# A Novel Progressive Lossy-to-Lossless Coding Method for Mesh Models of Images

Xiao (Joyce) Feng and Michael D. Adams
Dept. of Electrical and Computer Engineering, University of Victoria
Victoria, BC, V8W 2Y2, Canada
xiaofeng@uvic.ca and mdadams@ece.uvic.ca

***Abstract*—A novel progressive lossy-to-lossless coding method is proposed for mesh models of images whose underlying triangulations have arbitrary connectivity. For a triangulation $T$ of a set $P$ of points, our proposed method represents the connectivity of $T$ as a sequence of edge flips that maps a uniquely-determined Delaunay triangulation of $P$ to $T$. The coding efficiency of our method is highest when the underlying triangulation connectivity is close to Delaunay, and slowly degrades as connectivity moves away from being Delaunay. Experimental results show our method to significantly outperform a simple baseline coding scheme and suggest our method to be superior to more traditional coding approaches for meshes whose connectivity does not deviate too far from being Delaunay.**

## I. Introduction

In recent years, there has been a growing interest in image representations that utilize nonuniform sampling. By using nonuniform sampling, the position of sample points can be made adaptive to image content, allowing more accurate image representations to be obtained with fewer sample points. One popular class of image representations facilitating nonuniform sampling is that based on triangle meshes [1]. With a triangle mesh model of an image, the image domain is partitioned into triangular regions using a triangulation and then an approximating function is constructed over each such region. In order to be able to efficiently store and communicate (triangle) mesh models of images, effective coding schemes for such models are needed.

One highly effective approach for encoding mesh models of images is the **image tree (IT) method**, proposed in [2]. Unfortunately, the IT method only encodes the geometry of the mesh and not the connectivity, as the connectivity is assumed to be Delaunay. This renders the IT scheme useless in the many applications that employ mesh models with arbitrary connectivity. In this paper, we extend the IT method by adding to it a means for coding mesh connectivity. This allows mesh models with arbitrary connectivity to be coded, such as those produced in [1]. Through experimental results, we show that our proposed coding method is able to significantly outperform a simple baseline coding scheme. Furthermore, for meshes whose connectivity does not deviate too far from being Delaunay, our method is able to code mesh connectivity using less than the 3.67 bits/vertex [3] attained by traditional approaches. This result is of practical significance, since, in many applications, mesh connectivity is often not so far from being Delaunay, due to the good approximation properties of Delaunay triangulations.

978-1-4673-7788-1/15/$31.00 

The remainder of this paper is organized as follows. To begin, Section II provides some background information related to mesh models of images. In Section III, we propose our general framework for mesh coding, which has two free parameters. In Section IV, we consider how to best select the two free parameters of our framework, leading us to recommend a particular choice for these parameters and propose a specific coding method that employs this choice in our framework. The mesh models used for testing purposes in our work are also briefly introduced in Section IV. Through experimental results, the coding efficiency of our proposed method is evaluated in Section V. Finally, Section VI concludes the paper by summarizing the key aspects of our work.

## II. Background

Before proceeding further, we first introduce some of the basic notation and terminology used herein. The sets of integers and real numbers are denoted $\mathbb{Z}$ and $\mathbb{R}$, respectively. The cardinality of a set $S$ is denoted $|S|$. Similarly, the length of a (finite-length) sequence $S$ is denoted $|S|$. For two line segments $\overline{a_0a_1}$ and $\overline{b_0b_1}$, we say that $\overline{a_0a_1} < \overline{b_0b_1}$ in lexicographic order if and only if either 1) $a_0' < b_0'$, or 2) $a_0' = b_0'$ and $a_1' < b_1'$, where $a_0' = \min(a_0, a_1)$, $a_1' = \max(a_0, a_1)$, $b_0' = \min(b_0, b_1)$, and $b_1' = \max(b_0, b_1)$, and $\min(a, b)$ and $\max(a, b)$ denote the least and greatest of the points $a$ and $b$ in xy-lexicographic order, respectively. A **triangulation** $T$ of the set $P$ of points in $\mathbb{R}^2$ is a set $T$ of (nondegenerate) triangles such that: 1) the union of the triangles in $T$ is the convex hull of $P$; 2) the union of the vertices of all triangles in $T$ is $P$; and 3) the interiors of any two triangles in $T$ are disjoint. The sets of all vertices and edges in a triangulation $T$ are denoted $\mathrm{vertices}(T)$ and $\mathrm{edges}(T)$, respectively.

***Delaunay triangulations.*** One of the most common types of triangulations is the Delaunay triangulation [4]. Formally, a triangulation $T$ of a set $P$ of points is said to be **Delaunay** if the circumcircle of each face in $T$ contains no points from $P$. The Delaunay triangulation of a set of points is not necessarily unique, due to certain degeneracies that can arise. In some applications (such as in our work), it may be desirable to be able to obtain a unique triangulation of a given set of points. Fortunately, numerous methods exist for uniquely choosing one of all possible Delaunay triangulations of a point set. In our work, we employ the preferred-directions method [4], which yields a unique (Delaunay) triangulation of a point set, known as the **preferred-directions Delaunay triangulation (PDDT)**.

