh approximation error (if $\mathrm{vertexSig}(p) < 0$) or leaves the mesh approximation error unchanged (if $\mathrm{vertexSig}(p) = 0$). As it turns out, with our framework, when the target number of points is finally achieved and the mesh-generation process would normally terminate, there is the possibility that some bad points will be present in the mesh. Depending on the initial subset P_0 and the particular sequence of additions/deletions, the number of bad points could, in fact, be quite large.

To combat the degradation in mesh quality caused by the presence of bad points, we have devised a technique for eliminating such points called the **bad-point-replacement (BPR)** method. This method works by deleting bad points and substituting other new points in their place. This is done in such a way as to not result in any net change in the number of points in the mesh (i.e., the number of add operations and number of delete operations employed are equal). This method is intended to be performed as a final step in the mesh-generation process, once a mesh with the target number of points has been obtained. In more detail, the BPR method consists of the following steps: 1) while the point p that would be deleted by the next optimal-delete operation satisfies $\mathrm{vertexSig}(p) \leq 0$, perform an optimal-delete operation, and mark p as immutable; if no points were deleted in this step, stop; 2) perform n optimal-add operations, where n is the number of points deleted in step 1; 3) go to step 1. In step 1, p is marked as immutable in order to avoid p being added back to the mesh in subsequent iterations, which could cause the algorithm to become trapped in an infinite loop, repeatedly cycling through the same sequence of addition/deletions.

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

Although the BPR method constitutes a useful tool, it is really only intended to be employed as a postprocessing step used in conjunction with some standalone mesh-generation method. Since the BPR method does not change the number of points in the mesh, it has to rely on some other mesh-generation scheme to provide a mesh of the desired size, which can then be optimized. With the above in mind, we now turn our attention to a standalone mesh-generation method derived from our framework.

IDDT METHOD. Since the proposed mesh-generation framework is very flexible, many different mesh-generation methods can be constructed from it. In what follows, we present one of the more effective of these methods that we have found to date. This particular method we refer to by the name IDDT. The IDDT method selects, as the initial mesh P_0, the four corner points of the image

bounding box. Then, by using both optimal add and delete operations, the number of mesh points is increased until the target number of points is reached. Finally, the BPR method is used to remove any bad points. In more detail, the IDDT method consists of the following steps: 1) let $n = N - |P|$ (where P is the set of points currently in the mesh); if $n \leq 0$, go to step 5; 2) perform n optimal-add operations; 3) perform $\lfloor n/2 \rfloor$ optimal-delete operations; 4) go to step 1; 5) apply the BPR method to the mesh. The $\mathrm{candErr}$ function is chosen as $\mathrm{candErr}(p) = d(p)|\hat{\phi}(p) - \phi(p)|$, where $d(p)$ denotes the maximum magnitude second-order directional derivative of ϕ at p. As a practical matter, d is computed using the formula given in [1], where the associated partial-derivative operators are applied to a smoothed version of ϕ, and the smoothing operator employed is the tensor product of two one-dimensional filters with transfer function $z^4(\frac{1}{2} + \frac{1}{2}z^{-1})^8$.

4. IMPLEMENTATION

Although our proposed mesh-generation framework is conceptually simple, implementing it in a computationally efficient manner is tricky and requires careful software design. A naive implementation could easily require several orders of magnitude more computation than is strictly necessary. Below, we offer some guidance as to how our mesh-generation framework can be efficiently implemented, by describing the particular implementation strategy that we employed.