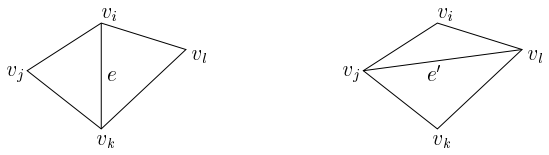



Fig. 1. An edge flip. The part of the triangulation associated with quadrilateral $v_i v_j v_k v_l$ (a) before and (b) after applying an edge flip to $e$.

***Mesh models.*** Consider an image function $\phi$ defined on $\Gamma = [0, W-1] \times [0, H-1]$ and sampled at points in $\Lambda = \{0, 1, ..., W-1\} \times \{0, 1, ..., H-1\}$ (i.e., a rectangular grid of width $W$ and height $H$). In the context of our work, a mesh model of $\phi$ is characterized by: 1) a set $P = \{p_i\}$ of sample points (where $P \in \Lambda$); 2) a triangulation of $P$; and 3) a set $Z = \{z_i\}$ of function values for $\phi$ at each point in $P$ (i.e., $z_i = \phi(p_i)$). The set $P$ must include all of the extreme convex hull points of $\Gamma$ (i.e., the four corners of the image bounding box) so that the triangulation of $P$ covers all of $\Gamma$. The mesh model is associated with a continuous piecewise linear interpolant $\hat{\phi}$ that approximates $\phi$. Typically, subject to a constraint on maximum model size, the mesh model is chosen to minimize the mean-squared error (MSE) $\epsilon$. For convenience herein, MSE is expressed in terms of **peak-signal-to-noise ratio (PSNR)** as given by $\mathsf{PSNR} = 20 \log_{10} (2^\rho - 1)/\sqrt{\epsilon}$, where $\rho$ is the sample precision in bits/sample. The quantities $P$ and $Z$ together comprise the geometry of the mesh, while the connectivity of $T$ constitutes the connectivity of the mesh. The work described herein addresses the problem of how to efficiently code a mesh model of the above form, with a focus on the connectivity coding.

***Local optimization procedure (LOP).*** One method of particular relevance to our work is the so called Lawson **local optimization procedure (LOP)** [5], a technique for finding an optimal triangulation of a given point set. In what follows, we provide a brief introduction to the LOP.

As a matter of terminology, an edge $e$ in a triangulation is said to be **flippable** if $e$ has two incident faces (i.e., is not on the triangulation boundary) and the union of its two incident faces forms a strictly convex quadrilateral $Q$. For a flippable edge $e$, an **edge flip** is an operation that replaces $e$ with the other diagonal $e'$ of $Q$, as shown in Fig. 1. As it turns out, every triangulation of a set of points can be transformed into every other triangulation (of the same set of points) by a finite sequence of edge flips [6]. This fact motivated Lawson to propose the LOP, an optimization scheme based on edge flips.

With the LOP, one must define an optimality criterion for edges. An edge $e$ being optimal means that the flipped counterpart of $e$ is not preferred over $e$ (i.e., the triangulation obtained by flipping $e$ is not more desirable than the original triangulation with $e$). The LOP deems a triangulation optimal if every flippable edge in the triangulation is optimal. Essentially, the LOP is an algorithm that simply keeps applying edge flips to flippable edges that are not optimal until all flippable edges are optimal (i.e., the triangulation is optimal). In more detail, the LOP works as follows. A priority queue, called the **suspect-edge queue**, is used to record all edges whose optimality is suspect (i.e., uncertain). Initially, all flippable edges in the triangulation are placed in the suspect-edge queue. Then, the following steps are performed until the suspect-edge queue is empty: 1) remove the edge $e$ from the front of the suspect-edge queue; 2) test $e$ for optimality; 3) if $e$ is not optimal, apply an edge flip to $e$ and place any newly suspect edges (resulting from the edge flip) on the suspect-edge queue. When the iteration terminates, the resulting triangulation is optimal. The LOP can be used to compute the PDDT of a point set by providing any valid triangulation as input to the LOP and specifying the PDDT criterion [4] as the optimality criterion for the LOP. In this case, the LOP will yield the PDDT as output. Although the PDDT produced is unique, the sequence of edge flips performed by the LOP is not and will depend on the particular priority function used for the suspect-edge queue.

## III. Proposed Mesh-Coding Framework

With the necessary background in place, we can now introduce the general framework from which our proposed mesh coding method is derived. As mentioned earlier, our approach is based on the IT scheme [2]. The IT method, although highly effective for coding mesh geometry, has no means for coding connectivity. Our proposed coding framework extends the IT coding scheme by adding to it a mechanism for coding connectivity. By providing the ability to code connectivity, our framework can be used to encode mesh models with arbitrary connectivity (unlike the IT scheme).

Our approach to connectivity coding is based on the idea of expressing the connectivity of a triangulation relative to the connectivity of a uniquely determined reference triangulation via a sequence of edge flips. More specifically, for a triangulation $T$ of a set $P$ of points, our approach represents the connectivity of $T$ as a sequence $S$ of edge flips that transforms the PDDT of $P$ to $T$. Since the PDDT of $P$ is unique, $P$ and $S$ completely characterize the connectivity of $T$. Given $P$ and $S$, we can always recover $T$ by first computing the PDDT of $P$ and then applying the sequence $S$ of edge flips to this PDDT. If $T$ is close to having PDDT connectivity, the sequence $S$ will be very short and the connectivity coding cost (in bits) will be very small. As $T$ deviates further from having PDDT connectivity, the sequence $S$ will grow in length and the connectivity coding cost will increase. In what follows, we describe the encoding and decoding processes in more detail.

***Encoding***. First, we consider the encoding process. As input, this process takes a mesh model, consisting of a set $P$ of sample points, a triangulation $T$ (of $P$), and the set $Z$ of function values (at the sample points). Given such a model, the encoding process outputs a coded bit stream, using an algorithm comprised of the following steps:

1) *Geometry coding.* Encode the mesh geometry (i.e., $P$ and $Z$) using the IT scheme.
2) *Sequence generation.* Generate a sequence $S$ of edge flips that transforms the PDDT of $P$ to the triangulation $T$.
3) *Sequence optimization.* Optionally, optimize the edge-flip sequence $S$ to facilitate more efficient coding.
4) *Sequence encoding.* Initialize the triangulation $\tau$ to the PDDT of $P$. Encode the edge-flip sequence $S$, updating the triangulation $\tau$ in the process.

In what follows, we explain each of steps 2 to 4 (from above) in more detail.

*Sequence generation (step 2).* In step 2 of our encoding framework, an edge-flip sequence is generated. We now explain in more detail how this is accomplished. To begin, we assign a unique integer label to each edge in the triangulation $T$, by numbering edges starting from zero using the lexicographic order for line segments defined in Section II. Next, we apply the LOP to the triangulation with the edge-optimality criterion chosen as the PDDT criterion, which will yield the PDDT of $P$. As the LOP is performed, each edge flip is recorded in the sequence $S' = \{s'_i\}_{i=0}^{L-1}$, where $s'_i$ is the label of the $i$th edge flipped. (Note that flipping an edge does not change its label.) After the LOP terminates (yielding the PDDT of $P$), each edge in the triangulation is assigned a new unique label using a similar process as above (i.e., by numbering edges starting from zero using the lexicographic order for line segments). Let $\rho$ denote the function that maps the original edge labels to the new ones. The edge-flip sequence $S = \{s_i\}_{i=0}^{L-1}$ that maps the PDDT of $P$ to $T$ is then given by $s'_i = \rho(s_{L-1-i})$. In other words, $S$ is obtained by reversing the sequence $S'$ and relabelling the elements of the sequence so that they are labelled with respect to the PDDT of $P$. The particular sequence $S$ obtained from the above process will depend on the specific priority scheme employed by the suspect-edge queue. In our work, the following three priority schemes were considered: first-in first-out (FIFO), last-in first-out (LIFO), and lexicographic (i.e., edges are removed from the queue in lexicographic order). As for which choice of priority scheme might be best, we shall consider this later in Section IV.

*Sequence encoding (step 4).* The sequence encoding process in step 4 of our encoding framework employs a scheme that numbers a subset of edges in a triangulation relative to a particular edge. So, before discussing the sequence encoding process further, we must first present this relative indexing scheme for edges. To begin, we first introduce some necessary terminology and notation. For an edge $e$ in a triangulation, $\mathrm{dirEdge}(e)$ denotes the directed edge oriented from the smaller vertex to the larger vertex of $e$ in xy-lexicographic order. For a directed (triangulation) edge $h$: 1) $\mathrm{opp}(h)$ denotes the directed edge with the opposite orientation (and same vertices) as $h$; 2) $\mathrm{next}(h)$ and $\mathrm{prev}(h)$ denote the directed edges with the same left face as $h$ that, respectively, follow and precede $h$ in counterclockwise order around their common left face; and 3) $\mathrm{edge}(h)$ denotes the (undirected) edge associated with $h$. The preceding definitions are illustrated in Fig. 2. With the preceding notation in place, we can now specify our relative indexing scheme for edges. Given a triangulation $\tau$ and a subset $\Theta$ of its edges and two distinct edges $e_1, e_0 \in \Theta$, the index of the edge $e_1$ relative to the edge $e_0$, denoted $\mathrm{relIndex}(e_1, e_0, \tau, \Theta)$, is determined as specified in Algorithm 1. (Note that $\mathrm{relIndex}(e_1, e_0, \tau, \Theta)$ is not necessarily equal to $\mathrm{relIndex}(e_0, e_1, \tau, \Theta)$.)