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

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

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

Table 1. Test Images

Image	Size	Bits/Sample	Description
`bull`	1024×768	8	computer-generated
`ct`	512×512	12	tomography [6]
`lena`	512×512	8	woman [7]
`peppers`	512×512	8	collection of peppers [7]

As the description above implies, the computation associated with updating the vertex priority queue (e.g., re-evaluating $\mathrm{vertexSig}$ values for vertices) is deferred until the result is absolutely needed (namely, when an optimal-delete operation is to be performed). In situations where multiple optimal-add operations are performed without an intervening optimal-delete operation, this deferred processing can save a considerable amount of computation. This savings results from avoiding vertex-priority updates that would later be rendered unnecessary by other optimal-add operations. Lastly, in order to further reduce computational complexity, we employ two additional optimizations: 1) the face priority queue is not initialized until the first optimal-add operation; and 2) the vertex priority queue is not initialized until the first optimal-delete operation. These optimizations save considerable time when the mesh-generation process begins with 1) a large number of optimal-add operations without an intervening optimal-delete operation; or 2) a large number of optimal-delete operations without an intervening optimal-add operation.

5. RESULTS

Having introduced our proposed mesh-generation framework, we will now demonstrate its utility by showing that: 1) the BPR scheme (derived from our framework) can be added as a postprocessing step to other previously-proposed methods to yield meshes of much higher quality; and 2) the IDDT method (also derived from our framework) produces meshes of better quality relative to other effective methods. For the purposes of evaluation herein, we consider 28 (grayscale) images, namely, the four listed in Table 1 plus the 24 images of the Kodak test set [5].

As stated earlier, given an arbitrary mesh-generation method (i.e., one that is not necessarily derived from our framework), it is often the case that method may yield meshes with some bad sample points. (Note that, in this context, we mean "bad" in the specific sense introduced before in Section 3.) Earlier (in Section 1), we mentioned two previously-proposed mesh-generation methods that have proven quite effective, namely the MGH and ED schemes. Through experimentation, we have discovered, perhaps surprisingly, that the MGH and ED methods both typically yield meshes with a significant number of bad sample points. In the case of the MGH method, it is not unusual for about 10% of the sample points to be bad; while in the case of the ED method, often about 50% of the sample points are bad. Thus, the MGH and ED methods could both potentially benefit from the use of our BPR scheme.

To allow us to evaluate the benefit of our BPR scheme, we consider modified versions of the MGH and ED methods, called **optimized MGH (OMGH)** and **optimized ED (OED)**, respectively, which include our BPR scheme as a postprocessing step. That is, the OMGH method first uses the MGH scheme to produce a mesh with the desired number of sample points, and then our BPR scheme is applied to the resulting mesh. Similarly, the OED method first employs the ED scheme to produce a mesh with the desired number of sample points, and then our BPR scheme is applied to the resulting mesh.

Table 2. Comparison of mesh quality for various mesh-generation methods. (a) Results for four specific images; and (b) results averaged across 28 images.

(a)

Image	Samp. Density (%)	PSNR (dB) MGH	OMGH	ED	OED	IDDT
lena	0.5	24.26	25.17	17.17	25.37	**25.81**
	1.0	26.87	27.77	21.13	27.92	**28.54**
	2.0	29.74	30.49	25.83	30.60	**31.09**
	3.0	31.33	32.01	28.06	32.06	**32.51**
peppers	0.5	24.68	25.66	16.03	25.59	**26.50**
	1.0	27.53	28.38	21.35	28.78	**29.15**
	2.0	29.85	30.76	26.09	31.09	**31.31**
	3.0	31.13	31.89	28.17	32.19	**32.45**
ct	0.25	29.74	31.25	17.81	30.72	**32.25**
	0.5	35.13	36.45	21.61	35.82	**37.59**
	1.0	39.70	40.45	29.45	40.44	**41.42**
	2.0	43.78	44.44	35.62	44.02	**45.39**
bull	0.25	35.29	36.55	20.59	36.44	**37.56**
	0.5	38.76	40.13	25.89	40.15	**40.48**
	1.0	41.07	42.09	33.34	42.21	**42.46**
	2.0	43.07	43.97	37.56	44.06	**44.38**

(b)