With the relative indexing scheme for edges having been introduced, we can now describe the edge-flip sequence encoding process in detail. Our coding scheme employs context-based adaptive binary arithmetic coding [7]. In our approach, the need to code nonbinary symbols arises, which requires the use of a binarization scheme to convert nonbinary symbols to sequences of binary symbols. For this purpose, the UI

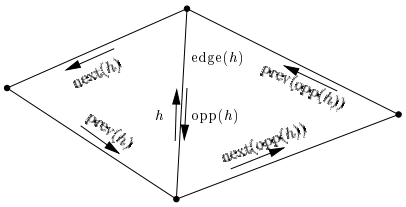



Fig. 2. Illustration of various definitions related to directed edges. An edge $e$ in a triangulation with two incident faces and the associated directed edges $h$ and $\mathrm{opp}(h)$.

**Algorithm 1** Calculating $\mathrm{relIndex}(e_1, e_0, \tau, \Theta)$ (i.e., the index of the edge $e_1$ relative to the edge $e_0$ in the set $\Theta$ of edges in the triangulation $\tau$.)

```
{q is FIFO queue of directed edges}
{h is directed edge and c is integer counter}
mark all edges in τ as not visited and h := dirEdge(e_0)
clear q, and insert opp(h) and then h in q
c := 0
while q not empty do
    remove element from front of q and set h to removed element
    if edge(h) ∈ Θ and not visited then
        mark edge(h) as visited
        if edge(h) = e_1 then
            output c as index of edge e_1 relative to edge e_0; and stop
        endif
        c := c + 1
    endif
    if opp(e) has left face then
        if edge(next(opp(h))) not visited then
            insert next(opp(h)) in q
        endif
        if edge(prev(opp(h))) not visited then
            insert prev(opp(h)) in q
        endif
    endif
endwhile
abort with error indicating e_1 ∉ Θ
```

binarization scheme described in [2] is employed. Given the set $P$ of vertices of the triangulation $T$ to be coded and the edge-flip sequence $S = \{s_i\}_{i=0}^{|S|-1}$ that transforms the PDDT of $P$ to $T$, the encoding process proceeds as follows:

1) Initialize the triangulation $\tau$ to the PDDT of $P$.
2) Encode $|S|$ as a 30-bit integer. If $|S| = 0$, stop.
3) Flip the edge $s_0$ in $\tau$.
4) Encode $s_0$ as an $m$-bit integer, where $m = \lceil \log_2(|\mathrm{edges}(\tau)|) \rceil$ (i.e., $m$ is the number of bits needed for an integer representing edge labels); if $|S| < 2$, stop.
5) For $i \in \{1, 2, \ldots, |S| - 1\}$:
   a) Let $r_i = \mathrm{relIndex}(s_i, s_{i-1}, \tau, \mathrm{flippableEdges}(\tau)) - 1$, where $\mathrm{flippableEdges}(\tau)$ denotes the set of all flippable edges in $\tau$.
   b) Flip the edge $s_i$ in $\tau$.
6) Encode $n = \left\lceil \log_2 \left(1 + \max\{r_1, r_2, \ldots, r_{|S|-1}\}\right) \right\rceil$ as a 30-bit integer (i.e., $n$ is the number of bits needed for an integer representing relative indexes).
7) Initialize the arithmetic coding engine, and start a new

arithmetic codeword.
8) For $i \in \{1, 2, \ldots, |S| - 1\}$, encode $r_i$ using UI$(n, 4)$ binarization (where $n$ and $r_i$ are as calculated above).
9) Terminate the arithmetic codeword.

In step 8 above, each element in the edge-flip sequence $S$ is coded relative to the previous element in the sequence. Such an approach is effective since the edge-flip sequence tends to exhibit locality (i.e., neighbouring elements in the sequence tend to be associated with edges that are close to one another in the triangulation).

*Sequence optimization (step 3).* As mentioned above, our coding scheme relies on the fact that the edge-flip sequence tends to exhibit some degree of locality. The purpose of the (optional) sequence-optimization step (i.e., step 3) in our encoding process is to attempt to improve the locality properties of the edge-flip sequence (prior to encoding) through optimization. In what follows, we describe this optimization process in more detail.

Before proceeding further, we must first introduce some notation and terminology related to the optimization process. Let $\mathrm{tri}_{T,S}(i)$ denote the triangulation obtained by applying the first $i$ edge flips in the sequence $S$ to the triangulation $T$; and let $\mathrm{tri}_{T,S}$ denote the triangulation obtained by applying all of the edge flips in the sequence $S$ to the triangulation $T$. Two edge-flip sequences $S$ and $S'$ are said to be **equivalent** if $\mathrm{tri}_{T,S} = \mathrm{tri}_{T,S'}$ (i.e., the application of each edge-flip sequence to the triangulation $T$ produces the same final triangulation). Let $\mathrm{swap}(S, i, j)$ denote the new sequence formed by swapping the $i$th and $j$th elements in the sequence $S$. Let $\mathrm{erase}(S, i, j)$ denote the new sequence formed by removing elements $s_i, s_{i+1}, \ldots, s_j$ from the sequence $S$ (where elements in $S$ with index greater than $j$ are shifted downwards by $j - i + 1$ positions to form the new sequence). Two adjacent elements of an edge-flip sequence $S$ with indices $i$ and $i + 1$ are said to be **swappable** if they correspond to edges that are not incident on the same face of the triangulation $\mathrm{tri}_{T,S}(i)$.

For a given edge-flip sequence, it is possible to find many (distinct) sequences that are equivalent (in the sense of "equivalent" as defined above). Some of these equivalent sequences, however, have better locality properties than others, and therefore lend themselves to more efficient coding. The optimization process attempts to produce an edge-flip sequence with better locality by applying a series of transformations to the sequence that results in an equivalent sequence. Two types of transformations are of interest. First, if the $i$th and $(i+1)$th elements in the sequence $S$ are swappable (as defined above), swapping these elements will yield an equivalent sequence (i.e., $\mathrm{tri}_{T,S} = \mathrm{tri}_{T,\mathrm{swap}(S,i,i+1)}$). Second, if the $i$th and $(i+1)$th elements in $S$ are equal, the deletion of these two elements will yield an equivalent sequence (i.e., $\mathrm{tri}_{T,S} = \mathrm{tri}_{T,\mathrm{erase}(S,i,i+1)}$). With the above in mind, the optimization process works as follows. The optimization algorithm makes repeated passes over the elements of the sequence $S$, until a full pass completes without any change being made to $S$. Each pass performs the following for $i \in \{0, 1, \ldots, |S| - 2\}$:

1) If $s_i = s_{i+1}$, $S := \mathrm{erase}(S, i, i + 1)$ (i.e., delete $i$th and $(i + 1)$th elements from $S$).
2) Otherwise, if $s_i$ and $s_{i+1}$ are swappable and the binary decision function $d_S(i) \neq 0$, $S := \mathrm{swap}(S, i, i + 1)$ (i.e., swap the $i$th and $(i + 1)$th elements in $S$) and $i := i + 1$. The binary decision function $d_S(i)$, which is used to determine if the $i$th and $(i + 1)$th elements in $S$ should be swapped, is a free parameter of our method and will be described in more detail shortly.

In our work, we considered three choices for the decision function $d_S$. To assist in specifying these choices, we introduce some additional notation in what follows. For $x, y \in \mathbb{R}$, $\mathrm{lt}(x, y)$ is 1 if $x < y$ and 0 otherwise; and $\mathrm{lte}(x, y)$ is 1 if $x \leq y$ and 0 otherwise (i.e., lt and lte are boolean-valued functions for testing the less-than and less-than-or-equal conditions). Let $c_S(i)$ denote the approximate cost of coding the $i$th edge flip in the sequence $S$, where $c_S(i) = \mathrm{relIndex}(s_i, s_{i-1}, \mathrm{tri}_{T,S}(i), \mathrm{flippableEdges}(\mathrm{tri}_{T,S}(i)))$ for $i \in \{1, 2, \ldots, |S| - 1\}$ and $c_S(i) = 0$ for $i \in \{0, |S|\}$, and $\mathrm{flippableEdges}(T)$ denotes the set of all flippable edges in the triangulation $T$. For convenience, let $c(i)$ and $c'(i)$ denote $c_S(i)$ and $c_{\mathrm{swap}(S,i,i+1)}(i)$, respectively (i.e., $c(i)$ and $c'(i)$ represent the cost without and with the $i$th and $(i+1)$th edges swapped, respectively). With this notation in place, the three choices for the decision function $d_S$ considered in our work are given by the following:

$$\text{rule 1:} \qquad d_S(i) = \mathrm{lt}\left(\sum_{k=i}^{i+2} c'(k), \sum_{k=i}^{i+2} c(k)\right)$$

(i.e., elements are swapped if this would strictly reduce the overall sum of the relative indexes to be coded);

$$\text{rule 2:} \quad d_S(i) = \left[\prod_{k \in I} \mathrm{lte}\left(c'(k), c(k)\right)\right] \left[\max_{k \in I} \mathrm{lt}(c'(k), c(k))\right],$$

where $I = \{\max\{1, i\}, \ldots, \min\{|S|-2, i+2\}\}$ (i.e., elements are swapped if this would not increase the cost of any of the three relative indexes to be coded that are affected by the swap and at least one cost would be strictly reduced); and

$$\text{rule 3:} \qquad d_S(i) = \begin{cases} \mathrm{lt}\left(c'(i), c(i)\right) & i \geq 1 \\ 0 & i = 0 \end{cases}$$

(i.e., elements are swapped if this would strictly reduce the $i$th relative index to be coded). As for which choice of $d_S$ might be best, we will explore this later in Section IV.