Samp. Density (%)	PSNR (dB) MGH	OMGH	ED	OED	IDDT
0.25	20.89	22.11	15.60	22.66	**22.78**
0.5	22.89	23.91	17.67	24.46	**24.61**
1.0	25.00	25.86	20.77	26.34	**26.56**
2.0	27.38	28.11	24.01	28.42	**28.73**
3.0	28.92	29.60	25.87	29.78	**30.14**

Now, we are ready to present some experimental results to compare the performance of the various mesh-generation methods introduced herein (namely, the MGH, OMGH, ED, OED, and IDDT methods). For numerous images and target sampling densities, each of the MGH, OMGH, ED, OED, and IDDT methods was used to generate a mesh and the resulting mesh quality was measured in terms of peak-signal-to-noise ratio (PSNR). A representative subset of the results, involving four specific images, is shown in Table 2(a), while the average results computed across all 28 test images are shown in Table 2(b). In each case (i.e., table row), the best result is indicated in boldface.

MGH VS. OMGH AND ED VS. OED. First, we compare the performance of the MGH and OMGH methods. A quick inspection of the results in Tables 2(a) and (b) shows that the OMGH method beats the MGH scheme in every case. In the case of the four images in Table 2(a) and the average results in Table 2(b), the OMGH method outperforms the MGH scheme by 0.6 to 1.6 dB and 0.6 to 1.22 dB, respectively. So, clearly, the addition of our BPR method in the OMGH scheme has resulted in a significant improvement in mesh quality. Next, we compare the performance of the ED and OED methods. A quick glance at the data in Tables 2(a) and (b) reveals that the OED method outperforms the ED scheme in every case. In the case of the four images in Table 2(a) and the average results in Table 2(b), the OED method outperforms the ED scheme by 4.0 to 15.9 dB and 3.9 to 7.1 dB, respectively. Obviously, the addition of our BPR method in the OED scheme has led to a very marked improvement in mesh quality. From these results, it is clear that our BPR method is extremely effective.

IDDT VS. MGH/OMGH/ED/OED. Next, we compare the performance of our IDDT method to all four of the other methods. From the results of Tables 2(a) and (b), it is evident that the IDDT method outperforms all of the other methods under consideration, yielding the best result in every case. In terms of the summary results in Table 2(b), the IDDT method performs best in every case and beats the second-place contender (which is either OMGH or OED) by 0.12 to 0.36 dB. Relative to the previously proposed MGH and ED methods, the IDDT method is vastly superior, beating the best of the MGH and ED methods in each case by 1.22 to 1.89 dB. The large margin by which our IDDT method outperforms previously-proposed methods, like the MGH and ED schemes, demonstrates the effectiveness of our method.

Lastly, we would like to briefly note that, for sampling densities of practical interest, all of the methods under consideration here take a relatively modest amount of computation time. For example, for the lena image at a sampling density of 2%, all of the methods take less than 3.1 seconds on a six-year old notebook computer (with a 3.4 GHz Intel Pentium 4 and 1 GB RAM). Undoubtedly, these execution times would be even shorter on more modern hardware.

6. CONCLUSIONS

In this paper, we have proposed a new very-flexible mesh-generation framework for image representation, along with two methods derived from this framework, namely the BPR and IDDT schemes. The BPR method was shown to be highly effective as a postprocessing step to improve upon the results produced by other mesh-generation methods. In particular, this postprocessing strategy was shown to yield much higher quality meshes when applied to the previously-proposed MGH and ED schemes. The IDDT method was shown to produce meshes having vastly superior quality to those produced with previously-proposed methods. The BPR and IDDT methods that we have proposed can benefit the numerous applications where adaptively-sampled image representations are needed. Moreover, by further exploring the many other algorithmic possibilities that our proposed framework affords, we are optimistic that it will be possible to derive even more effective mesh-generation schemes.

7. REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

[5] "Kodak lossless true color image suite," 2010, http://r0k.us/graphics/kodak.

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

[7] "USC-SIPI image database," 2010, http://sipi.usc.edu/database.