***Decoding***. Now, we consider the decoding process. Given a coded bit stream as input, this process outputs the corresponding mesh model, characterized by a set $P$ of sample points and their corresponding set $Z$ of function values, and a triangulation $T$ of $P$. The decoding process consists of the following steps:

1) *Geometry decoding.* Decode the mesh geometry using the IT scheme, yielding $P$ (and $Z$).
2) *Sequence decoding.* Initialize the triangulation $\tau$ to the PDDT of $P$. Label the edges in the triangulation in an identical manner as the encoder (i.e., lexicographic order starting from zero). Decode the edge-flip sequence, updating $\tau$ in the process. After each edge flip is decoded, the edge flip is applied to the (current) triangulation $\tau$. The final value of $\tau$ corresponds to the decoded triangulation $T$.

In step 2 of the decoding process above, the steps involved in the decoding of an edge-flip sequence simply mirror those for encoding (described earlier).

TABLE I. SEVERAL OF THE MESH MODELS USED IN OUR WORK

| Name | Category | Vertex Count | Edge Count | Sequence Length† | Relative Sequence Length (%)† |
|---|---|---|---|---|---|
| A1 | A | 15728 | 47080 | 1070 | 2.3 |
| A2 | A | 6881 | 20632 | 1421 | 6.9 |
| A3 | A | 5242 | 15374 | 2298 | 14.9 |
| B1 | B | 7864 | 23536 | 7326 | 31.1 |
| B2 | B | 7864 | 23514 | 6720 | 28.6 |
| B3 | B | 2621 | 7797 | 3235 | 41.5 |

† Sequence generated with lexicographic priority scheme.

## IV. PROPOSED MESH-CODING METHOD

In the previous section, we introduced our proposed framework for mesh coding. This framework has two free parameters, namely, the choice of priority scheme (used by the LOP) and the choice of decision function (used by sequence optimization). In this section, we study how different choices for these parameters affect coding efficiency. Based on these results, we recommend a particular choice for these parameters, which corresponds to the specific coding method proposed herein.

***Test data.*** Before proceeding further, a brief digression is necessary in order to introduce the test data used in our work. Herein, we employed a set of 50 mesh models of images (with arbitrary connectivity) that were produced by the state-of-the-art mesh-generation scheme proposed in [1]. Since the efficiency of our proposed coding approach depends on the extent to which mesh connectivity deviates from being Delaunay, we have grouped our meshes into two categories, A and B, based on the extent of this deviation. To quantify the extent to which the connectivity of a mesh deviates from being Delaunay, we used the length of the edge-flip sequence required to transform the mesh connectivity to PDDT connectivity, measured as a percentage of the total number of edges in the mesh. The 25 meshes in category A have connectivity relatively close to being Delaunay (i.e., the relative length of the edge-flip sequence is less than or equal to 15%), while the 25 meshes in category B have connectivity that deviates relatively more from being Delaunay (i.e., the relative length of the edge-flip sequence is greater than 15%). Herein, we present statistical results taken across all of our meshes as well as results for individual meshes. For consistency, when presenting results for individual meshes, we focus on the six representative meshes listed in Table I, where the meshes A1, A2, and A3 are from category A and the meshes B1, B2, and B3 are from category B.

***Choice of priority scheme.*** To begin, we consider how different choices of priority scheme used by the LOP (in step 2 of the encoding process) affect coding efficiency. For each of the 50 models in our test set, we coded the mesh using each of the three priority schemes (namely, LIFO, FIFO, and lexicographic, as introduced earlier) and measured the resulting bit rate. The results obtained are shown in Table II. In each case, the best result is highlighted in bold font. From both the results for individual meshes in Table II(a) and the overall statistical results in Table II(b), it is clear that the lexicographic priority scheme is most effective, consistently leading to the lowest bit rate in the individual cases as well as overall. In fact, a more detailed examination of the results shows that the lexicographic scheme performs best in all 50/50 of the test cases. A careful analysis shows that the superior performance

TABLE II. COMPARISON OF THE CONNECTIVITY CODING PERFORMANCE OBTAINED WITH THE VARIOUS PRIORITY SCHEMES. (A) INDIVIDUAL RESULTS. (B) OVERALL RESULTS.

(a)

| Dataset | Coded Size (bits/vertex) | | |
|---|---|---|---|
| | LIFO | FIFO | Lex.† |
| A1 | 1.02 | 1.02 | **0.86** |
| A2 | 2.54 | 2.53 | **1.98** |
| A3 | 4.01 | 3.95 | **2.74** |
| B1 | 8.36 | 8.42 | **7.70** |
| B2 | 7.63 | 7.87 | **7.24** |
| B3 | 9.81 | 9.93 | **9.03** |

† lexicographic

(b)

| Category | Mean Coded Size (bits/vertex) | | |
|---|---|---|---|
| | LIFO | FIFO | Lex.† |
| A | 1.41 | 1.41 | **1.16** |
| B | 8.60 | 8.80 | **8.02** |
| Overall | 5.00 | 5.10 | **4.59** |

† lexicographic

TABLE III. COMPARISON OF THE CONNECTIVITY CODING PERFORMANCE OBTAINED WITH THE VARIOUS DECISION FUNCTIONS. (A) INDIVIDUAL RESULTS. (B) OVERALL RESULTS.

(a)

| Dataset | Coded Size (bits/vertex) | | |
|---|---|---|---|
| | Rule 1 | Rule 2 | Rule 3 |
| A1 | 0.82 | 0.84 | **0.76** |
| A2 | 1.92 | 1.93 | **1.78** |
| A3 | 2.70 | 2.64 | **2.54** |
| B1 | 7.86 | 7.66 | **7.44** |
| B2 | 7.43 | 7.20 | **6.98** |
| B3 | 9.21 | 8.98 | **8.67** |

(b)

| Category | Mean Coded Size (bits/vertex) | | |
|---|---|---|---|
| | Rule 1 | Rule 2 | Rule 3 |
| A | 1.12 | 1.13 | **1.06** |
| B | 8.11 | 7.95 | **7.64** |
| Overall | 4.62 | 4.54 | **4.35** |

of the lexicographic priority scheme is due to its ability to produce edge-flip sequences with better locality properties.

***Choice of decision function.*** Next, we consider how different choices of decision function in the sequence-optimization step (i.e., step 3) of the encoding process affect coding efficiency. For each of the 50 models in our test set, we coded the mesh using each of the three decision rules (namely, rules 1, 2, and 3, as introduced earlier) and measured the resulting bit rate. The results obtained are given in Table III, with results for six individual meshes in Table III(a) and overall statistical results for all meshes in Table III(b). In each case, the best result is highlighted in bold font. (In this experiment, the lexicographic priority scheme was employed, although similar results were obtained with other priority schemes as well.) By examining the results of Tables III(a) and (b), it is clear that rule 3 consistently performs best, leading consistently to the lowest bit rate. As it turns out, rule 3 outperforms the other two rules in all 50/50 of the test cases. A more careful analysis of the results shows that rule 3 tends to perform more swap operations, allowing locality to be improved more than with rules 1 and 2.

***Proposed method.*** As demonstrated by the above experimental results, our proposed mesh-coding framework is most effective when the priority scheme is chosen as lexicographic and the decision function is chosen as rule 3. Therefore, we recommend that these particular choices be employed in our proposed framework. In the remainder of this paper, we will refer to our framework with these recommended choices as our proposed method for mesh coding.

## V. EVALUATION OF PROPOSED MESH-CODING METHOD

Having introduced our proposed mesh-coding method, we now evaluate its coding performance by comparing it to a simple baseline approach. This baseline coding scheme is identical to our proposed method, except that the edge-flip sequence (which conveys connectivity information) is coded using a very trivial approach. In particular, the baseline scheme encodes the edge-flip sequence as follows. First, the length of

TABLE IV. CODING PERFORMANCE COMPARISON OF THE PROPOSED AND BASELINE METHODS. (A) INDIVIDUAL RESULTS. (B) OVERALL RESULTS.

(a)

| | Connectivity Size (bits/vertex) | | Total Size (bits/vertex) | |
|---|---|---|---|---|
| Dataset | Proposed | Baseline | Proposed | Baseline |
| A1 | **0.76** | 1.09 | 15.07 | 15.40 |
| A2 | **1.78** | 3.11 | 14.86 | 16.19 |
| A3 | **2.54** | 6.15 | 11.38 | 14.99 |
| B1 | **7.44** | 13.98 | 21.84 | 28.38 |
| B2 | **6.98** | 12.83 | 22.59 | 28.44 |
| B3 | **8.67** | 16.08 | 24.28 | 31.69 |

(b)

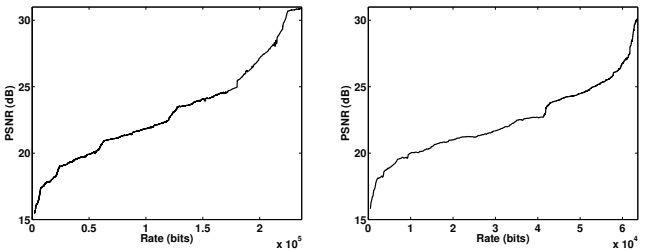
| | Mean Connectivity Size (bits/vertex) | | Mean Total Size (bits/vertex) | |
|---|---|---|---|---|
| Category | Proposed | Baseline | Proposed | Baseline |
| A | **1.06** | 1.72 | 14.87 | 15.54 |
| B | **7.64** | 13.92 | 24.83 | 31.11 |
| Overall | **4.35** | 7.82 | 19.85 | 23.32 |

the edge-flip sequence is output as a 30-bit integer. Then, each element in the edge-flip sequence is output as an $n$-bit integer, where $n = \lceil \log_2 |\text{edges}(T)| \rceil$ and $T$ is the triangulation associated with the mesh.

***Proposed vs. baseline.*** To compare the performance of the proposed and baseline methods, we proceeded as follows. For each of the 50 mesh models in our test set, we coded the model using each of the proposed and baseline methods and measured the number of bits required to encode the connectivity information as well as the complete mesh. The results obtained are shown in Table IV, with results for six specific meshes in Table IV(a) and overall statistical results for all 50 meshes in Table IV(b). Since the difference between the two methods is in the connectivity coding alone, we will focus our attention on comparing the connectivity coding numbers from these tables in what follows. Examining the individual results in Table IV(a), we can see that our proposed method outperforms the baseline method in all 6/6 of the test cases by margin of at least 0.33 bits/vertex (for connectivity coding). The overall results in Table IV(b) are consistent with the individual results, with our proposed method significantly outperforming the baseline scheme for meshes from both categories A and B. In fact, a more detailed analysis of the results shows that the proposed method beats the baseline scheme in all 50/50 of the test cases, by a margin of up to 11.56 bits/vertex (for connectivity coding).

***Proposed vs. traditional methods.*** Perhaps, it is also worth noting that in the case of all 25 meshes in category A, our proposed method consistently requires less than the 3.67 bits/vertex for connectivity coding often cited for more traditional connectivity coding methods. Thus, it seems likely that our method would also be competitive with these other approaches for meshes with connectivity that does not deviate too far from being Delaunay. Furthermore, our proposed method is progressive, unlike many traditional connectivity coding methods.

***Progressive performance.*** As indicated earlier, our proposed coding method is progressive. To illustrate the progressive capability of our coding scheme, we provide a brief example. For two of the meshes from Table I (namely, A1 and B3), we measured the reconstruction quality of the image produced from the decoded mesh model as a function of bit rate. The results obtained are shown in Figs. 3(a) and (b), with the right side of each graph corresponding to lossless decoding of the original mesh model. As we can see from the figures,




(a) (b)

Fig. 3. Progressive coding results for the (a) A1 and (b) B3 meshes.

an incrementally better quality image reconstruction (i.e., with higher PSNR) can be decoded as the bit rate increases. Such functionality is desirable in many applications.

## VI. CONCLUSIONS

In this paper, we have proposed a progressive coding method for mesh models of images that can handle meshes with arbitrary connectivity. Our method is such that its coding efficiency is highest for meshes that have connectivity close to being Delaunay, with performance degrading slowly as connectivity moves further away from Delaunay connectivity. Through experimental results, we showed that our method is significantly better than a simple baseline coding scheme. Furthermore, we showed that it is likely that our proposed method can outperform traditional connectivity coding methods for meshes that do not deviate too far from Delaunay connectivity (i.e., the relative length of the edge-flip sequence that transforms the mesh connectivity to PDDT connectivity is less than about 15%). Since Delaunay meshes have many desirable properties for approximation, it is quite common to encounter meshes that are close to being Delaunay in practice. Therefore, the excellent performance of our method for such meshes is of great practical value. Furthermore, our coding method yields very substantially more compact representations than a simple naive coding scheme like the baseline method described earlier. Also, our coding method is progressive, which can be beneficial in many applications. Thus, our proposed coding method can benefit applications that must efficiently store or transmit mesh models of images.

## REFERENCES


[1] P. Li and M. D. Adams, "A tuned mesh-generation strategy for image representation based on data-dependent triangulation," *IEEE Transactions on Image Processing*, vol. 22, pp. 2004–2018, May 2013.

[2] M. D. Adams, "An efficient progressive coding method for arbitrarily-sampled image data," *IEEE Signal Processing Letters*, vol. 15, pp. 629–632, 2008.

[3] D. King and J. Rossignac, "Guaranteed 3.67v bit encoding of planar triangle graphs," in *Canadian Conference on Computational Geometry*, Vancouver, BC, Canada, 1999.

[4] 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.

[5] 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.

[6] ——, "Transforming triangulations," *Discrete Mathematics*, vol. 3, pp. 365–372, 1972.

[7] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.